A Knowledge Based Engineering Approach for Design Automation

(A Case Study: Rectangular Turret of Transformer)

A Thesis submitted
In partial fulfillment of the requirement for
The award of degree of

MASTER OF ENGINEERING
in
CAD/CAM & ROBOTICS

Submitted By
Anshu Dwivedi
Roll No. 8048103

Under the supervision of
Mr. Shiv Prasad Subudhi
Consultant, TCS Limited, Noida

And guidance of

Mr. Chanpreet Singh
Lecturer, M.E.D., T.I.E.T.

Dr. D. Gangacharyulu
Associate Professor, Ch.E.D, T.I.E.T.

Mechanical Engineering Department
Thapar Institute of Engineering and Technology
(Deemed University), Patiala-147004 (INDIA)

June, 2006

CERTIFICATE

This is to certify that the thesis titled, **"A Knowledge Based Engineering Approach For Design Automation (A Case Study: Rectangular Turret of Transformer)",** submitted by **Mr. Anshu Dwivedi,** in partial fulfillment of the requirement for the award of **Master of Engineering (CAD/CAM & Robotics) at Thapar Institute of Engineering & Technology (Deemed University), Patiala** is a bonafide work carried out by him under our guidance and supervision and no part of this thesis has been submitted for the award of any other degree.

(Mr. Shiv Prasad Subudhi)
Consultant, Tata Consultancy Services Limited,
Noida – 201301

(Mr. Chanpreet Singh)
Lecturer, M.E.D.,
Thapar Institute of Engg. & Tech.
Patiala-147004 (Punjab)

(Dr. D. Gangacharyulu)
Associate Professor, Ch.E.D,
Thapar Institute of Engg. & Tech.
Patiala-147004 (Punjab)

Counter signed by:

(Dr. S. K. Mohapatra)
Professor & Head, M.E.D.,
Thapar Institute of Engg. &Tech.
Patiala-147004 (Punjab)

(Dr. T. P. Singh)
Dean Academic Affairs
Thapar Institute of Engg. & Tech.
Patiala-147004 (Punjab)

ACKNOWLEDGEMENT

I express my sincere gratitude to my guides, Mr. Shiv Prasad Subudhi, Consultant, Tata Consultancy Services Limited (TCSL), Dr. D. Gangacharyulu, Associate. Professor, Chemical Engineering Department, Thapar Institute of Engineering & Technology, Patiala, and Mr. Chanpreet Singh, Lecturer, Mechanical Engineering Department, Thapar Institute of Engineering & Technology, Patiala, for their valuable guidance, proper advice, painstaking and constant encouragement during the course of my work on this thesis.

I am greatly thankful to Mr. Sunil Kumar, I.T. Analyst, TCSL, and Mr. Anant Gupta, Assistant Consultant, TCSL, who devoted their valuable time and helped me in all possible ways towards successful completion of my thesis work. I am also thankful to Mr. Mayank Tyagi, I.T. Analyst, TCSL, for his inspiration and moral support which help me in completing the thesis.

I would like to thank entire staff of TCSL and M. E. D., T.I.E.T., Patiala, for their valuable help and support.

(Anshu Dwivedi)

Table of contents

List of Abbreviations

1. **ADS –** Automated Design Systems
2. **AI –** Artificial Intelligence
3. **API –** Application Programming Interface
4. **BOM –** Bill of Material
5. **CAD –** Computer Aided Design
6. **CAM –** Computer Aided Manufacturing
7. **CAE –** Computer Aided Engineering
8. **CBR –** Case Based Reasoning
9. **CAAD –** Computer Aided Architectural Design
10. **CBR –** Case Based Reasoning
11. **CBD –** Case Based Design
12. **CIMS –** Computer Integrated Manufacturing Systems
13. **DART--** Design Analysis Response Tool
14. **DFA.--**Design for assembly
15. **DFM --**Design for Manufacturing
16. **ES –** Expert systems
17. **ED –** Engineering design
18. **FEM --**Finite Element Method
19. **KBE –** Knowledge Based Engineering
20. **KBS –** Knowledge Based Systems
21. **KBDSS --**Knowledge-based design support system
22. **KEE –** Knowledge Enabled Engineering
23. **KF –** Knowledge Fusion
24. **MDA –** Mechanical Design Automation
25. **OOP –** Object oriented Programming
26. **UG/NX –** Unigraphics / Next Generation
27. **UI –**User Interface
28. **SMEs--** Small and Medium sized Enterprises
29. **2 D –** Two Dimensional
30. **3 D –** Three Dimensional

List of Figures

List of Tables

Abstract

The success of manufacturing companies depends on their ability to produce high-quality products at the lowest cost. This applies to a transformer industry that aims to create designs that are optimized for manufacture. In striving for lower cost, it may be worthwhile to focus on the design phase of product development, since most of the product cost is committed there. This cost is, however, not often seen until it is allocated later or downstream in the product development process. Currently, design phase is often conducted on conventional computer-aided solid modeling design tools in the transformer industry, which takes time and entails the risk of missing design flaws. Knowledge based engineering (KBE) has become a practical method of visualizing manufacturing cost and enabling product analysis by simulating product development activities in design for manufacturing support tools. The aim of this work is to discover how knowledge based engineering (KBE), an approach which includes engineering design, knowledge based engineering (KBE) and similar knowledge intensive methods, can be used to improve productivity of transformer industry.

Further advent of the UG/Knowledge Fusion proved to revolutionize the way an organization looks at solving and automating repetitive engineering problems. The presented work shows how an automated design system can be used to capture and present knowledge extracted from performance, manufacturing and maintenance activities, creating a better foundation for making decisions regarding conceptual design. Engineers can then change the design and directly assess the life cycle cost allowing fast iterations, based on knowledge from design, manufacturing and maintenance disciplines.

CHAPTER 1

Introduction

1.1. Background

The economic success of manufacturing firms depends on their ability to identify the needs of customers and quickly create products that meet these needs and can be produced at low cost [1]. This is applicable for manufacturers aiming at optimizing design activities for manufacturing.

Traditionally, design solid modeling tools are primarily used for activities that occur at the end of the design process. Such usage of solid modeling (CAD) tools, for instance, during detailing geometry of an artifact, is in generating a production drawing, or in documenting geometry in a digitized form. CAM systems are conventionally used to program machining or cutting instructions on the NC machines for a part whose mock-up design, clay or plaster prototype may already exist. CAE systems are used to check the integrity of the designed artifact (such as structural analysis for stress, thermal, etc.), when most of the critical design decisions have already been made. Studies have revealed that 75% of the eventual cost of a product is determined before any full-scale development or a CAD tool usage actually begins [1].

Most conventional computer-aided solid modeling design tools are not really "capture" tools. The need for "change" after a solid model is initially "built," however, is all but inevitable. Often, design solid models are built from scratch only when engineering activity is complete, and are validated via a series of design reviews. These results are generally completed by what we will call work groups. The Design Work Groups typically:

- Document the design through CAD software only after the completion of major engineering processes and after resolving all of the pressing engineering issues.
- Captures the geometry in a static form, such as lines and surfaces.

This static representation is actually a documentation that tells a designer what the final design looks like but not how it has come to be. If changes are required in the design, a new CAD solid model is recreated using some type of computer-aided "re-do" or

"back-tracking" methods. Such methods of activating change or modification (e.g., a redo or a backtracking) can be extremely time consuming and costly being that late in the life-cycle process [1].

In such static representations of solid geometry, configuration changes cannot be handled easily, particularly when parts and dimensions are linked. In addition to the actual process that led to the final solid design, most of the useful lessons learned along the way are also lost. In the absence of the latter, such efforts have resulted in loss of configuration control, proliferation of changes to fix the errors caused by other changes, and sometimes ambiguous designs. Hence, in recent days, during a product design, development and delivery process, emphasis is often placed on the methods used for capturing the lifecycle intent with ease of modifications in mind. The power of a "knowledge capture" tool comes from the methods used in capturing the design intent initially so that the anticipated changes can be made easily and quickly later if needed. By capturing "design intent" as opposed to a "static solid geometry," configuration changes could be made and controlled more effectively using the power of the computer than through the traditional solid model or CAD attributes (such as line and surfaces) [2].

1.2. Design Automation

Automation methodologies applied primarily to manufacturing operations have proven that it is possible to achieve dramatic improvements in cost, quality, and time by focusing on process performance. Automation is linked to speed, efficiency, and waste. The essence of automation is to concentrate effort on removing waste while improving operations. The simplified notion of design automation is to remove waste from all aspects of the product and associated development process before it ever gets to the manufacturing floor [2].

Design automation provides industrial companies with a complete arsenal to attack waste, both in how the product is produced (process improvement using lean manufacturing techniques) and in what product gets produced (product improvement using lean design).To be cost competitive, front-line staff must learn to see waste and relentlessly pursue its elimination not only on the manufacturing floor but also on their engineering drafting boards and computer screens [2].

1.3. Opportunities For Reduction Of Waste In Design

The areas of waste in manufacturing, which are relatively easy to identify and quantify, can be extrapolated to apply to other product development processes. The table 1.1 identifies and evaluates waste in product design, summarizing the findings of two studies [2].

Table 1.1: Opportunities for reduction of waste in design

Opportunities for reduction of waste in design	
Area of waste reduction	**% of design waste**
Design never used, completed, or delivered	Unknown
Downtime while finding information, waiting for test, results, etc	33-50 %
Unnecessary documents and prototypes	
Underutilization of design knowledge	18 %
Over design, such as features customers don't need	8 %
Validating manufacturing errors early in design process	17 %
Poor design producing product defects	15 %

- In a recent analysis of Automotive Tier 1 suppliers, the pre-discovery process identified between 33% and 50% of design cost spent on delays finding information, waiting for information and producing unnecessary information such as paper documents and physical prototypes.

- An A.T. Kearney study (The Line on Design, 2003) discovered a total of 58% of cost wasted in the design of railway vehicles including failure to exploit previous design experience, over design and poor design for manufacturing.

To illustrate how costs can be eaten up by design waste, consider the costs involved in manufacturing a railway vehicle. Of the total direct cost (which includes internal production and material), roughly 8 percent is often for gold plating. These are functions and features that the customer is not willing to pay for. Another 11 percent is for excessive safety factors and over specifications.

- These are requirements that are either unnecessary or left over from a previous product design and therefore irrelevant. For example, high-grade fire resistance on train seats (required on trains that travel through tunnels) on a train that never goes through a tunnel. About 15 percent of the costs to produce railway vehicles are often due to sub-optimal concepts, such as inferior solutions already in production

on other vehicles in-house or on competitor vehicles. An example might be a floor system that is 40 percent more expensive than in equivalent trains of competitors.

- Almost 7 percent of design cost is attributed to "lazy designs," referring to designs that do not fully use the capacity of components or use superior materials rather than required materials. Roughly 5 percent is due to poor design for manufacturing and assembly, such as a complex seat attachment that, if simplified, would reduce assembly time by 60 percent. Finally, 12 percent of design waste is due to failure to design for supply. These might be panels purchased from domestic suppliers with high factor costs. In all, 58 percent of the total cost is design waste, and the total cost of the vehicle could be reduced by 30 percent over two years by simply attacking it.

1.4. Benefits Of Reducing Waste In Product Development

The categories of waste in manufacturing can be mapped to associated areas targeted for waste reduction in product development [12], as shown in table 1.2.

Table 1.2: Benefits of reducing waste in product development

7 deadly wastes	**Applied to design**
Delays	**Reduce delays** • Finding information, waiting on test results • Unnecessary documents, physical prototypes
Movement and transport	
Excess production and early production	**Maximize design reuse** • Learning from past design process • Reducing unnecessary features • Designs never used, completed or delivered
Inventory	
Poor process design	**Improve process efficiency** • Underutilization of design knowledge • Early validating of manufacturing errors
Inefficient performance of process	**Reduce defects** • Poor design • Warranty issues
Making defective items	

When an objective is to cut lead times, reduce costs and increase quality, a focus on the concept phase can be valuable as a majority of the product cost is committed in the concept phase of engineering design. This cost is, however, often not seen until it is allocated later, or downstream, in the product development (PD) process, e.g., during manufacturing. Therefore, there is a risk that engineering designers commit more

manufacturing cost than needed. Even though design flaws that result in poor manufacturability might be found manually before the actual manufacturing operation begins, it is plausible that time could be saved if the design flaw was found during conceptual design. This extra time for redesign need to be reduced as a part in halving the time to market. To cope with this, the manufacturing firm must employ methods and tools that enhance the engineering design process [12].

Further the product development process proceeds; the more knowledge is gained from the performed activities. Because design is an open-ended problem, each new piece of added information affects what decision seems to lead to the best solution [13]. Decisions more or less constrain the amount of possible solutions, with those being taken early on having a major impact as they limit the solution space.

Due to the trends presented above, business is changing and creating new requirements on how to develop products in business-to-business relations, forcing companies to optimize their part of the total product system.

1.5. Problem Formulation

1.5.1 Motivation

A product development process consists of numerous activities to be performed. Developing a product in Business-to-Business relations requires information to be shared between companies. The information shared comes from activities done within each company in their respective processes. An activity to be performed needs information from other activities within or outside the company. The output from one activity may then be the input for another. The lead-time for the total product development process is then dependent on the lead-time for each activity and for sharing the information needed to perform the activities.

Little is known about the final product at the beginning of the product development process. Each performed activity gathers more knowledge about what the final properties of the product will be. When developing a commercial transformer, requirements and interface information are exchanged between several companies working together as well as between disciplines inside each company. The view of the

conceptual phase where companies work together has been left to the continuation of the research.

1.5.2. Research Gap

Don Runkle, Vice Chairman, Delphi Automotive, USA observed that "It is hard to see the waste in engineering design and easy to see it in manufacturing".

Product development drives business growth but at the same time, it is expensive. The industry average is one product success for every three thousand ideas. A structured and reliable approach is required in engineering to ensure high return on the innovation investment.

Transformers are custom designed products. Practically every order requires a separate design and new set of drawings. Though certain assemblies/components have been standardized, these constitute a small part of the total design effort. A typical transformer may consist of as many as 3000 components/sub-assemblies, and the complete engineering information requires 500 drawings/Bill of material. The effect of this huge design/drawing work is that the issue of engineering information takes as much as 25% of the total delivery cycle of the transformer [4].

All the components /sub-assemblies/assemblies that go into a transformer have to be so designed that they can be assembled properly; any mismatch identified at the time of manufacture is difficult to rectify. In addition, all modifications carry a certain penalty in terms of the cost of the product as well as the time taken to manufacture. Since a large number of drawings have to be made, they are made simultaneously, adding to the chances of mismatch [4].

Any engineering drawing has basically, two sets of dimensions that are most important part of drawing. These are [4]:

- Dimensions which are independent variables, i.e. they are derived from the specifications, ratings, customers requirement, etc.
- Dimensions which are dependent i.e. they are derived from either the independent dimensions based on certain formulae or are derived from certain standard practices or design rules.

Using this logic, it is possible, if the independent dimensions are available, to work out the dependent dimensions, and thus all the dimensions needed to make the drawing can be worked out.

If we take a top down view of the drawing process, we will see that an assembly consists of a number of components. Certain dimensions derived from the assembly are common to a number of components. When the drawings are being made, any mistake in the common dimensions, either for assembly or component drawing would create a mismatch. In the ordinary courses, all the drawings with common dimensions should be checked at the same time to ensure correctness. The fact that all such drawings may not be available, as well as the possibility of human error creates mismatch that have to be rectified later at a cost to the organization [4].

1.5.3. Research Problem and Solution

From above study a question can be formulated in order to define the scope of this research:

"How productivity of transformer industry can be enhanced in the design phase of the product development process by means of knowledge based engineering"

A new approach has been taken to maximize the productivity. Computer programs would be developed, which take the dimensions identified as an independent variables as inputs. The logic for calculating dependent variables is built into program, and dependent variables calculated. Based on these dimensions and the relations between them, a 3-D model of the Component/assembly is generated by the program. Finally, the program also generates the 2-D product drawings. Since separate programs would cover different assemblies, all the common dimensions are available while making the component drawings. The chances of errors/mismatch are thus reduced. Since all the data required making the Bill of material is available, the program generates the bill of material also. The advantages of such a system are:

- By giving one set of data for an assembly, a number of drawings and the Bill of material would be generated. Thus the time taken to make the drawings and the bill of material is reduced.

- Since the common dimensions would be given only once, errors due to mismatch is reduced.
- As a 3-D model would be available, the designers can rotate the model and visualizes it from different angles and thus make improvements.
- Fouling between different components /assemblies, excess clearance would be visible on the computer screen in 3-D, allowing correctness to be before the drawings are released, or components procured.
- The bill of material would be generated by the program, and can be stored in a data base for use by the planning groups.
- Since all the changes are built in the program, there would be commonality of logic. Mistakes due to adopting the wrong formulae by an individual are therefore eliminated.
- Any changes in the design philosophy would entail modifications in the program and need to ensure proper changes in the individual designs would be elminated.This would ensure better design control.

Based on above philosophy an Automated Design System (ADS) with the help of knowledge based engineering (KBE) would be developed. The system is suitable for the design of the transformers following any transformer manufacturer organization's design methods. The system would be very comprehensive and generate analyzable model and assembly, drawings and bill of material for a large variety of transformers with different ratings.

The system would be designed with menu driven prompts (as shown in figure 1.1) so that a user without any knowledge of 3-D modeling software or database package can use the system. The only need is that the user should know how to use this system. The basic modeling software used to develop this system is Unigraphics due to its strong knowledge based approach. The menu screens for making the system user friendly and the coding has been developed in Knowledge fusion and Ufunc.

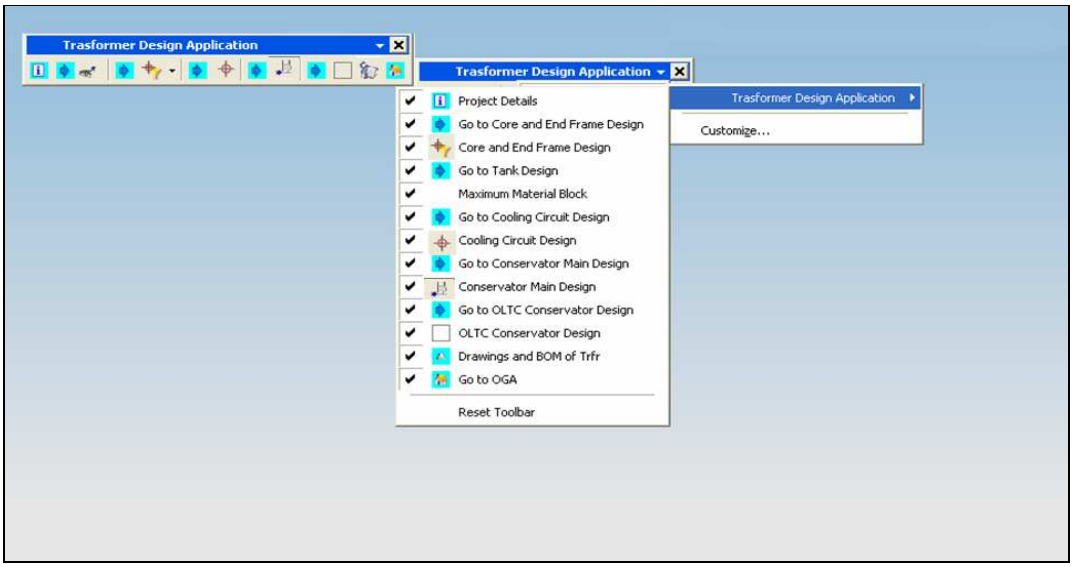


Figure 1.1: Transformer design application menu bar

My research will focus on this particular fact that how to realize different structures in the design knowledge bases according to which designers organize their knowledge. The role of knowledge bases (KB) in design is thus two fold:

- KB serves as a source of knowledge for the problem description; a vocabulary of shared terms that may be used to communicate design problem description.

