University of Alberta

Smart Modeling of Drilling-Well in an Integrated Approach

by

Shah Md Rajiur Rahman

A thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

Spring 2011

Edmonton, Alberta

Examining Committee

Dr. Zihui Xia, Mechanical Engineering

Dr. Ergun Kuru, Civil & Environmental Engineering

Dr. Amit Kumar, Mechanical Engineering

Dr. Yongsheng Ma, Mechanical Engineering

To my parents

Abstract

The current well planning practice is usually done section by section with limited help of some knowledge-based tools. This thesis presents an integrated approach and a software prototype developed for well planning. It considers the geological input, i.e. pore pressure, over burden etc., to generate a step by step interactive drilling plan. The implemented well planning stages include the casing setting depth, casing and hole size determination, casing selection and then drill string design and modeling. The system is integrated with a Computer Aided Design (CAD) system for generating three-dimensional parametric model. The conceptual design and CAD modeling system are integrated in such a way that any changes in the design will be reflected to the CAD model. Such intelligent CAD design practice is new in the drilling industry. An Operational Parameters module is also attached with the system to predict the drilling coefficients by using offset well data and determine the optimum weight on bit, and the drill string rotation that minimizes drilling cost per foot for a single bit run.

Based on this approach, an integrated well planning system can be fully developed and it will be very useful for the decision making of drilling companies.

Acknowledgement

I would like to take this opportunity to express my sincere thanks to my supervisor Dr. Yongsheng Ma for his continuous guidance, encouragement and advice on my academic works and life at the University of Alberta.

I would like to thank my colleagues; especially Md. Moin Uddin, Abiy Wubneh and Ganesh Gujarathi for their contribution to this study and for making the working delightful in Collaborative and Integrated Engineering Informatics Group (CIEIG) such as enjoyable experiences.

Last but not least, my sincere thanks goes to my family, my parents and my dearest wife Farzana Ahmed Risa, whose love, support and encouragement made this academic achievement possible.

Table of Contents

List of Tables

List of Figures

List of Symbols

α	Hole angle from vertical
γ_{mud}	Mud specific gravity
$\gamma_{fracture}$	Fracture pressure specific gravity
γ_{pore}	Pore pressure specific gravity
γ_{water}	Fresh water specific gravity
τ_B	Bearing constant, hr
τ_H	Formation abrasiveness constant, hr
$\Delta P_{S\_c\_surface}$	Surface casing collapse load at the surface, psi
$\Delta P_{S\_c\_shoe}$	Surface casing collapse load at the shoe, psi
$\Delta P_{S\_b\_surface}$	Surface casing burst load at the surface, psi
$\Delta P_{S\_b\_shoe}$	Surface casing burst load at the shoe, psi
$\Delta P_{I\_c\_surface}$	Intermediate casing collapse load at the surface, psi
$\Delta P_{I\_c\_shoe}$	Intermediate casing collapse load at the shoe, psi
$\Delta P_{I\_b\_surface}$	Intermediate casing burst load at the surface, psi
$\Delta P_{I\_b\_shoe}$	Intermediate casing burst load at the shoe, psi
$\Delta P_{P\_c\_surface}$	Production casing collapse load at the surface, psi
$\Delta P_{P\_c\_shoe}$	Production casing collapse load at the shoe, psi
$\Delta P_{P\_b\_surface}$	Production casing burst load at the surface, psi
$\Delta P_{P\_b\_shoe}$	Production casing burst load at the shoe, psi
a_1	Formation strength parameter
a_2	Exponent of the normal compaction trend
a_3	Undercompaction exponent
a_4	Pressure differential exponent
a_5	Bit weight exponent
a_6	Rotary speed exponent
a_7	Tooth wear exponent
a_8	Hydraulic exponent
b	Bearing constant
B	Fractional bearing wear
Bit_{wm}	Maximum weight on bit, 1,000lb
C_b	Cost of bit, dollars
C_i	Drilling cost per foot drilled, dollars/ft

C_r	Cost of rig, dollars/hr
d	Bit diameter, in
D	Well depth. Ft
g	Acceleration of gravity, ft/sec^2
g_p	Pore pressure gradient of the formation, lb/gal
h	Fractional tooth height worn away
h_1	Vertical depth of point 1, ft
h_2	Vertical depth of point 2, ft
H_1, H_2, H_3	Constant that depend on bit type
$HWDP$	Heavy weight drill pipe
k_b	Buoyancy factor
L_{dp}	Length of drill pipe, ft
L_c	Length of drill collars, ft
L_{hwdp}	Length of heavy weight drill pipe, ft
M	Molecular mass of gas, lb/mole
MOP	Margin of overpull, lb
NP	Neutral point design factor
P_p	Theoretical collapse pressure from tables
P_{ac}	Allowable collapse pressure, psi
P_c	Net collapse pressure, psi
$P_{fracture}$	Fracture pressure, psi
R	Ideal gas constant at standard gravity, lb-ft/mole- ºR
ROP	Rate of penetration
SF	Safety factor
T_{avg}	Average absolute temperature between two points, ºR
$T_{sruface}$	Temperature at the casing surface, ºF
T_{shoe}	Temperature at the casing shoe, ºF
TVD	True vertical depth, ft
t_b	Rotating time during bit run, hr
t_c	Non rotating time or connection time, hr
t_t	Trip time during bit run, hr
W_{dp}	Weight per foot of drill pipe, lb
W_{dc}	Weight per foot of drill collar, lb
W_f	Weight of drilling fluid, lb/cu ft

W_q	Weight of drilling fluid, lb/gal
W/d	Weight on bit per inch of bit diameter, 1,000lb/in
$(W/d)_{\max}$	Maximum weight on bit per inch of bit diameter, 1,000lb/in
$(W/d)_t$	Threshold bit weight, 1,000 lb/in
Z	Compressibility factor of gas

Chapter 1 Introduction

1.1. Motivation of the Research

Drilling is one of the most important and critical phases of petroleum industry. It is a process of making a hole in the ground for hydrocarbon to travel from the subsurface reservoir to the surface. The first step in well drilling is to plan the well. Well drilling planning has to follow a systematic approach. It involves several stages as shown in the Figure 1-1. Most of the well planning tasks at different stages depend on one or more other stages, such as casing selection requires input of casing setting depth and casing size; hence commonly, the planning stages are developed concurrently and interactively. Well planning is usually carried out by a team of experts. Very much specialized knowledge is required to achieve an economical and safe design. Although, many drilling software tools are available in the market to assist the planning team, but most of them are standalone to support certain stages. An integrated and comprehensive system is required for designing the well because data sharing and constraint management can be carried out in a coherent manner.

On the other hand, the success of a drilling work is very much dependent on design of the drill string. It is a well known fact that drill-string failure represents one of the major causes for "fishing" operations which may lead to millions of dollars in loss for the industry. So in order to reduce the risk of drill string failure, the design should be justified beforehand by simulation or finite element analysis. At present analytical models or finite element analysis for whole drill string are used to compute torque and drag. It will be very much useful if a three-dimensional (3D) engineering design model (including geometric 3D information) of drill string can be used in fluid flow simulation and load finite element analysis (Menand et al. 2006) to predict the behavior of the well. Austin (1993) stated that, 3D model of well schematic provides closer links between geoscientists and reservoir engineers while promoting an integration as well as interaction of the two. Therefore a 3D engineering design model of the drill string is required. However, it is cumbersome to develop the repetitive 3D models for each section of the well in order to perform such analyses.

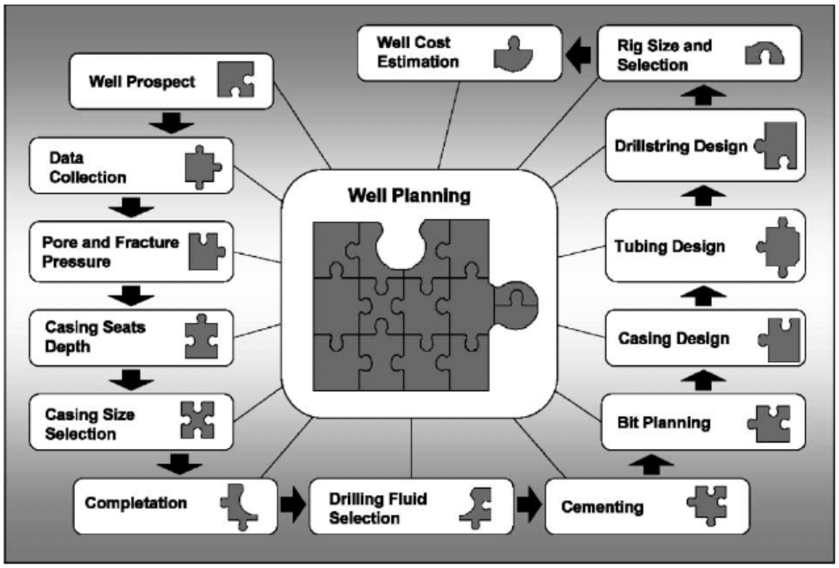


Figure 1-1: Well drilling planning stages (Morooka et al. 2001)

In view of the above discussion the aim of this research work is to develop an integrated well drilling system as well as a prototype computer program that integrates casing design and drill string design on top of a CAD system. The design output of these two stages can be used to generate 3D CAD models of well schematic and drill string. The system currently consists of three modules, (i) Casing Design, (ii) Drill String Design, and (iii) Operational Parameters. The prototype named as “DrillSoft” is implemented by using Microsoft Visual Basic application in Excel for user interfaces, engineering rules and calculations, and Siemens NX6 for CAD modeling via NX Open C programming functions.

1.2. Objectives of the Research

The objectives of this research are summarized as follows:

- To develop a framework for integrated well planning
- To explore the potential application of knowledge driven CAD in drilling
- To develop a smart knowledge driven CAD software prototype for well drilling

1.3. Organization of the Thesis

The thesis is organized as the follows. In Chapter two, the literature review of the application of knowledge based tools in drilling industry is reviewed, then current practice of casing design, drill string design, knowledge driven CAD, feature-based design and drilling optimization are presented. In Chapter three, the background of the research work is discussed and proposed approach is presented. In Chapter four, the background casing design is discussed and then proposed smart casing design model is described. In Chapter five, knowledge driven drill string design with generative approach is presented and the steps, i.e. conceptual design, configuration and specification generator, assembly modeling, component creation, array of components and 3D model realization, are illustrated in detail. In Chapter six, the proposed feature-based generative CAD modeling approach is discussed. In Chapter seven, Bourgoyne and Young's ROP model (1974) is presented and the proposed module is discussed. In Chapter eight, the results predicted by the newly developed "DrillSoft" system are compared with some published cases. In Chapter nine, the conclusions and future work are summarized. The engineering formulas, equations, and calculations are included in Appendix A while Appendix B shows the programming code written for this project.

Chapter 2 Literature Review

2.1. Introduction

This chapter will provide an insight of the current knowledge based and expert tools used in drilling industry. The casing design practice will be reviewed and the need of comprehensive casing design system will be discussed. The factors considered in drill string design and the selection parameters of drill pipe will be reviewed. One of the objectives of this research work is to explore the application of knowledge driven CAD in petroleum industry. So, current practice of knowledge driven CAD in different application areas will be analyzed. As in this research work, knowledge driven CAD system is implemented through feature-based modeling, a complete literature survey on feature-based CAD modeling will be presented. Finally the optimization of drilling parameters and drilling Rate of Penetration model will be discussed.

2.2. Knowledge Based Tools Used in the Drilling Industry

Knowledge based engineering is the process of capturing and structuring reusable knowledge bases to create and enhance solutions for a product during its entire life cycle (Chu et al. 2006). This knowledge base can exist in many forms such as spreadsheets, hand books, engineering formulas, drawing and documents. Drilling industry uses knowledge base and expert system from late eighties. Hayes-Roth (1987) has identified that the expert systems will play a dramatic role in the success of the outstanding performers in the petroleum industry. Mabile and Hamelin (1989) developed an expert system that helps in formation recognition. Martinez (1992) constructed a directional drilling expert system for the use of advisory tool which recommends changes in the Bottom Hole Assembly (BHA). An Expert Slurry-Design System (ESDS) was proposed by Kulakofsky et al.

(1993) to guide the selection process of cement slurry. Chiu and Caudel (1993) implemented an expert system for advising engineers of the proper base fluids and additives to be selected for a given set of well condition. Fear et al. (1994) created an expert system for drill bit selection; that system used a knowledge base of bit selection rules to produce a generic description of the most suitable bit for a particular set of drilling and geologic conditions. Their approach had several limitations like, the bit selection cannot demonstrate best use of past experience, and relies too heavily on data that is conveniently available rather than being fit for purpose. Shokouhi et al. (2009) proposed a case based reasoning system to integrate real time data with past experience to reduce operational problem. Al-Yami et al. (2010) created an expert system for optimal design of cement slurries. Their model can guide drilling engineers to formulate effective cement slurries for the entire well sections.

2.3. Knowledge Based Tools for Casing Design

One critical application of knowledge based and expert systems in the drilling industry is the design of casing strings as identified by Heinze (1993), he constructed an expert system to design the casing and hydraulic program of a drilling well. Jellison and Klementich (1990) proposed a rule based expert system for casing design but casing setting depth was not included in there model. Wojtanowicz and Maidla (1987) addressed the need of optimization in casing design and they proposed an optimization program for minimum cost casing design. But their method was not able to handle complex load condition. Roque et al. (1994) has developed optimized methodology and algorithm to minimize cost of combined casing design for complex loading condition. But they separated the load calculation from optimization model as a result the efficiency of the overall design process was sacrificed. In addition to that, their system required many hours to complete a single string design.

This problem has been solved by Halal et al. (1996); they proposed an efficient minimum cost casing design technique which employed a recursive branch-andbound search method together with a streamlined load generator for complex loading conditions. Their technique efficiently designed the casing string with a very small time. However, in their system, casing setting depth determination was not included.

Rabia (1988) pointed out that, no company has unlimited access to all grades and types of available casings. He argued that cost calculations come into play after the grades and weights are selected. Akpan (2005) created a computer program for selecting casing design using a graphical method, but his program does not automatically select the casing, instead each casing has to be chosen by the user and fed to the system manually for evaluation.

2.4. Common Practice of Drill String Design

There are many aspects need to be considered in drill string design, such as bottom hole assembly (BHA) and drilling pipe assembly. This work looks into drilling pipe design only due to the limitation of time and BHA is not part of the scope of this thesis. The common practice of drill string design is reviewed and summarized in this section. The success of drilling greatly depends on drill string design. Special care should be taken during the design of drill string. Many factors contributed in design decision, such as maximum expected load, accumulated fatigue, buckling, hydraulics, equipment availability (Cunha 2002). An optimum design can be achieved by considering all these factors.

As most of the part of the drill string is composed of drill pipe, so a selection process of drill pipe is required that can achieve the target depth successfully and safely. Drill pipe size, weight and grade should be selected based on three parameters – strength, size and cost (Kessler and Smith 2001). These three parameters are described in the following sections on the light of published literature.

2.4.1. Drill Pipe Strength

The drill pipe must be strong enough to handle the service loads during all phases of the drilling program. Bednarz (2004) pointed out that tensile force acting on the drill string is smaller during drilling than expected from the measured depth. This is because the string leans against the wall of the bore-hole, especially at a great deviation angle. He concluded that the maximum tensile force is a boundary load in vertical bore-holes, whereas in horizontal bore-holes it is the torsional strength.

The structural requirement of drill pipe should be defined by the torsional, tensile and buckling load that the pipe will experience during the service life (Mehra 1997). However, internal and external pressure capabilities are also factors that must be considered. Except for tension and buckling, these loads are often applied simultaneously.

Hill et al. (1993) describes that the operating torsion limit for a rotary shouldered connection is generally considered to be its makeup torque; however, with externally applied tension, it may be the tensile load capacity of the pin neck after taking makeup into consideration.

Tool joints are torsionally the weakest part of most drill strings (Hill et al. 1993). The pipe and tool joints must have the torsional strength needed to rotate the string when drilling and when coming out of the hole and it must have the buckling strength to transfer weight from the build zone to the bit (Mehra 1997).

So the strength of each member must be evaluated in terms of the forces and loads it will encounter under static, dynamic and fatigue conditions.

2.4.2. Drill Pipe Size

During size considerations in drill string design, the following points should be considered.

2.4.2.1. The Size of the Tool Joint in Relation to the Diameter of the Hole

Fishability is the main concern here. This consideration favors a tool joint with the smallest possible outside diameter.

2.4.2.2. The Size of the Pipe in Relation to the Diameter of the Hole

This relationship has several hydraulic consequences and most dominant are annular velocity (AV) and equivalent circulation density (ECD) (Kessler and Smith 2001). The larger the pipe size in relation to the hole, the lower the pump flow needed to attain the annular velocity for cuttings transport. However, increasing pipe size in relation to the hole size will increase the ECD, which in turn leads to pump pressure increase. AV and ECD must be optimized when selecting pipe size. If annular velocity is more important, the larger pipe size can be selected. Cunha (2002) suggested using of large diameter drill pipes because he argues that it may minimize hydraulic problems since it will imply in less friction loss inside the string and a more constrained annular.

A non-hydraulic consequence of the relationship between pipe size and hole diameter is the pipe’s buckling strength. Larger pipe contains higher buckling load. Cunha (2002) investigated the influence of torque and concluded that, torque can cause significant reduction of the buckling resistance on drill pipes with small diameter. On the other hand, for bigger pipes, that are the ones most used in extended reach wells, torque will have little influence on the buckling resistance. This consideration favors the largest possible pipe size for a given hole.

2.4.2.3. The Size of the Tool Joint in Relation to the Pipe

The elevator hoist capacity is dependent on the contact area between the elevator and the tool joint elevator shoulder, which is dependent on the difference between the tool joint OD and the pipe OD adjacent to the tool joint (Kessler and Smith

2001). The smaller the tool joint OD relative to the pipe OD, the lower the elevator hoist capacity.

2.4.3. Drill Pipe Cost

Cost is based on the benefits received by paying more for the pipe. These benefits fall into two categories (Kessler and Smith 2001):

- Increased performance such as high torque tool joints, high strength pipe body material and plastic coating for improved hydraulic efficiency
- Increased longevity like certain types of hard banding, plastic coating, and larger OD tool joints.

In addition to these factors, drill string components should be inspected before use and a method for qualifying the drill string is suggested by Hill et al. (1993). The drill string design prevents drill string failure and optimized the drilling operations to save rig time and lower bottom line cost.

The reported research work only considered vertical well and tensile load is the main determinant factor for strength calculation in this type of well as defined by American Petroleum Institute standard (API RP7G, 1998). Drill pipe size is considered as one of the inputs. So, the conceptual design of drill string is carried out by considering the cost and strength factors.

2.5. Knowledge Based CAD

The human expertise and knowledge are scarce and a need of knowledge embodiment within the geometric model is obvious (Kasravi 1994). However, most 3D CAD software offers only simple geometric modeling function and they fail to provide users with sufficient design knowledge (Lin et al. 2008). Therefore, the design of automatic, knowledge-based, and intelligent systems has been an active research topic for a long time (Lin et al. 2008). Zha et al. (2001) developed a knowledge-based expert design system for assembly oriented design. Koo and

Han (1998) constructed an expert paper feeding mechanisms system, where the physical part of the paper feeding mechanisms were represented as objects, and the design knowledge and design constraints are represented by rules and methods without the interface to CAD. Myung and Han (2001) proposed a design expert system to redesign assemblies of machine tool in a CAD environment. Roh and Lee (2006) created a hull structural modeling system for ship design, which was developed using C++ and build on top of 3D CAD software. Transferring KBE intelligence to CAD system is challenging because there is no mechanism to enable such information flow as identified by Ma et al. (2007). As introduced in (Kasravi 1994), parametric engineering uses the design requirements as the input data, and the output data consists of the parameters of the key features of the constituent components. Chu et al. (2006) constructed a computer aided parametric design system for 3D tire mold production in CATIA and CAA. Lee et al. (1999) developed a parametric computer-aided tool design system for cold forging using AutoLISP. Commonly, most researchers used parametric part templates to generate new 3D designs and changes are realized by setting values to the driving parameters (Siddique and Yanjiang 2002 ; Ma et al. 2003). Ma et al. (2003) considered the topological and configuration changes of parts.

2.6. Feature-based CAD Modeling

Solid modeller is the core element of CAD systems. By using solid modeller one can define the geometry of a product. A complete product definition is required to automate different CA'x application. But contemporary solid modeller is not capable of doing so due to two major deficiencies as identified by Shah and Rogers (1988) :

- Incomplete product definition and
- Low-level product definition.

Batanov and Lekova (1993) have identified three possible reasons that limit the integration of CA'x. These are:

- Lack of data standardization
- Functional difference and incompleteness of the data used for different CIM subsystem
- Lack of integration between data as a formal basis of given process in CIM and corresponding knowledge, necessary for realization of that process

In addition to that, Ma et al. (2007) identified a gap between knowledge based system and CAD. They argued that the current CAD system is not able to communicate with the knowledge based system and vice versa.

To eliminate these deficiencies in solid modeller or conventional CAD system, the concept of features came forward. As pointed out by Ma et al. (2007), feature-based design can be used as a bridge between CAD and KBE system. Shah and Rogers (1988) demonstrate three fundamental approaches to associate feature with solid modeller. These are: human assisted feature recognition, feature recognition and extraction and feature-based modeling. Among these three approaches, feature-based modeling is most popular and widely accepted in the literature.

The definition of features depends on the type of product, application and level of abstraction (Shah and Rogers 1988). No well established generic definition of features is observed till date. Different definitions based on these three factors appear in the literature. Most of the earlier attempts to define feature were related to geometry (form features) and machining.

Van't Erve (1988) defined feature as, "a distinctive or characteristic part of a work piece, defining a geometrical shape, which is either specific for a machining process or can be used for fixturing and/or measuring purposes"

Shah (1989) sorted out four essential requirements of features, these are:

- Physical constituent of a part (component).
- Mappable to a generic shape.
- Have engineering significance.

- Have predictable properties.

Cunningham and Dixon (1988) defined feature as any geometric form or entity that is used in reasoning in one or more design or manufacturing activities (i.e., fit, function, manufacturability evaluation, analysis interfacing, tool and die design, inspectability, serviceability etc).

According to Shah (1991) Features encapsulate the engineering significance of portions of the geometry of a part or assembly, and, as such, are important in product design, product definition, and reasoning, for a variety of applications.

In some literature, object-oriented approach has been implemented in feature-based modeling due to two major advantages it provides – data abstraction and inheritance. Though feature-based modeling uses the data abstraction concept from the beginning but inheritance was not present with the feature-based system. In object oriented approach feature is treated as object. According to Salomons et al. (1993), “features can be treated as design objects, belonging to a general class, which inherits properties of other classes”. Shah (1991) defined object as a cluster of knowledge of generic feature and according to his perspective this knowledge may be in the form of parameters, rules, procedures etc. Batanov and Lekova (1993) also defined features, based on object-oriented concept – ‘feature’ as a set of knowledge concerning an object (part or product) description suitable for different CIM processes. So, according to these definitions, feature contains all the information or knowledge required to integrate different applications programs.

Cunha and Dias (2002) defined feature as – Composition of design objects, which encapsulate own attributes and methods of each design phase, called phase’s design signature. According to Cunha and Dias (2002),

$$\mathit{Object} \equiv \mathit{Feature} \Leftrightarrow (\mathit{Geometry}, \mathit{Topology}) \cup \mathit{Semantic} \qquad \text{Eq. (2.1)}$$

In Eq. (2.1), the geometry and topology represent the physical model portion, which is independent, exact and quantitative, whereas the semantics represents the abstract model portion, which is context-dependent, subjective and qualitative.

Feature is also considered as a set of information. Shah and Rogers (1988) defined a feature as "a set of information related to a part's description". Ovtcharova and Jasnoch (1994) defined feature as – feature have been identified in the engineering community as meaningful abstractions with which human’s reason about products and processes. Huifen et al. (2003) stated that, Feature technology is the kernel technology of CAD/CAPP/CAM integration they further added that feature is the medium for transmitting information among CAD, CAPP and CAM systems. So in general, feature is a set of information related to the type of product, application and level of abstraction.

Several approaches towards feature-based design have been found in the literature. Shah and Rogers (1988) has demonstrated the functional requirements of Feature-based Modeling System (FBMS), but his demonstration is limited to only three different classes of features namely – form features, material features and precision features. However, an ideal feature-based system should be highly flexible in feature definition so that a designer can define features in any form, at any level and in any combination, as appropriate to their needs (Shah and Rogers 1988). Ovtcharova and Jasnoch (1994) argued that features are not product parts themselves, but rather, distinctive characteristics (properties) of these parts which exist only within the parts. Thus, feature-based design can be characterized as a method for designing product parts by defining and manipulating the properties of those parts (Ovtcharova et al. 1992).

Kim and O'Grady (1996) proposed domain independent representation formalism for feature-based design with the aims to make it possible to develop a feature-based design system for a specific design domain in a structured way. Kappert et al. (1993) noticed that, feature-based approach guides the designer in creating product models according to company rules. Schulte et al. (1993) introduces functional features, which are used as “vehicles” to introduce functional aspects into CAD systems and CAD techniques and the introduction of functional aspect in CAD system could open up a new range of functionalities for the CAD systems.

Chan and Nhieu (1993) incorporates an external user defined feature database with the CAD database to form the unified central database, which is used to capture the required information for downstream applications during the early design stage.

Ma and Tong (2003) argued that most of the features used on contemporary feature-based system are CAD application oriented and related to machining process or design geometry and their definitions are based on less flexible predefined parametric templates. They further added that a feature-based modeling system should be a special application of information modeling.

Shah and Rogers (1988) classified features to four different categories, namely – form feature, material features, precision features and technological features. Ovtcharova and Jasnoch (1994) classified features as generic and application features. According to their approach general features are not application specific rather defined by general properties of a product. Whereas application features are defined by assigning data on generic features and they are application specific.

There are ample of advantages of using feature-based system. This modeling system supports additional levels of information beyond those available in geometric modellers as pointed out by Shah (1991). He further added that the availability of high-level information makes the design environment more attractive, produces a richer definition of the product, and allows one to automate downstream application to a higher degree. The primary benefit of feature-based design systems is the ease of integration of conceptual modeling with downstream analyses and applications (Chan and Nhieu 1993). A feature-based representation of the part becomes necessary to automate the reasoning involved in the tasks in the product development cycle (Zha et al. 2001).

Feature-based generic modeling approach has been followed in this research work to integrate the engineering knowledge with the CAD system.

2.7. Drilling Optimization via Operational Parameters

The design of a well program must satisfy all the technical considerations. But only considering the technical aspect may not result an economical design. In order to be an optimum design the design should simultaneously satisfy the technical constraints and at the same time it should be the least cost possible from different alternatives. A comprehensive drilling optimization program was first applied in 1967 and it significantly reduced the drilling cost. After its first introduction, it becomes one of the most important research areas in drilling. Drilling optimizations are carried out by selecting the best combination of different drilling variables. Drilling variables are divided into two groups (Lummus 1970) – alterable and unalterable. The alterable drilling variables are: mud, hydraulics, bit type, weight on bit, rotary speed. The unalterable variables are: weather, location, depth, rig condition, etc.

Usually alterable drilling variables and few unalterable variables are used to optimize the drilling procedure. Several researchers have been developed algorithm, model and program to optimize the drilling performance by maximizing Rate of Penetration (ROP) and minimizing cost per foot. Galle and Woods (1963) investigated the effect of best constant bit weight and rotary speed for lowest cost. Reed (1972) constructed a variable weight-speed optimal drilling model which was being solved using Monte Carlo scheme for minimum cost per foot drilling. Bourgoyne and Young (1974) proposed a comprehensive drilling model to calculate formation pore pressure, optimum weight on bit (WOB), rotary speed, and jet bit hydraulics and also provided a multiple regression approach to determine the drilling coefficients in order to calibrate the drilling model with different fields. Maidla and Ohara (1991) tested a drilling model on offshore drilling data and compared the findings with Bourgoyne and Young’s model. They concluded that ROP for successive wellbores in the same area could be predicted based on the coefficients calculated from the past drilling data, resulting cost savings. Bjornsson et al. (2004) proposed a rule based bit selection expert system by employing Mechanical Specific Energy (MSE) concept. Their system efficiently increases the ROP, bit life and significantly reduce the drilling time. Dupriest and Koederitz (2005) effectively used mechanical Specific Energy (MSE) concept in evaluating drilling efficiency of bits in real time basis. Rashidi et al. (2008) used both Mechanical Specific Energy (MSE) and inverted rate of penetration models to develop a method for evaluating real time bit wear. That tool was useful to assist the field engineer in deciding when to pull the bit. Eren and Ozbayoglu (2010) constructed a model for real time optimization of drilling parameters during drilling operations.

Among all of these models, Bourgoyne and Young's model is adapted in this research for Operational Parameters determination because this model is one of the most complete mathematical drilling ROP model used in petroleum industry.

2.8. Summary

Although the drilling industry uses knowledge based tools for many years, an integrated system with knowledge driven CAD system is not common. Several casing design tools are available in the literature but a comprehensive solution is still desirable. Applications of knowledge driven CAD in different areas are reviewed and as a result it was identified that such advanced design tools can also be used in drilling. To achieve knowledge driven CAD system, feature-based design is the most common method for implementation, hence, this aspect has also been reviewed in details.

Chapter 3 The Proposed Method for Integrated Well Drilling Planning

3.1. Background

Well planning is the first step in drilling. It is the key to being able to safely and economically drill a usable hole for oil and gas production. Well planning requires many detailed studies evaluating every aspect that directly or indirectly influences the successful economic outcome of the project. Many stages are involved in the design process such as, logging program, casing program, mud program, cementing program, well control, hydraulic program, drill bit program, drill string program, drilling rig specification etc. Very much specialized knowledge in each of these stages is required to achieve an economical and safe design.

Commonly, the planning stages are developed concurrently and interactively by manual calculations or limited use of some software tools. However, manually undertaking the design process is mentally challenging, time-consuming and a lack of rule validation and data integration. An automated and integrated system is required.

On the other hand, knowledge driven CAD system has been a growing practice in the manufacturing industry. This approach improves the speed and effectiveness of the product development process, reduces the time taken to create new design, and automates the tedious and time consuming parts of design. It can capture the design knowledge and expertise, and can be applied to develop different engineering application models according to specific requirements.

Knowledge driven CAD system can also be applicable to the drilling industry, as there is a practical need to speed up the design process for drilling wells. Ideally, it can be expected that knowledge driven CAD approach can support the necessary automation of the design process of well planning, and at the same time produce the 3D engineering design models for different sections, such as casing and drill string. However, to the best of the candidate’s knowledge there is no such program that integrates the well planning with knowledge driven CAD system.

In this research work, the knowledge driven CAD system has been implemented with the help of feature-based design approach. Three different levels of features namely, assembly level features, component level features and form features are defined to realize the 3D CAD model of casing and drill string. Features are defined in such a way that they encapsulate the engineering significance of portions of the product geometry and might be useful for further analysis such as finite element analysis, simulation etc. Feature-based CAD modeling has been further discussed in more detail in chapter six.

3.2. Proposed Approach

The proposed approach integrates two important well drilling planning stages, i.e. casing design and drill string design, and linked the system with a knowledge driven CAD system to generate the parametric 3D CAD model of casing and drill string. These CAD models can be useful for finite element analysis, simulation and better visualization. In addition to these two stages an Operational Parameters module is also added with the system to determine the optimum weight on bit and rotary speed for minimum cost drilling. The proposed system has been shown in Figure 3-1. The following section briefly describes each component of the proposed system.

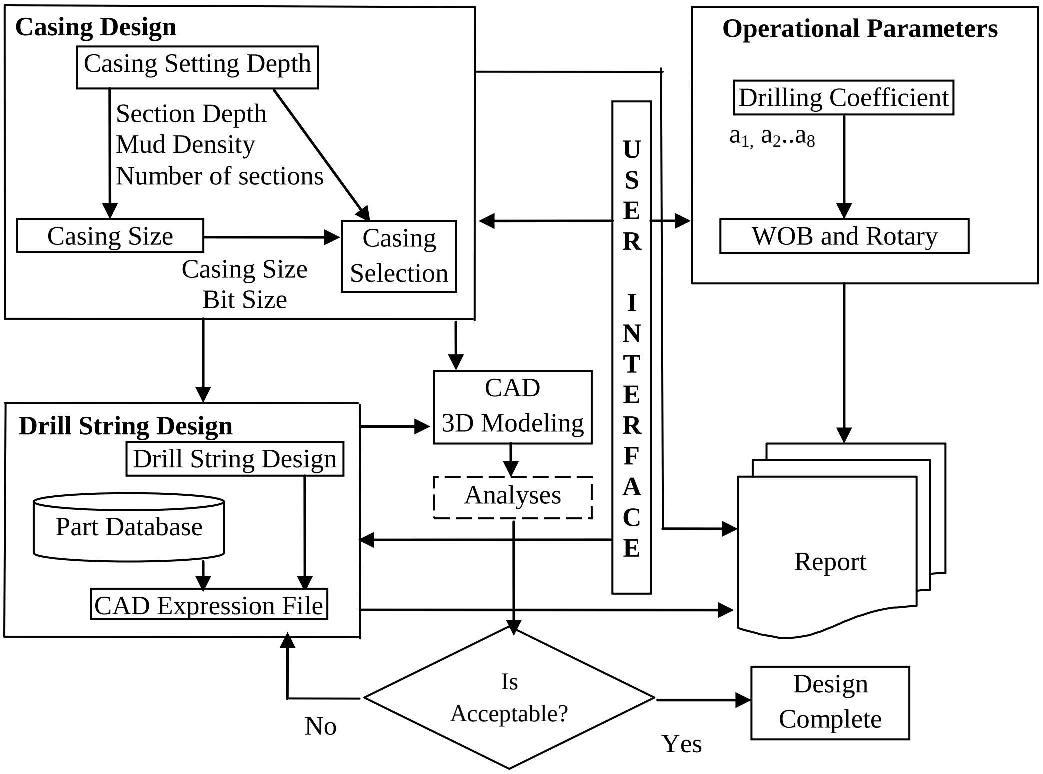


Figure 3-1: Proposed system

3.2.1. Casing Design Module

Casing design module calculates casing setting depth by using formation pore pressure and formation fracture pressure; determines the size of hole and casing of each section and generates a 3D model of different casing sections of the well. It then selects the optimum combination of casing string from the available inventory.

3.2.2. Drill String Design Module

In the drill string design module, the engineering design and 3D modeling processes are automated. Built-in rules and knowledge are used to develop the conceptual design and then the system automatically generates the assembly configuration based on the conceptual design and retrieves part specifications from the part database to generate the CAD parametric files.

3.2.3. Operational Parameters Module

Operational Parameters module works out drilling coefficients, the optimum weight on bit (WOB) and the drilling rotary speed (RPM). Drilling Coefficients are determined according to Bourgoyne and Young (1974) regression analysis procedure. Optimum WOB and RPM are determined for the minimum cost. This program also generates six different tables of economic performance as a function of weight on bit and rotary speed as an operational guide.

3.2.4. 3D CAD Modeling

This system produces the conceptual design by considering the built in rules and constraints; and the conceptual design is used to generate the parametric 3D model of drill string and casing. Unlike those efforts using CAD templates, which require predefined part libraries and are difficult to manage the data consistency related to various parametric constraints, this research work uses feature-based generative approach for 3D CAD modeling of casing and drill string. In generative approach, the part is created with parametric feature primitives in a programming environment; a predefined part template is not required for generating any model (Ma et al. 2003). This system takes input specifications and the application program creates the geometry of drill string assembly internally with the CAD NX functions. The advantages of using program instead of template files are:

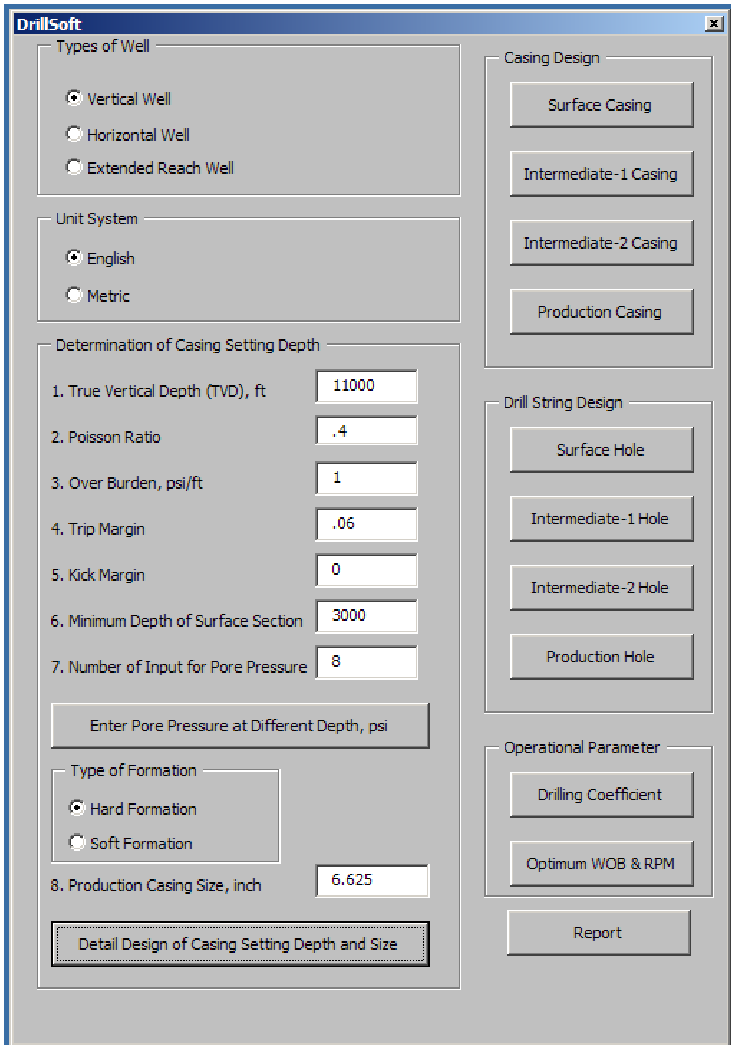
- Geometry and feature can be easily created and edited;
- Parameters can be created and manipulated in more controlled manner;
- Geometry analysis and part standardization can be easily achieved;
- Files can be managed more efficiently;
- Data access and family of parts creation are more convenient.

3.2.5. CAE Analysis

The proposed system can be integrated with a CAE analysis system but was not part of the thesis scope. CAE analysis might be useful to justify the conceptual design of the casing and drill string from the angle of loading evaluation, fluid flow pressure distribution and detailed selection of well structure elements. If the analysis does not show satisfactory results then design iterations and optimization should be carried out. This process continues until the acceptable design is achieved.

3.2.6. User Interface and Report Generation

User friendly interfaces have been created in order to integrate all the modules. The interfaces guide the users to develop the well plan with less effort and in an organized sequence. Figure 3-2 shows the main user interface of the prototype system. Type of the formation is an input selection by the user in order to calculate the hole and casing sizes (Byrom 2007). Currently, the User Interface (UI) is limited to a well configuration with two intermediate casings. The UIs are configurable by programming according to the algorithm requirement. More UIs are to be introduced in the following chapters. The proposed system also produces formal reports for design details.



DrillSoft

Types of Well

● Vertical Well

○ Horizontal Well

○ Extended Reach Well

Unit System

● English

○ Metric

Determination of Casing Setting Depth

1. True Vertical Depth (TVD), ft: 11000
2. Poisson Ratio: .4
3. Over Burden, psi/ft: 1
4. Trip Margin: .06
5. Kick Margin: 0
6. Minimum Depth of Surface Section: 3000
7. Number of Input for Pore Pressure: 8

Type of Formation

● Hard Formation

○ Soft Formation

8. Production Casing Size, inch: 6.625

Figure 3-2: Main user interface of DrillSoft

3.3. Information Flow

A systematic information flow model has been developed. The information flow model is shown in Figure 3-3. The system takes design requirement through a user interface and stored this information in a database.

There are separate knowledge bases for each module, which stores the engineering knowledge and rules. A method fires the appropriate rules based on the input during the design. Inference mechanism uses reasoning process to come up with a conceptual design and send it to the database for storing. Configuration and specification generator retrieves the part geometric specification from the data base, generates the assembly configuration based on the conceptual design and creates the parametric data file. CAD system uses this parametric file to generate the 3D model. This 3D model can be used for further analysis, if the design does not satisfy the requirements it then goes back to the design process for reiteration.

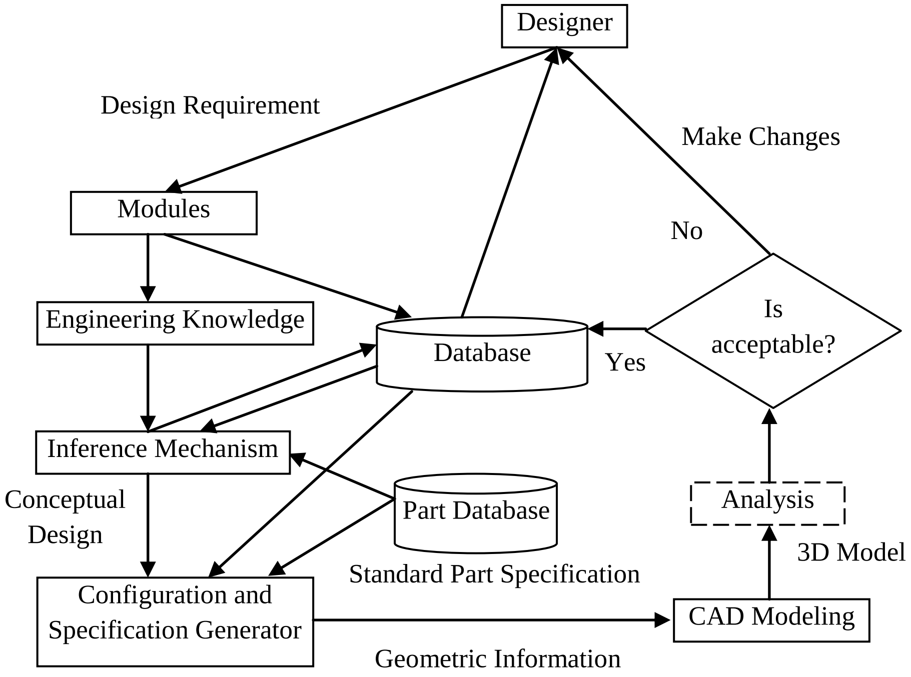


Figure 3-3: Flow of information

3.4. Advantages of the Proposed System

The advantages of the proposed system are given as follows –

- The proposed system uses integrated approach as a result; the flow of information is very smooth. Due to the information sharing among the modules less input is required to carry out the design.
- As the design process becomes automated, the cycle time can be reduced significantly.
- The proposed system is coupled with engineering knowledge and parametric modeling, design parameters required for 3D modeling can be created automatically.

- Associative relationships among the components have been created, that makes the model adaptable with changes.
- The proposed system uses generative approach; it eliminates the need of part template for creating the part library. This approach is very efficient for file management and part creation.
- Eliminates the need of repetitive CAD model generation.
- Automatic generation of bill of materials.
- Provides a visual environment.

3.5. Summary

This chapter presents the current practice of well drilling. It is identified that there is a need to automate the well drilling design process. An integrated knowledge driven CAD system for well drilling is proposed and the components are described briefly. Finally the advantages of the proposed system are presented.

Chapter 4 Smart Casing Design

4.1. Introduction

This chapter will first briefly describe the casing design process and then the comprehensive casing design approach will be presented.

4.2. Overview of Casing Design

A casing is a collection of steel tubes that becomes permanent part of an oil or gas well. Casing is required at certain stage during drilling to protect the well. Casing serves many important functions during the life of a well. The major functions of the casing are as follows (Byrom 2007; Mattiello 1992):

- Maintaining the structural integrity of the bore hole.
- Serving high strength flow conduit to surface for both drilling and production fluids.
- Providing support for wellhead equipment and blowout preventers.
- Preventing contamination from near fresh water zones.
- Facilitating of running wireline equipment for testing.
- Allowing isolated communication with selectively perforated formation(s) of interest.

A well consists of several sections of different diameters hole and a string of casing is run after each section of hole has been drilled. Such as a typical well may contain- conductor casing, surface casing, intermediate casing, production casing. The design, material of construction and purpose are different for each of these different sections. A well configuration with four casing sections has been shown in Figure 4-1.

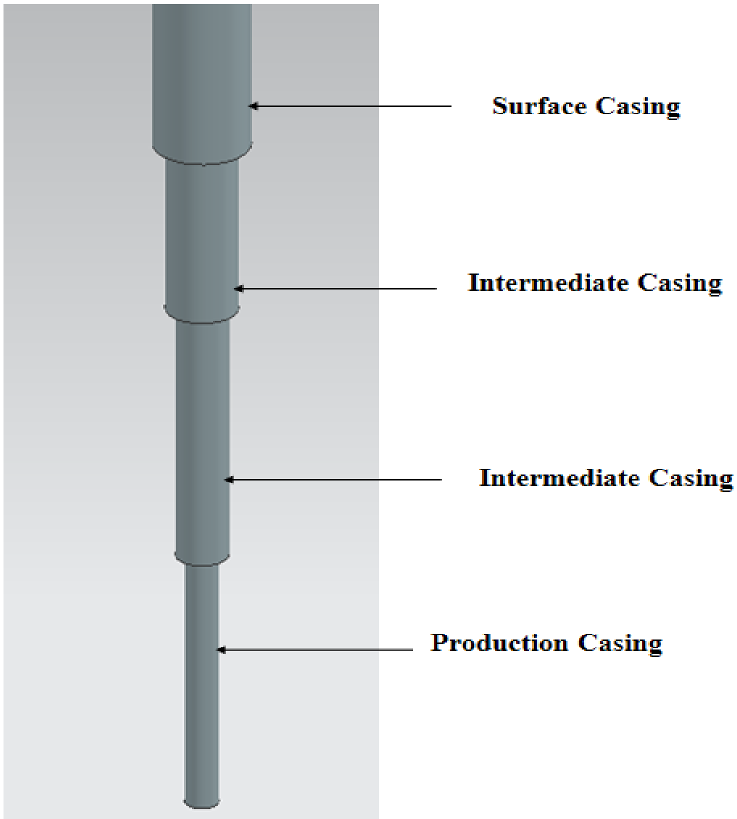


Figure 4-1: Different casing sections

Casing design is one of the most important tasks in well drilling plan. Casing program of a well represents a significant amount of overall well cost, almost 20% of it (Roque et al. 1994). So a small reduction in cost will result a huge saving. But at the same time, the casing program should satisfy all the constraint and loading requirements. Casing design can be divided into two phases – preliminary design and detailed design (Halal et al. 1996). In the preliminary design casing setting depth, mud density, casing size and hole size are determined. Selection of less expensive casing from the available inventory that satisfies the entire design requirement is considered in detailed design. The whole casing design process has been shown in Figure 4-2. The following sections will describe the design procedures.

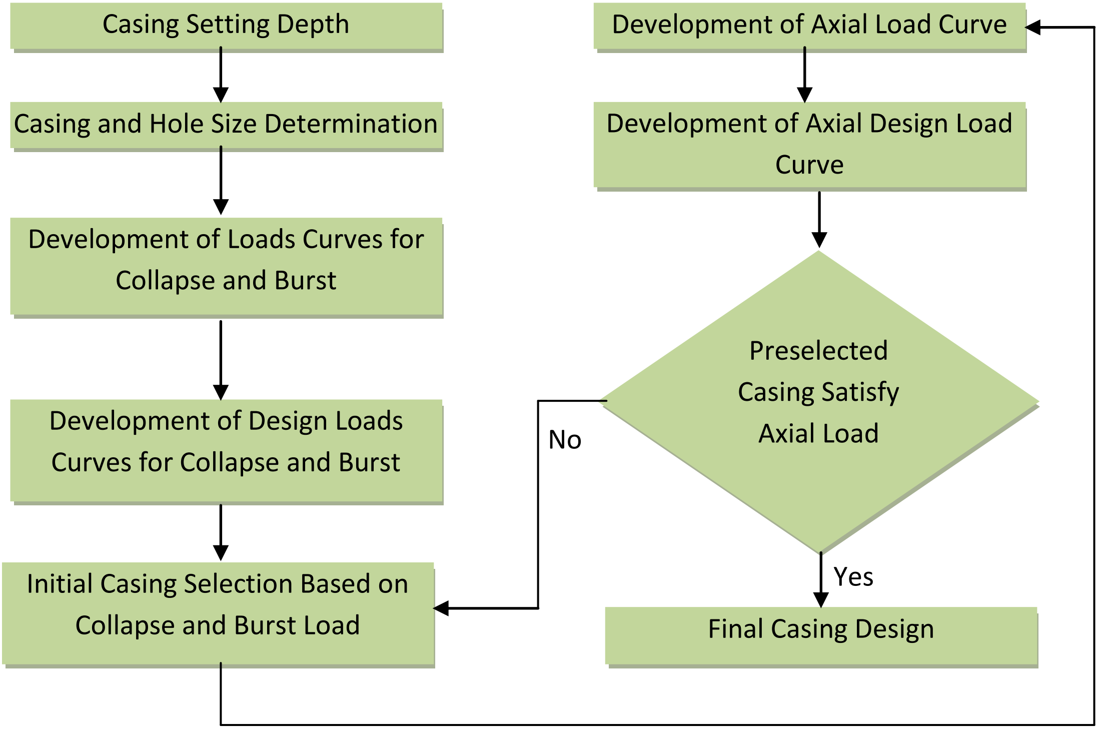

Casing Setting Depth
Casing and Hole Size Determination
Development of Loads Curves for Collapse and Burst
Development of Design Loads Curves for Collapse and Burst
Initial Casing Selection Based on Collapse and Burst Load
Development of Axial Load Curve
Development of Axial Design Load Curve
Preselected Casing Satisfy Axial Load
No
Yes
Final Casing Design

Figure 4-2: Casing design steps

4.2.1. Casing Setting Depth

Casing setting depth determination is the most critical steps in casing design. Several parameters are considered during casing setting depth, these are - pore pressure, fracture pressure, experience in an area, bore hole stability problem, corrosive zones, environmental consideration, regulations and company policy. Among these, pore pressure and fracture pressure are most widely used to determine the setting depth. These two parameters are also considered in this study as main determinant factor for casing setting depth calculation. Formation pore pressure and fracture pressure are described in the following sections.

4.2.1.1. Formation Pore Pressure

Formation pore pressure is defined as the pressure exerted by the formation fluids on the walls of the rock pores (Rabia 1985). The rocks inside the earth contain pore spaces and these spaces are filled with fluids – either in the form of gas or liquids. Formation pore pressures are divided into two types, normal and abnormal formation pore pressure (Bourgoyne et al. 1991). When formation pressure is approximately equal to the theoretical hydrostatic pressure for a given depth then it classified as normal; but if the formation pressure is greater than the normal hydrostatic pressure for a concerned depth, then it is treated as abnormal (Bourgoyne et al. 1991).

Two different methods are used to determine the formation pressure (Rabia 1985). These are – geophysical method and logging method. Geophysical method helps to predict the formation pore pressure before the well is drilled and logging method is applicable during drilling of the well. However, in this study pore pressure is not estimated, it is considered as an input.

4.2.1.2. Formation Fracture Pressure

The formation fracture pressure is defined as the pressure necessary to overcome the formation pressure and strength of the rock matrix. It is the pressure at which a formation matrix opens to admit whole liquid through an actual crack in the matrix of the rock as opposed to invasion through the natural porosity of the rock (Byrom 2007). Two approaches are used to determine the fracture pressure, these are – direct and indirect method (Rabia 1985). The pressure required to fracture the rock and the pressure required to propagate the resulting fracture is determined by direct method and stress analysis is used to predict the fracture gradient in indirect method. There are six different indirect methods available in the literature (Bourgoyne et al. 1991). These are - Hubber and Willis method, Matthews and Kelly method, Eaton method, Pennebaker correlation, Christman equation and MacPherson and Berry correlation. Eaton method has been chosen to predict the fracture pressure gradient calculation in this research work. Eaton (1969) argued that Poisson's ratio for a given field should be fairly constant and can be determined from the data obtained from the nearby well. However, in this program Poisson's ratio is also considered as an input. Eaton method is used to determine the fracture gradient of the prospective well:

$$FG= (v/1-v)(\sigma_v - P_f\,)/D + P_f\,/D \qquad \text{Eq. (4-1)}$$

Here, *FG*= Fracture gradient, psi/ft; *D*= Depth, ft; = Poisson's ratio; σ_v= Over burden, psi/ft; P_f= Formation pore pressure, psi/ft. As Eaton method is limited to certain geographical area, an option has been given to use fracture pressure as input.

4.2.2. Casing Size Determination

A well consists of several sections; it is an important task to determine the bore hole and casing size in each sections. The following points should be considered during the casing size determination (Byrom 2007):

- The bore hole must be large enough for the casing to pass freely with little chance of getting stuck
- There should be enough clearance around the casing to allow for a good cement job
- The bigger the bore hole, the more costly it is to drill

The casing and hole sizes determination usually carried out by following common practice in concerned geographical area (Byrom 2007). There are many charts and tables available in the literature, some of these are good for some areas but greatly lacking for other areas. However in order to develop an automated program for casing and hole size selection, two standard charts for hard and unconsolidated formation have been adopted in this research work (Byrom 2007).