- KB serves as a reference vocabulary supporting retrieval of familiar/similar cases and their modification (knowledge re-use).

CHAPTER 2

Knowledge Based Design

Design systems for engineering have been developed basically in two parallel directions [5] and those are:

- The automated design systems; also called intelligent CAD (MacCallum, 1990), whose aim was full or partial automation of the design process; the role of human designers is to give the initial requirements, evaluate solutions, build prototypes etc.

- The design support systems that aimed at assisting human designers in their tasks either by recalling past cases (Watson and Perera, 1997), critiquing and navigating (Fischer, 1992), reasoning and consistency maintenance (Smithers, Conkie et al, 1990; Tang, 1997), etc.

Engineering design was identified as a goal-oriented intellectual and knowledge-rich activity (MacCallum 1990; Smithers, Conkie et al, 1990). There were several attempts to develop a computer-based system supporting in one way or another designers and design process. For a further reference see e.g. (Smithers, Conkie et al, 1990; Tang, 1997; Watson and Perera, 1997). Knowledge-based design support system (KBDSS) may be basically defined as *a* decision support system to enable designers to explore the structures of design problems and their solutions by combining human design expertise with domain and design knowledge that might be stored in a system. As seen from the definition, one of the aims of KBDSS is to assist designers in the exploration of structure of both design process and domain knowledge. KBDSS thus need means that would clarify the structure and simplify the exploration process. The next section gives an overview of related research activities aimed at the design support systems and identifies current positions of research in this area. In the remaining sections the application of knowledge models as a core technology for the knowledge base construction in a KBDSS is justified [5].

2.1. Knowledge Based Design Support

There are three Knowledge modelling technique [21]:

2.1.1 Case-Based Reasoning

Cased-based reasoning (CBR) is a technique that takes the design requirements and finds a recent design case that gives the best match.CBR has many benefits in comparison with rule-based systems, which are often based on knowledge based engineering or sometimes expert systems techniques. Marefat and Britanik states that CBR is more efficient than rule-based systems, since the engineer does not have to start from scratch when generating the solution. Another drawback of rule-based systems is that it is difficult to maintain and ensure the consistency of the rule base as it gets larger. Finally, Marefat and Britanik claims that domain experts tend to explain their experience in terms of scenarios (cases) rather the rules. Although CBR has benefits, the number of solutions is dependent on the number of cases. Another issue is that when AI is included in a CBR system, as in, the quality of the result, i.e., how good the Programming Logic Controller (PLC) is simulated, may vary from time to time, as the logics often are dependent on algorithms. This sort of logic, sometimes denoted fuzzy logic, and can be used to simulate Product Development activities that change often.

2.1.2 Expert Systems

Expert systems (ES) are characterized as containing expert knowledge and are often used for analytic activities such as manufacturing evaluation. Venkatachalam presents an ES for manufacturing evaluation in terms of drilling and milling. As ES are often focused on analysis, automatic geometry generation is often omitted. Instead, it is quite common that ES are coupled to a geometry engine by which geometry definition is performed manually. Since, by definition, expert systems contain expert knowledge, they may become something of a black box, as some users may have difficulty understanding the expert knowledge these systems contain.

2.1.3 Knowledge Based Engineering

Knowledge based engineering (KBE) is a method for building product models containing experience of engineering design and evaluation in terms of rules coupled to geometric parameters. Stokes defines KBE as: “The techniques to capture and re-use product and process knowledge in an integrated way”.

KBE product models are often built up of objects in an object-oriented programming (OOP) hierarchy of classes and instances. When one object changes in OOP, demand-driven capabilities calculate only the dependent objects instead of all parameters, as for

procedural programming. KBE is suitable for products where the engineering content is more important than the industrial design content, as the objective is to capture engineering knowledge regarding repetitive and mundane activities. The smaller the difference is between product variants the greater is the reuse of engineering design experience by means of KBE systems.

Although both CBR and ES can be coupled to a geometry engine, it is more of a rule that KBE is coupled to a geometric engine. In contrast to ES, KBE focuses on routine, mundane and time-demanding knowledge instead of expert knowledge. KBE enables automatic creation of product definition in terms of geometry and manufacturing plans, for example. It is also possible to enable topological changes; a rectangular prism becomes a cylinder for instance, which is cumbersome if not impossible with traditional parametric solid modelling. Therefore, KBE includes both product synthesis and analysis. KBE systems can generate an infinite number of solutions, as the product model is not dependent on a finite number of cases. As all knowledge is represented as rules in the model, the output will always follow the real process, given that the real process has not changed since it was modeled. All these techniques are applied for following product as shown in table 2.1.

2.2. Design Automation With Knowledge Based Engineering

Knowledge Based Engineering (KBE) can be seen as a tool for capturing knowledge and reusing it. The concept of KBE is therefore very broad. KBE is a subset of Knowledge Based Systems (KBS), which is a spin off from artificial intelligence (AI). KBS is often referred to as “expert system” because they intend to capture expert knowledge and sometimes also generate creative solutions. KBE on the other hand is used to automate mundane time demanding tasks. By freeing people from this routine work more time could be used to come up with new innovative solutions [7].

Table 2.1: Knowledge modeling technique applied in industry

Knowledge modeling techniques	**Product**	**Discipline**
Expert System (ES)	Generic	Design, Manufacturing
Design Rationale (DR)	Kitchen design	Design
	Chemical plant	Design
Knowledge Based Engineering (KBE)	Wing structure	Performance and Manufacturing
	Wing structure	Design, Cost analysis
	Car body structure	Design, Analysis
	Aerospace	Design, Analysis,

		Manufacturing
	Buildings	Design, Analysis
Agents and Case Based Reasoning (CBR)	Moulding evaluation	Manufacturing, Analysis
	Insurance	Analysis
	Low power transformers	Design, Analysis
	Material selection	Design, Analysis
	Travel agency	Analysis
	Induction motors	Product support

Benefits with KBE are that optimization of product concepts is easier and product and process knowledge is stored. Drawbacks are that it is time demanding to develop KBE systems and it can sometimes be seen as a "black box" if the automated procedures are poorly explained to the user [7].

In Design Automation there is a strong group of researchers attempting to model human activities on knowledge level (introduced by Newell, 1982). The key assumption in knowledge-level modelling is that 'intelligent' behavior of an agent could be explained in terms of agent's knowledge about the situation. In other words there is a strong relationship between the knowledge and the behavior of its possessor when tackling a problem. In the last decade various knowledge-level models were introduced for different domains. I suggest applying knowledge-level modelling in the designers' support for the problem formalization in engineering design.

The design engineer's dilemma is to develop improved and less expensive products that can be manufactured in less time. These demands are the challenge for every design engineer. In order to achieve this, the product development process has to be refined and improved. In the late 70's design-drawing tables started to be changed for computers and products could be developed in a two-dimensional computer environment rather than on paper. This was a big step and made the modeling process easier. Around ten years later the first solid modeling systems were employed, i.e. CAE/CAD/CAM. This again was a big improvement and basically gave a new dimension to design work by enabling a better overview and an important step into the paradigm of virtual prototyping [17].

The nest big step for the product development process was when all its players were started to be integrated in the beginning of 90's. Since then the integrated product development process has been improved in different ways. One way which lately has

increased in popularity is utilizing a tool named Knowledge Based Engineering (KBE). It is stated that KBE will have same importance for companies in 2010 that CAE/CAD/CAM had in the 90's [17]. The concept is not brand new; it goes back to the 50's. At this time research were performed with the objective of developing a system which had an own intelligence, known as Artificial Intelligence (AI). The idea of AI was to implement adaptive solving strategies that could be used to solve a broad spectrum of tasks. However, the resulting system was a disappointment. The simple problems the system manages to solve humans could do faster. Today knowledge based systems (KBS) or expert systems are closest related to AI. The key to success was to let the engineer do the creative work and use the computer to automate routine work. In the beginning of 80's researchers started to store knowledge and rules in the systems which then managed to perform mundane tasks of the product development process. The concept of KBE was born.

An important issue in research about KBE-systems is how to reduce the development time. Therefore development methodologies have been proposed. It is also of interest to optimize the use of KBE for Small and Medium sized Enterprises (SMEs). Today, it is expensive for SMEs to implement KBE- systems and therefore methodologies for KBE development have been proposed. It is proposed that the organizational structure is mapped up and the fields where the company can gain from KBE are identified. Another research topic is how to expand the capabilities of process integration in KBE. Manufacturing evaluation has shown being possible using KBE [18].

2.3. Knowledge Based Engineering Approach

2.3.1. Capture Of Engineering Knowledge [8]

Knowledge acquisition was performed at the partner company through formal and informal interviews and company reports. Managers and engineers with design, manufacturing, and performance and maintenance functions were interviewed and are considered as representatives for the process.

Knowledge from many disciplines needs to be captured and managed within engineering design. The capture of engineering knowledge is not easily performed, as the knowledge exists in a number of disciplines from business to maintenance activities. A number of modelling techniques, in Table 2.1, have been used to capture, support or automate different engineering activities. No technique will capture all aspects within the engineering domain. Instead, the KEE approach supports the use of the best-suited technique for each knowledge asset. It is important to allow a continuous improvement where lessons learned are incorporated into the model of knowledge. A spread sheet is a tool for capturing knowledge by implementing equations, or rules, which enables knowledge recycling.

2.3.2. Knowledge Formalization

The acquired knowledge was interpreted into rules and formalized to a computer implementation-friendly format. Formalization was done using a partner-company approach. During this activity the implementation structure, i.e., a hierarchical class structure, takes shape.

2.3.3. Automation Of Engineering Activities

This phase is usually done simultaneously with the capturing of knowledge. It is an iterative process between the capturing of engineering knowledge and the automation of the engineering activities. The latter is considered a vital part of the KBE approach and allows fast iteration of engineering activities in the early phases to extract knowledge not normally available. The choices made can then be tested allowing engineers to design the life cycle properties of the product.

2.3.4. Quality Control Of Engineering Activities

If the process is captured in a computerized system, it can be replicated each time. Several hundred concepts can then be created when the process to generate and evaluate them is the same. This quality assurance provides engineers a reliable basis from where to compare concepts. A captured process is now an asset of the company and can be reused whenever needed.

2.4. KBE Fundamentals [21]

The main objective of KBE is to reduce lead time by capturing product and process knowledge. The core of the system is the product model where product and process

knowledge is stored. External databases are used for table data. Input to the KBE system is usually customer's specifications, which in turn gives several kinds of output when being processed. The system software is object-oriented and can therefore perform demand-driven calculations.

2.4.1. Objective

The main objective with a KBE system is to reduce lead-time by automating mundane work activities of the product development process as shown in figure 2.1. Capturing knowledge from staff is another objective. This makes the company more resistant to staff turnover.

It has also been stated, the KBE could be a useful tool to support and organize the new functional approach in product development where a product can be seen as a service rather than "just" a physical product. The product can then include maintenance, education and more "soft" properties than the traditional product. Today there are not other tools available for development of such functional products. By developing product model the objective can be met.

2.4.2. Product Model

The product model is the central part of the KBE system and contains the knowledge about how a certain product should be developed, in terms of rules.

The archived knowledge is typically categorized in three groups:

- Geometry
- Configuration
- Engineering knowledge

That in turn consists of rules and methods. The output from the product model can be a broad spectrum of results.

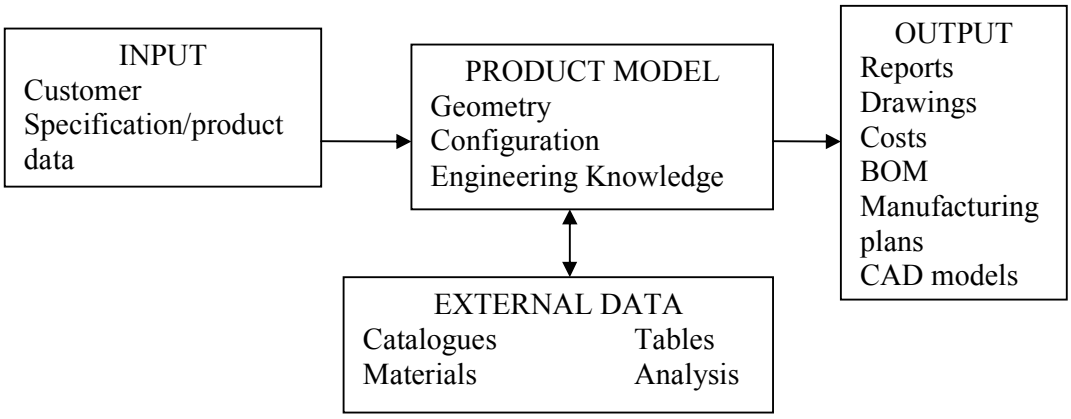


Figure 2.1: KBE system

- **Geometry**

In Computer Aided Design (CAD) it is common to create parametric models, which means that all or some dimensions of the model on a particular dimension of the model. For example when designing a pipe the thickness and the length can be dependent on the diameter. When the diameter is changed, the thickness and the length change with an amount that conserves the same proportions. The model is in other words scaled. This can be implemented in KBE giving the computer a diameter as input and the KBE system automatically scales the pipe.

- **Configuration**

The configuration aspects relate to the activity of selecting parts that the product consists of. If for instance a car manufacture develops a new sports car the system can chose properties like low profile tires, strong motor and double exhaust pipes.

- **Engineering Knowledge**

Engineering knowledge can be programmed into the KBE system consist of manufacturing properties for example. These properties can be rules that confirm manufacturability and also costing.

2.4.3. External Database

The external database holds information about standard parts and properties needed in the product development. This data is not to be confused with the knowledge of KBE system. The knowledge of the KBE system lies in the product model.

2.4.4. Input

The user interface is the only part of the KBE system that the user comes in contact with. This is where the customer specifications are read into system. In order to have an updated KBE system, the product model has to be continuously fed with new product data.

2.4.5. Output

The kind of result generated by the product model is dependent on the input. The most important factor is however how the product model is configured. Usual outputs are reports, drawings, manufacturing plans, costs, BOM and CAD models.

2.4.6. System Software

KBE systems are built utilizing an object-oriented approach. The implementation is often done with IF-rules.

- **Object-Oriented**

The KBE system is object oriented. This means that the knowledge in a product model is seen as objects which all belong to a "class". The class specifies what kind of properties the object has. The properties can be design constraints and manufacturing requirements. The classes are arranged in an order with sub and super classes and matched with hoe the parts are part if other parts, e.g. a spring is a part of a damper. A sub class can inherit the same properties as its higher classes. The class system implies that when a similar product is developed it is not needed to define a totally new class. Instead properties can be inherited which together with the new properties form a new class.

Every knowledge object is an instance of a class. An instance has values for all the properties of the class in a vector. These properties are visualized by a particular draw method. The relationship between classes and objects is shown in figure 2.2:

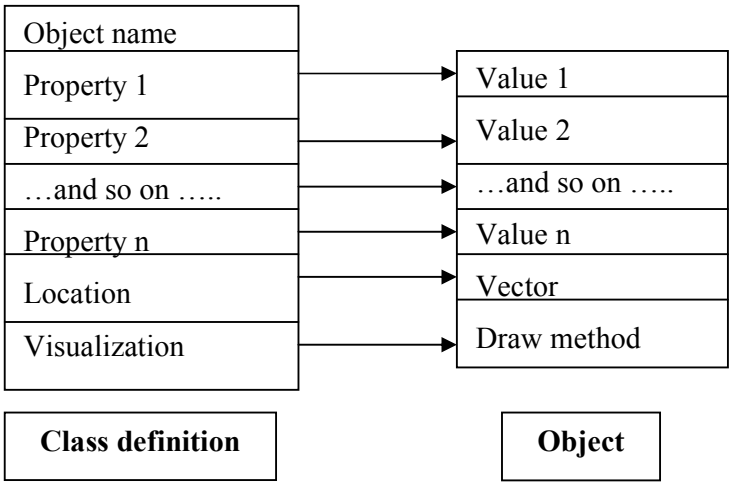


Figure 2.2: Class and object relationship

The object-oriented approach implies that when a single change is made in the product model it is not necessary to re-calculate the whole model like in procedural programming. Instead only the objects, which are connected to the change, are re-evaluated. This is called demand driven or calls by need computation.

- **Implementation**

The KBE rules are often implemented using a set of IF-rules.

2.4.7. Applications

The main application of KBE-systems in the fields of product development is to automatically generate product concept from specifications. The output were from the beginning often only the geometry of the product. Later on, the work has been taken a step further by including preparations for finite element analysis. A newer challenge is to integrate manufacturing aspects in the KBE-system enabling evaluation of manufacturability.

- **Concept Generation**

The original application of the KBE in the product development was generating a product concept in terms of geometry. The definition of concept in this case is a draft of the product showing its main features. The geometry was generated from the implemented rules in the product model together with a geometry engine.

- **Concept Evaluation Preparation**

To make a step further on KBE has been used to make a preliminary evaluation of solid mechanics by using the Finite Element Method (FEM). When using FEM to evaluate proposed concept the geometry is often subject to change in order to enable meshing. The iterative work between design and analysis is often slow and costly. A developed KBE-system can make this work faster.

The importance of design for Manufacturing (DFM) has during the last decades grown. Therefore integration of manufacturing aspects has also started to appear in KBE applications. The manufacturing assessment procedure is based on critical indices, which are defined from customer specifications. The geometry is modeled after the manufacturing assessment is performed. The framework is built up by smart elements.

2.4.8. Methodologies For Developing KBE Systems

The development of KBE systems often follows a pattern: First problem has to be understood at the conceptual level. Then the problem is shed into smaller objects that can be understood clearly. What follows is an iterative process that lasts until the outcome is satisfactory.

- **Development Of KBS**

KBE is a subset of Knowledge Based Systems (KBS) and has more geometrical features than the latter, because it handles product modeling in engineering. KBS are often called “expert systems” and holds a broad spectrum of knowledge in a centralized database. KADS is perhaps the most common methodology for development of KBS. KADS is an acronym that has many interpretations e.g. Knowledge Acquisition Documentation System or Knowledge-based system Analysis and Design Support.

KADS contains a lot of functions and is therefore sometimes seen as large and complex to learn and not suitable for Small to Medium Enterprises (SMEs). It is also stated that successful KBE developers seldom follow the steps of for instance KADS and similar methodologies because they are not always easy to apply on the dynamic process of engineering design.

- **MOKA**

MOKA (methodology and tools orientated to knowledge based engineering applications) is a project that aims to develop a methodology that will serve as an international standard for KBE system development. MOKA aims to state the common patterns instead to create a methodology that fits every industrial scenario. The methodology of MOKA follows six steps according to figure 2. IDENTIFY, JUSTIFY, CAPTURE, FORMALIZE, PACKAGE and ACTIVATE. The focus of MOKA is the CAPTURE and FORMALIZES steps.

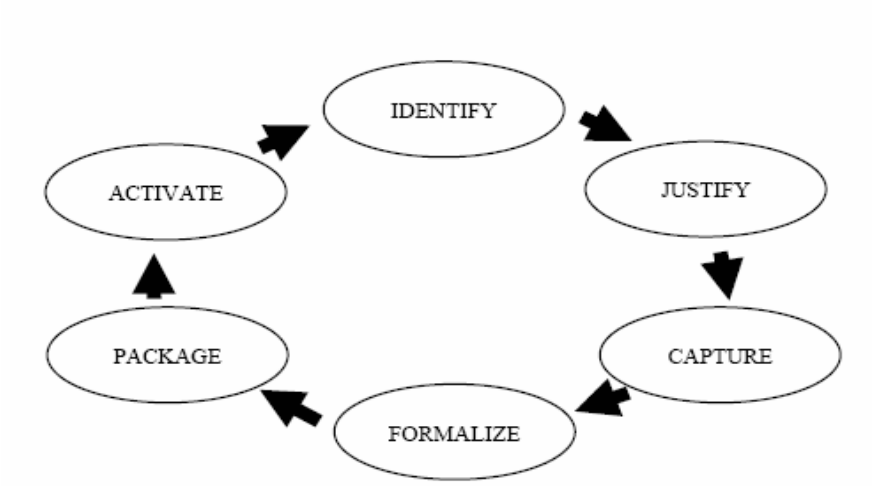


Figure 2.3: KBE developing lifecycle according to MOKA

- **Identify**

This is the step where type of KBE system is determined, if it is needed at all. The outcome from this step is a conceptual specification of the KBE application. IDENTIFY is also triggered by ACTIVATE if more information is needed about the implementation environment.