4.2.3. Casing Selection

Devereux (1998) identified two important aspects of casing design; casing should be designed to resist the forces or conditions that are imposed on it during drilling and it should sustain throughout the life of the well to meet the well objectives without requiring a workover. Three basic loads are considered –collapse, burst and axial. Brief descriptions of these loads are given as follows:

4.2.3.1. Collapse Load

Collapse loads are the differential pressure loads between the external and internal pressure of the tube or casing (Byrom 2007). The primary collapse loads are generated by the hydrostatic head of the fluids column outside of the casing (Rahman and Chilingarian 1995). These fluids are usually drilling fluids or cement slurry. During the design, the worst case scenario is considered. Such as, when collapse load is calculated, the minimum internal pressure and the maximum external pressure are considered. The collapse loads in different casing sections considered in this research work are provided in Appendix A.

4.2.3.2. Burst Load

Burst loads are defined by the difference between the internal and external pressures in which the internal pressure exceeds the external pressure tending to cause casing to rapture or burst (Byrom 2007). Burst loads are normally caused by the mud hydrostatic pressure inside the casing. Fluids on the outside of the casing supply a hydrostatic pressure that helps resist pipe burst.

4.2.3.3. Axial Load

The axial load in a casing string at any point due to gravity or weight is a function of the buoyancy of the drilling fluid and the inclination of the well bore (Byrom 2007). Axial load is highest at the top of the string; it decreases with depth toward the bottom of the string.

4.2.3.4. Design Safety Factor

Many uncertainties exist during the service life of a well. As casing is the permanent part of well, it may subject to forces that are not foreseen during the design. Besides, casing strength may also deteriorate with time due to wear, erosion and corrosion. So to produce a safe design, proper safety factor should be considered. However, policies on safety factor can be quite confusing and widely varying as identified by Devereux (1998). Every company maintains their own set of standard towards selecting the safety factor. In this module options have been provided for the users to select safety factor based on their requirement.

4.3. Proposed Approach for Casing Design

The proposed system provides a comprehensive model that combines both preliminary and detailed design, i.e. casing setting depth, casing size, casing selection. In addition to that, a 3D parametric CAD model is added with the system. The complete system is shown in Figure 4-3. A systematic approach is proposed to carry out the design task interactively. A specific sequence should be followed during the design. Because casing selection requires input from casing setting depth and casing size. So, casing setting depth and casing size determination should be carried out first followed by casing selection. A well consists of several casing sections; so, it is required to design the casing individually for each section. Finally the casing setting depth and casing size output can be used to generate the 3D CAD model of the well.

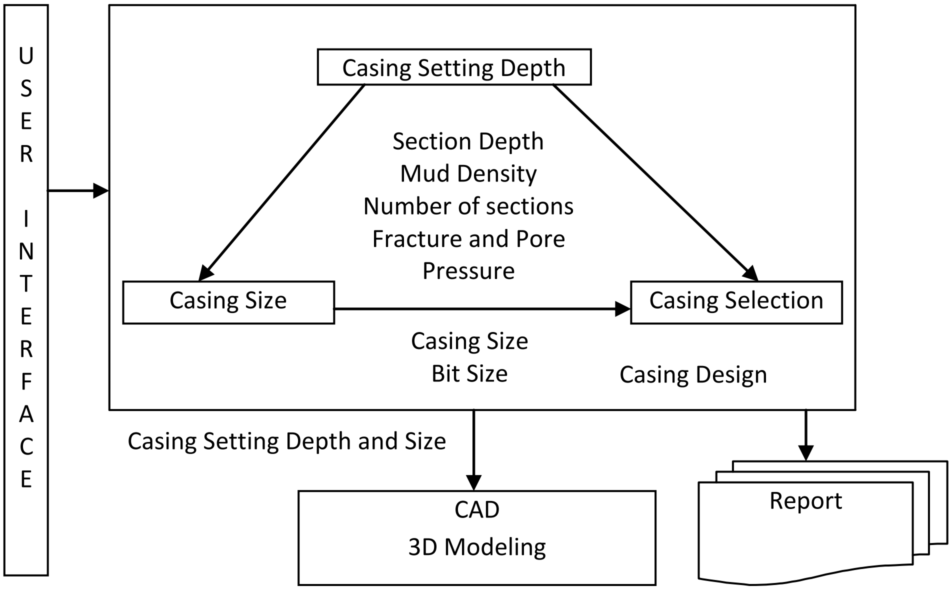


Figure 4-3: Casing design modules

4.3.1. Casing Setting Depth

The developed algorithm for casing setting depth is described in this section. The flow chart for casing setting depth and size determination has been shown in Figure 4-4. The program first takes input from the user through the user interface. The following inputs are required:

- Type of Well
- Unit system
- Number of pore pressure input
- Pore pressure at different depth
- Kick margin
- Trip margin and
- Minimum casing setting depth

The fracture pressure needs to be input since Eaton's method is not generalized one. This matter has been discussed in the subsection 4.2.1.2. Two safety factors should be considered during mud density determination:

- Mud density should be slightly higher than the formation pressure

- The density should be less than the fracture pressure

These two safety factors are known as trip margin and kick margin respectively. The mud densities are chosen to provide an acceptable trip margin above the anticipated formation pore pressure to allow for reduction in effective mud weight (Bourgoyne et al. 1991). Mud pressure may be decreased due to several reasons such as drop in mud level, swab effect, etc. Swab effect is produced during tripping of the pipe because when making a trip the pipe is pulled upward and due to this pulling action a negative pressure inside the hole is created, and that results the reduction of hydrostatic pressure. On the other hand when the drill string is returning back to the hole a positive surge pressure is produced. If the pressure is more than the fracture pressure of the well then the stability of well will be hampered. So a kick margin is subtracted from the design fracture gradient line. If no kick margin is provided, it is impossible to take a kick at the casing setting depth without causing hydrofracture and it is possible to have underground blowout (Bourgoyne et al. 1991).

The minimum casing setting depth is for surface casing. Besides pore and fracture pressures, surface casing depth is also affected by two more sets of factors, the depths of freshwater bearing zones, and legal regulations and requirements. The minimum casing setting depth is determined by considering these four groups of factors.

The system first estimates the fracture pressure and determines the MSG and FSG based on the input provided. MSG is the required mud density to drill the well. It can be calculated as the sum of pore pressure and trip margin, i.e. pore pressure+ trip margin. FSG can be defined as the difference between fracture pressure and kick margin, i.e. fracture pressure-kick margin. It then finds out the mud density at the true vertical depth. According to the theory this is the density of mud required to drill the well to the final depth. Here it is said production section of the well. The next step is to determine the depth at which the fracture gradient is equal to the mud specific gravity, in other words the depth at which the vertical line drawn from the mud density curve touches the fracture gradient curve (Figure

4-5). Once the depth of the next section has been known the program then determines the mud density required to drill in this depth. In this way the process continues until the mud density becomes smaller than the minimum fracture gradient or the depth become smaller than the minimum casing setting depth. The designed casing setting depth information is stored in a database and can be shared to other modules. The proposed method of casing setting depth has been described in more detail with case study in chapter eight, Section 8.2.

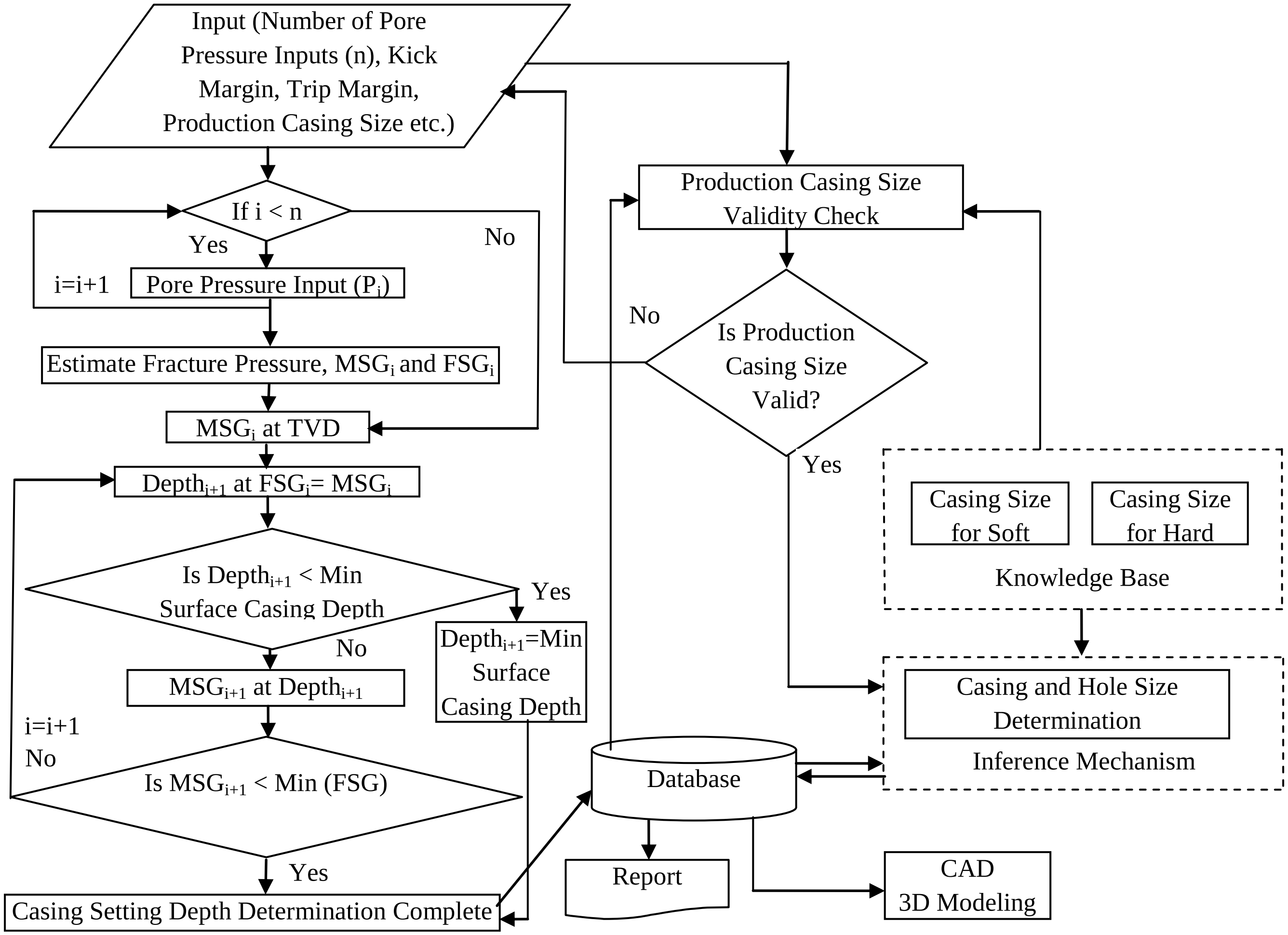

Input (Number of Pore Pressure Inputs (n), Kick Margin, Trip Margin, Production Casing Size etc.)
If i < n
Yes
No
i=i+1
Pore Pressure Input (P_i)
Estimate Fracture Pressure, MSG_i and FSG_i
MSG_i at TVD
$Depth_{i+1}$ at FSG_i= MSG_i
Is $Depth_{i+1}$ < Min Surface Casing Depth
Yes
No
$Depth_{i+1}$=Min Surface Casing Depth
MSG_{i+1} at $Depth_{i+1}$
i=i+1
No
Is MSG_{i+1} < Min (FSG)
Yes
Casing Setting Depth Determination Complete
Production Casing Size Validity Check
No
Is Production Casing Size Valid?
Yes
Casing Size for Soft
Casing Size for Hard
Knowledge Base
Casing and Hole Size Determination
Inference Mechanism
Database
Report
CAD
3D Modeling

Figure 4-4: Flow chart of casing setting depth and size

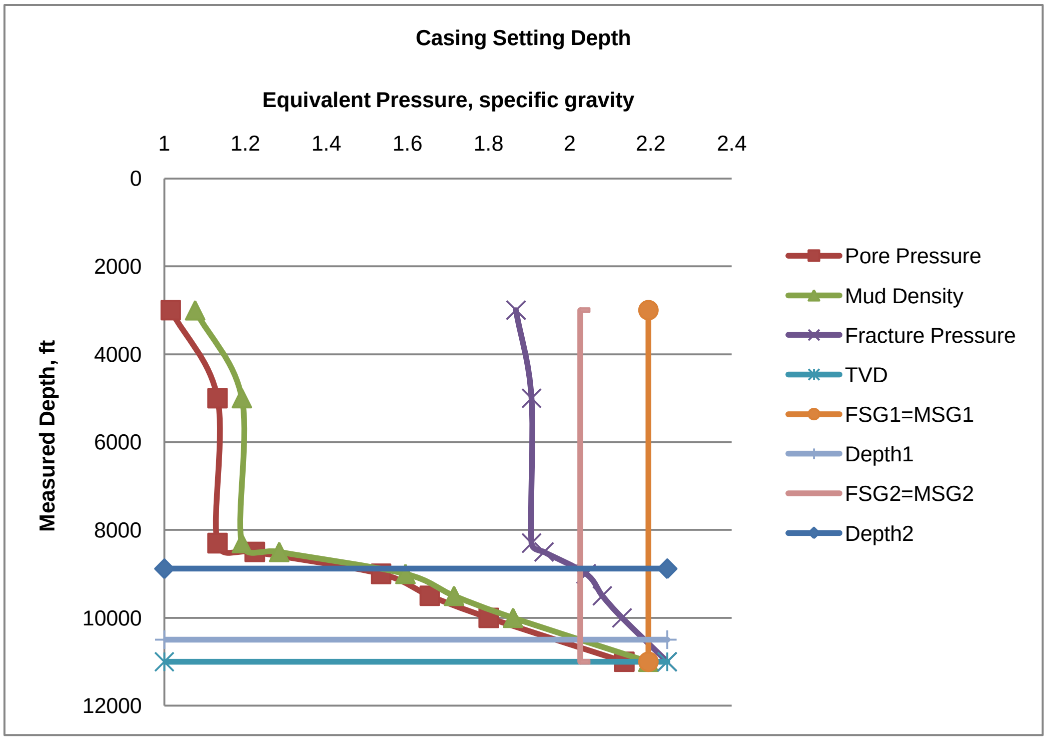


Figure 4-5: Casing setting depth

4.3.2. Casing Size

A prototype knowledge base has been created to determine the casing and hole sizes. The knowledge base contains different configurations of casing sizes for hard and unconsolidated formation. The casing and hole size configuration for hard and unconsolidated formation have been shown in Figure 4-6 and Figure 4-7. In this research work, the casing and hole sizes are selected by considering the formation types (chapter 3, Byrom 2007), number of casing sections and production casing size. The inference mechanism sorts out the hole and casing sizes for each section of the well based on the users input. Inference mechanism is used to derive the proper size of casing and hole by executing the reasoning process. Different combinations of casing sizes are possible. If the specific production casing size is not available in the system knowledge base then it recommends the sizes that are available in the knowledge base.

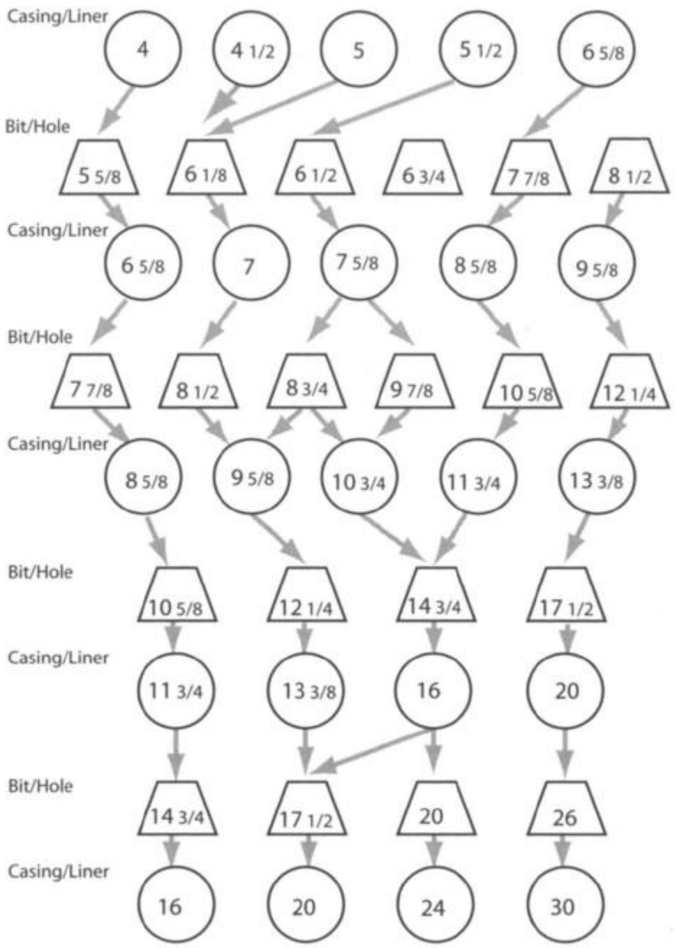


Figure 4-6: Casing and bit size selection for hard rock formation (Byrom 2007)

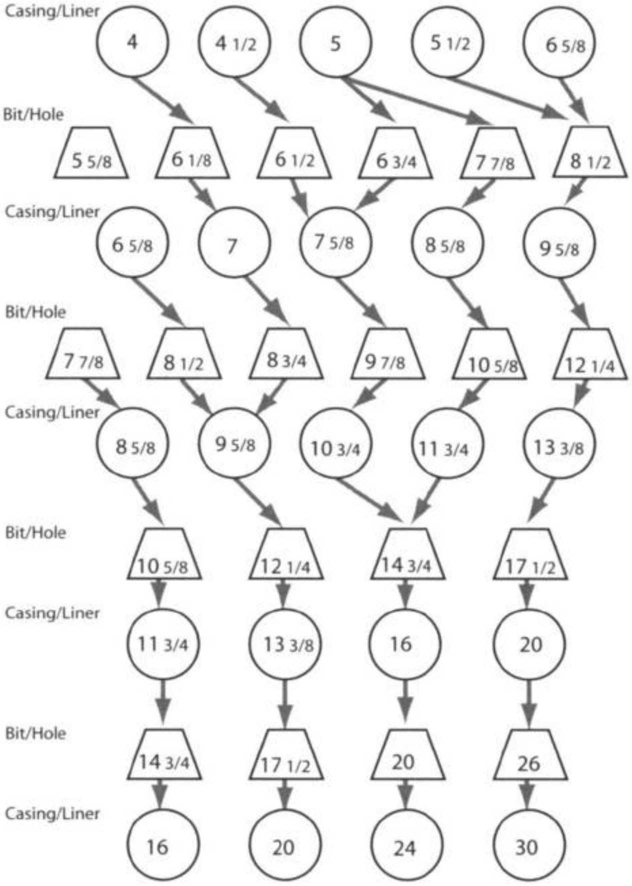


Figure 4-7: Casing and bit size selection for soft formation (Byrom 2007)

4.3.3. 3D CAD Model

A parametric 3D model of well structure has been developed, which takes the casing setting depth and casing size output to generate different casing sections of the well.

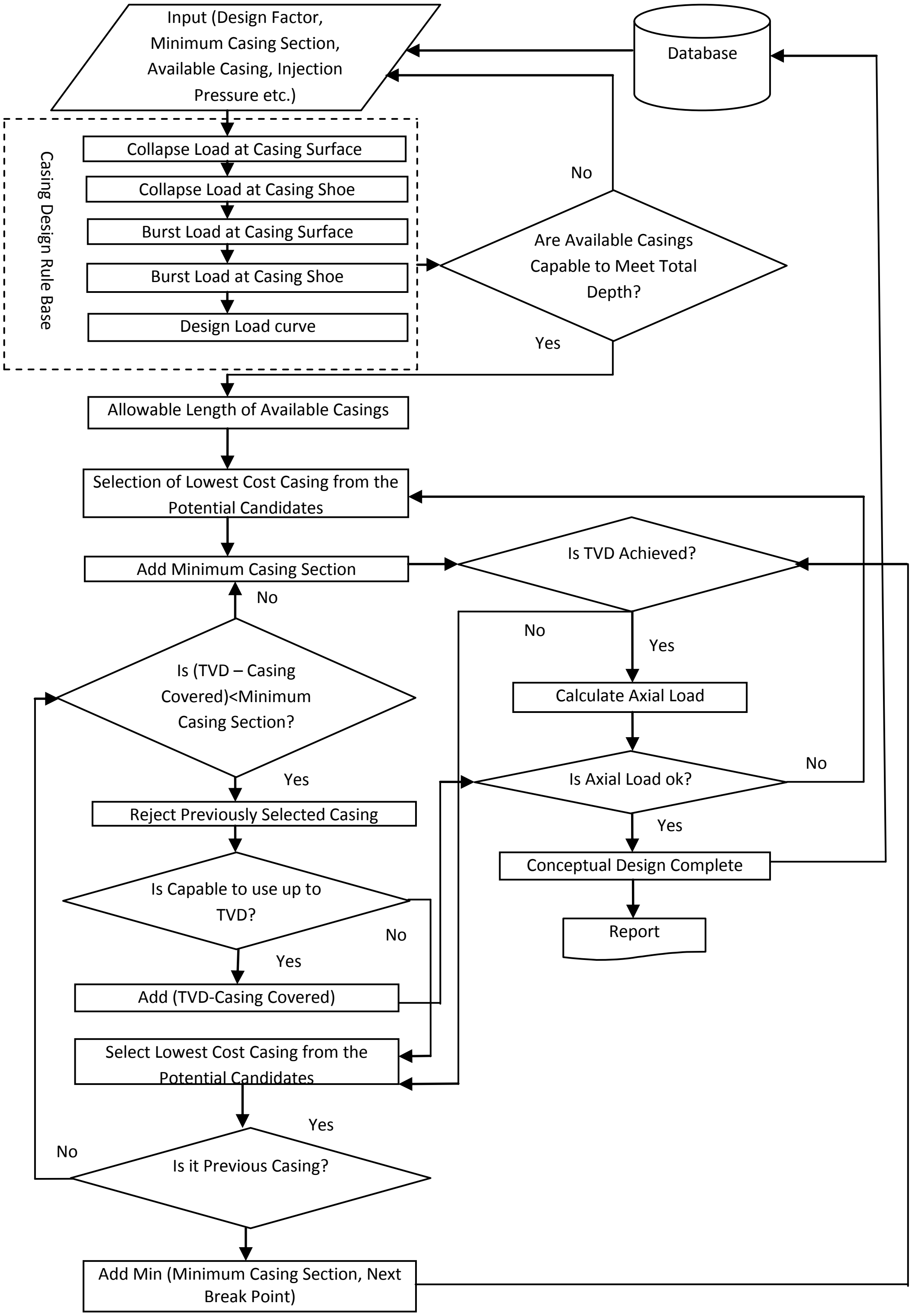

Input (Design Factor, Minimum Casing Section, Available Casing, Injection Pressure etc.)
Database
Casing Design Rule Base
Collapse Load at Casing Surface
Collapse Load at Casing Shoe
Burst Load at Casing Surface
Burst Load at Casing Shoe
Design Load curve
No
Are Available Casings Capable to Meet Total Depth?
Yes
Allowable Length of Available Casings
Selection of Lowest Cost Casing from the Potential Candidates
Add Minimum Casing Section
Is TVD Achieved?
No
No
Yes
Is (TVD – Casing Covered)<Minimum Casing Section?
Calculate Axial Load
Yes
Is Axial Load ok?
No
Reject Previously Selected Casing
Yes
Conceptual Design Complete
Is Capable to use up to TVD?
No
Report
Yes
Add (TVD-Casing Covered)
Select Lowest Cost Casing from the Potential Candidates
Yes
No
Is it Previous Casing?
Add Min (Minimum Casing Section, Next Break Point)

Figure 4-8: Flowchart of casing selection

4.3.4. Casing Selection

A well has several sections and casing design is required for each hole. The program contains a rule base to calculate collapse, burst and axial load. The rule base for surface, intermediate and production casing load calculation are given in the Appendix A. Every company has their own set of standards to calculate these loads during the casing design. Flexible software is very much useful to adopt with companies need. This flexibility can be achieved by adding the new rule to the rule base.

Casing selection process flow chart has been shown in Figure 4-8. During casing selection the program first calculates collapse and burst load and set their values as constraints. Depending on the type of casing section, the program selects the appropriate rules from the rule base for load calculation. Such as, the collapse load calculation for surface and intermediate casing are not same, different procedure has to be followed. The system identifies which rules to be fired by considering types of section.

After determining the collapse and burst rating the program checks whether the available casings are capable to meet the total depth. If it does not meet whole depth then the program asks to provide more casings. Once it finds that the total depth is achievable with these casings the program then determines the allowable length for each available casing based on collapse and burst rating. The allowable length is used as break points for the algorithm. The system determines the potential candidates in each break point. The potential candidates can be defined as the type of casing that can be used safely in the concerned depth interval. It is assumed that the available casing input is provided sequentially starting from the lower costs to higher costs. Usually costs are related to the grade of the casing. Higher grade casings are more expensive than the lower grade casings. However, the developed program does not check the grades and type of joints of the casing during casing selection process. It is assumed that the user has a better judgment. It only considers the sequence of casing input. The program selects the (most economical) lowest cost casing type from the potential candidates and adds a length equal to minimum casing section. Minimum casing section is a very important factor in this algorithm; it limits the number of different types of casing used in a combined casing string. Byrom (2007) suggested this minimum casing section length should not be less than 500 ft. The system selects the first casing type and adds a length equal to the minimum casing section length. The next step is to check whether the casing string achieve the total depth. If the total depth has not achieved yet then the program chooses again the lowest cost casing type from the available candidates. If the candidate selected in that stage is similar to the previous casing type then the system determines the minimum value of the minimum casing section and difference between the next break point and casing covered, i.e. *Minimum (Minimum casing section, (Next break point-casing covered))*. The minimum of these two values is taken and added to the casing and checked again whether the casing string achieves the total depth. If it does not achieve the total depth then the process continues until the whole depth has been covered.

Once the casing selection has been done based on burst and collapse requirements the next step is to check whether the selected casing is capable to sustain with the axial load. The rule base is used to calculate the axial load of each section of casing. The axial load is checked on the joint of the casing. If any portion of the casing string fails to satisfy the axial load condition then the casing selection starts again from the beginning. If the axial load is satisfactory with the designed casing then the conceptual design of casing is completed. This conceptual design will be stored in a database and can be further used by other modules. At the end of casing design a formal report is generated. This process can be repeated for other section of the hole. An example case study of the proposed approach has been provided in chapter eight, section 8.3.

4.4. Summary

In this chapter the comprehensive casing design approach is described. The algorithms for casing setting depth and casing size determination and casing selection have been presented.

Chapter 5 Smart Drill String Design

5.1. Introduction

A drill string is a collection of drill pipes, drill collars, heavy weight drill pipe, crossover sub and bit sub that transmits drilling fluid and rotational power to the drill bit. They are hollow shafts through which drilling fluid can be pumped down and through the annulus the fluid and cutting, i.e. drilling mud with rock chips, brought back to the surface. According to Cunha (2002), drill string design is the most important part for operations in drilling engineering.

In this chapter, a brief description of drill string components and their functions will be provided then the proposed approach and detailed description of the model will be discussed.