- **Justify**

The aim of this step is to create a project plan and to seek management approval by examining possible risk concerning, cultural and technical matters for example.

- **Capture**

In this step the raw knowledge is collected and structured into the "Informal Model" by using ICARE forms (Illustration, Constraint, Activity, Rules and Entity forms). The relations between the ICARE elements are described by the process and hierarchy charts.

- **Formalize**

The “Formal model” is created from the informal Model. The modeling language used is MOKA Modeling Language (MML), which is based on the Universal Modeling Language (UML).

- **Package**

Now the Formal Model will be implemented to a KBE-system.

- **Activate**

This step includes installation, distribution and support of the KBE-system. It may also trigger the IDENTITY step if it is concluded that more information about the implementation environment is needed.

2.4.9. Commercial Systems

ICAD were launched in the beginning of the 80’s as the first commercial CAD software for KBE applications. Since then several systems have been developed and today the major vendors of CAD software, EDS among others, have integrated KBE-systems close to the geometry engine.

- **ICAD and INTENT!**

KTI’s ICAD (Intelligent Computer Design) acme in the early 1980’s and is the first CAD software for KBE applications. It consists of two interfaces where the first handles the actual geometry and the second deals with the programming language; ICAD Design Language (IDL), which is a LISP (List Processing Language), based language.

INTENT! Is in some ways similar to ICAD; it is LISP based and developed by people who have worked with the development of ICAD. INTENT! uses AutoCAD as its geometry engine.

- **Unigraphics Knowledge Fusion**

EDS Unigraphics is one of the leading CAD software’s on the market and has a KBE application, which is called Knowledge Fusion. Unigraphics uses INTENT! as modeling language and Para solids as geometry engine. The user can build interfaces for different geometric parts. Databases are easy to access by using a Open Database

Connectivity (ODBC) method. Earlier on, Cad automating programmers had to write complex code to achieve this.

2.5. Revolution or Evolution [16]

Even though the discipline of engineering design can benefit from KBE systems there are still some drawbacks that need to be removed through research.

2.5.1. Benefits

The most obvious benefit from using a KVE system is the reduced lead-time. It is also easier to optimize products. By capturing knowledge, staff turnover is not a big problem. Because time demanding routine tasks are automated more time for creative solutions are given.

- **Reduced Lead-time**

The major benefit is that the KBE system reduces the lead-time. This is highly pronounced for product development of the products with the following properties:

- Products with high degree of similarity between versions.
- The higher similarity the more knowledge can be re-used.
- Products requiring a large amount of design configurations (e.g. geometry configurations material alternatives, etc).
- Design configurations are suitable to be controlled by rules.
- Product with a large number of design processes (e.g. FEA, cost calculation, weight calculation).
- These design processes can be performed automatically in a “one button push” matter if all needed input is given to the KBE-system.

- **Product Optimization**

Optimizing the design is easy - trial and error goes fast. The user can try many “what - ifs” and come to a conclusion after a radically smaller amount of time than before. The computer can also search for best configuration within a specified range.

- **Knowledge Capture**

Staff turnover is no longer a big problem for companies KBE because base knowledge is stored in the product model. This also implies that companies can reduce their

outsourcing activities when basic knowledge is stored and handled by the KBE - system.

- **More Time For Innovation**

By automating mundane, routine work more time for the product developer to concentrate on innovative solutions is given. On a long-term view this is increasing the product value and enhances the companies business.

2.5.2. Drawbacks

The major drawback with KBE is that it takes a great deal of time to build up the product models. An important aspect, which sometimes is neglected, is the knowledge transfer, how knowledge is spread in the company.

- **Time Demanding Building Product Models**

A KBE system can reduce the lead-time when implemented correctly. This implementation process is however quite time demanding. A KBE system can be seen as a new engineer joining the department. The system has to be educated like the new engineer who in the beginning will learn the basic tasks that are performed often. It takes a while before the new engineer can perform tasks that the company can benefit from.

- **Knowledge Transfer**

A danger with any computer system is if it becomes a “black box”. You put something in and you get something out, but what happens in between, nobody knows. A result generated from a computer system is often regarded with more respect that it deserves. This gives a false security and can lead to problems if a system contains faults. When a system becomes a “black box” there is a problem to transfer the captured knowledge to new employees. Therefore it is of great importance that the systems provide an understanding of the process of reaching the results by explaining what has been done.

CHAPTER 3

UGS Solutions Applied To Design Automation

UGS is a leading global provider of product lifecycle management (PLM) software and services with nearly 4 million licensed seats and 46,000 customers worldwide. Headquartered in Plano, Texas, UGS vision is to enable a world where organizations and their partners collaborate through Global Innovation Networks to deliver world-class products and services while leveraging UGS open enterprise solutions, fulfilling the mission of enabling them to transform their process of innovation [10].

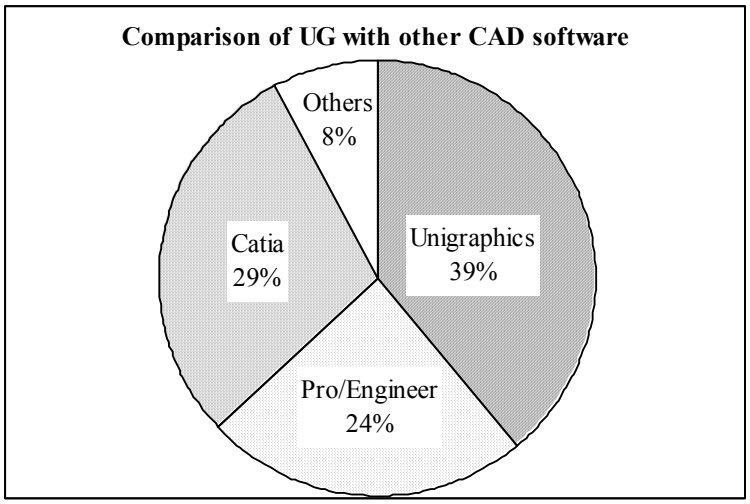


Figure 3.1: Comparison of UG with other CAD software

3.1. Why UGS

There is no single road to innovation, but there are signs you are headed in the right direction. Leading innovators get to market faster, manage compliance, optimize resources and achieve globalization. They are also four times more likely to use PLM software to plan, define, build and support their products. UGS family of PLM solutions helps businesses establish Global Innovation Networks that transform their process of innovation. Drive your business to greater innovation and accelerate your growth. Throughout its broad product application suite, NX leverages key attributes that help companies achieve business objectives of waste reduction, quality improvement, shorter cycle times and greater product innovation. These unique attributes directly support business process initiatives aimed at transforming product development [10].

UGS technologies uniquely support design automation initiatives, helping companies reduce delays, maximize design re-use, reduce defects, and improve process efficiency. The primary solution components enabling Lean Design are NX Digital Product Development and Team center product suites. Together they provide a managed development environment; a unified solution that addresses all phase of product development, from concept through manufacturing; knowledge driven automation; simulation, validation, and optimization; and system-based modeling. All of these tools and technologies can be directly applied to eliminating waste and successfully implementing lean development initiatives [11].

3.1.1. A Managed Development Environment

With the managed development environment provided by NX and Team center, companies can reduce delays through fully integrated, synchronized management of all product data and process knowledge. A key component is a structured repository for all products and process data required in product development-not only the data created in computer-aided design, engineering, and manufacturing, but also a wealth of related information needed to execute development processes.

With this managed environment, companies can capture workflows and processes that automatically route information to team members when it is needed, eliminating the time wasted looking for, waiting for, or re-creating product data. Not limited to CAD models, drawings, and Bill of Materials, the development environment also manages application data from simulation and testing, NC programming, process planning, and manufacturing, and company-specific knowledge bases to get the right information where and when it is required to expedite development decisions. These capabilities eliminate the delays inherent in paper-based, non-automated, or inefficient information distribution and approval processes [12].

The managed environment includes a robust suite of collaboration tools that help reduce waste and delays encountered by large development teams, even those extending beyond the enterprise to customers and suppliers. With integrated visualization and collaboration aids, all stakeholders in product development can instantly find, access, manage, view, and share product and process data. The managed development environment supports geographically distributed development teams, even

when they are using disparate systems and technologies. The result is fewer delays due to data transfer and translation through the development process [19].

3.1.2. Unified Solution From Concept To Manufacture

The NX solution integrates a comprehensive range of digital product development applications on a unified technology platform. Sharing the same foundation, NX applications automate development processes from initial concept through manufacturing using the same, dynamically associative architecture and data structures. With NX associativity, changes to the product design model, for example, are automatically propagated to documentation, engineering analysis and numerical control programming processes.

With a unified solution for all development processes, manufacturers can reduce the waste of creating documents to communicate design changes, and eliminate data translation or re-creation from the critical path of product development. This capability directly supports Lean initiatives by eliminating waste, delays, errors, and data loss through all development stages, with dramatic improvements in overall process efficiency and cycle time [11].

3.1.3. Knowledge Driven Automation

With NX, companies can apply product and process knowledge across all elements of product development to automate processes and maximize re-use. NX enables manufacturers to readily capture product and process knowledge, and easily re-use it to automate even very complex development tasks.

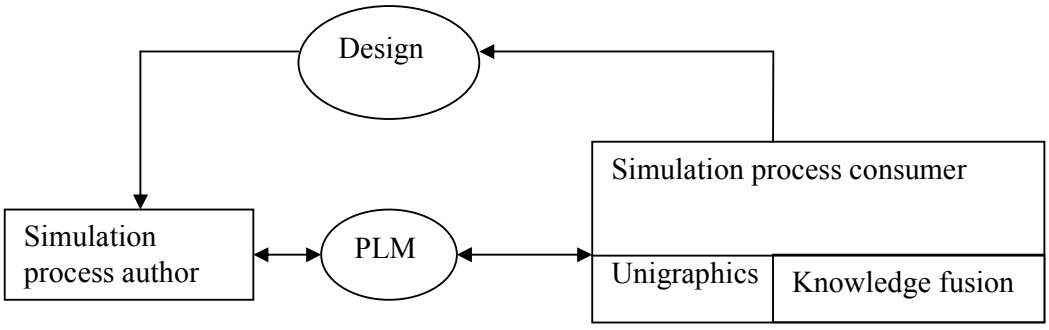


Figure 3.2: Knowledge based automation

Knowledge-driven automation supports Lean Design by enabling companies to re-use product designs and development processes instead of re-creating them for each new

product challenge. With NX knowledge-enabled capabilities, companies can create templates that can be instantly loaded and automatically executed as stored processes. As an example, a wheel manufacturer can capture its best practices for designing and analyzing various types of standard wheels and define that process in a template file [13].

They need only drag that template onto the geometry and start the process. Process assistants can be developed by simulation experts who enable the same process to be executed accurately by new simulation employees or even design engineer users in a wizard-like tool. These same automation techniques enable preferred simulation processes to be followed for each type of workflow or product evaluation activity leading to higher staff productivity, higher product quality and consistency of engineering results. This effectively brings fundamental performance simulation activity into the upfront design process and supports enterprise wide initiatives to capture in-house knowledge and proven repeatable simulation methods [14].

Companies can take advantage of NX knowledge-driven automation in several ways. First, the NX suite includes several applications that have already captured industry knowledge and best practices for specialized tasks including structural analysis and mold and tooling designs. These prepackaged tools apply the knowledge of experts to streamline and automate complex development sequences, enabling companies to reduce effort wasted in assimilating standard practices or performing detailed tasks associated with complex processes. NX also includes Knowledge Fusion, a knowledge-based engineering rules evaluation engine embedded directly inside NX that eliminates the traditional wall between knowledge systems and detailed design systems. With Knowledge Fusion, companies can drive designs “up-front” using external requirements and knowledge databases, even those specific to the company and its products. Knowledge Fusion can encapsulate stimuli/results sequences by embedding knowledge rules inside existing designs or do both simultaneously. The result is a higher level of automation that both reduces wasted or repeated engineering effort, especially for complex, costly products, and an automated means of ensuring that products and processes are of higher quality and adhere to engineering, customer, or company requirements [15].

3.1.4. Simulation, Validation & Optimization

Integrated tools for simulation, validation, and optimization in NX support Lean Product Development initiatives by reducing defects and associated warranty and support costs, improving performance and manufacturability, and reducing the lengthy and costly design-build-test cycles of physical prototypes.

The comprehensive simulation solutions in NX include sophisticated but easy-to-use tools for design engineers that predict product performance early in the development cycle, when it is easiest and least expensive to alter the design. Knowledge-enabled wizards and design integrated structural and motion analysis tools enable early-stage simulation without requiring engineering analysis specialists, and provide valuable performance and manufacturability feedback that guides development toward higher quality, lower cost, and better function. For more detailed simulation, NX offers advanced system-level digital prototyping and functional simulation capabilities for engineering analysis specialists. These support Lean Design by accurately representing the physical world through digital, rather than physical, prototypes. Physical prototyping requires significant expense in both time and materials, and can unnecessarily lengthen the overall development cycle. By digitally simulating performance, companies can minimize product failures and their associated service, support, and replacement costs [15].

3.1.5. System Based Modeling

NX provides a system-based modeling capability can help companies reduce development waste in a number of ways. It provides an effective method for capturing subsystem and component relationships using a system-level conceptual model and product control structure. With product definition templates, companies can standardize design practices and rapidly create product variants, effectively re-using knowledge and eliminating engineering re-work. System-based modeling also helps reduce engineering errors and defects among product options, variants, and derivative platforms by preserving high-level design parameters and the interfaces between systems and components [15].

CHAPTER 4

Unigraphics Knowledge Fusion

4.1. KBE and KBE Languages

The concept of "Knowledge Based Engineering" is a very broad one. The basic idea is to find ways of recording different kinds of knowledge about how to engineer, design and configure a product in a way that allows it to be easily found, understood, reused, and maintained. A KBE language is one of many kinds of tools that can be used to help achieve this. A KBE language allows knowledge to be captured in the form of "rules," statements written in the language. The following statements may be made of a general KBE language [15].

- The programmed rules specify how various parameters and attributes of objects in a model are related to each other. The relationships are commonly expressed as math formulas.
- They may also be captured in many other ways such as by data base relations or programmed procedures. A KBE language program defines a model of a physical or abstract system. An application developed in a KBE language allows the end-user to enter the rules to calculate the effect of the parameter values and produce results derived from them. As with electronic spreadsheets, the language is declarative rather than procedural. This means that the order of operations need not be specified. The system figures out the order needed from the relationships defined by the formulas. Just as a spreadsheet program does when it recalculates.
- Unlike spreadsheets, KBE programs are object-oriented. Instead of using a matrix of cells to organize data, the KBE program developer organizes data into named objects. All the major attributes of the object-oriented paradigm: abstraction, encapsulation, modularity, and hierarchy are included. KBE languages differ from most general-purpose object oriented languages in being primarily declarative rather than procedural in nature.

Computer Aided Design in general (including CAM, CAE, etc.) has traditionally meant the use of a graphical, interactive system for modeling product or process geometry. The object model is for the most part designed into the system. The possible kinds relationships that exist between objects are often build into the internal data model of the system. There may be a way to write custom programs for the system. Such

programs are generally procedural in nature, following the way the interactive system works: First do this, and then do that, etc. The main strengths of common CAD/CAM system are in built- in solid geometry processing and the ease of use of geographical interaction for building and operating on complex solid geometric models [15].

4.2. KBE Single Object Representation In A Master Model [15]

A single representation of the object in a master model is achieved when there is a deep integration of KBE language with a CAD/CAM system. The objects are created and viewed from either the language or the CAD interaction system. A deep integration needs to include the following components.

- The Extendible Object Model: The CAD solid model data needs to be presented in a way that is compatible with the idea of a declarative KBE language. The model needs to allow natural user- defined extension and the capture of user- defined relationships between objects.
- The User Programming Tool-High Level Language: The language itself provides a textual way of expressing a model.
- Visual Development Environment: A Visual Development Environment simplifies the development of applications or models using the language.

4.3. Overview Of UG/Knowledge Fusion [23]

The Unigraphics UG/Knowledge Fusion application is based on Intent!, a software toolkit, including the programming language, owned exclusively by M/s LoftTech Inc and marketed by M/s Heide Corporation, Medfield, MA.

UG/Knowledge Fusion is designed for use in building engineering automation solutions. This implies that there is a repetitive "problem" which requires automating as the solution, which is generally a requisite condition for KBE implementation. UG/Knowledge Fusion is most useful where the solution requires a combination of configuration (the selection and assembling of components to make a coherent whole product), engineering (the decision-making process related to validity and applicability of components), and geometry (the components physical organization).

UG/Knowledge Fusion can also be described as:

- Engineering spread sheet that allows you to record and reuse engineering formulas. The formulas can refer to values of other formulas.
- A configuration system for selecting and assembling compatible parts.
- A very powerful computer- aided design (CAD) programming language.
- A system with which you can define and create “smart parts” that can be assembled and/or configured in ways that are not programmed.

The UG/Knowledge Fusion language, combined with the Knowledge Fusion Navigator user interface, can support you, the end user, at a wide range of skill levels. It allows one to go from making simple adjustments to an existing knowledge based engineering (KBE) application (by changing parameters or adding dynamic rules to an existing UG part file), to the creation of complex assemblies by a design team, such as for example, a parametric family of automotive transmissions.

4.4 Basic Concepts Of UG/Knowledge Fusion [15]

UG/Knowledge Fusion has certain basic features, which should be kept in mind in order to maximize the functionality possible. These features are described below. UG/Knowledge Fusion is much like a spreadsheet, in that:

- You can write which can refer to the results of other formulas.
- It is not necessary to be programmer to write formulas.
- You can simple or complex formulas.
- The formulas have syntax.

Solutions to the complex problems generally cannot be developed with a simple CAD like interface. They require statements, relationships, formulas and condition.

4.4 1. UG/Knowledge Fusion Is Demand- Driven And Declarative

Again, like a spreadsheet, formulas do not have to be written in any particular order. If another formula is referenced (demanded), the system automatically finds and executes it. This is not case with procedural languages, like Basic, C, FORTRAN, etc.

4.4 2. UG/Knowledge Fusion Is Object Oriented

Like modern computing languages such as C++ there are classes and instances, and there are multiple inheritances.

4.4 3. UG/Knowledge Fusion Is Hierarchical

In much the same way engineers think about designing products, in UG/Knowledge Fusion:

- Components are built into sub-assemblies, and sub-assemblies are combined to become assemblies.
- There is consistent treatment at all levels.

4.4 4. Geometry

UG/Knowledge Fusion is like a CAD system, but is not interactive in the same way:

- Some UG/Knowledge Fusion Parts will be raw geometry.
- Geometric parts in the language create geometric entities in the CAD system.
- UG/Knowledge Fusion controls relationships between its parts, and automatically propagates them to the CAD system.
- UG/Knowledge Fusion Supports the geometry of Unigraphics (UG).

4.5 Why UG/Knowledge Fusion

There is a reason to integrate Intent! into UG/Knowledge Fusion application as it provides a means to accomplish what UG alone cannot do, namely determining in advance the performance and behavior of the final product.

With the integration of UG/Knowledge Fusion, a single data model can be used to manage both parametric and knowledge- based part information. It can also capture rules/requirements from the engineering designer and convey this information seamlessly into the design definition through the instantiation of class files into part files.