5.2. Drill String Components and Functions

A drill string composed of kelly, drill pipe, heavy weight drill pipe, drill collar, stabilizer, drill bit, reamer, crossover sub, drilling jars etc. Each of these components serves some basic functions. Kelly is used to transmit the rotation of the rotary table to the drill string and conducts drilling fluid from the swivel to the drill stem. Drill pipe transmits rotary motion and drilling mud under high pressure to the drill bit. Drill collar provides weight on bit for drilling and keeps the drill pipe in tension. Crossover sub connects last joint of drill string to the first drill collar. Stabilizer keeps the hole straight.

5.3. Proposed Drill String Design Approach

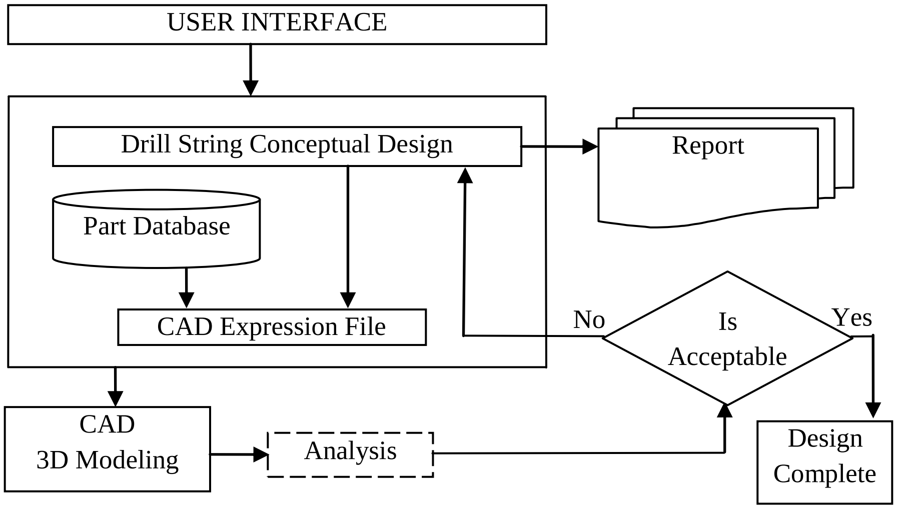


Figure 5-1: Drill string design

The approach is to automate the design and 3D modeling process for a drill string. Built-in rules and knowledge are used to develop the conceptual design of a drill string and then the system automatically generates the assembly configuration based on the conceptual design and retrieves part specifications from the part database to generate the CAD parametric files. These parametric files are used to develop the CAD model. The proposed model is presented in Figure 5-1.

The following sections describe step by step drill string design process (Figure 5-2) from conceptual design to 3D model realization.

5.3.1. Drill String Operational Requirements

Drill string design starts after the operational requirements have been defined based on the casing design output and the customer input. The operational requirements include type of the well, depth, mud specific gravity, maximum WOB, margin of overpull, safety factor for collapse, type of drill pipe available in the inventory, drill collar, heavy weight drill pipe size, etc.

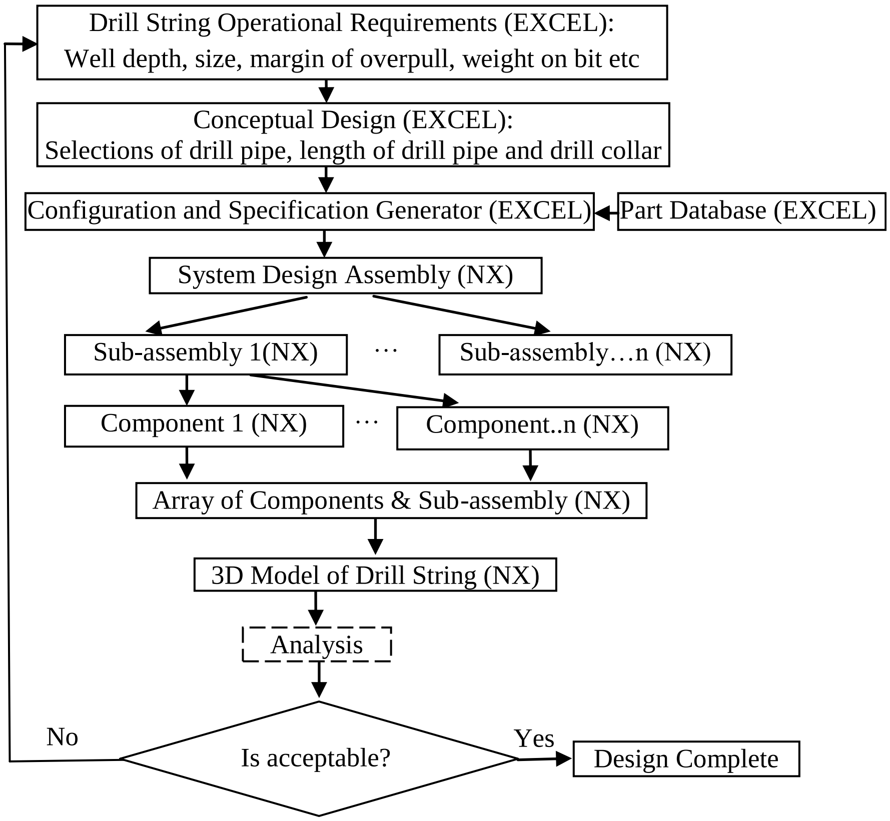

Drill String Operational Requirements (EXCEL):
Well depth, size, margin of overpull, weight on bit etc
Conceptual Design (EXCEL):
Selections of drill pipe, length of drill pipe and drill collar
Configuration and Specification Generator (EXCEL)
Part Database (EXCEL)
System Design Assembly (NX)
Sub-assembly 1(NX)
…
Sub-assembly…n (NX)
Component 1 (NX)
…
Component..n (NX)
Array of Components & Sub-assembly (NX)
3D Model of Drill String (NX)
Analysis
No
Is acceptable?
Yes
Design Complete

Figure 5-2: Drill string design and modeling process

5.3.2. Conceptual Design

The drill string design module of the proposed system generates the conceptual design based on a set of built-in engineering rules embedded in the module following the recommended practice for drill stem design standards (API RP 7G, 1998) For example, one such rule is that "*drill pipe should always be under effective tensile stress, neutral point of buckling should be in the drill collar*". In the conceptual design stage calculating the length of drill collar requires the WOB data. Then the system selects the cheapest (in most of the cases also weakest) drill pipe type from the available inventory and check against the loading criteria. If the type is not safe to run the whole length of the drill string, the allowed maximum length of that drill pipe type is worked out. The selection cycle of the drill pipes continues until the whole length of the drill string is achieved. Hence, the algorithm selects the cheapest drill string assembly based on the lowest grade and the unit weight of the pipes in the inventory.

5.3.3. Configuration and Specification Generator

The configuration and specification generator converts the conceptual design into expression files. Knowledge about parts, knowledge about the relations between parts, attributes, and constraints of parts are the configuration knowledge (Myung and Han 2001). Once the conceptual design of drill string is completed the next step is to determine the drill string configurations and component specifications. The configuration design determines the number and types of components, their orientation and position in the drill string assembly. As the drill string is a symmetric vertical column, all components of the string possess same origin for x and y coordinate, i.e. (0, 0). Only the vertical coordinate, i.e. z coordinate, changes when a new part is added to the assembly. Rules have been created to determine the origin or position of a new component. The following rule determines the z coordinate of drill pipe:

Origin (z-coordinate) of Drill pipe= HWDP_z+ Drill_collar_z+ Bit_sub_z
+ Drill_bit_z+Cross_over_sub_z Eq. (5-1)

Here,

HWDP_z= Number of HWDP Length of HWDP+ Origin (z-coordinate)* Eq. (5-2)

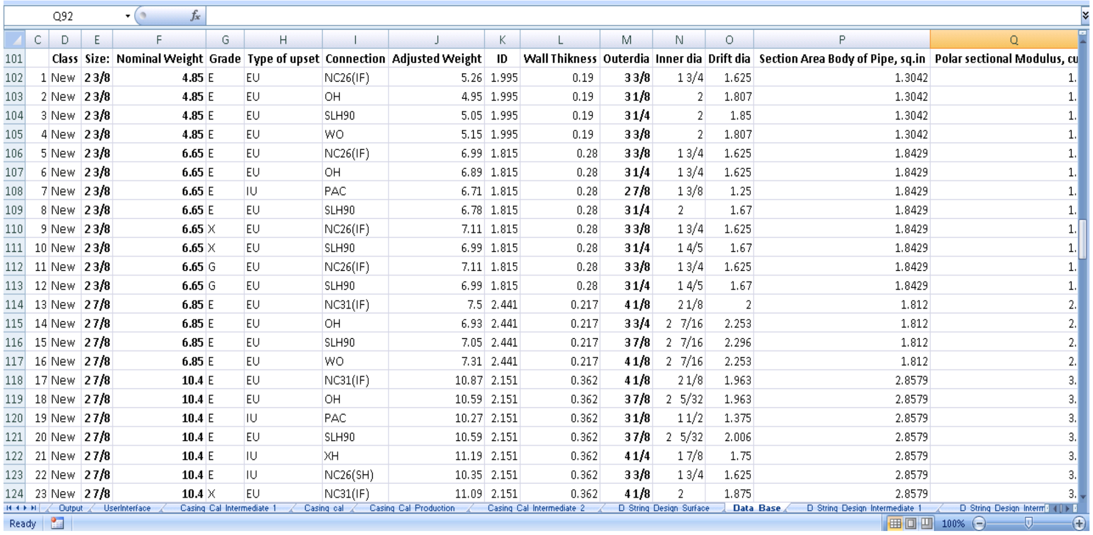
*Drill_collar_z= Number of Drill_collar * Length of Drill_collar*
+ Origin (z-coordinate) Eq. (5-3)

Bit_sub_z= Length of Bit_sub + Origin (z-coordinate) Eq. (5-4)

Dirll_bit_z= Length of Drill_bit + Origin (z-coordinate) Eq. (5-5)

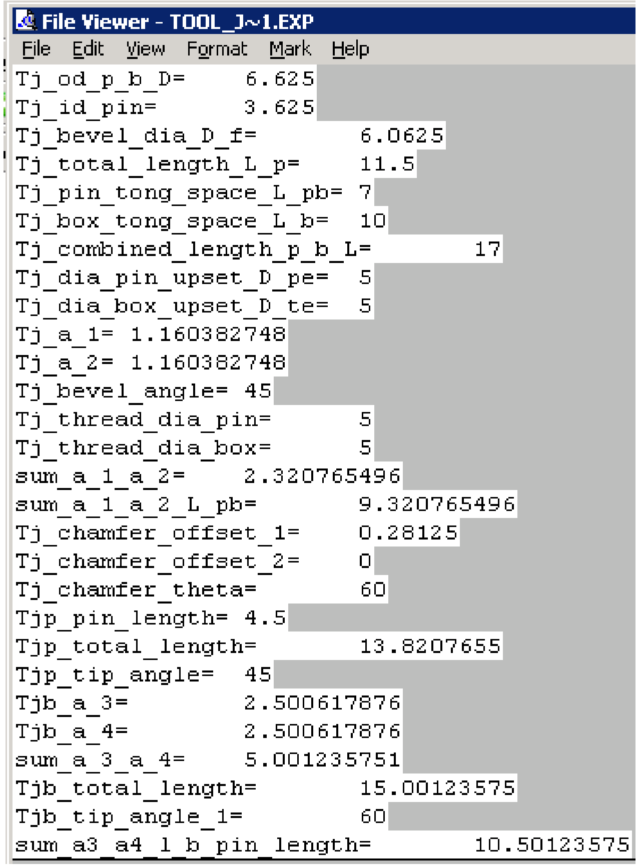
Cross_over_sub_z= Length of Cross_over_sub + Origin (z-coordinate)   Eq. (5-6)

The specifications of different components are retrieved from part data base. For that reason, a part database prototype has been created. Figure 5-3 shows partly the database of drill pipe. In this database, each drill pipe is defined by six factors- size, class, nominal weight, grade, type of upset and connection. The values of these six factors are unique for each different type of drill pipe. Based on this unique combination of these six factors the specification generation method automatically retrieves rest of the geometric and non-geometric specifications of each component according to the library specifications. This information is then converted into expression files. As for example, tool joint pin and box expression file is shown in Figure 5-4.



	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
101		Class	Size:	Nominal Weight	Grade	Type of upset	Connection	Adjusted Weight	ID	Wall Thikness	Outerdia	Inner dia	Drift dia	Section Area Body of Pipe, sq.in	Polar sectional Modulus, cu
102	1	New	2 3/8	4.85	E	EU	NC26(IF)	5.26	1.995	0.19	3 3/8	1 3/4	1.625	1.3042	1.
103	2	New	2 3/8	4.85	E	EU	OH	4.95	1.995	0.19	3 1/8	2	1.807	1.3042	1.
104	3	New	2 3/8	4.85	E	EU	SLH90	5.05	1.995	0.19	3 1/4	2	1.85	1.3042	1.
105	4	New	2 3/8	4.85	E	EU	WO	5.15	1.995	0.19	3 3/8	2	1.807	1.3042	1.
106	5	New	2 3/8	6.65	E	EU	NC26(IF)	6.99	1.815	0.28	3 3/8	1 3/4	1.625	1.8429	1.
107	6	New	2 3/8	6.65	E	EU	OH	6.89	1.815	0.28	3 1/4	1 3/4	1.625	1.8429	1.
108	7	New	2 3/8	6.65	E	IU	PAC	6.71	1.815	0.28	2 7/8	1 3/8	1.25	1.8429	1.
109	8	New	2 3/8	6.65	E	EU	SLH90	6.78	1.815	0.28	3 1/4	2	1.67	1.8429	1.
110	9	New	2 3/8	6.65	X	EU	NC26(IF)	7.11	1.815	0.28	3 3/8	1 3/4	1.625	1.8429	1.
111	10	New	2 3/8	6.65	X	EU	SLH90	6.99	1.815	0.28	3 1/4	1 4/5	1.67	1.8429	1.
112	11	New	2 3/8	6.65	G	EU	NC26(IF)	7.11	1.815	0.28	3 3/8	1 3/4	1.625	1.8429	1.
113	12	New	2 3/8	6.65	G	EU	SLH90	6.99	1.815	0.28	3 1/4	1 4/5	1.67	1.8429	1.
114	13	New	2 7/8	6.85	E	EU	NC31(IF)	7.5	2.441	0.217	4 1/8	2 1/8	2	1.812	2.
115	14	New	2 7/8	6.85	E	EU	OH	6.93	2.441	0.217	3 3/4	2 7/16	2.253	1.812	2.
116	15	New	2 7/8	6.85	E	EU	SLH90	7.05	2.441	0.217	3 7/8	2 7/16	2.296	1.812	2.
117	16	New	2 7/8	6.85	E	EU	WO	7.31	2.441	0.217	4 1/8	2 7/16	2.253	1.812	2.
118	17	New	2 7/8	10.4	E	EU	NC31(IF)	10.87	2.151	0.362	4 1/8	2 1/8	1.963	2.8579	3.
119	18	New	2 7/8	10.4	E	EU	OH	10.59	2.151	0.362	3 7/8	2 5/32	1.963	2.8579	3.
120	19	New	2 7/8	10.4	E	IU	PAC	10.27	2.151	0.362	3 1/8	1 1/2	1.375	2.8579	3.
121	20	New	2 7/8	10.4	E	EU	SLH90	10.59	2.151	0.362	3 7/8	2 5/32	2.006	2.8579	3.
122	21	New	2 7/8	10.4	E	IU	XH	11.19	2.151	0.362	4 1/4	1 7/8	1.75	2.8579	3.
123	22	New	2 7/8	10.4	E	IU	NC26(SH)	10.35	2.151	0.362	3 3/8	1 3/4	1.625	2.8579	3.
124	23	New	2 7/8	10.4	X	EU	NC31(IF)	11.09	2.151	0.362	4 1/8	2	1.875	2.8579	3.

Figure 5-3: Partial view of drill pipe database



```
Tj_od_p_b_D=     6.625
Tj_id_pin=       3.625
Tj_bevel_dia_D_f=       6.0625
Tj_total_length_L_p=    11.5
Tj_pin_tong_space_L_pb= 7
Tj_box_tong_space_L_b=  10
Tj_combined_length_p_b_L=       17
Tj_dia_pin_upset_D_pe=  5
Tj_dia_box_upset_D_te=  5
Tj_a_1= 1.160382748
Tj_a_2= 1.160382748
Tj_bevel_angle= 45
Tj_thread_dia_pin=      5
Tj_thread_dia_box=      5
sum_a_1_a_2=     2.320765496
sum_a_1_a_2_L_pb=       9.320765496
Tj_chamfer_offset_1=    0.28125
Tj_chamfer_offset_2=    0
Tj_chamfer_theta=       60
Tjp_pin_length= 4.5
Tjp_total_length=       13.8207655
Tjp_tip_angle=  45
Tjb_a_3=         2.500617876
Tjb_a_4=         2.500617876
sum_a_3_a_4=     5.001235751
Tjb_total_length=       15.00123575
Tjb_tip_angle_1=        60
sum_a3_a4_l_b_pin_length=       10.50123575
```

Figure 5-4: Expression file of a drill pipe tool joint pin and box

5.3.4. System Design Assembly

An assembly is a collection of components arranged in a specific manner. Two approaches for assembly generation is available, bottom up and top down. In bottom up approach, component parts are designed and edited apart from their usage in higher assembly. In this approach, the part solid model is first created, and then it combines with the sub-assemblies followed by assemblies. On the other hand, in top down approach, the hierarchy of assemblies and sub assemblies is designed first, and then part solid models are designed in place.

A “top-down” assembly approach has been followed, where the structure of the whole assembly of the drill string is first created; the generic configuration of drill string assembly contains all possible components of drill string. For example, a drill string assembly composed of drill pipe, drill collar, heavy weight drill pipe, bit sub, cross over sub, drill bit etc. Depending on the operational requirements some components may not be required. For example, sometimes “heavy weight drill pipe” is not used in the drill string; then in that case, the module which creates assembly structure, will suppress the “heavy weight drill pipe” from the assembly.

5.3.5. Sub-Assembly

The next level is sub-assembly generation. The program first finds out which member of the assembly contains sub-assembly from a configuration definition and then fire the rule to initiate sub-assembly creation. A drill pipe sub assembly contains three parts: drill pipe body, tool joint pin and tool joint box. That is shown in Figure 5-5. Such structure generation algorithm continues iteratively until all the configuration of the whole assembly is completed. It should be worth mentioning that, in the “top-down” approach, though assembly structure is first created but in each of the structure members, no physical geometry entities are created until the program reaches to the next stage. The next stage is to create the component geometry entities.

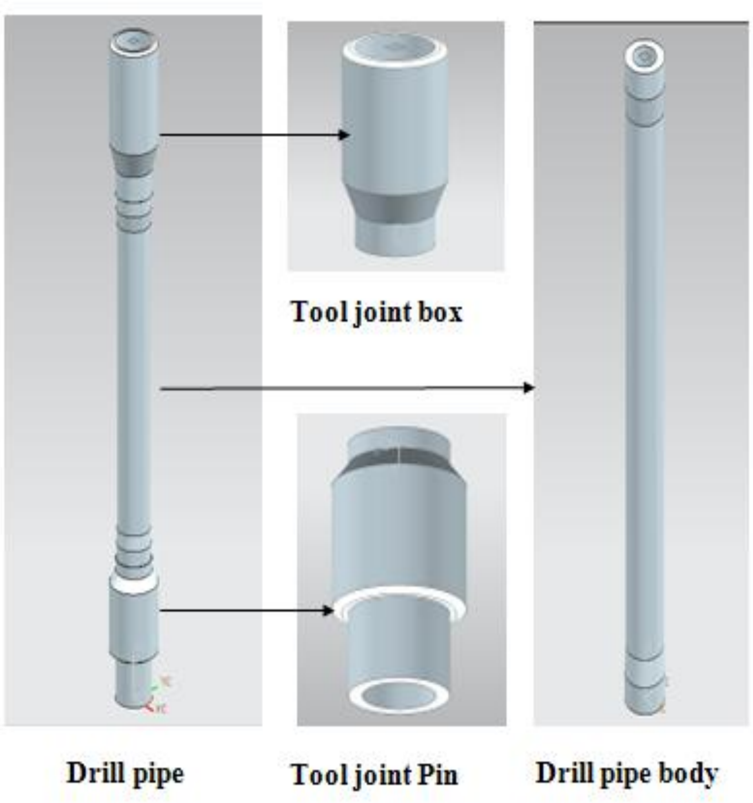


Figure 5-5: Drill pipe sub-assembly and components

5.3.6. Component Creation

Constructive Solid Geometry (CSG) approach is followed during component creation. In CSG method, the solid primitives are progressively cut and joined to form a new shape. Primitive such as cylinder, block, cone, surface etc are used to generate the component. Most common CSG operations are union, intersection and subtraction. The hierarchal construction to achieve the final components of drill pipe tool joint pin is shown in Figure 5-6. The construction of drill pipe tool joint pin is initialized with cylinder; this is the first level primitive of the drill pipe tool joint pin. Then it is connected to second level primitives, a cone, followed by the third level and so on.

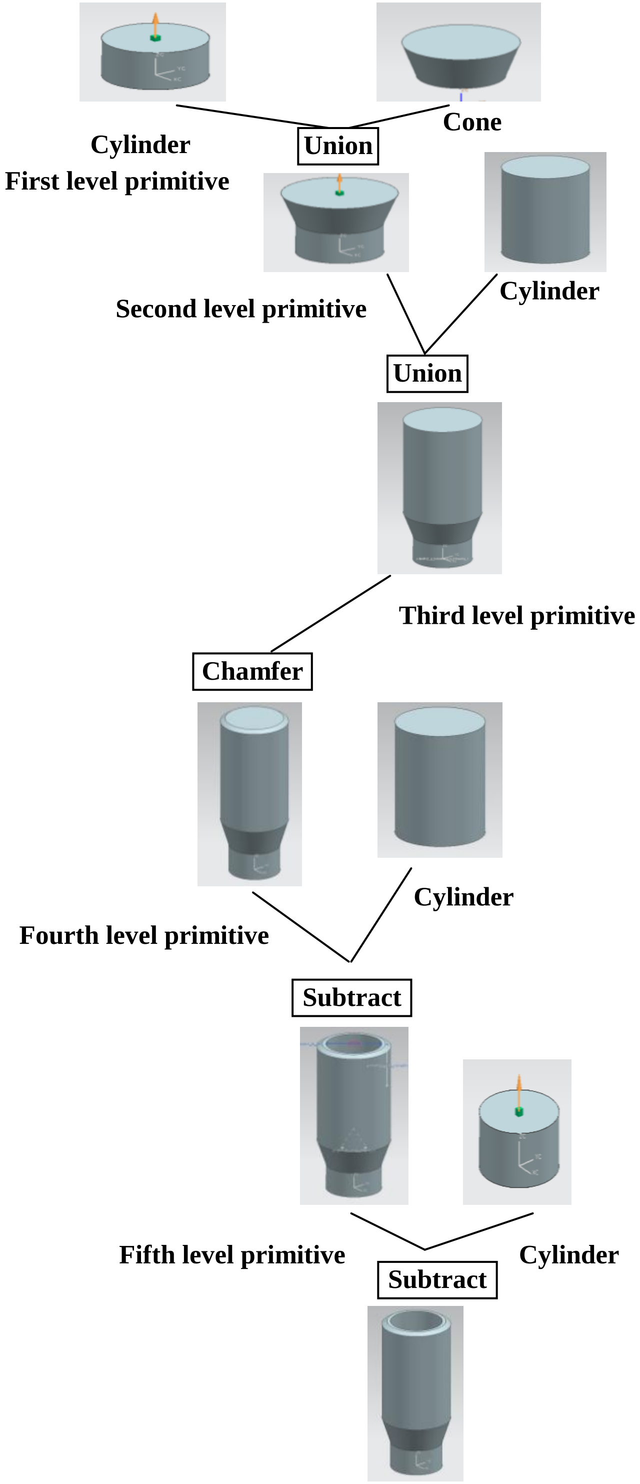
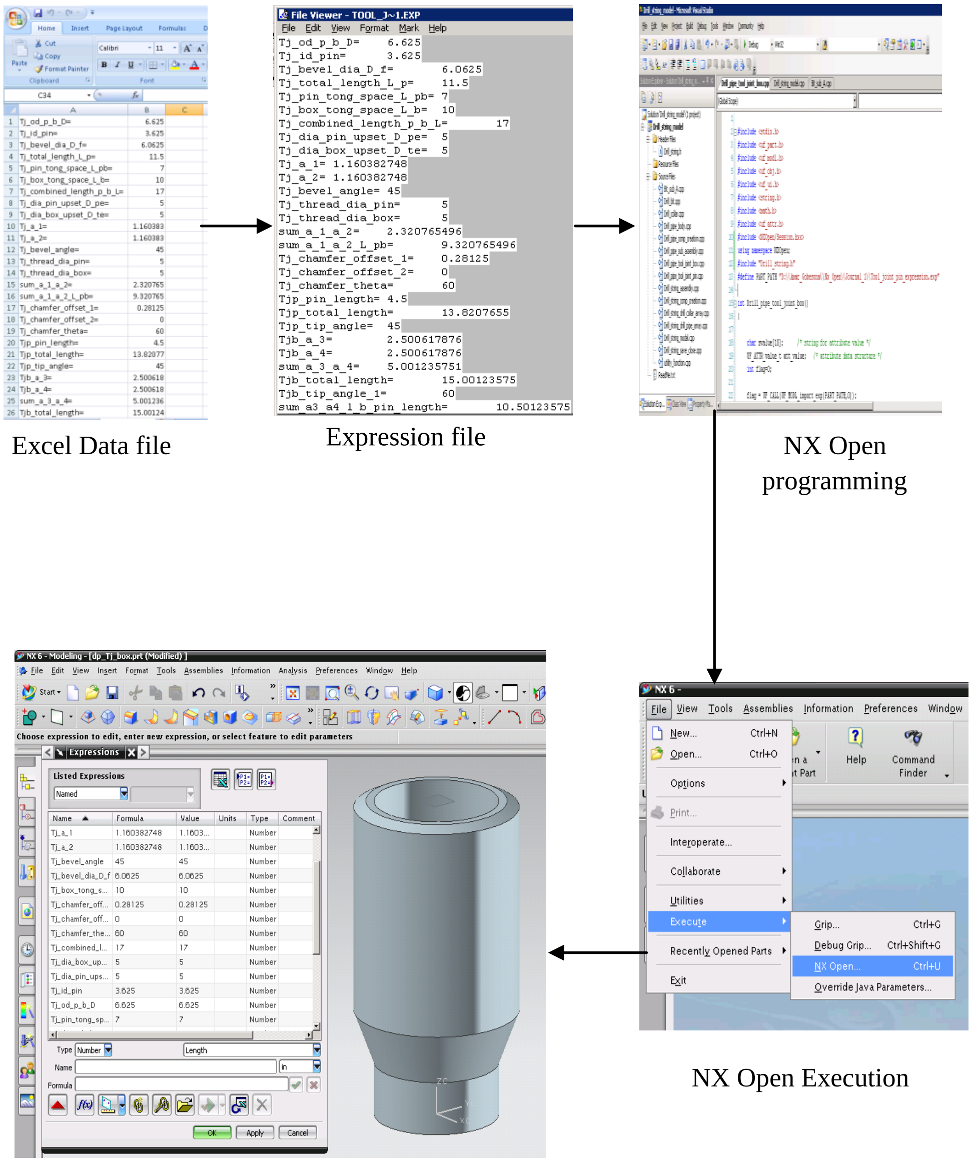


Figure 5-6: CSG approach for component creation

Individual parametric program has been written for each component to generate the generic 3D model. Although the model is created in NX environment but the parameter is controlled by configuration and specification generator. The steps required for three dimensional model realization of drill pipe tool joint box component has been shown in Figure 5-7.

The configuration and specification generator first retrieves the parts specification from the part data base and generates the excel data file for parts specification and configuration of the assembly based on the conceptual design. Then excel data files are converted into expression (.exp) files by the configuration and specification generator. Next step involves the creation of dll (dynamic link library) file. This dll file is a library of executable. NX required a dll file to execute the program. During execution NX invokes the expression files and creates the component based on design requirements.



	A	B	C
1	Tj_od_p_b_D=	6.625	
2	Tj_id_pin=	3.625	
3	Tj_bevel_dia_D_f=	6.0625	
4	Tj_total_length_L_p=	11.5	
5	Tj_pin_tong_space_L_pb=	7	
6	Tj_box_tong_space_L_b=	10	
7	Tj_combined_length_p_b_L=	17	
8	Tj_dia_pin_upset_D_pe=	5	
9	Tj_dia_box_upset_D_te=	5	
10	Tj_a_1=	1.160383	
11	Tj_a_2=	1.160383	
12	Tj_bevel_angle=	45	
13	Tj_thread_dia_pin=	5	
14	Tj_thread_dia_box=	5	
15	sum_a_1_a_2=	2.320765	
16	sum_a_1_a_2_L_pb=	9.320765	
17	Tj_chamfer_offset_1=	0.28125	
18	Tj_chamfer_offset_2=	0	
19	Tj_chamfer_theta=	60	
20	Tjp_pin_length=	4.5	
21	Tjp_total_length=	13.82077	
22	Tjp_tip_angle=	45	
23	Tjb_a_3=	2.500618	
24	Tjb_a_4=	2.500618	
25	sum_a_3_a_4=	5.001236	
26	Tjb_total_length=	15.00124	

Excel Data file

File Viewer - TOOL_J~1.EXP

```
Tj_od_p_b_D=    6.625
Tj_id_pin=      3.625
Tj_bevel_dia_D_f=       6.0625
Tj_total_length_L_p=    11.5
Tj_pin_tong_space_L_pb= 7
Tj_box_tong_space_L_b=  10
Tj_combined_length_p_b_L=       17
Tj_dia_pin_upset_D_pe=  5
Tj_dia_box_upset_D_te=  5
Tj_a_1= 1.160382748
Tj_a_2= 1.160382748
Tj_bevel_angle= 45
Tj_thread_dia_pin=      5
Tj_thread_dia_box=      5
sum_a_1_a_2=    2.320765496
sum_a_1_a_2_L_pb=       9.320765496
Tj_chamfer_offset_1=    0.28125
Tj_chamfer_offset_2=    0
Tj_chamfer_theta=       60
Tjp_pin_length= 4.5
Tjp_total_length=       13.8207655
Tjp_tip_angle=  45
Tjb_a_3=        2.500617876
Tjb_a_4=        2.500617876
sum_a_3_a_4=    5.001235751
Tjb_total_length=       15.00123575
Tjb_tip_angle_1=        60
sum_a3_a4_l_b_pin_length=       10.50123575
```

Expression file

NX Open programming

NX Open Execution

NX 6 - Modeling - [dp_Tj_box.prt (Modified)]

Choose expression to edit, enter new expression, or select feature to edit parameters

Listed Expressions: Named

Name	Formula	Value	Units	Type	Comment
Tj_a_1	1.160382748	1.1603...		Number	
Tj_a_2	1.160382748	1.1603...		Number	
Tj_bevel_angle	45	45		Number	
Tj_bevel_dia_D_f	6.0625	6.0625		Number	
Tj_box_tong_s...	10	10		Number	
Tj_chamfer_off...	0.28125	0.28125		Number	
Tj_chamfer_off...	0	0		Number	
Tj_chamfer_the...	60	60		Number	
Tj_combined_l...	17	17		Number	
Tj_dia_box_up...	5	5		Number	
Tj_dia_pin_ups...	5	5		Number	
Tj_id_pin	3.625	3.625		Number	
Tj_od_p_b_D	6.625	6.625		Number	
Tj_pin_tong_sp...	7	7		Number	

Type: Number; Length; Name: ; in; Formula:

CAD model realization and expression file

Figure 5-7: Steps involved in component creation

5.3.7. Arrays of Components

An array method has been implemented to repeat the component and subassembly instantiation. As shown in Figure 5-8, a large quantity of similar type of drill pipe may exist in the drill string; this array method helps to reproduce the drill pipe and other components throughout the assembly.

5.3.8. Three Dimensional Model of Drill String and Analysis

After the arrays of components and sub-assemblies are carried out, the 3D model of the whole drill string assembly is realized (Figure 5-8). They might be useful for finite element analysis and simulation. If the result of the analysis is not satisfactory then the program redesigns the drill string by following the same steps mentioned earlier; then the whole design loop is integrated.

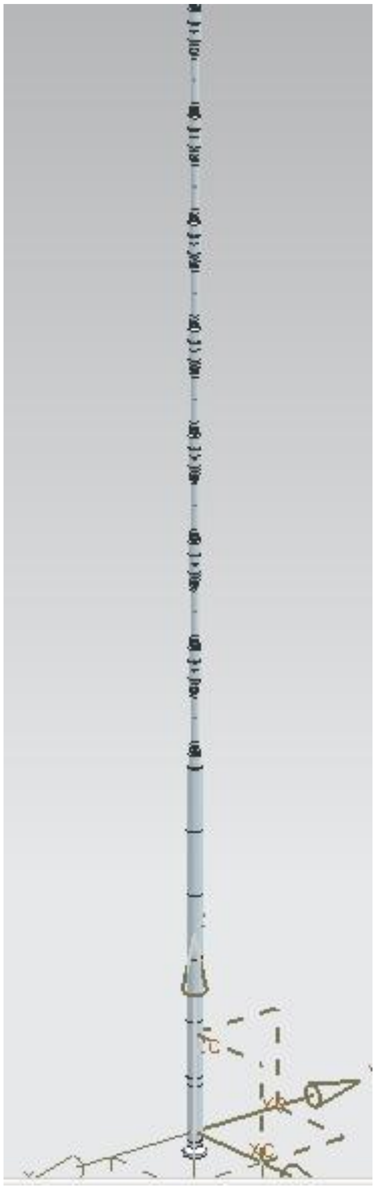


Figure 5-8: Partial assembly of drill string

5.4. Summary

This chapter provides a detailed description of the generative approach of drill string modeling. Conceptual design, assembly modeling, configuration design have been discussed. In order to maintain consistency in the assembly an example rule has been presented.

Chapter 6 Feature-based Generative CAD Modeling

6.1. Introduction

This chapter introduces the feature-based generative approach of CAD modeling for well casing and drill string design. Features at different levels are defined and discussed with application examples.

6.2. Feature-based Design

The proposed system uses knowledge driven CAD system to design the well casing and drill string. It has two parts – design and modeling. The conceptual design is carried out in Excel Visual Basic Application environment and then design output is transferred to the CAD system to generate the three dimensional CAD model. In order to smooth the integration of these two systems, i.e. design and modeling, feature-based concept is employed. One advantage of using feature comes from the abstraction of feature information that retains not only the associative geometric information but also much useful non-geometric information that reflects engineering semantics within different applications (Liang and Grady 2002). Another advantage of using feature is that, it can be used as a bridge between CAD and KBE system (Ma et al. 2007).

A well consists of several sections of different diameter holes. Each section of the well contains a casing string. A casing string is composed of casing and coupling. Couplings are used to join the casing string. On the other hand, a drill string composed of drill pipe, drill collar, heavy weight drill pipe, bit sub, drill bit etc. In this research work three different features levels are defined. These are assembly and sub-assembly level feature, component level feature and form feature. Figure 6-1 shows feature classes at different levels.

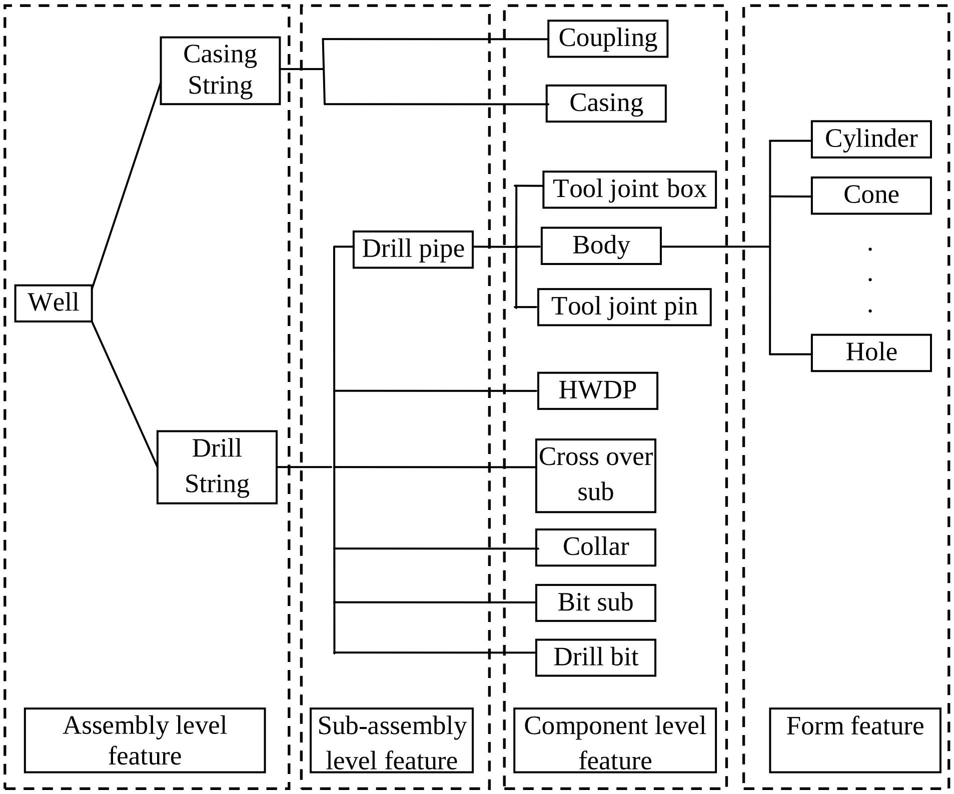


Figure 6-1: Features at different levels

6.3. Feature Representation

The generative CAD model of drill string and casing are realized by considering three feature levels. These are assembly or sub-assembly level feature, component level feature and form feature. These features are described in detail in the following section.

6.3.1. Assembly Level Feature

Figure 6-2 (a) shows the generic definition of assembly level feature. The attributes or properties include list of components and sub-assemblies, number of array of components, distance between arrays, constraints list, configuration etc. Related geometric entities can be assemblies, components, solids, form features, faces, edges, vertices etc. The sub-assembly level feature is also defined in the similar manner.

Assembly

Attributes:
Component or Subassembly;
Number of array;
Distance between arrays;
Configuration;
Mating conditions;
Length;
Constraints;

Methods:
Engineering calculation ();
Configurations generator ();
Array of components ();
Feature dimension control mechanism ();
Configuration control mechanism ();
Validity check ();
Feature generation ();
Modification ();
Delete ();

(a)

Drill String

Attributes:
Component 1: Drill pipe;
Component 2: Drill collar;
Component 3: Cross over sub;
Component 4: Bit sub;
Component 5: Drill bit;
Drill pipe array: 403;
Drill collar array: 21;
Distance between drill pipes: 360 inch;
Distance between drill collars: 360 inch;
Drill pipe position (inch): (0, 0, 7582.5);
Drill collar position (inch): (0, 0, 22.5);
………
Mating condition 1: Concentric;
………
Constraints 1: dc_o=db_o+db_l+bs_o+bs_l;
Constraints 2: cs_b_d=dp_tjp_d;
………
Length: 12700 ft;

Methods:
Engineering calculation ();
Configurations generator ();
Array of components ();
Feature dimension control mechanism ();
Configuration control mechanism ();
Validity check ();
Feature generation ();
Modification ();
Delete ();

(b)

Figure 6-2: Assembly level feature definition (a) Generic, (b) Drill string instantiation

Figure 6-2 (b) shows drill string assembly level feature instantiation. It is composed of drill pipe, collar, crossover sub, bit sub and drill bit. The number of drill pipe and drill collar array and distance between the arrays are also required to generate the CAD model. The position of drill pipe and drill collar in the assembly level feature represents the position of first drill pipe and drill collar in the assembly. The drill string is a vertical column and all the components are concentric. This is one example of mating condition of the assembly level feature. Other mating conditions such as coplanar, center distance, mate align angle are also implemented in the assembly level feature.

Associative relationships among the drill string components have been created to maintain the consistency in the assembly level feature. Such as, crossover sub is used in between drill collar and drill pipe. This is because the size of drill pipe tool joint pin is not similar to the size of drill collar joint. So, an associative relationship is required to solve this issue. The box and pin joint diameter of crossover sub are linked with the diameter of drill pipe tool joint pin and collar joint box respectively. This associative link of crossover sub has been shown in Figure 6-3. Similar associative links have been created for other components in the assembly level.

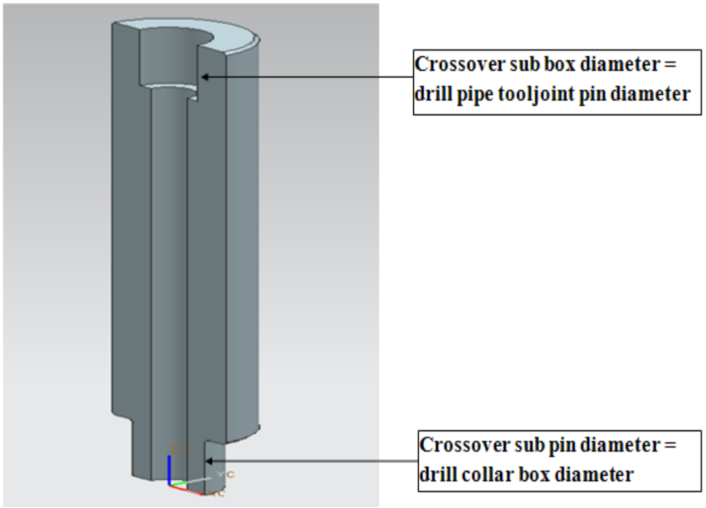

Crossover sub box diameter =
drill pipe tooljoint pin diameter
Crossover sub pin diameter =
drill collar box diameter

Figure 6-3: Associative link of cross over sub

6.3.2. Component Level Feature

Component

Attributes:
Non-geometric;
Geometric;
Dimensions;
Parameters,
Form features;
Constraints;

Methods:
Feature dimension control
Mechanism ();
Topological variation ();
Validity check ();
Feature generation ();
Modification ();
Delete ();

(a)

Drill pipe body

Attributes:
Grade: E 75;
Class: 2;
Weight: 16.6 lb;
Type of upset: IEU;
Range: 2;
Outer diameter: 4.5 inch;
Inner diameter: 3.826 inch;
Length: 335.8 inch
Upset diameter: 4.75 inch;
……..
Cylinder 1;
Cone1;
Cylinder 2;
Hole 1;
……..
Constraint 1: cn_1_dia=c_1_dia;
Constraint 2: h_1_dia<c_1_ dia;
………

Methods:
Feature dimension control Mechanism ();
Topological variation ();
Validity check ();
Feature generation ();
Modification ();
Delete ();

(b)

Figure 6-4: Component level feature definition (a) Generic, (b) Drill pipe body instantiation

Figure 6-4 (a) shows the definition of component level feature; the properties include geometric, non-geometric data, list of form features, and list of constraints.

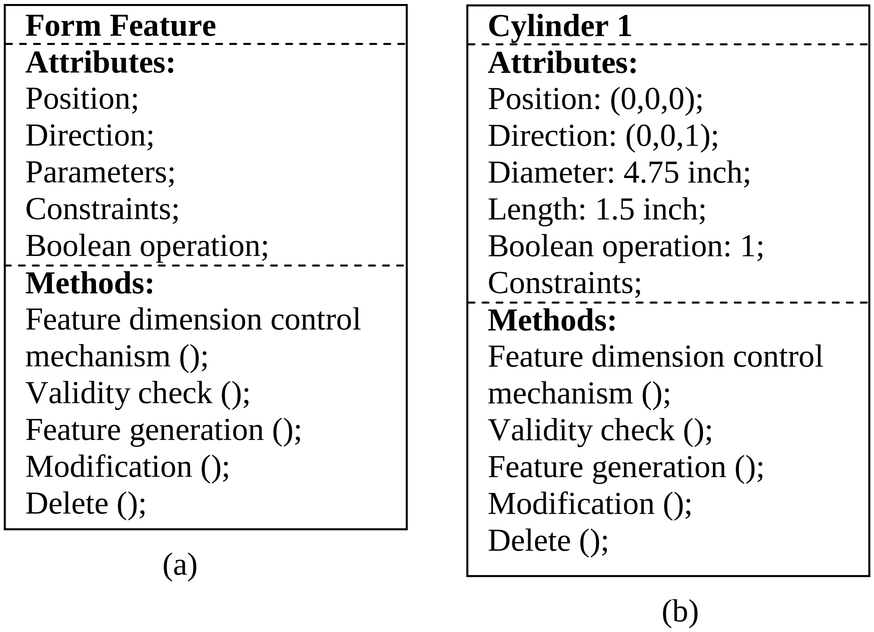
An example component level feature, drill pipe body, has been shown in Figure 6-4 (b). The drill pipe body feature contains both geometric and non-geometric information. Geometric information such as outside diameter, inside diameter, length, upset diameter etc are required by the CAD system to generate the 3D model. Non-geometric information such as grade, class, weight, material properties, i.e. collapse rating, axial load rating etc., are required for further analysis such as CAE analysis, simulation etc.

As mentioned in the previous chapter, constructive solid geometry approach has been followed in the creation of 3D model. So a list of form features is required. The drill pipe body consists of the following form features cylinder 1, cone 1, cylinder 2, cone 2, cylinder 3, hole 1, hole 2, hole 3. In order to maintain consistency in the component level several constraints have been created. As for example:

Constraint 1: Cone 1 top diameter should be equal to the cylinder 1 diameter (cn_1_dia= c_1_dia).

Constraint 2: Hole 1 diameter should be always less than cylinder 1 diameter (h_1_dia<c_1_ dia).

6.3.3. Form Feature



Form Feature

Attributes:
Position;
Direction;
Parameters;
Constraints;
Boolean operation;

Methods:
Feature dimension control mechanism ();
Validity check ();
Feature generation ();
Modification ();
Delete ();

(a)

Cylinder 1

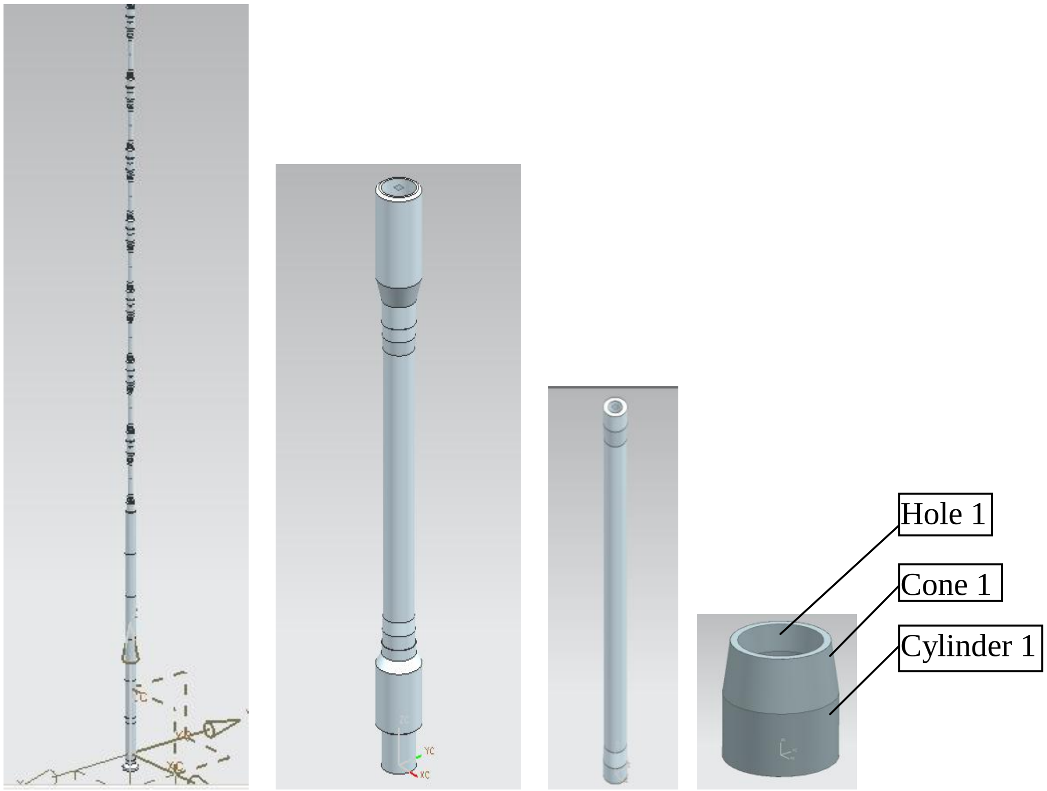
Attributes:
Position: (0,0,0);
Direction: (0,0,1);
Diameter: 4.75 inch;
Length: 1.5 inch;
Boolean operation: 1;
Constraints;

Methods:
Feature dimension control mechanism ();
Validity check ();
Feature generation ();
Modification ();
Delete ();

(b)

Figure 6-5: Form feature definition (a) Generic, (b) Cylinder 1 instantiation

Figure 6-5 (a) shows the definition of form feature. It contains the position, direction, parameters, list of constrains and type of boolean operations. An example instantiation of cylinder 1 form feature has been shown in Figure 6-5 (b). It is the first form feature of the drill pipe body, so it should be created at the origin. Other form features such as cone, hole etc are also defined in the similar manner. Figure 6-6 shows the 3D model realization of assembly and sub-assembly, component and form level features.



(a) Drill string (b) Drill pipe (c) Pipe body (d) Form features

Figure 6-6: Different levels of feature instantiation (a) Assembly, (b) Sub-assembly, (c) Component, (d) form feature.

6.4. Feature-based CAD Modeling Methods

These three levels of features are implemented by applying several methods. Some of the methods used in the different feature levels are described in the following sections.

6.4.1. Engineering Calculations

All the engineering calculations are carried out in Excel VBA environment. Several rule bases for engineering calculations are created. A reasoning mechanism sorts out which rules to be fired based on the users input. Engineering calculations method executes the calculations and generates the conceptual design.

6.4.2. Configuration Generation

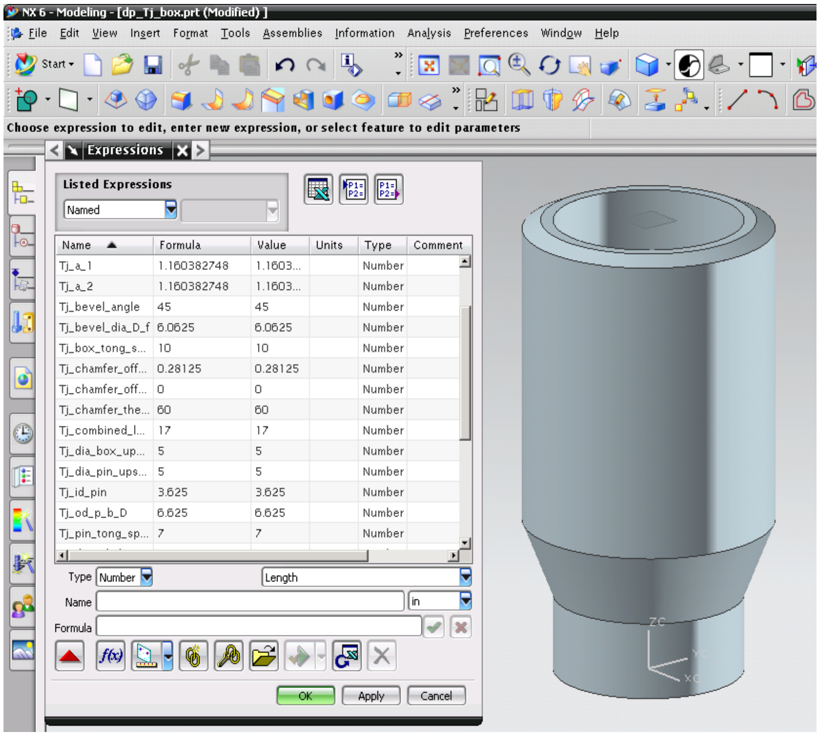
An assembly level feature requires the list of components, their specifications, orientation and positions in the assembly. Configuration generation method carries out these tasks. This method determines the configurations and specifications of components. As for example, in Figure 6-2 (b) drill string assembly level feature, the positions of drill pipe and drill collar are calculated as (0, 0, 7582.5) and (0, 0, 22.5).

6.4.3. Array of Components

The assembly may contain multiple numbers of similar types of components. Such as drill string assembly level feature contains multiple numbers of drill pipes, drill collars, heavy weight drill pipes. Similarly, casing string assembly level feature contains multiple numbers of casings and couplings. This repetition of similar type of component is termed as array in this work. Array of component method determines the number of array in the assembly based on the conceptual design. In Figure 6-2 (b), the instantiation of drill string assembly level feature contains 21 drill collars and 403 drill pipes array.

6.4.4. Feature Dimension Control Mechanism

Most of the design and engineering calculations are carried out in the Excel VBA environment. A method is required to import this information into the CAD system. The feature dimension control mechanism method takes care of this function. This method is responsible to retrieve the necessary expression for 3D model creation.