UG/Knowledge Fusion is a knowledge base language, and can therefore capture both geometry and non-geometric attributes of a given part or assembly, and write the rules which describe the process to create it. This in essence gives the ability to capture intelligence and engineering knowledge- how with in part file [23].

4.6 Some Key Definition

- **Class**

  This is the name given to the text file of rules that describe and define some object/part being designed. A class bears a "**.**dfa" file extension; hence, the term "DFA file". Instances of the class files become the building blocks of UG parts and assemblies, but once instantiated; these files reside in the UG part file, and no longer in a separate DFA file.

- **Rules**

  Rules are what describe the object or part that is being designed. A rule will state a dimension, a radius, a length, a start angle, end angle, start point, end point, etc. A rule might also be a mathematic expression or formula, or, it might address such items as design stress, resultant volume, position, mating or orientation. There are class rules and "dynamic" rules, which are rules that have been instantiated into a UG part file.

- **Attributes**

  That part of the rule syntax that assigns values to the rule through input parameters, such as a dimension, a radius, a length, etc.

- **Instance**

  An iteration of a class, of which there could be many.

- **Instantiation Or Instantiate**

  Creation of a class Unigraphics.

CHAPTER 5

Problem Description

The main components of the transformer are as follows:

- Turret assembly
- Core and end frame assembly
- Header pipe work assembly
- Conservator pipe work assembly
- Outer fitting assembly.

5.1. Turret

The turrets in the transformer are used for the installing the bushing on the top cover of the transformer tank body. The turrets are used where some space is required for installing the bushing on tank surface. In the case of high voltage bushing when the electrical loss will be higher, bushing has to be installed on the tank body top cover then the turrets are used.

There are basically two types of turrets:

- Rectangular Turret.
- Round Turret.

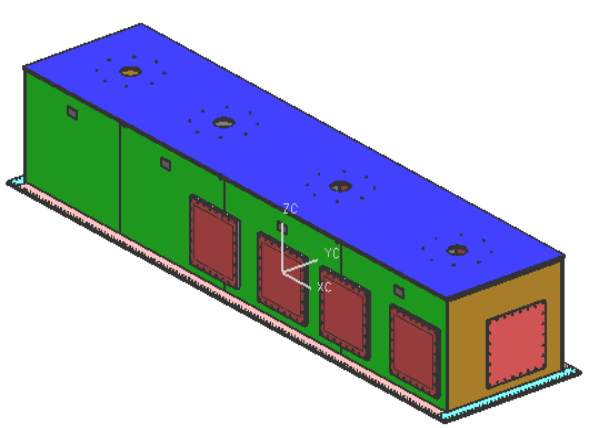


Figure.5.1: Rectangular Turret

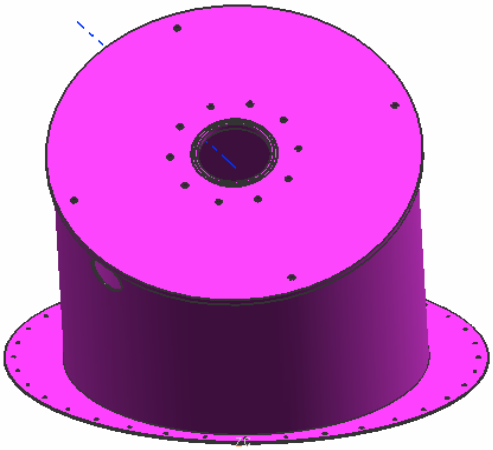


Figure 5.2: Round Turret

5.1.1. Rectangular Turret

For Cable Box / segregated phase bus duct only rectangular turret can be used. In this case where the bushing is to be installed in less space and the top surface of the tank body is straight; the rectangular turret can be used for installing the bushing. The rectangular turrets as specified by the name have the rectangular shape.

5.1.2. Round Turret

The round turret is used for the isolated phase bus duct. When the high voltage bushing is required and there should be significant distance between the two subsequent bushing then the round turret are used for decreasing the electrical loss between the core winding and the bushing. As the bushing is installed on the turret where the high voltage bushing are used, so when there is electrical loss occurs in-between the bushing itself then the round turret are installed. Turret can be tilted at two different angles for keeping the subsequent distance between the core winding and the bushing bottom and

the two bushings on the surface of the tank body. The basic shape of the round turret is of hollow cylinder, which can be tilted at two different angles.

5.2. Case Study

For a case study rectangular turret is taken. Main components of rectangular turret are:

- Turret Body
- Bushing Mounting Pad
- Bottom Flange
- Equalizing Pipes
- Fastening for Equalizing Pipe
- Gasket
- Lifting lugs
- Cooling Terminal Mounting

As an example to show increase in productivity using design automation with KBE, let us take the case of the model shown in figure 5.1.To make the complete assembly twenty drawings of the individual components would have to be made. All twenty drawings and Bill of material can be made through one program by giving 80 independent variables as an input. The program calculates various formulae and about 256 different dimensions relevant to the twenty drawings, generate one model of the assembly, and twenty models of individual components and the 2-D production drawings and bill of material are generated automatically. The bill of material data is also stored in a database for future requirements. Normally, about 5 weeks were required to make all the model, drawings, Bill of material etc, but using this system same can de done in two weeks only.

The design and automation of the round turret is done in following steps:

5.2.1. Knowledge Acquisition For Rectangular Turret

Specifications are the basic document on the basis of which the design is finalized, and final product is realized. To begin with, the designer has to study all site condition requirements and spell out the critical parameters. After that acquired knowledge is tabulated in the form of External Data base for turret and it’s component on the basis of design and manufacturing requirements.

For turret, it is shown below:

1. Turrets

- Type of turret: Rectangular /Round
- Welded or without welded type
- For Cable Box / segregated phase bus duct only Rectangular Turret can be used
- For isolated phase bus duct, round turret can be used but in this case its top flange will be of the size to accommodate the receiving holes of the bus duct. Hence when calling the turret user to ask whether Cable box / Bus duct required

2. Rectangular Turret

- This turret will always be vertical to the plane, which it will be placed on. Only vertical case is possible. Top flange and bottom flange are always parallel to each other
- If the receiving planes of the flanges of the bushings are not on the same plane then it will give alarm and user will have to reposition bushing to make the receiving pad on same plane and the center of the flanges should be in line if it is not then user will reposition the bushing
- If the bushing is inclined to the mounting plane then the option of rectangular turret will be disabled
- User Input
  - Whether for one bushing or three bushing
  - Whether bus duct will be required or not. If required then No. Of holes, hole diameter, Pitch circle diameter will be user input
  - Top flange SS or BS
  - If three bushings then user will select the other two bushings, which is to be placed on turrets
  - User will input the values of radial clearance between bushing receiving pad and turret inner walls, Radial Clearance
  - Thickness = 5 cm, 8 cm
  - Type of bus duct, drop down menu for segregated bus duct / phase bus duct
  - Segregated bus duct Length & Width of rectangle, no of holes along length & width together with its Pitch Circle Diameter
  - User to select fastening arrangement. From drop down between M12 and M 16

- User to choose the type of inspection cover

3. Side Plate

- X1 and X2 are the offset distance from the center of the outer bushings towards the inside of the side plate
- Thickness=T

4. Front / Back Plate

- Y1 and Y2 are the offset distance from the center of the outer bushings towards the inside of the front/back plate
- X=X1=X2=Y1=Y2= receiving pad id + RC
- X should be reported to the user and user will accordingly accept the value of RC or modify it
- Turret walls height=distance between the cover/wall plane and receiving plane of bushing flange-pad thickness on cover*(45) - gasket thickness - bottom flange thickness of turret-receiving pad thickness for bushing - top flange thickness of turret - gasket thickness (only for non welded type turret)
- How bushing mounting pad will be welded on cover / wall
- Turret walls height=distance between the cover/wall bottom plane and receiving plane of bushing flange-receiving pad thickness for bushing-top flange thickness of turret-gasket thickness - 25 (only for welded type turret)

5. Welding Detail For Side And Front Plate

- Front palate is extended by 15mm+ thickness of side plate for T welding.

6. Top Flange

- Thickness=18
- Width of the top flange is defined as sum of sidewall width + front and back wall thickness+30 in case bus duct is not there
- In case of phase bus duct width = Pitch Circle Diameter of holes for bus duct +80
- If this width >= width of top flange for non bus duct case then accept this else width of the top flange for non bus duct case to be accepted
- In case of segregated bus duct holes are not in circular pattern. They are along the periphery of rectangle

- Then width of flange = center distance of extreme holes along the width of the rectangle+ 90
- Length = length of the turret, in case bus duct is not there
- In case of phase bus duct length = distance between outer bushing center+Pitch Circle Diameter of holes for bus duct +80
- If this length >= length of top flange for non bus duct case then accept this else length of the top flange for non bus duct case to be accepted
- In case of segregated bus duct holes are not in circular pattern. They are along the periphery of rectangle
- Then length of flange = center distance of extreme holes along the length of the rectangle+ 80
- If this length >= length of top flange for non bus duct case then accept this else length of the top flange for non bus duct case to be accepted
- In case of bus ducts rectangular or circular profile used for holes need to be defined by sketch
- Circular profile for holes will be placed in line with the respective bushing axis
- Center of mass of the rectangular profile will be in line with the middle-bushing axis
- Holes for bus duct will be 12 deep in the top flange
- Bushing mounting pad it will be same as the bushing-mounting pad for cover
- Gasket and its hardware will also be same as discussed for bushing mounting pad for cover a flame cut on the top flange of diameter = bushing mounting pad id + 30

7. Side Bottom Flange (Required only if turret is welded type)

- Thickness = 28 mm after machining, post machining 20mm for Bill of material
- Width = 128
- Length= distance between front and back inner wall - 15 +235
- Its is 5 mm offset from the inner wall of the turret towards the inner side

8. For Front And Back Of The Bottom Flange

- Thickness = 18 mm after machining, post machining 15mm for Bill of material
- Width = 116
- Length= distance between front and back inner wall - 20 +230
- There is 5 mm offset from the inner wall of the turret towards the inner side

- Front and back of the bottom flange is required only if Turret is of not welded type
- Holes will be placed at corners 35 mm centrally from both outer edges. Other holes will be placed with the same logic as discussed in tank rim
- L1= L2+ 85

9. Air Release Plug (If equalizing pipe is not required)

- **For Lifting Lug**

▪ Case 1

♦ If (width of Top flange - distance between item 02 outer surface)/2 < = 100 then lifting lug will be placed on the of center side walls else case 2

♦ Two quantity of this will be placed with an offset distance of 50 from bottom of top flange

▪ Case 2

♦ It will be placed on top of the top flange and 110 offsets from the outer edge of the front and backside.

♦ In case of one bushing it is placed at the center along the front and back end side

♦ In case of three bushings it is placed between two bushings

♦ User can reposition it

10. For Boss

- This is called if bus duct is required and its hole on the top flange are coming 40 mm outside the wall
- For isolated phase bus duct it is placed along the plane of minor axis of each bushing and 25 offset from the turret wall outer surface
- For segregated bus duct it is placed at central bushing and 25 mm offset from the minor wall
- A hole corresponding to it of diameter 23 will come
- User can reposition it

11. For Pad Of Polarity Disc

- It's quantity = no. Of bushing
- It is placed on the front wall of the turret, 100 offset from the top flange bottom surface and center plane of each bushing

- User should be able to reposition it

12. For Pads Of Inspection Cover

- User to define its place and quantity and a cut corresponding to it will be there on the wall
- Stiffener ask by the user whether required or not
- Thickness = turret plate thickness
- Its width and length is same
- It is used in case of more than one bushing and is placed between the bushings centrally
- A 100 mm hole be will cut on the plate centrally and another semi circular hole of 100 mm will be cut with it's center on the center of the top edge of the stiffeners

13. For Drain Plug 3/4 Inch BS

- It will be placed below the (boss).
- It's variant and style no. Will depend upon the type of polarity disc.
- It is hardware for polarity disc.

14. For Gasket Of Bottom Flange (Required only if turret is of not welded type)

- Thickness = 8mm
- It will be along the rectangular profile of the bottom flange.

15. For Blanking Plate Of Bottom Flange

- It is always BS
- Thickness=5
- It's dimension come in the description column in Bill of material
- It is for blanking for turret bottom flange and on pad for turret on tank.
- It has hole for fastening it with flange
- It's dimension come in the description column of Bill of material together with "holes to be matched accordingly with flange" in remark column

16. For Handle Of The Blanking Plate For Bottom Flange

- Quantity = 1 in case of single bushing else it is 2

- It is to be welded with the blanking plate and a note to this effect will appear in the drawing.
- Length=45

5.2.2. Knowledge Formalization For Rectangular Turret

The acquired knowledge was interpreted into rules and formalized to a computer implementation-friendly format. Formalization was done using knowledge fusion approach one part of this is shown below

Distance_outer_face_Plate=2*UI_RC+2*UI_Turret_Thickness

DrainPlug_Boss=if(((2*offset_along_width+Width_Top_Flange_wo_bus_duct)/2)-(UI_RC+UI_Turret_Thickness))>40)(1)else(0)

Width_Top_Flange_wo_bus_duct=2*UI_RC+2*UI_Turret_Thickness+30

Moreover other acquired knowledge is formulated in the form of design database, Bill of material database, Expression database etc.

5.2.3. Design Automation For Rectangular Turret

Once the full Knowledge formalization is completed, this should be embedded in solid model. Now when the modelling is completed the interfacing of the model is done with the user using the knowledge fusion where the placement of the turret assembly is done on the tank body and the expression value which are required to update are updated and the user interface is done. For the user interface the user interface files are made (see figure 5.3, 5.4, 5.6 and 5.7).

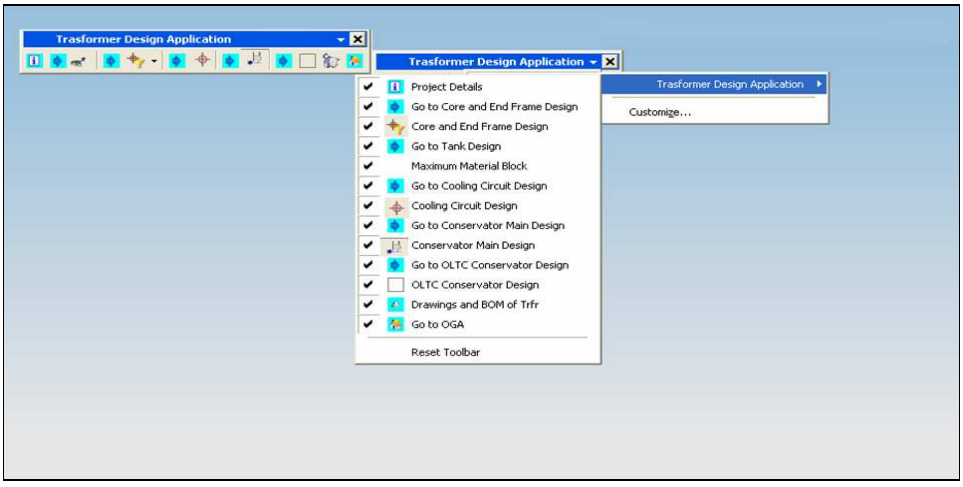


Figure 5.3: Transformer design menu bar

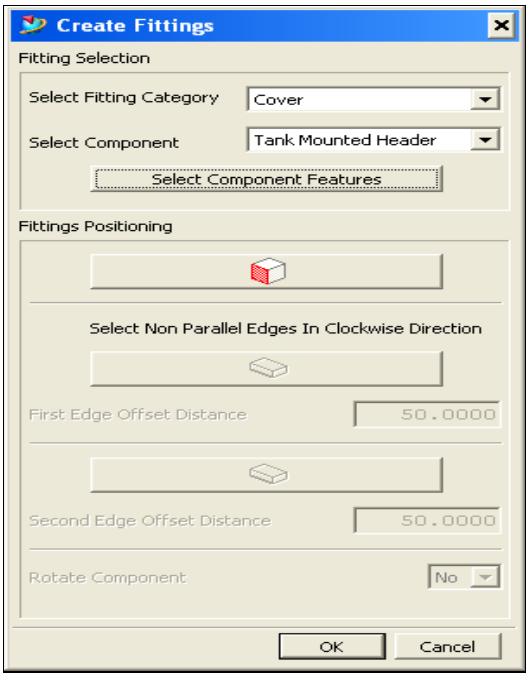


Figure 5.4: UI for tank fittings

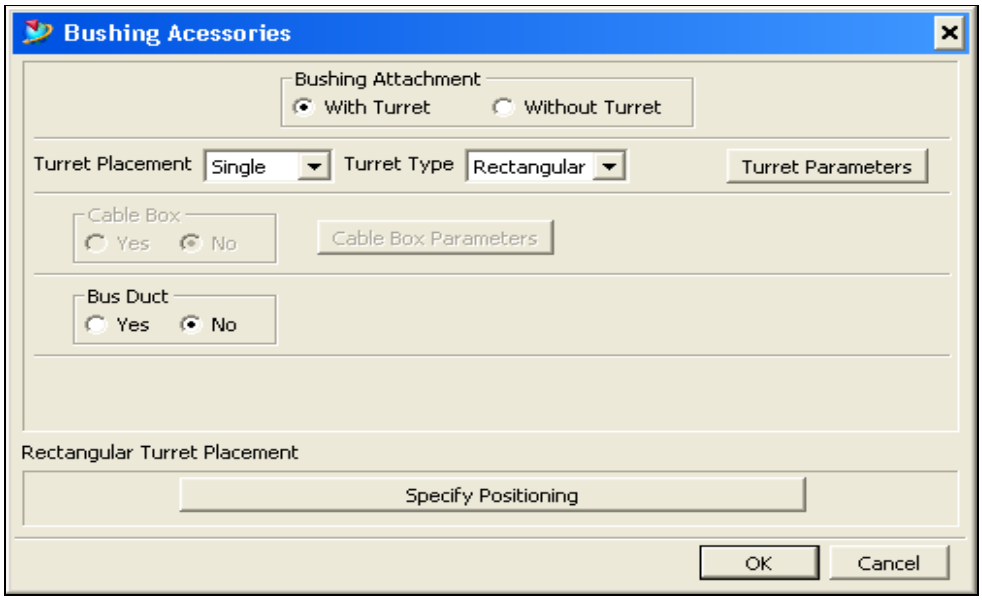
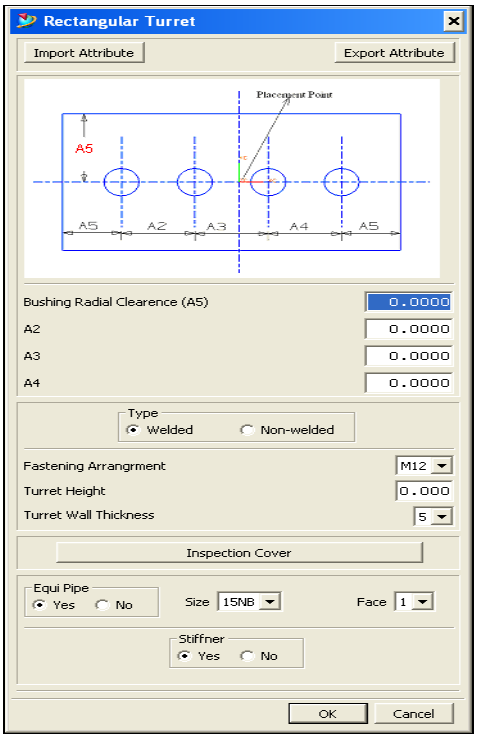


Figure 5.5: UI for selecting the type of turret and placements



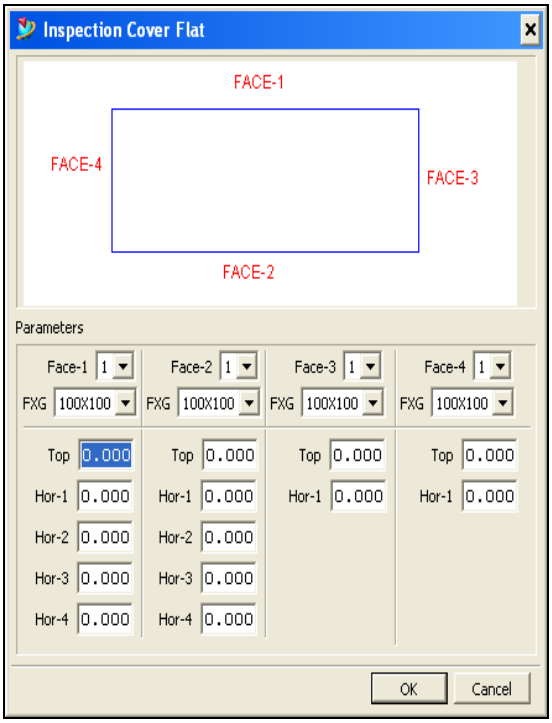


Figure 5.6: UI for updating (a) turret parameter (b) inspection cover parameter

The figure above shows the user interface for selecting the turret type and whether the turret is required or not. And if the turret is used then on which face and what will be distance from different edge and to enter the turret parameter. After selecting the turret parameter the following dlg file will come on the screen. In this the various parameter

are asked from the user, which are required for updating the turret. The figures shown are the user interface files, which are created in Unigraphics User Styler Interface module, which are then linked with the part with the help of coding. These files are integrated with the UG using the knowledge fusion. And the coding is done in the Ufunc for integrating it with the UG. And remove the use of entering the UG native by the customer. After the assembly the rectangular turret is shown in figure below.

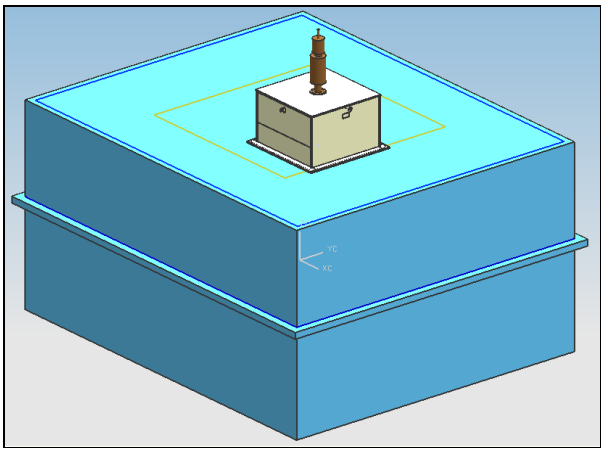


Figure 5.7: Assembly of tank body and the rectangular turret

In the similar way the different turret based on number of bushing can be placed. The bushing parameters can be updated as per the user requirement by just giving the parameter value in the user interface data file for the turret. The assembly of rectangular turret with three bushing is shown below.

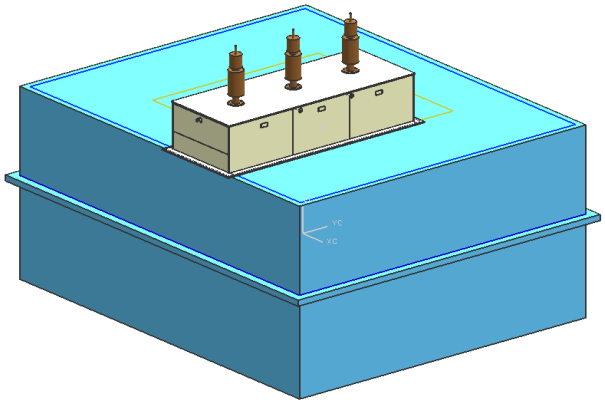


Figure 5.8: Assembly of tank body and rectangular turret

So finally the steps for automation of rectangular turret and integrating it with the automated transformer design wizard are as follows:

- Collect the required data for the modelling of the turret
- Create the model and give the required expression to the model according to design problem
- Create the dlg files and define the identifier to each parameter according to the requirement.
- Create the DFA file for the turret.
- Define the various parameters, which are required to update, and which are required for the assembly of the component as given in the user interface file.
- Define the various lists for the list parameters, which are given by the option menu to the user.
- Define the different name to the component as per the no. Of bushing as the clone file name.
- As the same component cannot be used for updating the component each time so the cloning of the component is required. So make a function as clone assembly using Ufunc.
- Define the full path for the component and name of the component given while modelling.
- Define the base string of the real component and the replace string, which is required for the clone part name.
- Clone the part as per the number of bushing required in the transformer assembly.
- Get the expression to be updated from the clone part file.
- Give the value of the parameter defined in the dlg file to the parameter of the clone file.
- As to update the values of the part file create a function in the Ufunc to update the expressions.
- Update the expression of the clone parts.
- Now for the assembly here the point is required. As the assembly in knowledge fusion works on the basis of the co-ordinate system.
- Use the select face and select edge function for getting the point for the assembly.
- Pass these points to the ug_add_component and the component at the required point.
- Give the attribute to each part of the turret as specified by the collected data for the bill of material.
- Integrate the component DFA with the main transformer DFA.

By following the above steps the automation and user interface for the parts can be achieved. To complete full turret wizard design automation for following assembly also has been done:

- Header pipe work
- Cooler pipe work

5.3. Software Development

UG/open is software designed that allows access to program functionality without requiring access to the source code. UG/Open API is a toolkit of C language callable routines for application programmers to write programs that work directly with Unigraphics shown in figure 5.9. Unigraphics solutions provide a library of C language functions; include files and FORTRAN Subroutines that interface with Unigraphics object database (Unigraphics Object Model).

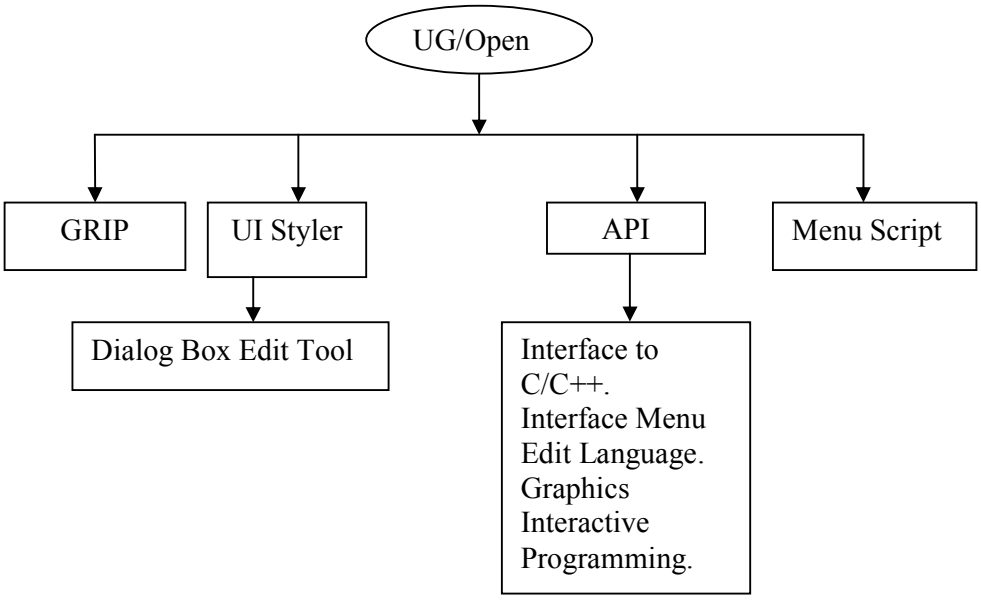


Figure 5.9: Structure of UG/Open

As shown above that using API ,we can customize the library of UG as per our convenience, by making the required programs in C/VC++ and then converting them to DLL with the help of COM/DCOM i.e. Component Object Modeling or Distributed Component Object Modeling a facility provided in API.

5.3.1 Forms Of UG/Open API

UG/Open API programs can be developed in two forms i.e.

- Internal UG/Open API Program
- External UG Open API Program

The Internal UG/Open API program is a real locatable image that runs within the Unigraphics interactive process .Programs can be developed to create and access geometry, analyze geometry, create and edit features, create and edit expressions and manage data. Routines to interact with users using standard Unigraphics Motif widgets are provided. Access to general motif calls is supported with in the UG/Open API.

An External UG Open API application is a program that runs without displaying a graphics window. External UG/Open API application program tend to be more for data management than for geometry manipulation. External UG/Open API application programs can construct and edit models, where no model display is necessary.

So, from above one can summarize that UG open architecture programs can be written as “External” or “Internal” programs. External programs run independently from the UG executable. That is, it is not necessary to start UG to run External programs. External programs use UG code simply for the numerous libraries. Since UG is not running, no graphical interaction is possible unless it is specially programmed in. On the other hand, internal programs are compiled as dynamically linked libraries and must have a function that matches the prototype:

The Application Programmer Interface (API) is basically used to convert /linking a basic c code for user interface library. Using this we create our Database as well as our library which could be used further during assembly and modeling for performing the different tasks.

Some useful applications of UG/Open API includes

- Geometry/ Feature creation and editing
- Expression Creation/Manipulation
- Geometry Analysis
- Part standardization
- File management
- Data Access

- Family of parts
- Create and interact with UI Styler Dialogs
- Create/maintain user defined and smart objects

5.3.2 GRIP (Graphics Interactive Programming)

GRIP is used to create FORTRAN like programs to operate the Unigraphics systems. Many operations that can be performed interactively can also be performed by using the GRIP program. Commands are available to create, geometric and drafting entities, control preferences, perform file management functions and modify existing geometries. GRIP also provides the interactive commands. These commands display messages in a motif dialog, allow the user to interact with a GRIP program while it is running. The message displayed by these commands can be programmed to fit the user's specific needs. There are interactive commands to control entity selection, menu option selection, data entry, text entry and generic point sub function.

API CAM's functionality increases significantly. GRIP NC code is now almost redundant. GRIP NC is still officially required to assign Point-to-Point geometry and unofficially required to fill in the blanks for API CAM's bugs missing parameters and missing UDE's. Almost all of the operation parameters for milling are in place and Point to Point is close behind. GRIP NC will soon be redundant. API is rapidly gaining on GRIP NC as the CAM automation language of choice. GRIP as discussed above is mainly used in UG for developing the machining sequences which could further be optimized but now most of the work have carried out in API. So from above discussion and also in the project GRIP is mainly used for creating just user interfaces instead of dealing with big conceptual programs.

Some useful applications of GRIP Application

- **Grip/Unigraphics**

  Special Geometric Functions

  Calculation and Analysis

  Plotting

  Part Standardization

  File Management

  Data Access

Family-of-part-programming

- **GRIP/NC Machining**
  Basic Machining
  Planar Milling
  Surface Milling
  Lathe Machining

- **Grip/ Finite Element Modelling**
  Node Creation and editing
  Element creation and editing
  Mesh Node/ Element creation
  Mesh definition by surface

5.3.3 Comparison Of GRIP And API [10]

API is the preferred language to any C programmer. It has more overhead than GRIP but it executes faster and can interface with the world out there through the Internet, HTML files, etc. While the instruction sets are long and tedious, it can be easily customized. Generic functions can be created and easily accessed from any other API program or function. This is a little more tedious than the single line GRIP command that does the same thing. The C language is very friendly, however, when it comes to sharing code among routines. The point is that this type of function can be created for all the machining operation types and subtypes and you can call them with a single line command packed with the appropriate parameters. I store these functions in a header file.

Using API we can create libraries of common functions and call them with C++ classes. Either way, the transition from GRIP, while difficult at first, can be done easily. One of the major benefits of API CAM is the ability to access the ON interactively during program execution. This makes it possible to select CAM objects in a similar manner to selecting geometry and drafting objects. This facilitates the use of an API program for testing the integrity of operations prior to output. Some companies require this as part of their QA. Currently the ON cannot be refreshed until exiting the API. This is an inconvenience but according to Tue Doan at GTAC it has been proved and will be fixed.

In the section 5.2, and 5.3 procedure for designing and integrating of automated rectangular turret assembly is discussed. The DFA (design for assembly) files are used to integrate the rectangular turret assembly to the transformer assembly. The software named Transformer design wizard is developed. The transformer design wizard is an expert system for the transformer design automation. As shown in the figure below the transformer design wizard can be started by clicking just one menu button in the screen. For installing the rectangular turret, as it comes under the tank body fitting (highlighted in menu below) is selected. The figure below shows the menu location in transformer design wizard. By selecting tank body fitting the menu for the tank body fitting will come on the screen. The turret is installed with the bushing so the bushing is selected from menu.

After selecting the bushing (shown in figure 5.6) the menu of low voltage and high voltage bushing will appear on the screen and from that menu we can select the installing accessory for the bushing. Selecting the rectangular turret for installing and giving the placement plane and distance from the edge the bushing can be placed on the tank body.

As the numbers of bushing selected are three because of that the placement point for the three turrets is asked from the user. And for changing the turrets the icon Turret Parameter (as shown in figure 5.7) is clicked and the turret parameter can be changed as per the requirement by giving the appropriate value to these parameter. And the drawing can created easily by just click one icon and similarly the bill of material can be created very easily.

CHAPTER 6

Results and Discussion

This software program tested for different rectangular turret design conditions. The following input parameter (as shown in table 6.1) are required to generate full design wizard for rectangular turret:

Table 6.1: Input parameters for rectangular turret

SI No.	Input For Rectangular Turret
1	Transformer Rating
2	Type of turret(Rectangular or Round)
3	Turret Welded or non welded on tank body
4	Turret Positioning
5	Type of fastening arrangement
6	Equalizing pipe required or not
7	Position of Equalizing pipe on turret face
8	Stiffener Required or not
9	Position of inspection cover on turret

After giving the above design input parameters following design output parameters are generated for different rated transformer:

6.1. Design Output Parameters For 3 Phase 250 MVA Transformer

Design Output parameters for 3 phase 250 MVA transformer have been obtained (as shown in Table 6.2) after giving the input (as shown in Table 6.1) and corresponding graphs are shown below.

Table 6.2: Output for rectangular turret of 3 Phase 250 MVA Transformer

SI No	Design Parameter For 250 MVA Transformer	With out Automation	With Automation
1	No of turret	1	1
2	No of bushing	3	3
3	No of Inspection cover	5	5
4	No of equalizing pipe	3	3
5	No of stiffener	2	2
6	Turret thickness	5	4.98
7	Welding distance	15	14.97
8	Equalizing pipe diameter	15	14.98
9	Bushing mounting pad diameter	150	149.8
10	Distance between 1st and 2nd bushings	1000	999.78
11	Distance between 2nd and third bushings	1000	999.87
12	Distance of inspection cover from top face	400	399.78
13	Radial clearance of bushings	600	599.87

14	Distance of inspection cover from horizontal face	600	599.94
15	Length of Top Flange in isolated bus duct	2246	2245.78
16	Length of Top Flange in segregated bus duct	5523	5522.87
17	Offset length for segregated bus duct	2599	2598.97
18	Distance of outer face from 2nd plate	1216	1215.89