NX 6 - Modeling - [dp_Tj_box.prt (Modified)]

Choose expression to edit, enter new expression, or select feature to edit parameters

Expressions

Listed Expressions: Named

Name	Formula	Value	Units	Type	Comment
Tj_a_1	1.160382748	1.1603...		Number	
Tj_a_2	1.160382748	1.1603...		Number	
Tj_bevel_angle	45	45		Number	
Tj_bevel_dia_D_f	6.0625	6.0625		Number	
Tj_box_tong_s...	10	10		Number	
Tj_chamfer_off...	0.28125	0.28125		Number	
Tj_chamfer_off...	0	0		Number	
Tj_chamfer_the...	60	60		Number	
Tj_combined_l...	17	17		Number	
Tj_dia_box_up...	5	5		Number	
Tj_dia_pin_ups...	5	5		Number	
Tj_id_pin	3.625	3.625		Number	
Tj_od_p_b_D	6.625	6.625		Number	
Tj_pin_tong_sp...	7	7		Number	

Type: Number | Length

Name: | in

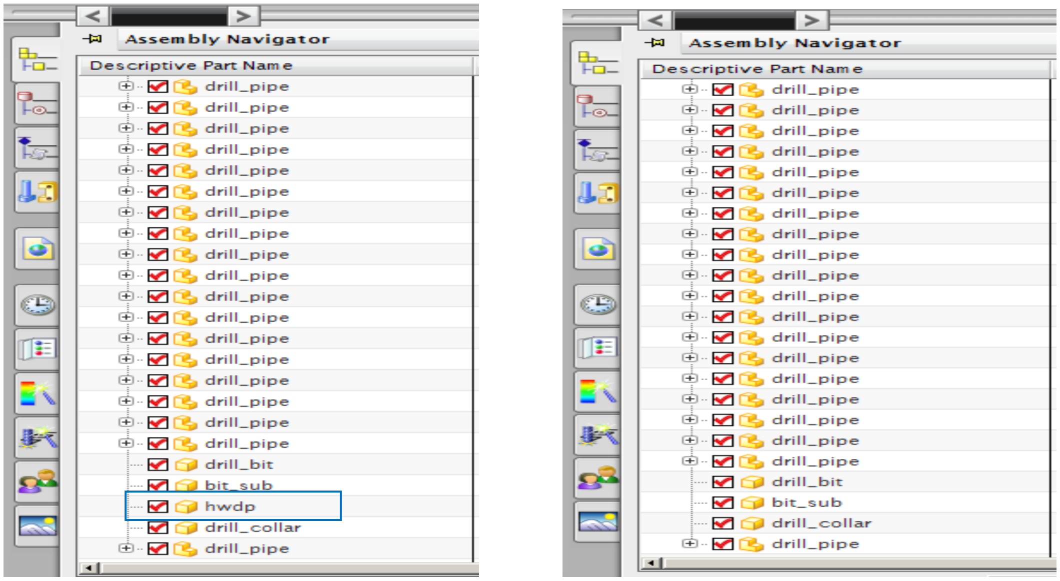
Formula:

Figure 6-7: Drill pipe tool joint box instantiation

Figure 6-7 shows the instantiation of drill pipe tool joint box, a component level feature, with necessary expression file. The feature dimension control mechanism retrieves the dimensional parameter from a predefined location. A macro code has been written for the path definition. The path or predefined location can be changed according to the requirements.

6.4.5. Configuration Control Mechanism

Configuration control mechanism decides which generation functions of the components in the assembly to be run. A top down assembly modeling approach has been followed to develop the assembly model. So, the default assembly configuration model contains all the possible components. In some occasions, one or more components are not considered in the design process. In that case, the configuration control mechanism suppresses the generation function of these particular components which are not considered. As a result these components will not include in the assembly model. As for example, in drill string assembly level feature, the heavy weight drill pipe is not considered in some occasion. In that situation the suppress method deactivate the heavy weight drill pipe generation function. Figure 6-8 (a) shows the drill string assembly configuration with heavy weight drill pipe (hwdp) and Figure 6-8 (b) shows the assembly configuration without heavy weight drill pipe.



Assembly Navigator

Descriptive Part Name

☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_bit
☑ bit_sub
☑ hwdp
☑ drill_collar
☑ drill_pipe

Assembly Navigator

Descriptive Part Name

☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_pipe
☑ drill_bit
☑ bit_sub
☑ drill_collar
☑ drill_pipe

(a) (b)

Figure 6-8: Drill string assembly configuration

6.4.6. Validity Check

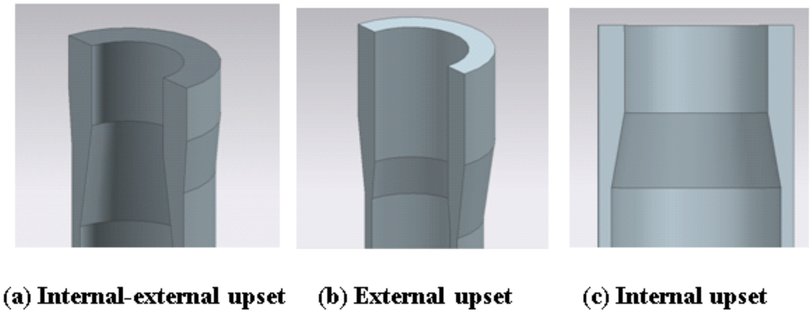
Validity check method keeps a feature to be self-contained and well-defined. It verifies the input of users, and invokes the next process if the inputs are acceptable, or provide feedback to the users if necessary.

6.4.7. Feature Generation

Each level of feature has its own generation method. Such as assembly generation method is responsible to create the assembly structure, the Component creation method creates the components 3D model and form feature generation method is used to create the specific form.

6.4.8. Topological Variation

Topological variation can be realized with this method. Based on the user input the method selects which generation functions of different topology should be run. As for example, as shown in Figure 6-9, a drill pipe body may have three different topologies – internal-external upset, external upset and internal upset.



(a) Internal-external upset (b) External upset (c) Internal upset

Figure 6-9: Drill pipe features

6.7. System implementation

The prototype of the well design system has been implemented using a intel® Core 2 Duo compatible as the hardware. This prototype system uses a commercial

CAD system (UG NX 6) and is developed using the Microsoft Excel with Visual Basic language and NX programming functions in a Windows environment.

6.8. Summary

Knowledge driven CAD system has been implemented by applying feature-based concept. Three levels of features, i.e., assembly feature, component feature and form feature are defined with application examples.

Chapter 7 Operational Parameters

7.1. Introduction

Bourgoyne and Young's Rate of Penetration model will be presented at the beginning of this chapter and then the proposed system will be discussed.

7.2. Bourgoyne and Young's Model

Bourgoyne and Young's (1974) model considered the effect of the formation strength, compaction, differential pressure, WOB, rotary speed, tooth wear and bit hydraulics and used a multiple-regression technique to calculate the constants of the model. Bourgoyne and Young's ROP model is given as follows.

$$\frac{dD}{dt} = f_1 \times f_2 \times \ldots\ldots\ldots\ldots f_8 \qquad \text{Eq. (7-1)}$$

Where *f_1* is the effect of rock drillability which is proportional with formation rock strength and is given by:

$$f_1 = e^{a_1} \qquad \text{Eq. (7-2)}$$

Then term *f_2* represents the effect of depth and *f_3* represents the effect of compaction:

$$f_2 = e^{(10000-D)} \qquad \text{Eq. (7-3)}$$

$$f_3 = e^{a_3 D^{0.69}(g_p - 9)} \qquad \text{Eq. (7-4)}$$

f_4 represents the effect of differential pressure:

$$f_4 = e^{a_4(g_p - p_c)} \qquad \text{Eq. (7-5)}$$

f_5 is the function for bit diameter and weight applied into the bit:

$$f_5 = \frac{\frac{w}{d_b} - (\frac{w}{d_b})_t}{4 - (\frac{w}{d_b})_t} \qquad \text{Eq. (7-6)}$$

f_6 represents the effect of rotary speed and given by:

$$f_6 = \left(\frac{N}{100}\right)^{a_6} \qquad \text{Eq. (7-7)}$$

f_7 represents the effect of tooth wear, where h is the fractional tooth height that has been worn away.

$$f_7 = e^{-a_7 h} \qquad \text{Eq. (7-8)}$$

The term f_8 represents the effect of hydraulics:

$$f_8 = (\frac{f_j}{1000})^{a_8} \qquad \text{Eq. (7-9)}$$

The constant a_2 through a_8 can be determined by multiple regression analysis of detailed data taken over short depth intervals.

Optimum weight on bit and rotary speed can be calculated by calculating the cost per foot for a given situation. Cost per foot can be calculated by:

$$c_f = \frac{c_b + c_r(t_t + t_c + t_b)}{\Delta D} \qquad \text{Eq. (7-10)}$$

The footage can be calculated as:

$$\Delta D = J_1 J_2\, \tau_H \left[\frac{1-e^{-a_7 h_f}}{a_7} + \frac{H_2\left[1-e^{-a_7}h_f -^{-a_7} h_f\ e^{-a_7}h_r\right]}{a_{7^2}} \right] \qquad \text{Eq. (7-11)}$$

Where,

$$J_1 = Exp(a_1 + \sum_{j=2}^{6} a_j x_j + a_8 x_8) \qquad \text{Eq. (7-12)}$$

$$J_2 = \frac{\tau_H}{H_3}\left[\frac{(\frac{w}{d})_{\max} - \frac{w}{d}}{(\frac{w}{d})_{\max} - 4}\right]\left[\frac{100}{N}\right]^{H_1}\left[\frac{1}{1+\frac{H_2}{2}}\right] \qquad \text{Eq. (7-13)}$$

The time to wear the bit completely is given by:

$$t_h = J_2 \tau_H [(1 + H_2)/2] \qquad \text{Eq. (7-14)}$$

The time to wear the bearing completely is given by:

$$t_b = \tau_B \left[\frac{4d}{w}\right]^b \left[\frac{100}{N}\right] \qquad \text{Eq. (7-15)}$$

The bit life is given by the smallest time values of Eq. (6-14) and Eq. (6-15). If $t_h > t_b$, then the final tooth war is calculated:

$$hf = \sqrt{(1/H_2)(2t_b / H_2 J_2 \tau_H} - (1/H_2) \qquad \text{Eq. (7-16)}$$

That value is then substituted into Eq. (6-11) to calculate the footage. The cost per foot is then calculated by Eq. (6-10).

7.3. Operational Parameters

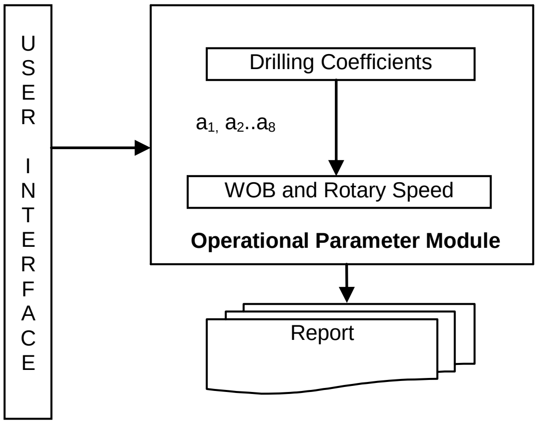
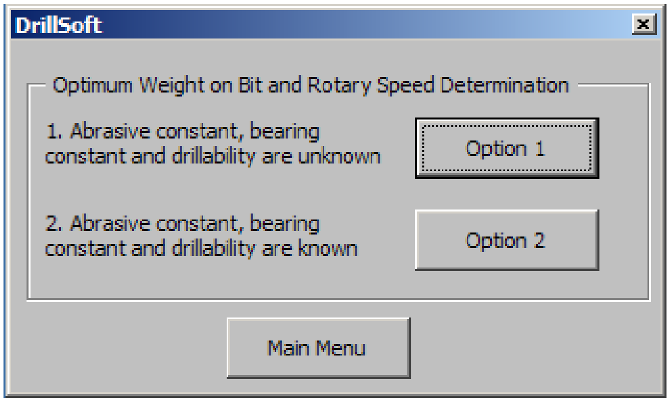


Figure 7-1: Operational parameters

The proposed approach of Operational Parameters determination has two parts as shown in Figure 7-1. In the first part, drilling coefficients are determined by using offset well data and in the second part; these coefficients are used to determine the optimum WOB and RPM.

The program first determines the drilling coefficient and then these values are used in optimum WOB and RPM calculation. The user interface of WOB and RPM determination is shown in Figure 7-2. It provides two options, in option 1- abrasive constant, bearing constant and drillability should be calculated first from the offset bit data, and then these values are used in the rest of the calculations. In this option the list of required input data is shown in Figure 7-3. In option 2, abrasive constant, bearing constant and drillability should be given as input and other required inputs are shown in Figure 8-12.



DrillSoft

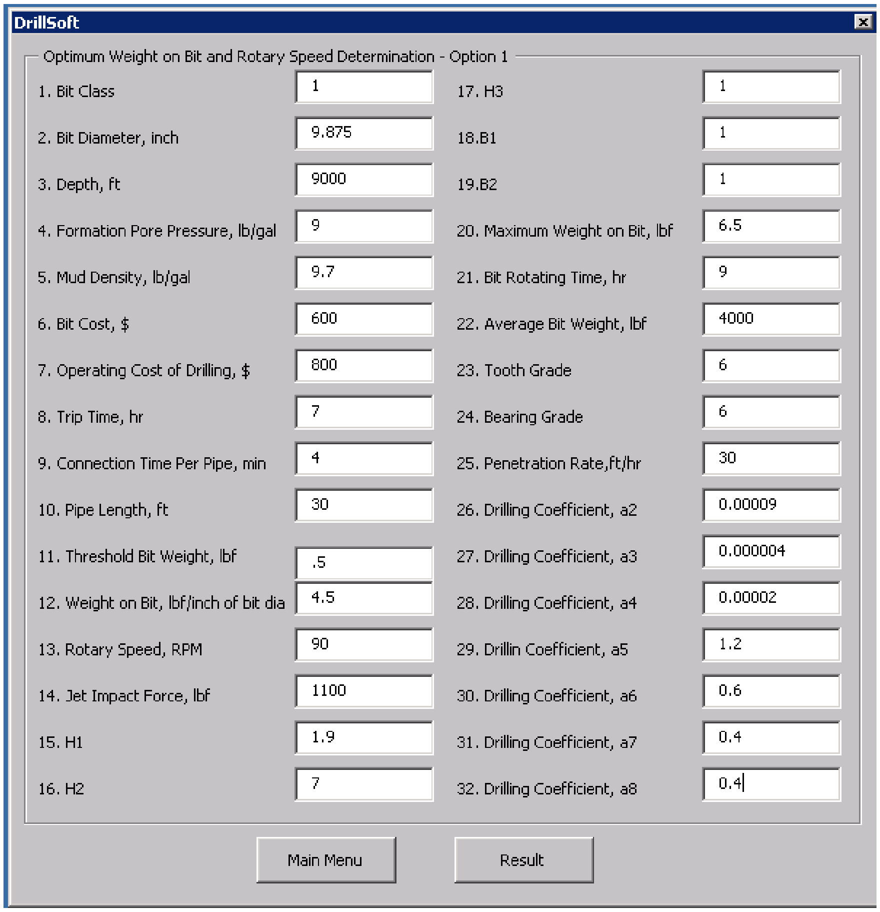
Optimum Weight on Bit and Rotary Speed Determination

1. Abrasive constant, bearing constant and drillability are unknown

2. Abrasive constant, bearing constant and drillability are known

Figure 7-2: User interface for Operational Parameters module

The program also generates six tables these includes – cost per foot, bit life, footage drilled, final tooth wear, final bearing wear, penetration rate. Cost per foot table can be used to quickly identify (Bourgoyne and Young 1974): (1) the best combination of bit weight and rotary speed; (2) the best rotary speed for a given bit weight; and (3) the best bit weight for a given rotary speed.



DrillSoft

Optimum Weight on Bit and Rotary Speed Determination - Option 1

1. Bit Class	1	17. H3	1
2. Bit Diameter, inch	9.875	18.B1	1
3. Depth, ft	9000	19.B2	1
4. Formation Pore Pressure, lb/gal	9	20. Maximum Weight on Bit, lbf	6.5
5. Mud Density, lb/gal	9.7	21. Bit Rotating Time, hr	9
6. Bit Cost, $	600	22. Average Bit Weight, lbf	4000
7. Operating Cost of Drilling, $	800	23. Tooth Grade	6
8. Trip Time, hr	7	24. Bearing Grade	6
9. Connection Time Per Pipe, min	4	25. Penetration Rate,ft/hr	30
10. Pipe Length, ft	30	26. Drilling Coefficient, a2	0.00009
11. Threshold Bit Weight, lbf	.5	27. Drilling Coefficient, a3	0.000004
12. Weight on Bit, lbf/inch of bit dia	4.5	28. Drilling Coefficient, a4	0.00002
13. Rotary Speed, RPM	90	29. Drillin Coefficient, a5	1.2
14. Jet Impact Force, lbf	1100	30. Drilling Coefficient, a6	0.6
15. H1	1.9	31. Drilling Coefficient, a7	0.4
16. H2	7	32. Drilling Coefficient, a8	0.4

Figure 7-3: Operational Parameters user interface option 1

7.4. Summary

The Operational Parameters module is discussed in this chapter. The system can predict the drilling coefficients by using offset well data and determines the optimum weight on bit, and the drill string rotation that minimizes drilling cost per foot for a single bit run.

Chapter 8 Case Study

8.1. Introduction

The validation of the various modules of the proposed system was based on previously published data. In this chapter, several case studies published in different literature have been considered and compared with the result generated by the proposed system. Step by step process of casing setting depth and sizes, casing selection and drill string design have been described with cases. Drilling coefficients values are compared with published results.

8.2. Case Study for Casing Setting Depth and Size Determination

Casing setting depth and size of a well is to be designed. The inputs are provided at the Table 8-1.

Table 8-1: Casing setting depth and size input (Rabia 1985)

Input Type	Input Value	Unit
True vertical depth, TVD	11,000	ft
Rock poisson’s ratio, υ	0.4	
Over burden, σ_v	1	psi/ft
Trip margin	0.06	specific gravity
Kick margin	0	specific gravity
Minimum depth of surface section	3000	ft
Number of pore pressure input	8	

In order to calculate the casing and hole sizes, formation type and production casing size are required as input. These two values are not given in the case study.

So, assumptions have been made, formation type: hard and production casing size: 6.625 inch.

The system first estimates the fracture pressure and determines the MSG (*Pore Pressure+ Trip Margin*) and FSG (*Fracture Pressure-Kick Margin*) based on the input provided, Table 8-2 shows the estimated value.

Table 8-2: Fracture pressure, FSG and MSG estimation

Input Number	Depth (ft)	Pore Pressure (psi)	Pore Pressure (SG)	Mud Density, MSG (SG)	Fracture Pressure (SG)	FSG (SG)
Input Value (Rabia 1985)			**Calculated Value**			
1	3000	1320	1.01	1.076	1.878	1.878
2	5000	2450	1.131	1.191	1.916	1.916
3	8300	4067	1.131	1.191	1.916	1.916
4	8500	4504	1.223	1.283	1.947	1.947
5	9000	5984	1.535	1.595	2.051	2.051
6	9500	6810	1.655	1.715	2.091	2.091
7	10000	7800	1.801	1.860	2.139	2.139
8	11000	10171	2.135	2.195	2.251	2.251

Mud density at the true vertical depth (TVD), i.e. 11 000 ft, is MSG_1 = 2.195. The depth at which MSG_1=FSG_1 can be found by linear interpolation. In this program it is assumed that the connecting line between the two neighboring points is linear. So, the program now determines the depth at which FSG_1=2.195. The two neighboring points of FSG_1 are, FSG_2= 2.250 and FSG_3 = 2.139 and corresponding depths are $Depth_2$ =11 000 ft and $Depth_3$ = 10 000 ft respectively.

The equation for linear interpolation is as follows:

$$Depth_1 = \frac{(FSG_1 - FSG_2)(Depth_3 - Depth_2)}{FSG_3 - FSG_2} + Depth_2 \qquad \text{Eq. (8.1)}$$

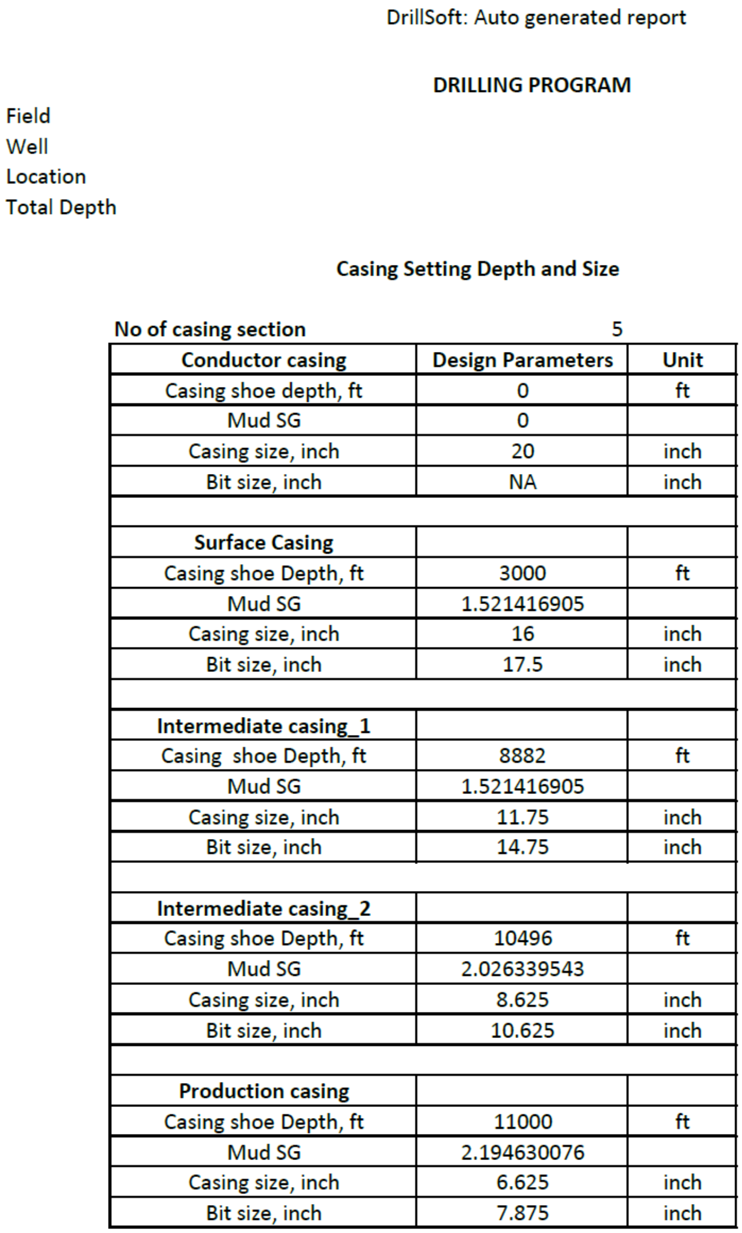
$$Detph_1 = 10496\ ft$$

So, the next casing setting depth should be at 10 496ft. Now it is required to determine the mud density above 10 496ft. Again the program uses linear interpolation and the following equation should be used:

$$MSG_1 = \frac{(MSG_3 - MSG_2)(Depth_1 - Depth_2)}{Depth_3 - Depth_2} + MSG_2 \qquad \text{Eq. (8.2)}$$

Here, $Depth_1$ =10 496ft, $Depth_2$ =11 000ft $Depth_3$ =10 000ft and MSG_2=2.195 MSG_3= 1.861. So, MSG_1= 2.026. These processes continue until the mud density becomes smaller than the minimum value of the fracture specific gravity or the depth becomes smaller than the minimum surface casing depth. The results generated by DrillSoft have been shown in Figure 8-1. These results are closely matched with the published result as presented in Table 8-3. Note that the report can be configured via programming, e.g. more intermediate casing stages can be accommodated.

The parametric 3D model of different casing sections depend on casing setting depth’s output, this output now converted into expression files and used for 3D model generation. Figure 8-2 shows the partial view of the well schematic.



DrillSoft: Auto generated report

DRILLING PROGRAM

Field
Well
Location
Total Depth

Casing Setting Depth and Size

No of casing section 5

Conductor casing	Design Parameters	Unit
Casing shoe depth, ft	0	ft
Mud SG	0	
Casing size, inch	20	inch
Bit size, inch	NA	inch
Surface Casing		
Casing shoe Depth, ft	3000	ft
Mud SG	1.521416905	
Casing size, inch	16	inch
Bit size, inch	17.5	inch
Intermediate casing_1		
Casing shoe Depth, ft	8882	ft
Mud SG	1.521416905	
Casing size, inch	11.75	inch
Bit size, inch	14.75	inch
Intermediate casing_2		
Casing shoe Depth, ft	10496	ft
Mud SG	2.026339543	
Casing size, inch	8.625	inch
Bit size, inch	10.625	inch
Production casing		
Casing shoe Depth, ft	11000	ft
Mud SG	2.194630076	
Casing size, inch	6.625	inch
Bit size, inch	7.875	inch

Figure 8-1: Casing setting depth and size

Table 8-3: Published result for casing setting depth (Rabia 1985)

	Surface Casing	Intermediate 1 Casing	Intermediate 2 Casing	Production Casing
Depth (ft)	3000	8850	10500	11000
Mud Specific Gravity	1.567	1.567	2.031	2.170

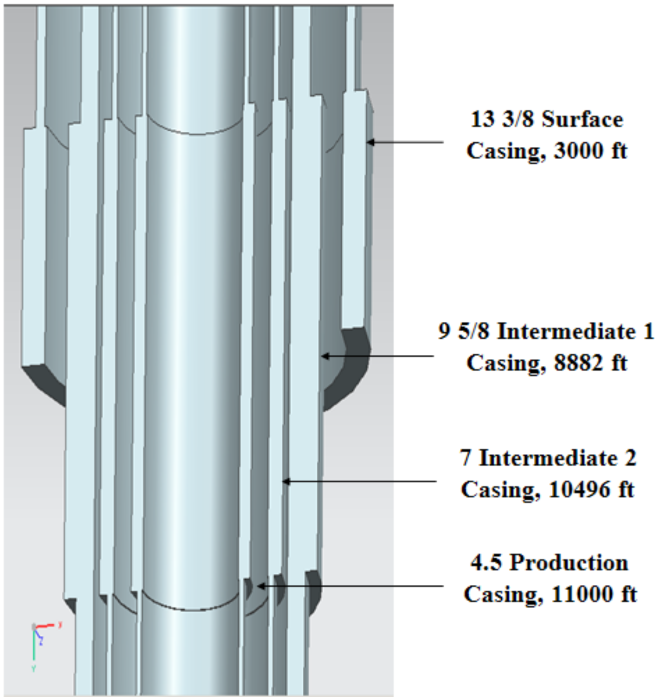
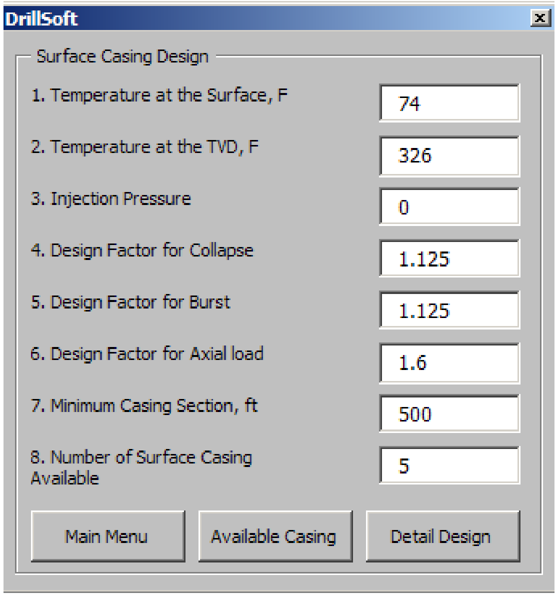


Figure 8-2: Partial cutaway view of different casing sections

8.3. Case Study for Casing Selection

A surface casing is to be designed. The case is taken from (Byrom 2007). The necessary inputs are, Depth= 3000ft; Mud density= 1.11; Casing size= 13 3/8. Figure 8-3 shows the user interface of surface casing design filled with input. It should be noted that the option for depth and mud density are not provided in the user interface. It is due to the fact that, the different parts of the software are integrated to each other. This integration helps the program to retrieve necessary information from the system data base. In this particular case, the program automatically retrieves the depth and mud density value from the casing setting depth database.