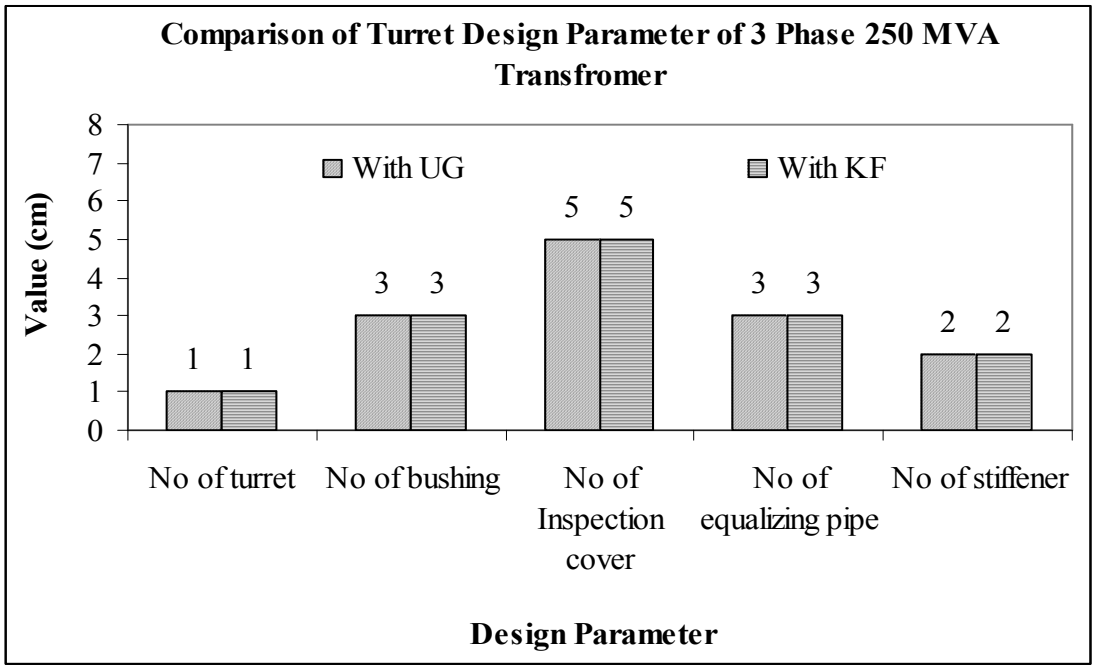


Figure 6.1: Comparison of turret parameter of 3 Phase 250 MVA transformer

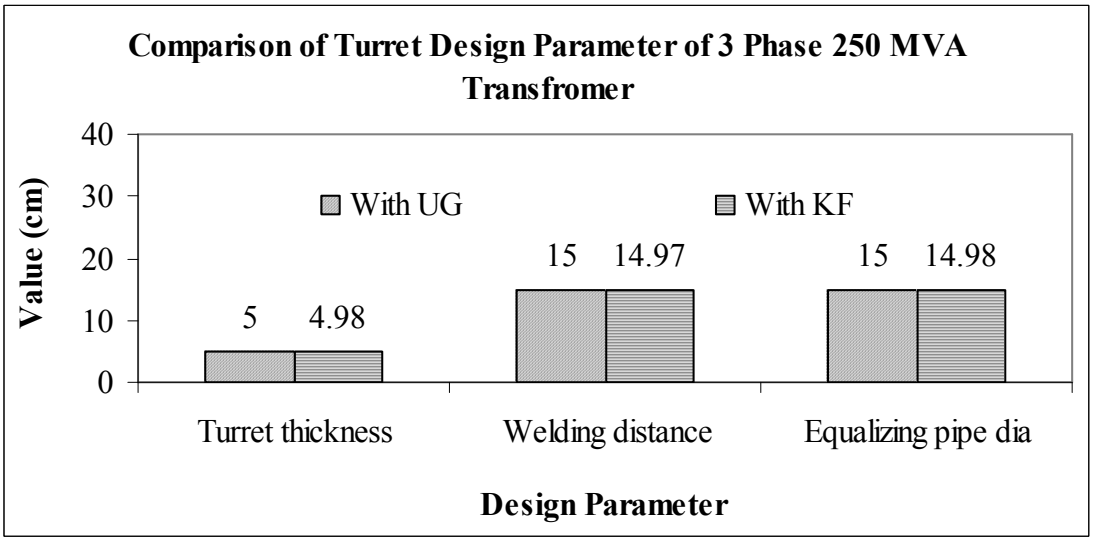


Figure 6.2: Comparison of turret parameter of 3 Phase 250 MVA transformer

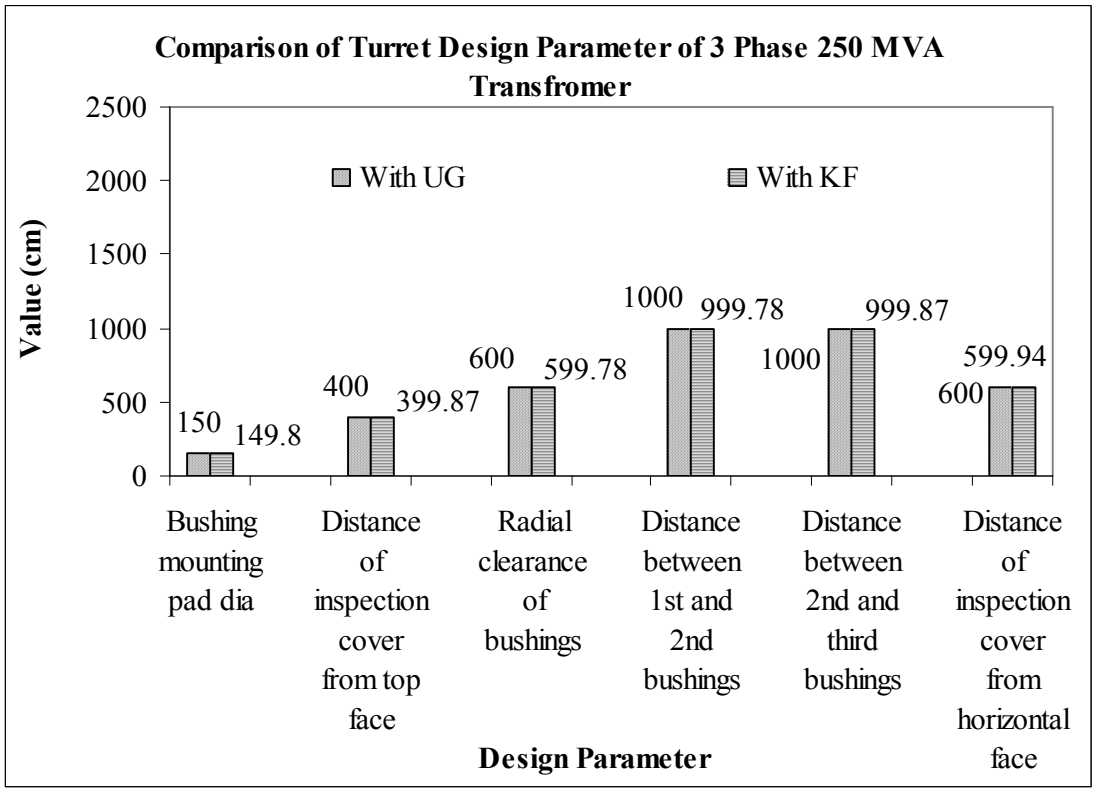


Figure 6.3: Comparison of turret parameter of 3 Phase 250 MVA transformer

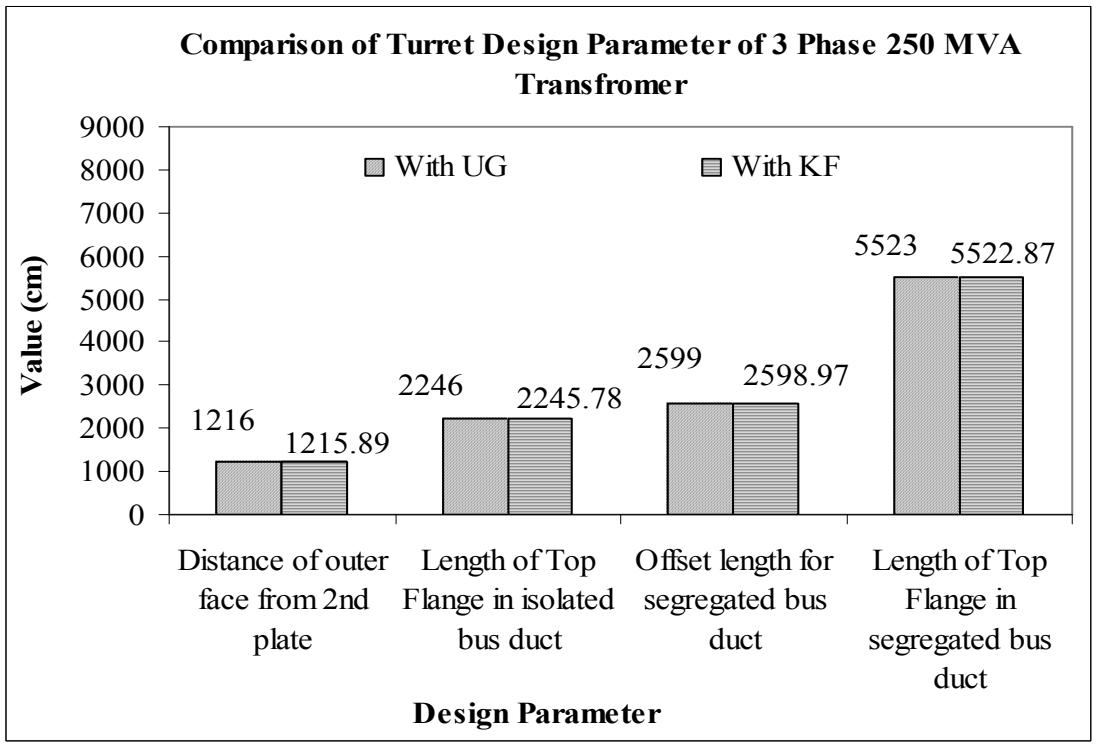


Figure 6.4: Comparison of turret parameter of 3 Phase 250 MVA transformer

6.2. Design Output Parameter For 3 Phase 400 MVA Transformer

Design Output parameters for 3 phase 400 MVA transformer have been obtained (as shown in Table 6.2) after giving the input (as shown in Table 6.1) and corresponding graphs are shown below.

Table 6.3: Output for rectangular turret of 3 Phase 400 MVA Transformer

SI No	Design Parameter For 400 MVA Transformer	With out Automation	With Automation
1	No of turret	2	1
2	No of bushing	4	3
3	No of Inspection cover	6	5
4	No of equalizing pipe	4	3
5	No of stiffener	3	2
6	Turret thickness	8	7.98
7	Welding distance	25	24.96
8	Equalizing pipe diameter	25	24.89
9	Bushing mounting pad diameter	200	199.8
10	Distance between 1st and 2nd bushings	1200	1199.78
11	Distance between 2nd and third bushings	1200	1199.87
12	Distance of inspection cover from top face	500	499.87
13	Radial clearance of bushings	800	799.78
14	Distance of inspection cover from horizontal face	800	799.94
15	Length of Top Flange in isolated bus duct	2746	2745.78
16	Length of Top Flange in segregated bus duct	5863	5862.89
17	Offset length for segregated bus duct	2989	2988.78
18	Distance of outer face from 2nd plate	1781	1780.698

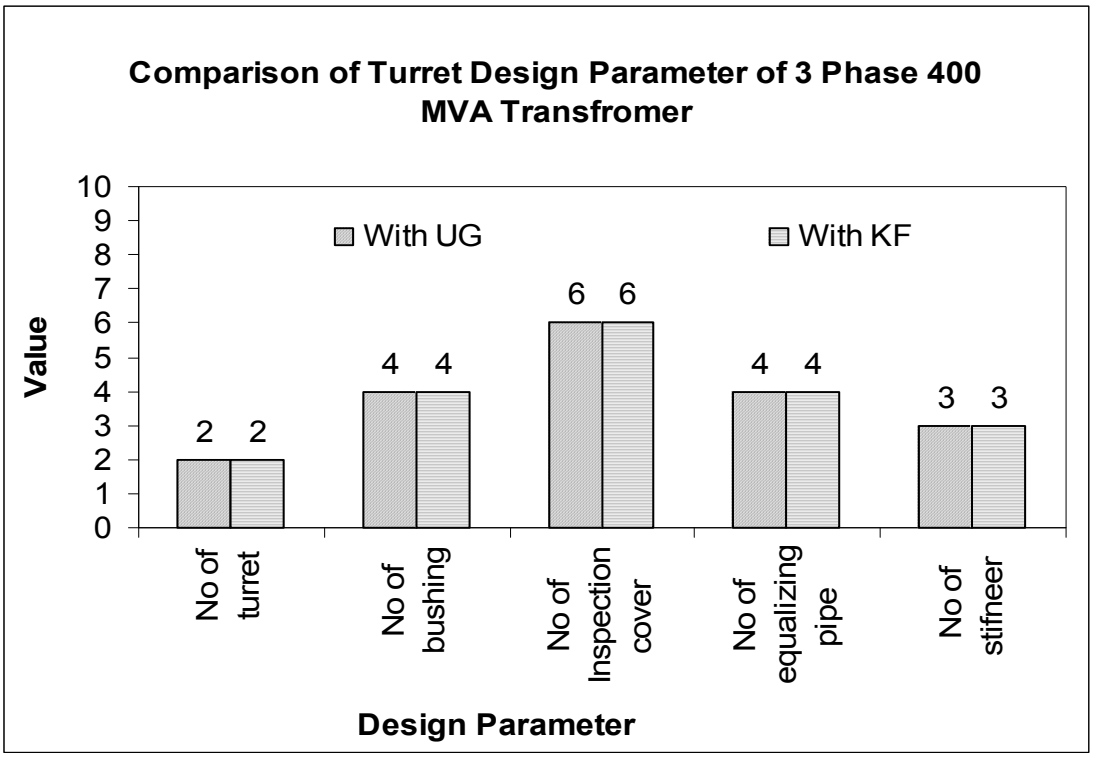


Figure 6.5: Comparison of turret parameter of 3 Phase 400 MVA transformer

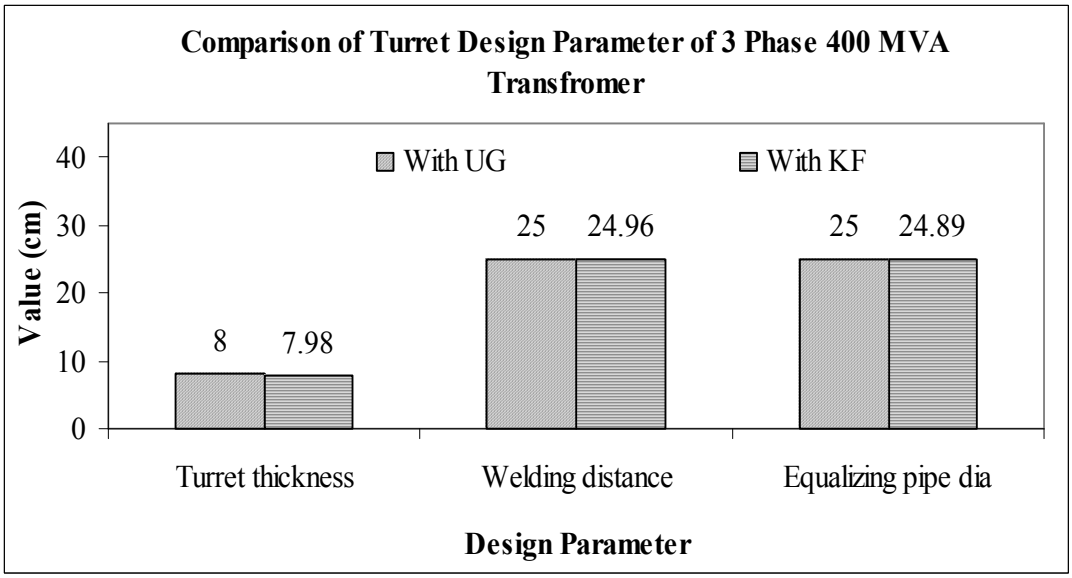


Figure 6.6: Comparison of turret parameter of 3 Phase 400 MVA transformer

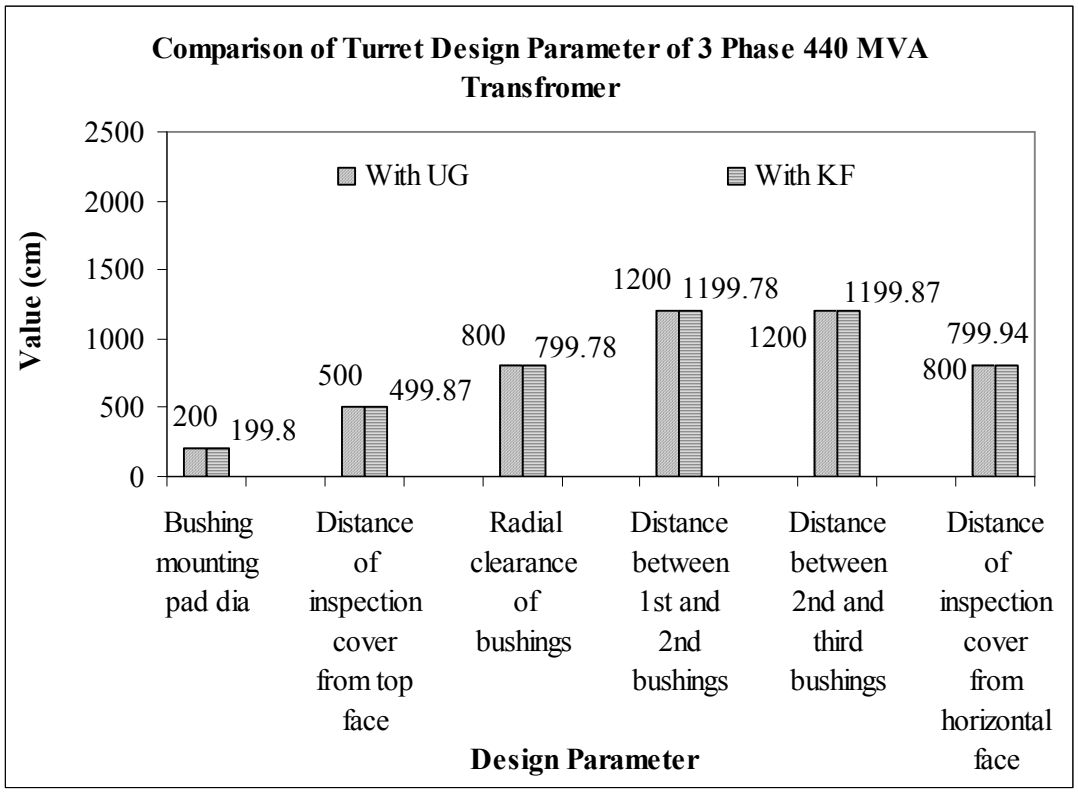


Figure 6.7: Comparison of turret parameter of 3 Phase 400 MVA transformer

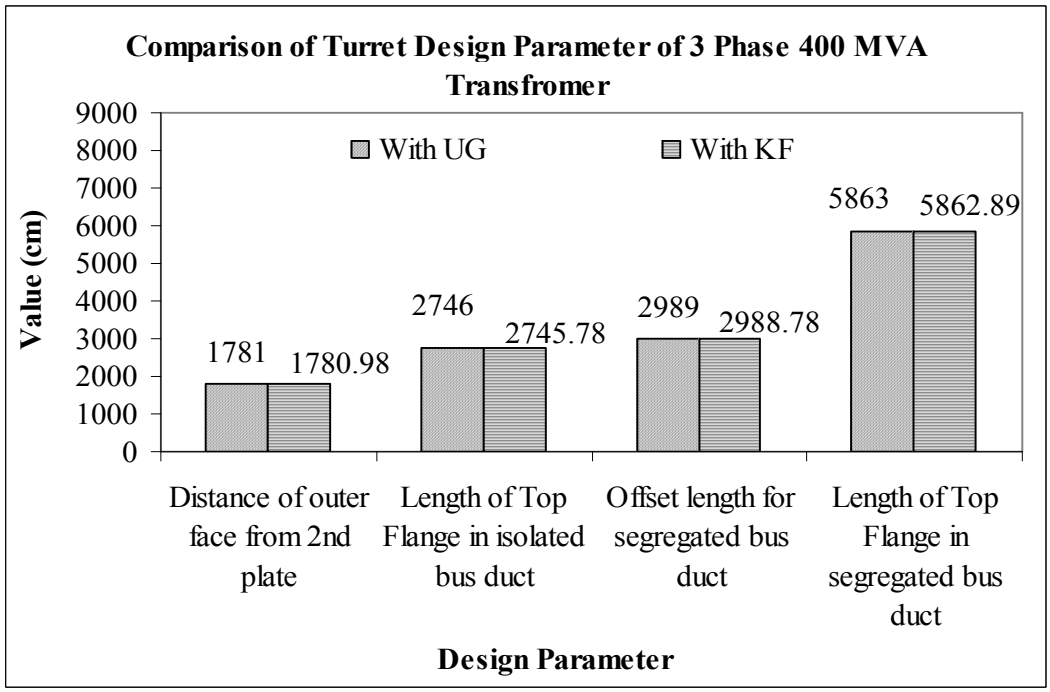


Figure 6.8: Comparison of turret parameter of 3 Phase 400 MVA transformer

6.3 Productivity Results

Product development process contains repetitive design tasks, which requires automation. The following design tasks are taken for comparison between model with

out design automation and with design automation so that increased productivity can be realized:

1. Knowledge Database Creation

Earlier when different orders from different customers came and they had requirement of different design of transformers on the basis of various voltage ratings. At that time designers made different database for different companies, but when design automation is implemented no need to make many numbers of databases. A single database can be used for various design problems with some minor modifications. An example is shown below for material properties. In this example, Knowledge Fusion interface to material databases is shown.

Step 1

Create a database.

1. Create a new database, using Microsoft Access, and name it materials.
2. Create a new table named properties:

Table 6.4: Material database

Material	Density, ρ (kg/m^3)	Ultimate Strength, σ_u ($N/m^{2)}$	Yield Strength, σ_y (N/m^2)	Modulus of elasticity, E (N/m^2)	Coefficient of thermal expansion, α ($per^{\cdot o}C$)
Oak	0.084	35000	22000	9000000	0.0000165
A242	0.254	70000	50000	29000000	0.0000065
302	0.286	125000	75000	28000000	0.0000096
Fiber-36	0.051	7000	42000	10600000	0.0000128
6061-T6	0.980	4200	27000	10000000	0.0000131

Step 2

Define a data source.

1. Choose Start→Settings→Control Panel.
2. Double click on Administrative Tools.
3. Double click on Data Sources.
4. Choose User Data structure.as shown in table 6.4.
5. Select Microsoft Access Driver material data base.
6. Enter materials in the Data Source Name box.
7. Find and select the database you created in step 1.
8. Choose OK  .

Step 3

Retrieve door.prt and add a material attribute.

1. If door.prt is not open, retrieve this part and open the KF Navigator.
2. Enter material in the Name box.
3. From the Type list, choose String.
4. In the Formula box, enter “6061-T6”.
5. Select the Input Parameter check box (it should be checked). Verify that Modifiable is unchecked .
6. Choose OK  .

Step 4

Add the database child rule.

1. In the Name Box, enter database.
2. From the Class List box, select ug_odbc_database.
3. From the Input Parameters box, select database.
4. In the Rule for Parameter box, enter “materials”.
5. Choose Apply  .

Step 5

Add the table child rule.

1. In the Name Box, enter table.
2. From the Class List box, select ug_odbc_recordset.
3. From the Input Parameters box, select database.
4. In the Rule for Parameter box, enter database.
5. In the Rule for Parameter box, enter “select from properties of material.
6. Choose OK  .

Step 6

Add properties and density attributes.

1. Move the cursor over root and click MB3. Choose Add Attribute dialog displays.
2. Enter properties in the Name box.
3. From the Type list, choose List.
4. In the Formula box, enter

   { table:movefirst:();

```
    table:getrecord:();
};
```

5. Verify that Modifiable is unchecked.
6. Choose Apply  .
7. Enter density in the Name box.
8. From the Type list, choose Number.
9. In the Formula box, enter nth (2, properties :);.
10. Choose OK  .

Step 7

Edit the door body to reference the density attribute.

1. Move the cursor over door_body and click MB3. Choose Edit. The Edit Child Rule dialog displays.
2. From the Input Parameters box, select Density.
3. In the Rule for Parameter box, enter density.
4. Choose OK  .
5. Save the part.