Surface Casing Design

1. Temperature at the Surface, F: 74
2. Temperature at the TVD, F: 326
3. Injection Pressure: 0
4. Design Factor for Collapse: 1.125
5. Design Factor for Burst: 1.125
6. Design Factor for Axial load: 1.6
7. Minimum Casing Section, ft: 500
8. Number of Surface Casing Available: 5

Figure 8-3: Surface casing design user interface

The user should provide the specifications of the available casing. The system takes casing specifications as a text file format and retrieves the required information from the text file. An example file with the available surface casing specifications is tabulated in Table 8-4.

Table 8-4: Specification and priority sequence of available casing

Casing number	OD(inch)	ID(inch)	Weight (Kg/ft)	Grade	Connection
1	16	12.615	54.5	K-55	ST&C
2	16	12.515	61	K-55	ST&C
3	16	12.415	68	K-55	ST&C
4	16	12.415	68	N-80	ST&C
5	16	12.347	72	N-80	ST&C

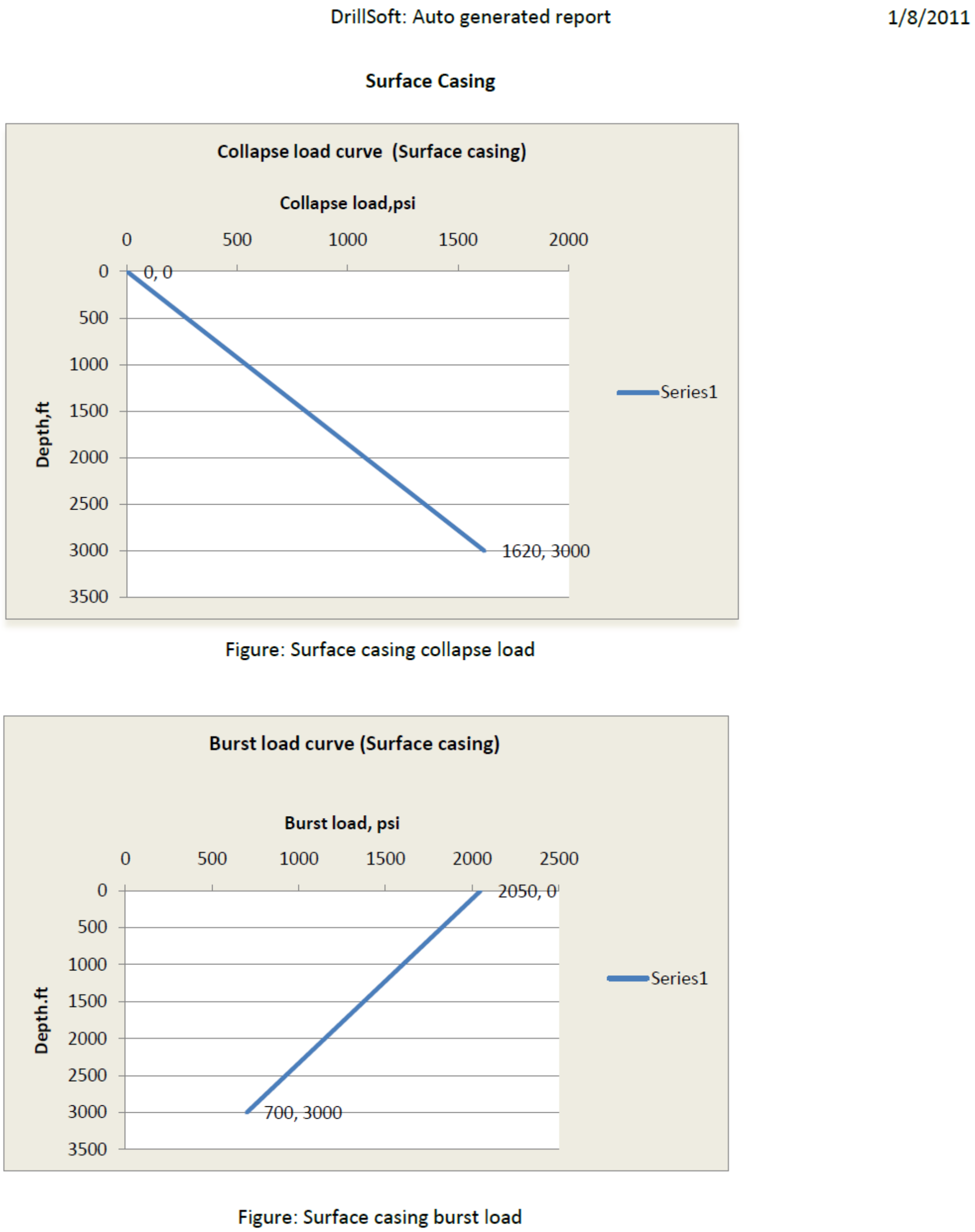
After getting the input, the system calculates the collapse and burst rating at the surface and casing shoe. In this particular case the design collapse pressure at surface and shoe are 0 and 1620 psi, Design burst pressure at surface and shoe are 2180 and 750 psi. Now the program determines the break points and potential candidates that satisfy both the collapse and burst rating. The break points and potential candidates are shown in Table 8.5.

Table 8-5: Casing selection break points and potential candidates

No	Break Points (ft)	Potential Candidates
1	0	1,2,3,4,5
2	2092	2,3,4,5
3	2851	3,4,5
4	3000	3,4,5

According to the algorithm, the first break point (0 ft) contains all 5 available casing as potential candidates. As available casings were listed according to the priority of the users, it assumes that, the first casing is more economical and then the next one and so on. The system selects the 1st candidate, i.e. the number 1 casing with grade K-55 and weight 54.5, and adds minimum casing section (500ft). Now the program checks the total depth has not achieved yet so it selects again the number 1 casing from the potential candidates. As it is similar to the previous casing so this time it will add a minimum value of (Minimum casing section or (Next break point-Casing covered)), i.e. Min (500 ft, (2092-500=1592 ft)). It is worked out that the minimum casing section (500ft) is the minimum between these two values. So another 500 ft will add to the previous casing. Now its length becomes 1000ft. As the closest break point is at 2092ft, this process continues until it reaches to 2000ft. At this point the system worked out that, number 1 casing is still a potential candidate and it is similar to the previous casing. So, minimum value (500, 2092-2000=92) = 92ft. Based on these conditions, the system selects casing 1 and adds a length of 92ft with the exiting length. So the total length of casing covered is 2092 ft. As the desired depth is 3000ft and it is not achieved yet the system checks the available potential candidates and selects the number 2 casing and adds a length of minimum casing section 500 ft with the previous length. The new length becomes 2592ft still less than the total depth. The system again selects number 2 casing and worked out the minimum value between (500, 2852-2592 = 260ft) = 260ft. So after adding this value the length becomes 2852ft. Another 148ft is required to complete the casing string for surface section. But this remaining section is less than the minimum casing section length and previously used casing (Number 1 and 2) are not allowed to use the full depth. Another casing type should be selected. But if the system selects a new casing type it will not satisfy the minimum casing section length. To solve this problem a re-evaluation of the design is required. The system re-evaluate the design and concluded that, instead of using number 2 casing from 2093ft to 2852ft, casing 3 should be used to the total depth. In that case, all the design criteria are satisfied. So the preliminary design based on collapse and burst is completed.

The next step is to check whether the designed casing string satisfy the axial load criteria or not. It is found that the designed casing string has axial load safety factors of 3.11 and 11.62 for number 1 and number 3 casing respectively. This concludes the surface casing conceptual design. Figures 8-4 and 8-5 provide the casing design report, generated by the program. Table 8-6 shows published result for surface casing selection. It is observed that the proposed system selects the similar types of casing as those selected in the published work by Byrom (2007).



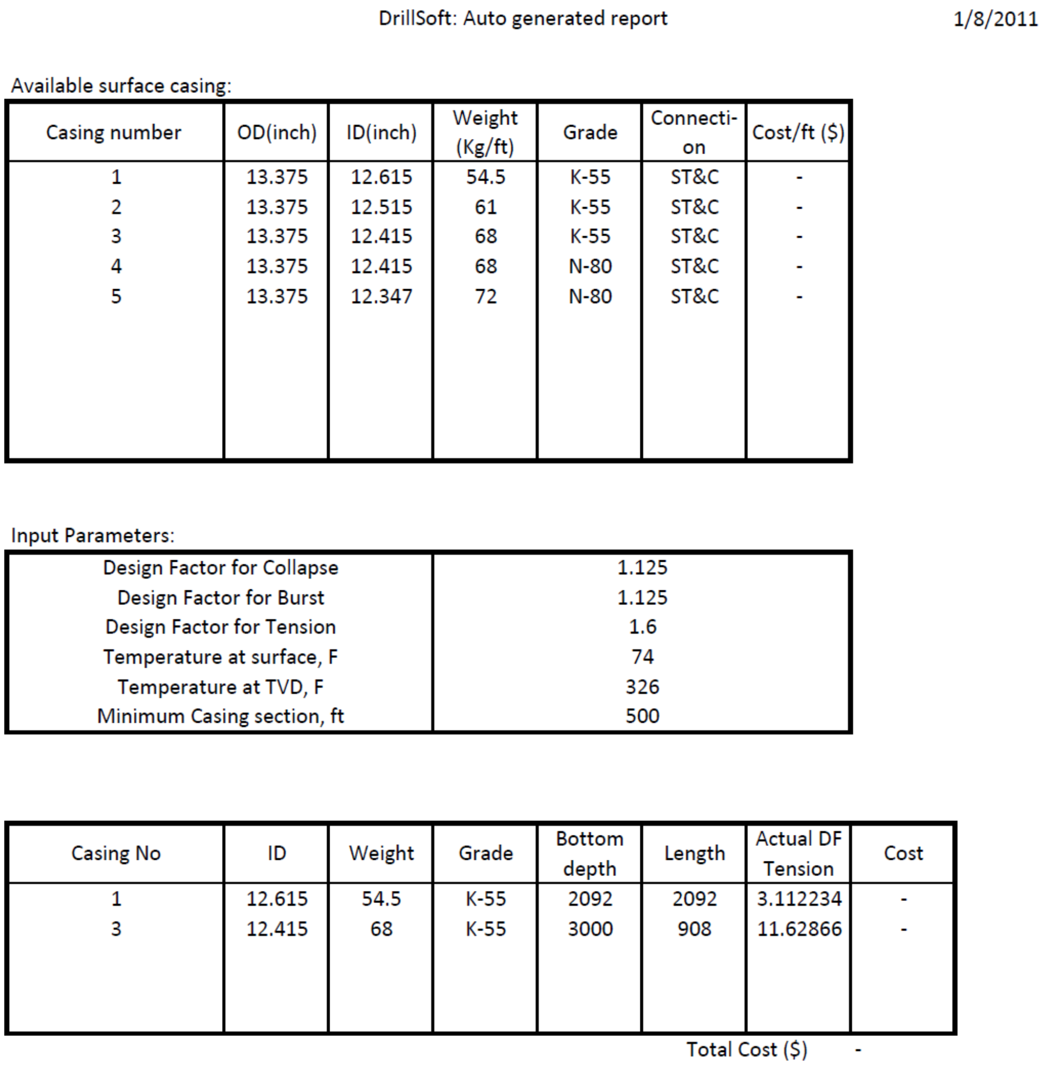
DrillSoft: Auto generated report 1/8/2011

Surface Casing

Figure: Surface casing collapse load

Figure: Surface casing burst load

Figure 8-4: Program generated report (Collapse and Burst load)



DrillSoft: Auto generated report 1/8/2011

Available surface casing:

Casing number	OD(inch)	ID(inch)	Weight (Kg/ft)	Grade	Connecti-on	Cost/ft ($)
1	13.375	12.615	54.5	K-55	ST&C	-
2	13.375	12.515	61	K-55	ST&C	-
3	13.375	12.415	68	K-55	ST&C	-
4	13.375	12.415	68	N-80	ST&C	-
5	13.375	12.347	72	N-80	ST&C	-

Input Parameters:

Design Factor for Collapse	1.125
Design Factor for Burst	1.125
Design Factor for Tension	1.6
Temperature at surface, F	74
Temperature at TVD, F	326
Minimum Casing section, ft	500

Casing No	ID	Weight	Grade	Bottom depth	Length	Actual DF Tension	Cost
1	12.615	54.5	K-55	2092	2092	3.112234	-
3	12.415	68	K-55	3000	908	11.62866	-

Total Cost ($) -

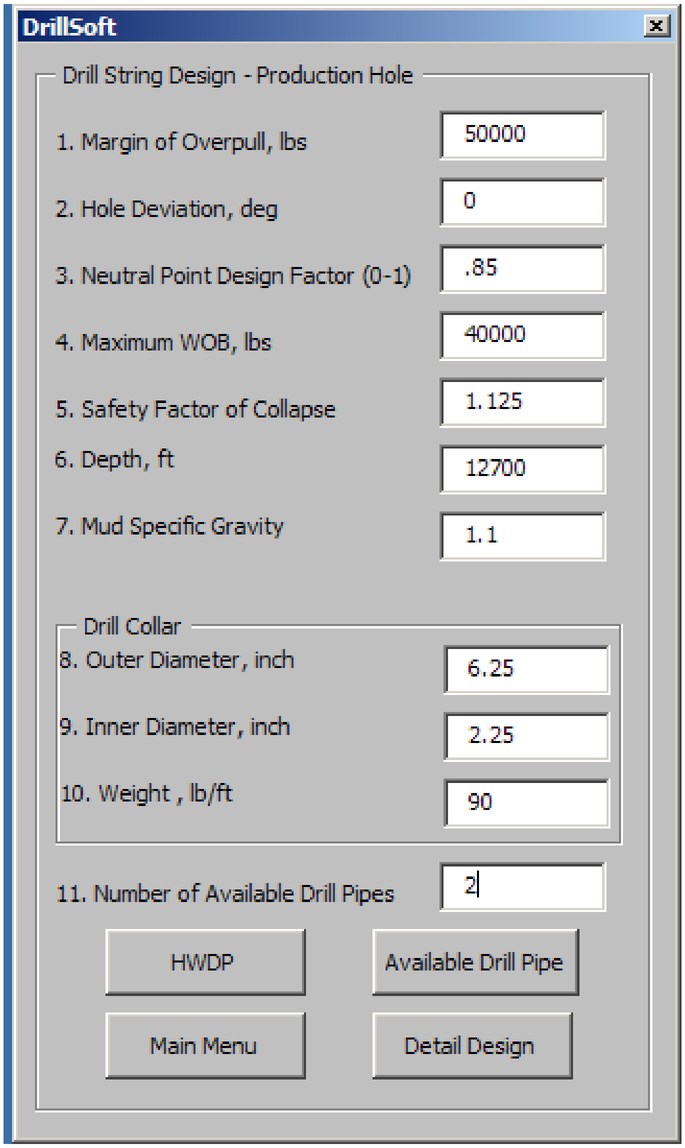
Figure 8-5: Program generated report for surface casing design

Table 8-6: 3/8" Surface casing design published result (Byrom 2007)

Casing No.	ID	Weight	Grade	Conn.	Bottom depth	Length	Tension
1	12.615	54.5	K-55	ST&C	2100	2100	3.11
3	12.415	68	K-55	ST&C	3000	900	10.38

8.4. Case Study for Drill String Design

A drill string is to be designed. The case is taken from API standard handbook (API RP 7G, 1998). Figure 8-6 shows the user interface of drill string module filled with input. Table 8-7 shows two drill pipe types available in the inventory.



Drill String Design - Production Hole

1. Margin of Overpull, lbs: 50000
2. Hole Deviation, deg: 0
3. Neutral Point Design Factor (0-1): .85
4. Maximum WOB, lbs: 40000
5. Safety Factor of Collapse: 1.125
6. Depth, ft: 12700
7. Mud Specific Gravity: 1.1

Drill Collar

8. Outer Diameter, inch: 6.25
9. Inner Diameter, inch: 2.25
10. Weight , lb/ft: 90

11. Number of Available Drill Pipes: 2

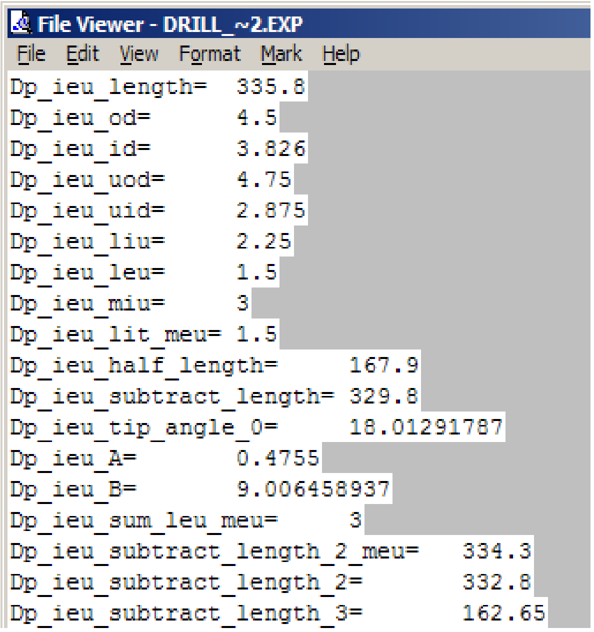
Figure 8-6: Drill string design user interface

Table 8-7: Conceptual design parameters for a drill string

Drill String Components	**Calculated Length, (ft)**	**No. of array**	**Length, (ft) (API RP 7G , 1998)**
Drill Collar: 6 ¼"OD X 2 ¼"ID	630	21	630
Drill Pipe Type 1: 4 ½" X 16.6lb, Grade E75, Class2	6750	225	6759
Drill Pipe Type 2: 4 1/2 " X 16.6lb,Grade X95, Premium Class	5320	178	5311

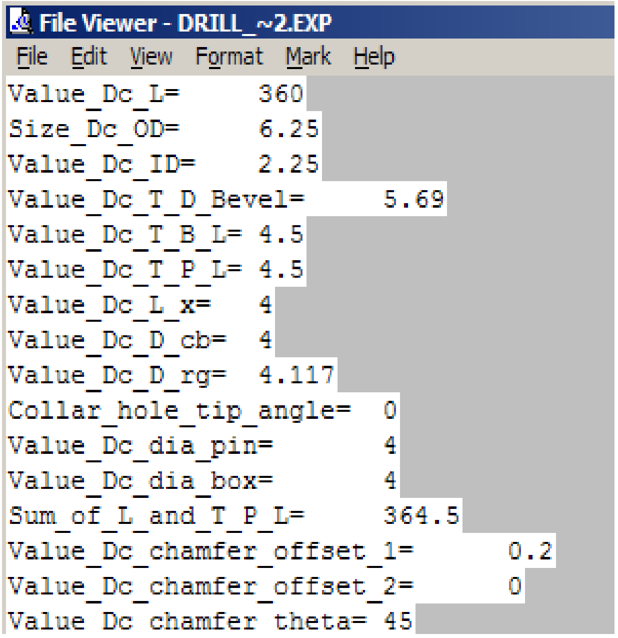
Based on the operational input the rule based system designed the drill string that uses two different types of drill pipes. As mentioned earlier, the program first considers the most economic drill pipe type among the available two; it first chooses grade E75 and determines the safe length of 6750 ft. After checking the length of 21 drill collars and the drill string developed has reached a length of 7380 ft, which is less than required depth of 12700 ft. So the program considers the second pipe type, grade X95 and determines to use the type for rest of 5320 ft. The conceptual design of the drill string and API standard hand book result are presented on Table 8-7. When considering the collapse loading the program generates messages for the users; in this case it is worked out as 10267 ft and drill string should not be run dry below this depth otherwise it may cause damage in the string.

Based on this conceptual design, the system generates the necessary configuration and specification files and converts these files into expression files. The expression files for drill pipe, drill collar, and array have been shown in Figure 8-7, 8-8 and 8-9 respectively. The 3D model of the drill string assembly is presented in Figure 8- 10.



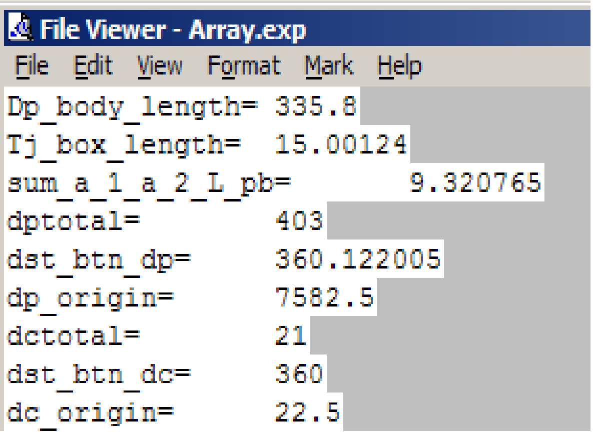
```
Dp_ieu_length=  335.8
Dp_ieu_od=      4.5
Dp_ieu_id=      3.826
Dp_ieu_uod=     4.75
Dp_ieu_uid=     2.875
Dp_ieu_liu=     2.25
Dp_ieu_leu=     1.5
Dp_ieu_miu=     3
Dp_ieu_lit_meu= 1.5
Dp_ieu_half_length=     167.9
Dp_ieu_subtract_length= 329.8
Dp_ieu_tip_angle_0=     18.01291787
Dp_ieu_A=       0.4755
Dp_ieu_B=       9.006458937
Dp_ieu_sum_leu_meu=     3
Dp_ieu_subtract_length_2_meu=   334.3
Dp_ieu_subtract_length_2=       332.8
Dp_ieu_subtract_length_3=       162.65
```

Figure 8-7: Expression file for drill pipe



```
Value_Dc_L=     360
Size_Dc_OD=     6.25
Value_Dc_ID=    2.25
Value_Dc_T_D_Bevel=     5.69
Value_Dc_T_B_L= 4.5
Value_Dc_T_P_L= 4.5
Value_Dc_L_x=   4
Value_Dc_D_cb=  4
Value_Dc_D_rg=  4.117
Collar_hole_tip_angle=  0
Value_Dc_dia_pin=       4
Value_Dc_dia_box=       4
Sum_of_L_and_T_P_L=     364.5
Value_Dc_chamfer_offset_1=      0.2
Value_Dc_chamfer_offset_2=      0
Value_Dc_chamfer_theta= 45
```

Figure 8-8: Expression file for drill collar



```
Dp_body_length= 335.8
Tj_box_length=  15.00124
sum_a_1_a_2_L_pb=       9.320765
dptotal=        403
dst_btn_dp=     360.122005
dp_origin=      7582.5
dctotal=        21
dst_btn_dc=     360
dc_origin=      22.5
```

Figure 8-9: Expression file for array of components

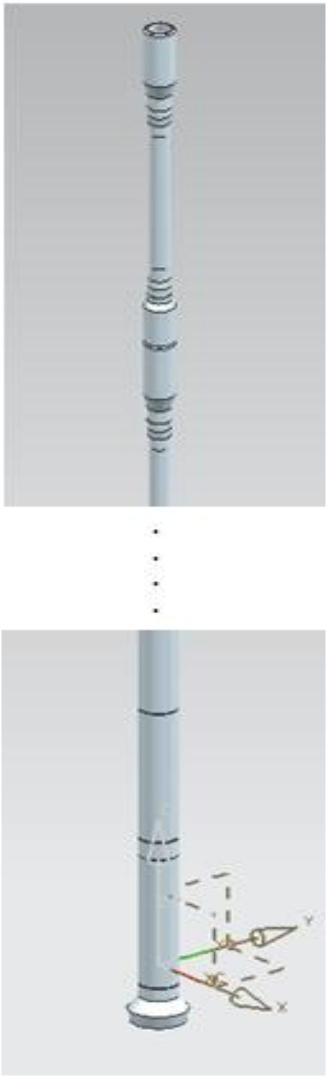


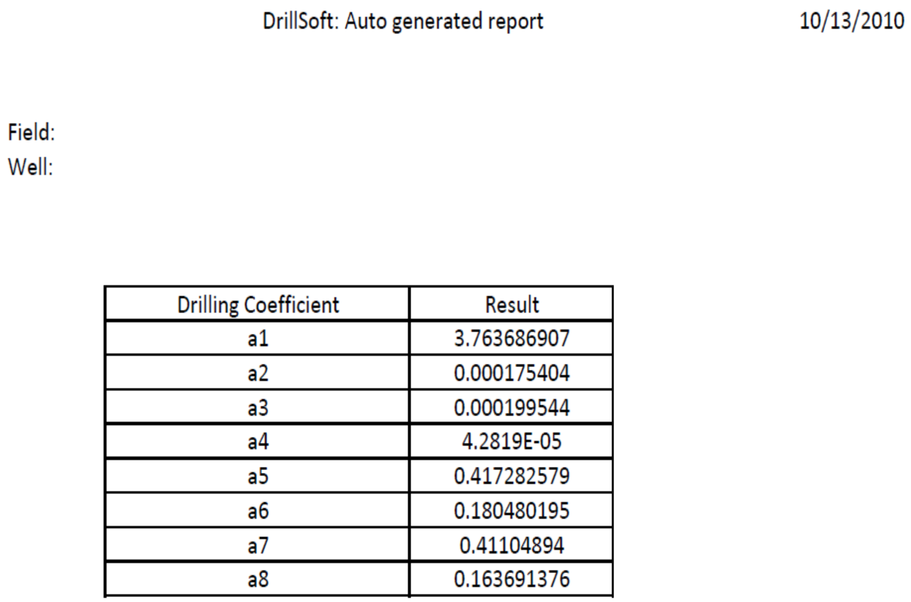
Figure 8-10: Drill string assembly

8.5. Case Study for Operational Parameters

Table 8-8 presents the comparison of the published result with the system generated result of drilling coefficients. Figure 8-12 shows the input for Operational Parameters and Figure 8-13 shows the system generated result.