Step 8

Edit the material.

1. Move the cursor over the material attribute, click MB3 and choose Edit. The Edit Attribute dialog displays.
2. Edit “6061-T6” to be any value in the Material column of the database created in step 1. Choose Apply. Notice the affect on the mass attribute of the door_body object.
3. Close the part.

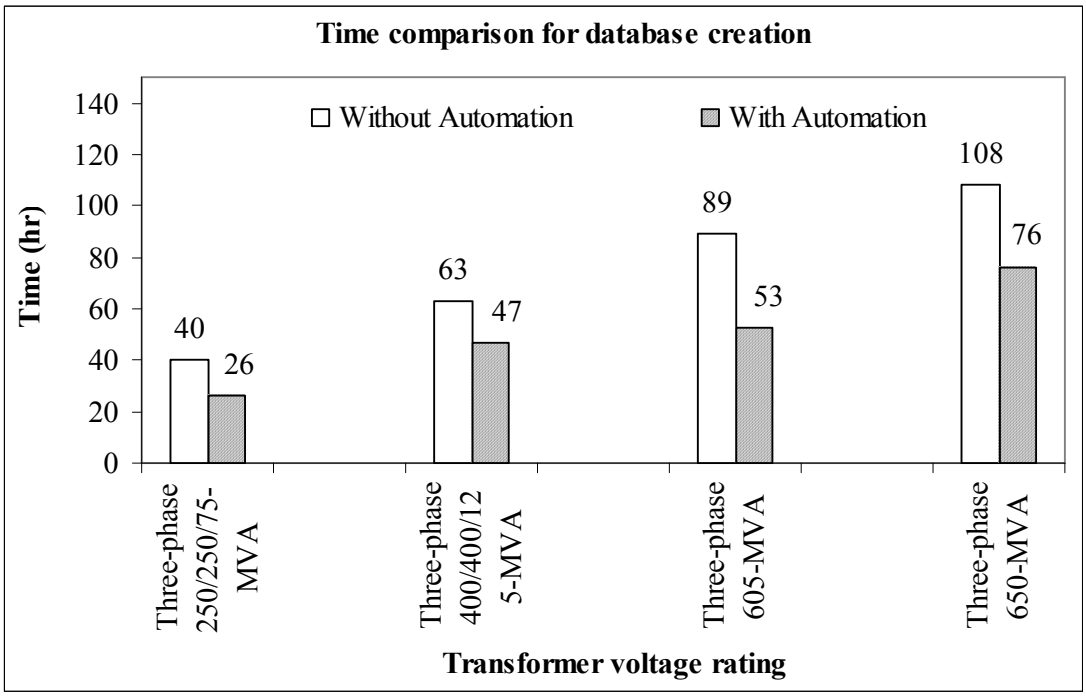


Figure 6.9: Time comparison for database creation

2. CAD Modeling

Like database, in modeling we have to make lot of changes with in models according different design requirement of transformer. However most of the CAD software provides parametric modeling, but it takes lot of time as a typical transformer may consist of as many as 3000 components/sub-assemblies. Through this design automation no need to remodel the various model to meet the requirements. It enables the designer to generate the set of parts/models with the help of coding. This is achieved by the user interface, which takes the necessary input from the designer and processes it according to the options specified. An example is shown below:

Further this example shows Knowledge Fusion interface to UG spreadsheets.

Step 1

Open a new part and create a spreadsheet.

1. Choose File → New. Enter spreadsheet in the File name box. Accept Units as inches. Choose OK .
2. Choose Tools → Spreadsheet.
3. Enter the data from spread sheet.
4. Save and exit the spreadsheet: File → Close & Return to Gateway.

5. Save the part.

Step 2

Open a new part and access the KF Navigator.

1. Choose File → New. Enter block in the File name box. Accept Units as inches. Choose OK 
2. From the graphics window, click MB3. Choose Replace View → TFR-TRI.
3. Choose View → Knowledge Fusion Navigator. Click the + sign on Attributes under the root node.

Step 3

Add a row attribute.

1. Move the cursor over root and click MB3. Choose Add Attribute dialog displays.
2. Enter row in the Name box.
3. From the Type list, choose Integer.
4. In the Formula box, enter 1.
5. Choose OK 

Step 4

Add the spreadsheet child rule.

1. Move the cursor over root and click MB3. Choose Add Child Rule.
2. From the Class List box, select ug_spreadsheet.
3. From the Input Parameters box, select part_file:.
4. In the Rule for Parameter box, place the cursor between the quotes and enter spreadsheet.prt.
5. Choose Apply 

Step 5

Add the block child rule.

1. In the Name Box, enter block.
2. From the Class List box, select ug_block.
3. From the Input Parameters box, select Length<1 ;>.
4. Backspace over the 1 so that only the semicolon displays.
5. Enter ss:ask_number:(row:,1).

6. From the Input Parameters box, select Width<1 ;>.
7. Backspace over the 1 so that only the semicolon displays.
8. Enter ss:ask_number:(row:,2).
9. Choose OK 
10. Save the part.

Step 6

Edit the row.

1. In the KF Navigator expand the root attributes by clicking the "+" sign.
2. Move the cursor over the row attribute, click MB3 and choose Edit. The Edit Attribute dialog displays.
3. Edit 1 to be any value from 1 to 6. Choose Apply. Notice that the block dimensions change.
4. Close the part.

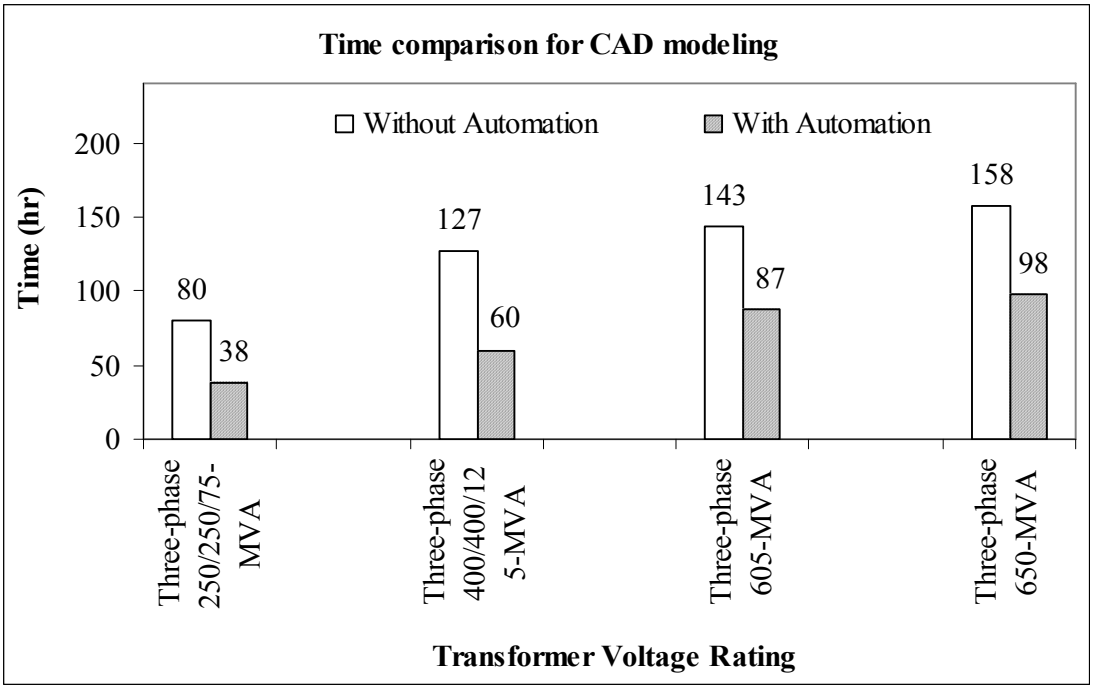


Figure 6.10: Time comparison for modeling

3. Assembly

As earlier when requirement for any assembly change had been come, designer started to reassemble it accordingly. Re-assembly of any parts always takes time, but this design automation provides the flexibility to assemble the parts through coding. Designer can reassemble the parts by changing only the coordinates in 3D space.

With this example, we will become familiar with creating an assembly using Knowledge Fusion.

Step 1

Create a new component parts and access the KF Navigator.

1. Choose File → New. Enter bushing in the File name box. Accept Units as inches. Choose OK 
2. Choose File → New. Enter turret in the File name box. Accept Units as inches. Choose OK 
3. Choose File → New. Enter tank body in the File name box. Accept Units as inches. Choose OK 
4. From the graphics window, click MB3. Choose Replace View → TFR-TRI.
5. Choose View → Knowledge Fusion Navigator.

Step 2

Create a cylinder in bushing.prt.

1. Move the cursor over root and click MB3. Choose Add Child Rule.
2. In the Name box, enter bushing.
3. In the Class list select ug_child_in_part.
4. From the Input Parameters box, select Target_File_Name:.
5. In the Rule for Parameter box, place the cursor between the quotes and enter bushing.prt.
6. From the Input Parameters box, select Parameters:.
7. In the Rule for Parameter box, place the cursor between the braces and enter class, ug_cylinder, diameter, 3.0.
8. Choose OK  .

Step 3

Create a cylinder in turret.prt.

1. Move the cursor over root and click MB3. Choose Add Child Rule.
2. In the Name box, enter turret.
3. In the Class list select ug_child_in_part.
4. From the Input Parameters box, select Target_File_Name:.

5. In the Rule for Parameter box, place the cursor between the quotes and enter turret.prt.
6. From the Input Parameters box, select Parameters:.
7. In the Rule for Parameter box, place the cursor between the braces and enter class, ug_cylinder, height, 12.0.
8. Choose OK  .

Step 4

Create bushing components in turret.prt.

1. Move the cursor over root and click MB3. Choose Add Child Rule.
2. In the Name box, enter right_bushing.
3. In the Class list select ug_child_in_part.
4. From the Input Parameters box, select Target_File_Name:.
5. In the Rule for Parameter box, place the cursor between the quotes and enter turret.prt.
6. From the Input Parameters box, select Parameters:.
7. In the Rule for Parameter box, place the cursor between the braces and enter class, ug_component, file_name, “bushing”, origin, point(0,0,-1).
8. Choose Apply  .
9. In the Name box, enter left_bushing.
10. In the Class list select ug_child_in_part.
11. From the Input Parameters box, select Target_File_Name:.
12. In the Rule for Parameter box, place the cursor between the quotes and enter turret.prt.
13. From the Input Parameters box, select Parameters:.
14. In the Rule for Parameter box, place the cursor between the braces and enter class, ug_component, file_name, “bushing”, origin, point (0, 0, 12).
15. Choose OK  .

Step 5

Create turret components in tank body.prt.

1. Move the cursor over root and click MB3. Choose Add Child Rule.
2. In the Name box, enter rear_turret.
3. In the Class list select ug_component.

4. From the Input Parameters box, select File_Name:.
5. In the Rule for Parameter box, place the cursor between the quotes and enter turret.prt.
6. From the Input Parameters box, select X_axis:<Vector(1,0,0);>.
7. In the Rule for Parameter box, change Vector(1,0,0) to Vector(0,-1,0).
8. From the Input Parameters box, select Y_axis:<Vector(0,1,0);>.
9. In the Rule for Parameter box, change Vector(0,1,0) to Vector(0,0,1).
10. Choose Apply  .
11. In the Name box, enter front_turret.
12. In the Class list select ug_component.
13. From the Input Parameters box, select File_Name:.
14. In the Rule for Parameter box, place the cursor between the quotes and enter turret.prt.
15. From the Input Parameters box, select Origin:<Point(0,0,0);>.
16. In the Rule for Parameter box, change Point(0,0,0) to Point(0,6,0).
17. From the Input Parameters box, select X_axis:<Vector(1,0,0);>.
18. In the Rule for Parameter box, change Vector(1,0,0) to Vector(0,-1,0).
19. From the Input Parameters box, select Y_axis:<Vector(0,1,0);>.
20. In the Rule for Parameter box, change Vector(0,1,0) to Vector(0,0,1).
21. Choose OK  .

Step 6

Display the Assembly Navigator.

1. Choose View → Assembly Navigator.
2. Save the parts.

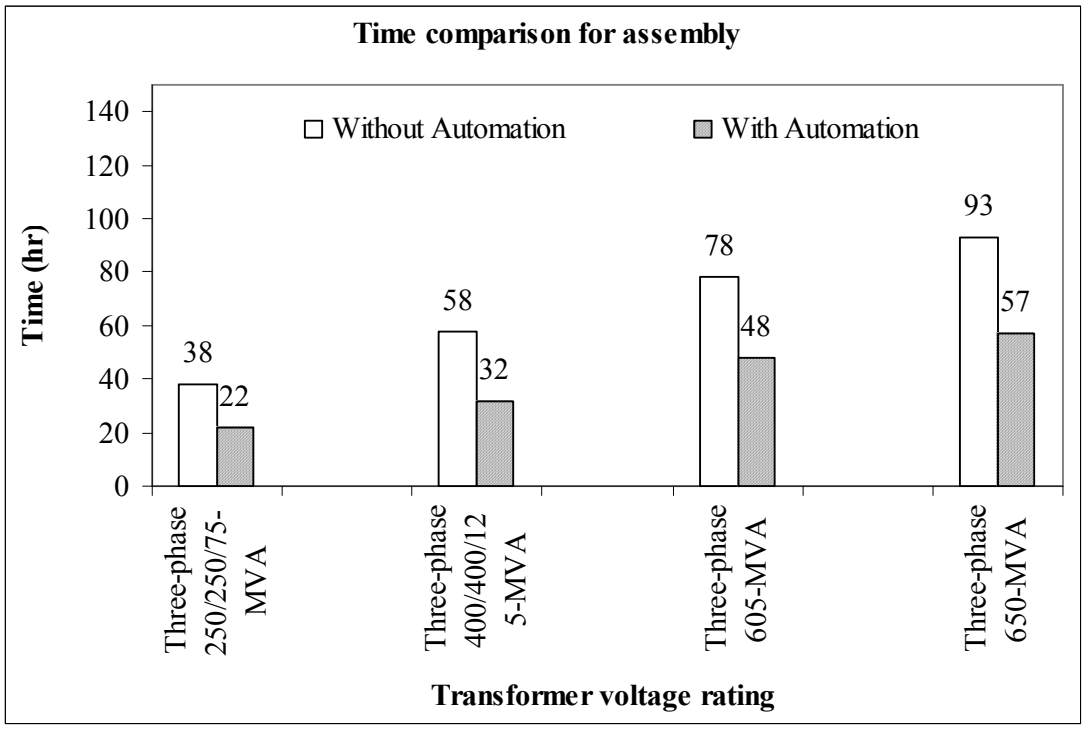


Figure 6.11: Time comparison for assembly

4. Drawings And Bill Of Materials

The complete engineering information requires 500 drawings/Bill of material. Earlier designer depended on handmade drawings made by draftsman. Moreover any fault in bill of material may cause a serious problem. But through this design automation drawings and bill of materials can be generated automatically. Further any error can be corrected easily without taking the significant time.

How to Add Attributes

This example demonstrates how to add attributes using the KF Navigator. In this exercise, you create four attributes: height, width, thickness, and position. These attributes are used in the next exercise, How to Add Child Rules.

Step 1

Open a new part and access the KF Navigator.

1. Choose File → New. Enter door in the File name box. Accept Units as inches. Choose OK.
2. From the graphics window, click MB3. Choose Replace View → TFR-TRI.
3. Choose View → Knowledge Fusion Navigator. Click the "+ "sign on Attributes under the root node.

Step 2

Add the height attribute.

1. Move the cursor over root and click MB3. Choose Add Attribute. The Add Attribute dialog displays.
2. Enter height in the Name box.
3. From the Type list, choose Number.
4. In the Formula box, enter 84.0.
5. Select the Input Parameter check box (it should be checked). Verify that Modifiable is checked and that Evaluate It should be checked. The Add Attribute dialog parameters should match the figure shown below.

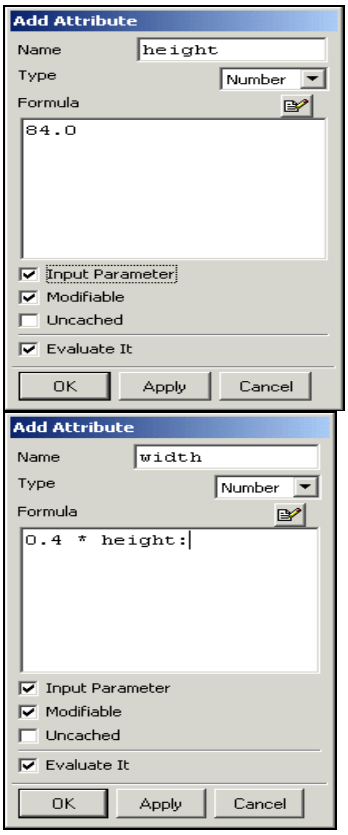


Figure 6.12: Add (a) attributes for height (b) attributes for width

6. Choose Apply  .

Step 3

Add the width attribute.

1. Enter width in the Name box. The Type list should read Number.
2. In the Formula box, enter 0.4*height.

3. In the KF Navigator, Move the cursor over the height attribute. Click MB3. Choose Reference. Note that height: appears in the Formula box. The colon after height references the value of the height attribute. The dialog should appear similar to the one shown below.
4. Choose Apply .

Step 4

Add the thickness attribute.

1. In the Name box, enter thickness.
2. In the Formula box, enter 2.5.
3. Choose Apply .

Step 5

Add the position attribute.

1. In the Name box, enter position.
2. From the Type list, choose Point.
3. In the Formula box, enter point (0, 0, 0). The dialog should appear similar to the one shown in the figure below.
4. Choose OK .
5. The KF Navigator should appear similar to the one shown in the figure below. Note the new attributes.

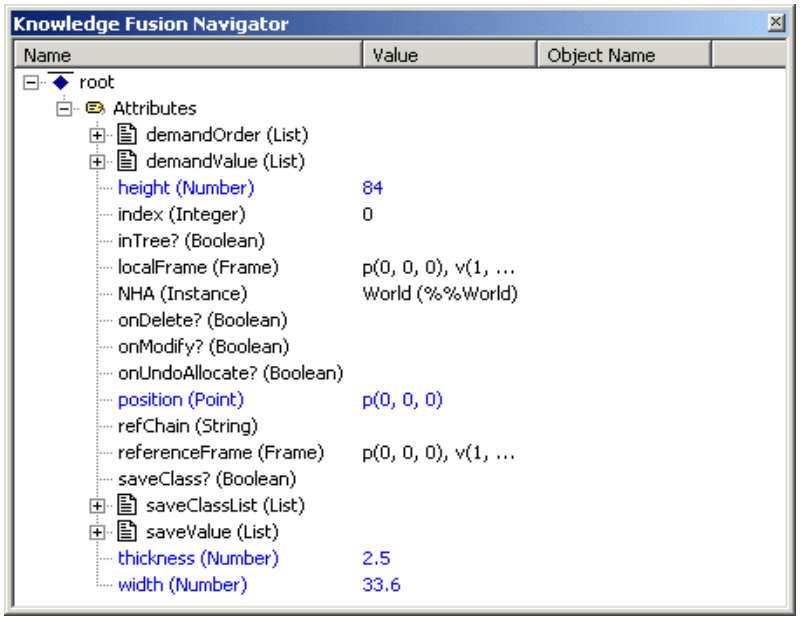


Figure 6.13: Knowledge Fusion Navigator

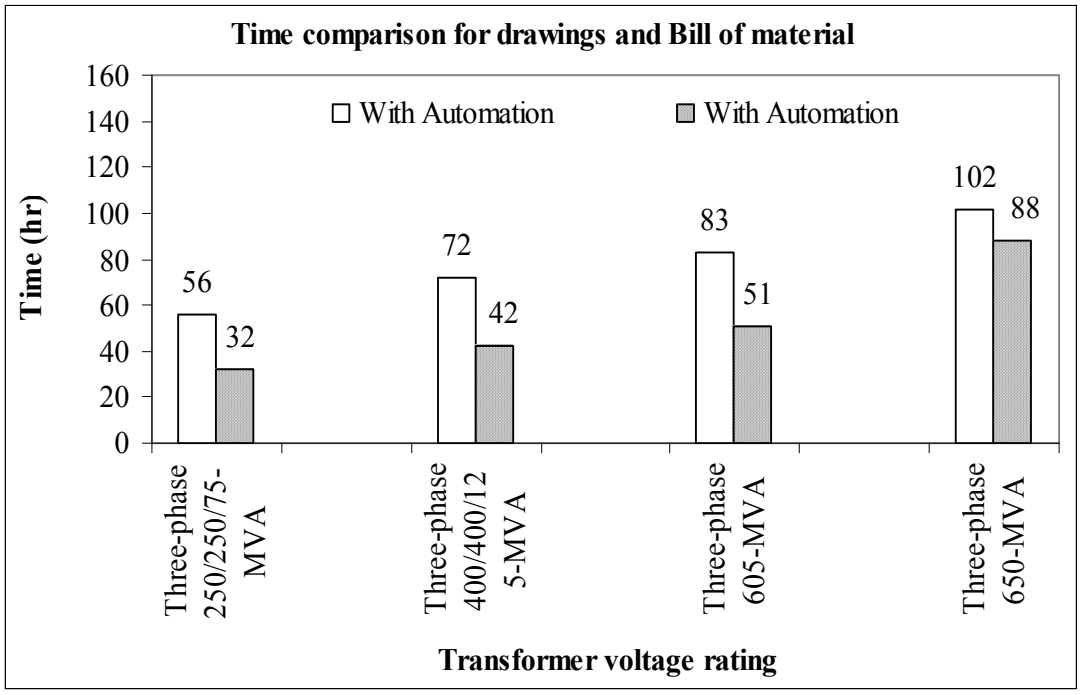


Figure 6.14: Time comparison for drawings and BOM

So following steps have to be executed to build design automation:

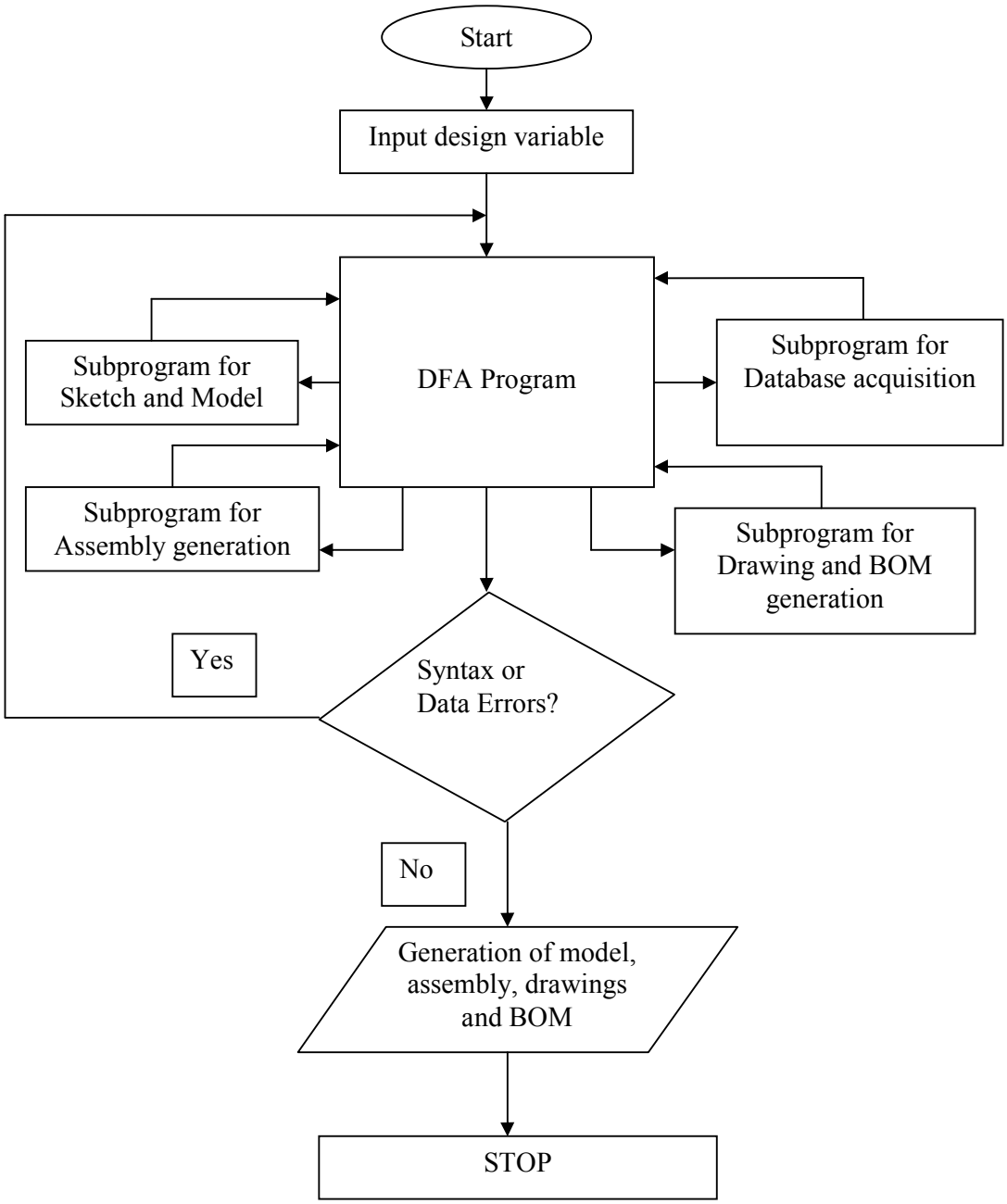


Figure 6.15: Flow diagram for design automation

CHAPTER 7

Conclusion

This research gives an idea about Knowledge Based Engineering (KBE) in product development. KBE can be seen as a tool for capturing knowledge and reusing it. This automated design tool helps in bridging the gap between design engineers and computational experts when analyzing product development process. An automated design system has been developed as a case study for rectangular turret a component of transformer. The following conclusions can be drawn from this research regarding the design automation and KBE:

- KBE is used in product development to automate mundane time demanding tasks.

- Design automation though KBE allows freedom to designer from above routine work so that more time could be used to come up with new innovative solutions.

- KBE offers optimization of product concepts in conceptual design phase.

- It is KBE tools which often are coupled to a geometry, model and assembly to enable automatic generation of product concepts in terms of virtual prototyping.

- Drawbacks are that it is time demanding to develop KBE systems and it can sometimes be seen as a black box" if the automated procedures are poorly explained to the user.

Future Scope

There future scope of research work may be as follows:

- KBE is a subset of Knowledge Based Systems (KBS) which is a spin off from artificial intelligence (AT). KBS is often referred to as expert system" because they intend to capture expert knowledge and sometimes also generate creative solutions. So Enhanced scope of KBE to append the KBE up to KBS so that it would be used as either an expert system or AI in product development process.

- It can also serve as an aid in the manufacturing planning to evaluate manufacturability and life cycle simulation of product for cost estimation.

- It is possible to prevent design conflicts due to parallel product development activities common in concurrent engineering, as engineers can jointly use the KBE.

References

1. Nytomt F. and Persoon F., “Functional Jet-Engine Component Concept and evaluation”, Department of Applied Physics and Mechanical Engineering, Division of Computer Aided Design, Lulea University of Technology, Sweden 1996.

2. “The Line on Design: How to Reduce Material Cost by Eliminating Design Waste.”,AT Kearney ,2003.

3. Sandberg, M., Boart, P. and Larsson, T., “Product life-cycle simulation application for cost estimation and conflict prevention in conceptual design of jet-engine components”, Journal Concurrent Engineering: Research and Applications, CERA, 1996.

4. “Transformers Second Edition”, Bharat Heavy Electricals Limited, Bhopal, TMH publication, 2003.

5. Boart, P., Sandberg, M., Nergård, H. and Isaksson, O. “A knowledge enabled engineering approach for conceptual design of life cycle properties”, Submitted to Journal of Computing and Information Science in Engineering, 1998.

6. Craig B. and Pinfold M., “The application of a knowledge based engineering approach to the rapid design and analysis of an automotive structure”, Advances in engineering softwares, 32, pp. 903-912, 1998.

7. Marcus S., “Knowledge Based Engineering in Product Development”, Division of Computer Aided Design Sirius Laboratory, Lulea University of Technology, Sweden, 1997.

8. Marcus S., “Knowledge enabled engineering design tools for manufacturability evaluation of jet engine components”, Division of Computer Aided Design, Department of Applied Physics and Mechanical Engineering, Luleå University of Technology, Sweden,2000.

9. Boart, P., Nergård, H., Sandberg, M. and Larsson, T. “A multidisciplinary design tool with downstream processes embedded for conceptual design and evaluation”, Accepted for presentation at International Conference of Engineering Design, Melbourne, August, 1998.

10. “Unigraphics NX technical Briefing, for Total Product Engineering”, August 2004.

11. “NX Digital Simulation, Experimentation is the key to innovation”, 2004.

12. . Bojcetic N., Storga M. and Marjanovic D., “Integrated Engineering Environment”, International Design Conference- Design, May 1999.

13. Pablo B. and Shing F.,” A KBE system for the design of wind tunnel models using reusable Knowledge components”, International Design Conference-Design 2000.

14. Dirk C., “The use of software system to implement case based reasoning enabled intelligent component for production briefing and design”, October 2001.

15. “Unigrhaphics Knowledge Fusion Participant Guide”, 2005 Users Group, Orlando, May 21-May 24, 2005.

16. Zimmermann M. and. Bronsart R, “Application of knowledge-based engineering methods for standardization and quality assurance in ship structural design”, September 20001.

17. Thomas M. Korman and Tatum C.B., “Development of a Knowledge-Based System to Improve Mechanical, Electrical system”, CIFE Technical Report 129, 2002.

18. Daniel E. Whitney, James L. Nevins and Thomas L. De Fazio, “Problems and Issues in Design and Manufacture of Complex Electro-Mechanical Systems”, The Charles Stark Draper Laboratory, Inc. Cambridge MA 02139, 2002.

19. John S. Gero,”An overview of knowledge engineering and its relevance to CAD”, Futures digital proceedings, pp 107-119, 1986.

20. Vigain H., Mats N., Derrick Tate, and Nam P. Suh,”Decision Making and Software Tools for Product Development”, 2002.

21. Mario S., Davor P. and Dorian M.,”Reducing Design Development Cycle By Data Management Within The Design Office”, 2002.

22. Zeng S., Edward J. Kim, Gregory M. Mocko, Xiao A., Russell P. and Farrokh M.,” Systematic Design Method For Information Modeling In CAD/CAE”, 2003.

23. Zdenek Z., Mulholland P., Domingue J. and Hatala M.,”Sharing engineering design knowledge in a distributed environment”, Knowledge Media Institute, The Open University, Milton Keynes, MK7 6AA, UK, 2003.

24. Boart P., “Life Cycle Simulation Support for Functional Products”, Division of Computer Aided Design, 2003.

Appendix

Example of code for sketch, model, assembly, drawing and BOM generation

```
/*  Include files */
#include <stdio.h>
#include <uf.h>
#include <uf_error_bases.h>
#include <uf_kf.h>
#include <uf_obj.h>
#include <uf_clone.h>
#include <uf_part.h>
#include <uf_modl.h>
#include <uf_ui.h>
#include <stdlib.h>
#include <windows.h>
#include "clone.h"

(child) line_4: {
                design; line;
                thruPoint1; origin:;
                thruPoint2; Point(1; 0; 0);
                };

(child) const_1: {
                design; UgSketchPerpendicularConstraint;
                line1; line_1:;
                line2; line_2:;
                };

(child) const_2: {
                design; UgSketchPerpendicularConstraint;
                line1; line_2:;
                line2; line_3:;
```

```
                    };

(child) const_3: {
                    design; UgSketchPerpendicularConstraint;
                    line1; line_3:;
                    line2; line_4:;
                    };

DefClass: block  (UgBody );
          (instance) mainBlock: {
                    class; UgBlockFeature;
                    length; 10;
                    width; 10;
                    height 10;
                    };

(instance) boss: {
                    class; UgBossFeature;
                    targetBody; mainBlock:resultBody:;
                    basePoint; Point(5; 5; 10);
                    radius; 1;
                    height; 2;
                    };

(instance) blend: {
                    class; UgBlendFeature;
                    edges;UgGetAllEdgesOfFace(UgFaceOnPoint(boss2:resultBody:;
                    Point(1; 1; 10)));
                    radius; 0.25;
                    };

Defclass: UgAssemblyComponent (...);
          (List Parameter) constraints: ;
          (Name Canonical) constraintLevel: NotUsingConstraints;
          (Group) use_constraints_solver: <constraints:> @{
```

```
        $mcs << GetAllMatingConstraints();
        $data << UGMCSolver($mcs);
        localFrame: << first($data);
        constraintLevel: << second($data); # perhaps
        # raise errors under some conditions, etc.
        };

DefClass: %TURRET_TDA_ui_rectangular_turret ( ug_base_part );
#####################################
# For First Turret Parameters
#####################################
# Radio button for selection of type of the turret
( integer Parameter modifiable)  ui_radio_turret_type:            2;

# Number parameter to get the ID
 ( Number Parameter modifiable)  ui_real_rectangular_turret_id:          900;

# Option menu to get the half theta between the holes of the bottom flange
( Integer parameter modifiable)  ui_opt_theta_flange_holes:        1;

# Radio button to select the equilising pipe
( Integer Parameter Modifiable ) ui_radio_equi_pipe:              1;
#############################################################
# UI Option Menu Lists
#############################################################
# List of option menus for half theta between the holes of the bottom flange
( List )  ui_opt_theta_flange_holes_list:        { 5, 6, 7.5, 9, 10, 12 };

# List of option menus for Bore Diameter
( List )    ui_opt_bore_diameter_list:           { "15NB","25NB" };

# List of option menus for number of CT
( List )    ui_opt_no_of_ct_list:                { 1,2,3,4 };

#############################################################
```

UI Control Option

##

(Boolean) ui_callback: TRUE;

(Boolean) ui_callback_visi: FALSE;

(Boolean) ui_sw_for_equilising_param_visi: (ui_radio_equi_pipe: = 1);

(Boolean) ui_sw_ct_param_visi: (ui_radio_ct: = 1) & (ui_radio_turret_type: = 2);

(Boolean) ui_sw_ct1_visi: (ui_radio_ct: = 1) & (ui_radio_turret_type: = 2) &
(ui_opt_no_of_ct: >= 1);

#(Integer) ui_real_ct1_height_sens: if (num_ct:>=1) then 1 else 0;

##

UI Buttons (children and attributes)

##

(Integer) ui_label_ct_terminal_board_visi: if (ui_radio_turret_type: = 2 &
ui_radio_ct: = 1) then 1 else 0;

(Integer) ui_real_angle_tb1_visi: if (ui_radio_turret_type: = 2 & ui_radio_ct: = 1 &
ui_opt_ct_term_board: = 1)|(ui_radio_turret_type: = 2 & ui_radio_ct: = 1 &
ui_opt_ct_term_board: = 2)|(ui_radio_turret_type: = 2 & ui_radio_ct: = 1 &
ui_opt_ct_term_board: = 3)then 1 else 0;

(Integer) ui_real_angle_tb3_visi: if (ui_radio_turret_type: = 2 & ui_radio_ct: = 1 &
ui_opt_ct_term_board: = 3)then 1 else 0;

##

Attributes referenced by parent

##

(Number) half_theta: nth(ui_opt_theta_flange_holes:, ui_opt_theta_flange_holes_list:
);

(Integer) int_weld_type: if(ui_radio_turret_type:=1) then 1 else 0;

(Number) num_bore_dia: if(ui_opt_bore_diameter:=1) then 15 else 25;

(Number) num_ct1_ht:: if (num_ct:=0) then 0 else ui_real_ct1_height:;

(Integer) int_W15: if (ui_radio_turret_type: =1) then 0 else if((ui_radio_turret_type: =2) & (ui_opt_ct_term_board:>=1)) then 1 else 0;

(Number)num_no_of_holes:nth(4,List_expression_values_rectangular_turret_app:);

(Number) num_inner_di:nth(1,List_expression_values_rectangular_turret_app:);

(Number) num_PCD: nth(2,List_expression_values_rectangular_turret_app:);

##############For cloning#################

(String) str_part_path_turret:: getenv("UGII_SITE_DIR") + "\parts\seedparts\Fittings\Rectangular_turret\";

(string) str_replace_string1:"TURRET1_";

(String) str_part_name_bom: "TURRET_RECTANGULAR_TURRET_BODY.prt";

(String) str_full_part_path_bom: str_part_path_turret: + str_part_name_bom:;

(String) final_dir_path1:getenv("UGII_TURRET_PROJECT_DIR") + "\Tank_Body\Rectangular Turret\";

################Axis Calculation ############

(Vector) vec_x_turret1: Rotatevector(vec_x:,ui_real_phi_bush1:,vec_z:);
(Vector) vec_y_turret1: Perpendicular(vec_x_turret1:);

(Vector) vec_x_turret2: Rotatevector(vec_x2:,ui_real_phi_bush2:,vec_z2:);

```
( Vector ) vec_y_turret2:        Perpendicular(vec_x_turret2:);

 ##################Add Children#############
( Child ) add_component_rectangular_turret:
{
class,if ((ui_radio_bushing_attachement:=1)&( turret_type: = "Rectangular" )
&(bushing_number:>=1)) then ug_component else Nulldesign;
File_Name,                str_clone_file_name_turret:;
Component_Name,                ui_opt_select_fittings:;
Origin;           if (ui_radio_turret_type:=1) then orgin_point1:+ (vec_z:)*(10) else
orgin_point1:+ (-vec_z:)*(10);
Reference_Set_Name,           "ASSEMBLY"
Color,            5;
X_axis,           vec_x_turret1:;
Y_axis,              vec_y_turret1:;
suppress?,                  FALSE;
};

#-------------------------------------------------------------------------------
# For BOM of Bottom Flange
#-------------------------------------------------------------------------------
(String )str_path1:       if(bushing_number:>=1) then getenv(
"UGII_TURRET_PROJECT_DIR" ) + "\Tank_Body\Rectangular Turret" +
"\TURRET1_TNF_RECTANGULARTURRET_ASSEMBLY_BLANKING_BOTTO
MFLANGE.prt" else if(bushing_number:>=2) then getenv(
"UGII_TURRET_PROJECT_DIR" ) + "\Tank_Body\Rectangular Turret" +
"\TURRET2_TNF_RECTANGULARTURRET_ASSEMBLY_BLANKING_BOTTO
MFLANGE.prt" else if(bushing_number:>=3) then getenv(
"UGII_TURRET_PROJECT_DIR" ) + "\Tank_Body\Rectangular Turret" +
"\TURRET3_TNF_RECTANGULARTURRET_ASSEMBLY_BLANKING_BOTTO
MFLANGE.prt" else getenv( "UGII_TURRET_PROJECT_DIR" ) +
"\Tank_Body\Rectangular Turret"
+\TURRET4_TNF_RECTANGULARTURRET_ASSEMBLY_BLANKING_BOTTO
MFLANGE.prt";
```

(Integer)int_set_Blanking_bot_flange_part_attribute:
FUNC_Turret_set_part_attributes(str_path1:,list_attr_names_Blanking_bot_flange:,list
_attr_values_Blanking_bot_flange:);

(List) list_attr_names_Blanking_bot_flange: { "TURRET_TDA_INDEX_DB" };

(String) str_Blanking_bot_flange_index:"527";

(list) list_attr_values_Blanking_bot_flange:{ str_Blanking_bot_flange_index:};