Table 8-8: Comparison of drilling coefficients

Drilling Coefficients	**Calculated Result**	**Published Result (Bourgoyne et al. 1974)**
a_1	3.76368	3.78
a_2	0.1754 10^{-3}	0.17 10^{-3}
a_3	0.1995 10^{-3}	0.20 10^{-3}
a_4	0.4281 10^{-4}	0.43 10^{-4}
a_5	0.41728	.43
a_6	0.1804	0.21
a_7	0.411048	0.41
a_8	0.16369	0.16

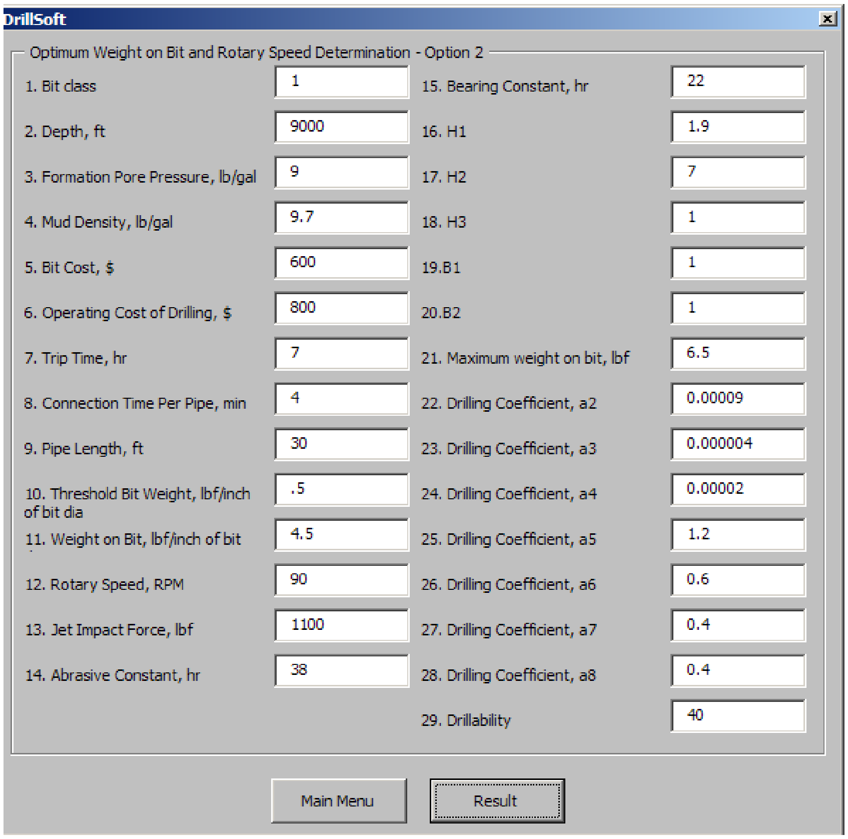


DrillSoft: Auto generated report 10/13/2010

Field:
Well:

Drilling Coefficient	Result
a1	3.763686907
a2	0.000175404
a3	0.000199544
a4	4.2819E-05
a5	0.417282579
a6	0.180480195
a7	0.41104894
a8	0.163691376

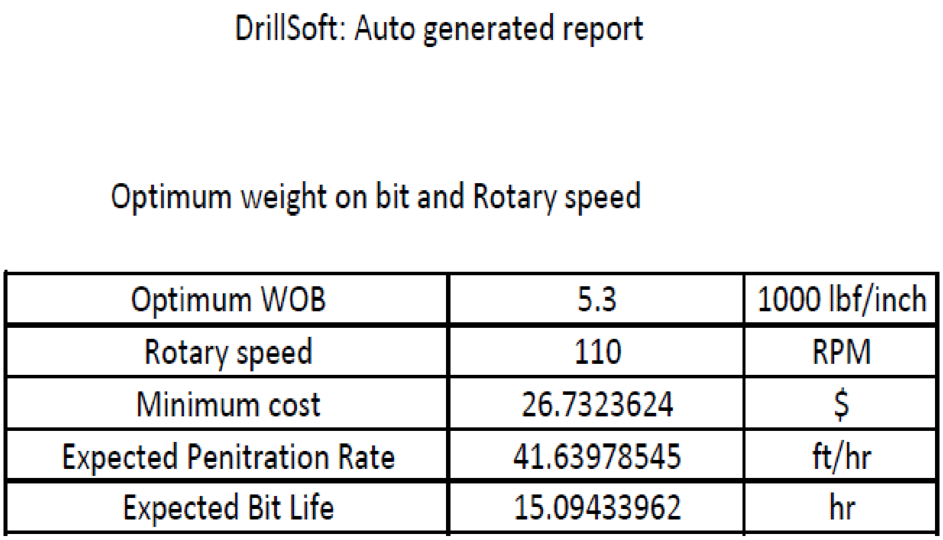
Figure 8-11: System generated output for drilling coefficient



Optimum Weight on Bit and Rotary Speed Determination - Option 2

1. Bit class	1	15. Bearing Constant, hr	22
2. Depth, ft	9000	16. H1	1.9
3. Formation Pore Pressure, lb/gal	9	17. H2	7
4. Mud Density, lb/gal	9.7	18. H3	1
5. Bit Cost, $	600	19.B1	1
6. Operating Cost of Drilling, $	800	20.B2	1
7. Trip Time, hr	7	21. Maximum weight on bit, lbf	6.5
8. Connection Time Per Pipe, min	4	22. Drilling Coefficient, a2	0.00009
9. Pipe Length, ft	30	23. Drilling Coefficient, a3	0.000004
10. Threshold Bit Weight, lbf/inch of bit dia	.5	24. Drilling Coefficient, a4	0.00002
11. Weight on Bit, lbf/inch of bit	4.5	25. Drilling Coefficient, a5	1.2
12. Rotary Speed, RPM	90	26. Drilling Coefficient, a6	0.6
13. Jet Impact Force, lbf	1100	27. Drilling Coefficient, a7	0.4
14. Abrasive Constant, hr	38	28. Drilling Coefficient, a8	0.4
		29. Drillability	40

Figure 8-12: Operational Parameters determination user input



DrillSoft: Auto generated report

Optimum weight on bit and Rotary speed

Optimum WOB	5.3	1000 lbf/inch
Rotary speed	110	RPM
Minimum cost	26.7323624	$
Expected Penitration Rate	41.63978545	ft/hr
Expected Bit Life	15.09433962	hr

Figure 8-13: System generated report for optimum WOB and RPM

8.6. Summary

Published cases are compared with proposed system generated result. From these comparisons it is concluded that, the result provided by the proposed program is quite satisfactory.

Chapter 9 Conclusions and Future Work

9.1. Conclusions

The research work has pioneered a proposed integrated approach for well drilling planning with case studies. A prototype system that integrates three important well drilling planning stages, i.e. casing design, drill string design and operational optimization, has been developed. The system generated results for casing setting depth, casing sizes, casing selections are tested with published result and the results are promising.

A comprehensive approach for casing design has been proposed that combines the conceptual and detailed design. The developed prototype produces the casing setting depth, casing size and selects the economical casing string from the available inventory. The casing setting depth and casing size output then used to generate the 3D model of the well configuration. Although at present the proposed system is capable of handling two intermediate casings only but it is possible to calculate more intermediate casings by making some customization in the system knowledge base and user interfaces.

The potential application of knowledge driven CAD in drilling industry is explored and applied successfully in the drill string design. A parametric and smart oil well drill string modeling CAD tool has been prototyped that enables generation of 3D models with built-in engineering rules, constraints and controls on different application cases with the changing situations throughout each well-drilling lifecycle. A prototype of common part database has been integrated into the system so that standard parts can be reused from a well defined library. This software tool can help the drilling engineer to interactively design and model the drill string. The drill string design module is capable of handling different configurations as well as different topologies of components. As the design and CAD modeling is integrated with each other, any changes in the design will be reflected in the CAD model.

An Operational Parameters module is added with system, this can be useful tool for prediction of optimum weight on bit and rotary speed during drilling.

The similar concept can be applicable to integrate other planning modules such as hydraulic program, bit program, time and cost estimation, etc., to a single system. Such an integrated system will be a very useful decision making tool to the drilling companies.

9.2. Limitation and Future Work

The developed prototype determines the casing setting depth based on formation pore pressure and fracture pressure, this process does not always guarantee well bore stability. Another limitation of the system is that it does not consider the combined load effect on the casing during casing selection.

At this moment, the reported software tool can only handle vertical oil well and analysis part of the proposed model is not completed. More research work should be carried out to develop a generic model which can equally applicable to horizontal, extended reach and multilateral wells.

Bibliography

Al-Yami, A.S., Schubert, J., Medina-Cetina, Z., and Yu, O.-Y. 2010. Drilling Expert System for the Optimal Design and Execution of Successful Cementing Practices. Paper SPE 135183-MS presented at the IADC/SPE Asia Pacific Drilling Technology Conference and Exhibition, Ho Chi Minh City, Vietnam, 1-3 November. doi: 10.2118/135183-MS

Akpan, H.O. 2005. Efficient Computational Method for Casing String Design. Paper SPE 98790-MS presented at the Annual SPE International Technical Conference and Exhibition, Abuja, Nigeria, 1-3 August. doi: 10.2118/98790-MS.

Au, C. K., and Yuen, M. M. F. 2000. A semantic feature language for sculptured object modelling. Computer-Aided Design 32(1) : 63-74. doi:10.1016/S0010-4485(99)00085-8.

Andrews P. T. J., Shahin T. M. M., and Sivaloganathan S. 1999. Design Reuse in a CAD Environment-Four Case Studies. Computer and Industrial Engineering 37(1999): 105-109. doi:10.1016/S0360-8352(99)00033-9.

API RP 7G, Recommended Practice for Drill Stem Design and Operating Limits, 16th Edition. 1998. Washington, DC: API.

Austin, A.Z.1993. Benefits of 3D Visualization to Reservoir Simulation. Paper SPE 26113-MS presented at the SPE Western Regional Meeting, Anchorage, Alaska, USA, 26-28 May. doi: 10.2118/26113-MS

Byrom, T.G. 2007. Casing and Liners for Drilling and Completion. Houston, Texas: Gulf Publishing Company.

Bednarz S. 2004. Disgn and Exploitation Problems of Drill String in Directional Drilling. Acta Motanistica Slovaca 9 (3): 152-155.

Bjornsson, E., Hucik, B., Szutak, G., Brown, L.A., Evans, H., Curry, D., and Perry, P. 2004. Drilling Optimization Using Bit Selection Expert System and ROP Prediction Algorithm Improves Drilling Performance and Enhances Operational Decision Making by Reducing Performance Uncertainties. Paper SPE 90752-MS presented at the SPE Annual Technical Conference and Exhibition, Houston, Texas, 26-29 September. doi: 10.2118/90752-MS.

Batanov, D. N., and A. K. Lekova. 1993. Data and knowledge integration through the feature-based approach. Artificial Intelligence in Engineering 8 (1): 77-83. doi:10.1016/0954-1810(93)90033-C.

Bourgoyne, A. T. Jr., Millhein, K. K., Chenevert, M. E., and Young, F. S. Jr. 1991. Applied Drilling Engineering. Richardson, Texas: Society of Petroleum Engineers Textbook Series.

Bourgoyne, A. T. Jr., and Young, F. S. Jr. 1974. A Multiple Regression Approach to Optimal Drilling and Abnormal Pressure Detection. SPE Journal 14(4): 371-384, Trans. AIME, SPE-4238-PA. doi: 10.2118/4238-PA.

Chu, C. H., Song, M. C., Luo, C. S. 2006. Computer-aided parametric design for 3D tire mold production. Computers in Industry 57 (1): 11-25 doi:10.1016/j.compind.2005.04.005:

Cunha, J.C. 2002. Drill-String and Casing Design for Horizontal and Extended Reach Wells Part 1. Paper SPE 79001-MS presented at the SPE International Thermal Operations and Heavy Oil Symposium and International Horizontal Well technology Conference, Calgary, Canada, 4-7 November. doi: 10.2118/79001-MS

Cunha, R. R. M., and Dias, A. 2002. A Feature-Based Database Evolution Approach in the Design Process. Robotics and Computer-Integrated Manufacturing 18(3-4):275-281. doi:10.1016/S0736-5845(02)00018-2.

Chan, K. C., and Nhieu, J. 1993. A framework for feature-based applications. Computers & Industrial Engineering 24 (2): 151-164. doi:10.1016/0360-8352(93)90004-H

Chiu, T.J., and Caudel, F.L.W. 1993. Development of an Expert System to Assist With Complex Fluid Design. Paper SPE 24416-PA presented at the SPE Computer Application, Houston, USA, 19-22 July. doi: 10.2118/24416-PA.

Cunningham, J. J., and Dixon, J. R. 1988. Designing with features: The origin of features. Presented at the Computers in Engineering Conference, San Francisco, CA, USA, 31 July.

Devereux, S. 1998. Practical Well Planning and Drilling Manual. Tulsa, Oklahama: Penwell Publishing Company.

Dupriest, F. E., and Koederitz, W. 2005. Maximizing Drill Rates with Real-Time Surveillance of Mechanical Specific Energy. Paper IADC/SPE 92914-MS presented at the SPE/IADC Drilling Conference, Dallas, Texas, USA, 9-12 February. doi: 10.2118/92194-MS.

Eren, T., and Ozbayoglu, M. E. 2010. Real Time Optimization of Drilling Parameters During Drilling Operations. Paper SPE 129126-MS presented at the SPE Oil and Gas India Conference and Exhibition, Mumbai, India, 20-22 January 2010, doi: 10.2118/129126-MS.

Eaton, B. A. 1969. Fracture gradient prediction and its application in oilfield operations. Journal of Petroleum Technology 21 (10): 1353–1360. SPE-2163-PA. doi: 10.2118/2163-PA.

Fear, M.J., Meany, N.C., and Evans, J.M. 1994. An Expert System for Drill Bit Selection. Paper SPE 27470-MS presented at the IADC/SPE Drilling Conference, Dallas, Texas, 15-18 February. doi: 10.2118/27470-MS.

Galle, E. M., and Woods, H. B. 1963. Best Constant Weight and Rotary Speed for Rotary Rock Bits. Drilling and Production Practice, API (1963): 48-73.

Huifen, W., Youliang, Z., and Jian, C. 2003. Feature-based collaborative design. Journal of Materials Processing Technology 139 (1-3): 613-618. doi:10.1016/S0924-0136(03)00502-8.

Halal, A. S., D. J. Warling, and R. R. Wagner. 1996. Minimum Cost Casing Design. Paper SPE 36448-MS presented at the SPE Annual Technical Conference and Exhibition, Denver, CO, USA, 6-9 October. doi: 10.2118/36448-MS

Heinz, L.R. 1993. CHES – Casing Hydraulic Expert System. SPE Computer Applications 4(2): 26-31. SPE-24420-PA. doi: 10.2118/24420-PA

Hill, T.H., Guild, G.J., Summers, M.A., T.H. Hill Assocs. 1993. Design Extended Reach Wells. SPE Drilling & Completion 11 (2): 111-117, SPE-29349-PA. doi: 10.2118/29349-PA

Hayes-Roth, F. 1987. Part 1: Expert Systems Applied to the Petroleum Industry Upstream Portion. Paper SPE 22411 presented at the 12th World Petroleum Congress, Houston, USA, April 26 - May 1.

Jellison, M.J. and Klementich, E.F. 1990. An Expert System for Casing String Design. Paper SPE 20328-MS presented at the fifth SPE Petroleum Computer Conference, Denver, Colorado, USA, 25-28 June. doi: 10.2118/20328-MS.

Kessler, F. and Smith, J. E. 2001. Component Balanced 4 – inch Drillpipe for 6 – inch Holes. Available at: http://www.nov.com/GrantPrideco/Drilling/news_and_articles/componentbalanc.pdf

Koo, D.-Y., and Han, S.-H. 1998. An object-oriented configuration design method for paper feeding mechanisms. Expert Systems with Applications 14 (3): 283-289. doi:10.1016/S0957-4174(97)00081-X.

Kim, C., and O'Grady, P. J. 1996. A representation formalism for feature-based design. Computer-Aided Design, 28 (6-7): 451-460. doi:10.1016/0010-4485(95)00042-9.

Kasravi, K. 1994. Understanding knowledge-based CAD/CAM. Computer-Aided Engineering 13(10): 72-78.

Kappert, J. H., Houten, F. J. A. M. V., and Kals, H. J. J. 1993. The Application of Features in Airframe Component Design and Manufacturing. CIRP Annals - Manufacturing Technology 42 (1): 523-526. doi:10.1016/S0007-8506(07)62500-1.

Kulakofsky, David, Henry, S.R., and Porter, D. 1993. PC-Based Cement Job Simulator Improves Primary Job Design. Paper SPE 26110-MS presented at the SPE Western Regional Meeting, Anchorage, Alaska, USA, 26-28 May. doi: 10.2118/26110-MS.

Lin, B. T., Chan, C. K., and Wang, J. C. 2008. A Knowledge-based Parametric Design System for Drawing Dies. The International Journal of Advanced Manufacturing Technology 36 (7-8): 671-680. doi: 10.1007/s00170-006-0882-y.

Liang, W. Y. and Grady, P. O. 2002. An Object-Oriented Formalism for Feature-based Distributed Concurrent Engineering. Concurrent Engineering 10 (1): 41-53. doi: 10.1106/106329302024055.

Lee, R. S., Hsu, Q. C., and Su, S. L. 1999. Development of a parametric computer-aided die design system for cold forging. Journal of Material Processing Technlology 91: 80-89. doi:10.1016/S 0924-0136(98) 00427-0.

Lummus, J. L. 1970. Drilling Optimization. Journal of Petroleum Technology 22(11): 1379-1388. SPE-2744-PA. doi: 10.2118/2744-PA.

Ma, Y. -S., Britton, G. A., Tor, S. B., and Jin, L. Y. 2007. Associative assembly design features: Concept, implementation and application. International Journal of Advanced Manufacturing Technology 32 (5-6): 434-444. doi: 10.1007/s00 170-005-0371-8.

Menand, S., Sellami, H., Tijani, M., and Stab, O. 2006. Advancements in 3D Drillstring mechanics: From the Bit to the Topdrive. Paper SPE 98965-MS presented at the IADC/SPE Drilling Conference, Miami, USA, 21-23 February. doi: 10.2118/98965-MS

Ma, Y. -S., and Tong, T. 2003. Associative feature modeling for concurrent engineering integration. Computers in Industry 51(5) : 51-71. doi:10.1016/S0166-3615(03)00025-3.

Ma, Y. -S., Tor, S.B., and Britton, G.A. 2003. The development of standard component library for plastic injection mould design using an object oriented approach. International Journal of Advance Manufacturing Technology 22(9-10): 611-618.

Ma, Y. -S., Britton, G. A., Tor, S. B., Gunawan, E., and Lee, C.H. 2003. Standard Component Library Design and Implementation for Plastic Injection Mold Design with CAD Tool. Presented at the Fourth International Conference on Control and Automation (ICCA' 03), Montreal, Canada, 10-12 June

Mendes, J.R.P., Morooka, C.K., and Guilherme, I.R. 2003. Case based Reasoning in Offshore Well Design. Journal of Petroleum Science and Engineering 40(1-2): 47-60. doi: 10.1016/S0920-4105(03)00083-4.

Morooka, C.K., Guilherme, I.R., and Mendes, J.R.P. 2001. Development of intelligent systems for well drilling and petroleum production. Journal of Petroleum Science and Engineering 32 (2-4): 191-199. doi: 10.1016/S0920-4105(01)00161-9.

Myung, S., and Han, S. 2001. Knowledge-Based Parametric Design of Mechanical Products Based on Configuration Design Method. Expert Systems with Applications 21(2): 99-107.

Mehra, S. 1997. Replacing 5 inch and 3-1/2inch Drill Pipe with a Single String of 4 inch Drill Pipe. Paper SPE 37648 presented at the SPE/IADC Drilling Conference, Amsterdam, The Netherlands, 4-6 March.

Mattiello, D., and Sansone, A. 1992. CASCADE: A Knowledge-Based Drilling Engineering Software Tool. Paper SPE 24273-MS presented at the SPE European Petroleum Compute Conference, Stanvanger, Norway, 24-27 May. doi: 10.2118/24273-MS.

Mattiello, D., Piantanida, M., Schenato, A., and Tomada, L. 1993. Casing shoe depths accurately and quickly selected with computer assistance. Oil & Gas Journal 91(40):86-93.

Martinez, E. 1992. Directional Drilling Expert System. Paper SPE 23664-MS presented at the SPE Latin America Petroleum Engineering Conference, Caracas, Venezuela, March 8-11. doi: 10.2118/23664-MS.

Maidla, E.E., and Ohara, S. 1991. Field Verification of Drilling Models and Computerized Selection of Drill Bit, WOB, and Drillstring Rotation. SPE Drilling Engineering 6(3): 189-195, SPE-19130-PA. doi: 10.2118/19130-PA.

Mabile, C.M., and Hamelin, J-P.A. 1989. An Expert System Helps in Formation Recognition. Paper SPE 19132-MS presented at the Petroleum Computer Conference, San Antonio, Texas, USA, 26-28 June. doi: 10.2118/19132-MS.

Ovtcharova, J., and Jasnoch, U. 1994. Featured-based design and consistency management in CAD applications: a unified approach. Advances in Engineering Software 20 (2-3): 65-73. doi:10.1016/0965-9978(94)90050-7.

Ovtcharova, J., Pahl, G., and Rix, J. 1992. Proposal for feature classification in feature-based design. Computers and Graphics 16(2): 187-195. doi:10.1016/0097-8493(92)90046-X.

Rahman, S.M.R., and Ma, Y.-S. 2011. Smart CAD Modeling for Well-drilling String Design. a paper being developed.

Rahman, S.M.R., and Ma, Y.-S. 2011. Knowledge driven generic drill string modeling. submitted to CANCAM 2011, 23rd Canadian Congress of Applied Mechanics, 2011, Vancouver, BC, Canada

Rashidi, B., Harland, G. and Nygaard, R. 2008. Real-Time Drill Bit Wear Prediction by Combining Rock Energy and Drilling Strength Concepts. Paper SPE 117109-MS presented at the Abu Dhabi International Petroleum Exhibition and Conference, Abu Dhabi, UAE, 3-6 November. doi: 10.2118/117109-MS.

Rahman, S. S., and Chilingarian, G. V. 1995. Casing Design Theory and Practice. Amsterdam, The Netherlands: Elsevier.

Roque, J. L., E. E. Maidla, and R. R. Wagner. 1994. Casing Cost Optimization for Complex Loading Situations. SPE Computer Applications 12(4): 24-29, SPE-28224-PA. doi: 10.2118/28224-PA.

Rabia, H. 1988. Discussion of Minimum Cost Casing Design for Vertical and Directional Wells. Journal of Petroleum Technology 504-506, SPE-17100.

Rabia, H. 1985. Oil Well Drilling Engineering Principles and Practice. London: Graham and Trotman Inc.

Reed, R. L. 1972. A Monte Carlo Approach to Optimal Drilling. SPE Journal 12 (5): 423-438. SPE-3513-PA. doi: 10.2118/3513-PA.

Shokouhi, S. V., Skalle, P., Aamodt, A., and Sørmo, A. 2009. Integration of Real-time Data and Past Experiences for Reducing Operational Problems. Paper SPE 13969-MS presented at the International Petroleum Technology Conference, Doha, Qatar, 7-9 December 2009. doi: 10.2523/13969-MS.

Siddique, Z., and Yanjiang, Z. 2002. Automatic Generation of Product Family Member CAD Models Supported by A Platform Using a Template approach. ASME DETC Conference, Montreal, Canada.

Schulte, M., Weber, C., and Stark, R. 1993. Functional features for design in mechanical engineering. Computers in Industry 23 (1-2): 15-24. doi:10.1016/0166-3615(93)90111-D.

Salomons, O., Houten, F. V., and Kals, H. 1993. Review of Research in Feature-Based Design. Journal of Manufacturing Systems 12(6):514-515. doi:10.1016/0278-6125(93)90012-I.

Shah, J. J. 1991. Assessment of features technology. Computer-Aided Design 23 (5): 331-343. doi:10.1016/0010-4485(91)90027-T.

Shah, J. J. 1989. Philosophical development of form feature concept. Presented at the NSF Engineering Design Research Conference. Amherst, MA, USA.

Shah, J. J., and Rogers, M. T. 1988. Expert form feature modeling shell. Computer Aided Design 20 (9): 515-24. doi:10.1016/0010-4485(88)90041-3.

Shah, Jami J. 1988. Feature transformations between application-specific feature spaces. Computer-Aided Engineering Journal 5 (6): 247-255.

Shah, Jami J., and Rogers, M. T. 1988. Functional Requirements and Conceptual Design of the Feature-based Modeling System. Computer-Aided Engineering Journal 5 (1): 9-15.

Van't Erve, A. H. 1988. Generative computer aided process planning for part manufacturing: An expert system approach. Ph.D. dissertation, University of Twente, The Netherlands.

Wojtanowicz, A.K., and Maidla, E.E, 1987. Minimum-Cost Casing Design for Vertical and Directional Well. Journal of Petroleum Technology 39 (10): 1269-1282, SPE 14499-PA. doi: 10.2118/14499-PA.

Zha, X.F., Du, H.J., and Qiu, J.H. 2001. Knowledge-Based Approach and System for Assembly-Oriented Design, Part II: The System Implementation. Engineering Applications of Artificial Intelligence 14(2): 239-254. doi:10.1016/S0952-1976(00)00061-0.

Appendix A Engineering Calculations

A.1 Casing Design Load Calculation (Byrom 2007):

Surface casing collapse load at the casing surface -

- Internal pressure: Zero
- External pressure: Zero

$$\Delta P_{S\_c\_surface} = 0 \qquad \text{Eq. (A-1)}$$

Surface casing collapse load at the casing shoe -

- Internal pressure: Atmospheric pressure or zero
- External pressure: Mud pressure when run

$$\Delta P_{S\_c\_shoe} = \gamma_{mud} h - 0 \qquad \text{Eq. (A-2)}$$

Surface casing burst load at the casing surface -

- Internal pressure: Surface casing full of gas, all the way from shoe to the top.
- External pressure: Zero

$$\Delta P_{S\_b\_surface} = P_{fracture} e^{\frac{gM(h_2 - h_1)}{ZRT_{avg}}} \qquad \text{Eq. (A-3)}$$

Surface casing burst load at the casing shoe -

- Internal pressure: Equivalent of gas kick that fracture and flows into formation below casing shoe
- External pressure: Freshwater gradient

$$\Delta P_{S\_b\_shoe} = (\gamma_{fracture} - \gamma_{water})h \qquad \text{Eq. (A-4)}$$

Intermediate casing collapse load at the casing surface –

- Internal pressure: Zero
- External pressure: Zero

$$\Delta P_{I\_c\_surface} = 0$$ Eq. (A-5)

Intermediate casing collapse load at the casing shoe–

- Internal pressure: Fresh water on the inside
- External pressure: Mud pressure when run

$$\Delta P_{I\_c\_shoe} = (\gamma_{mud} - \gamma_{water})h$$ Eq. (A-6)

Intermediate casing burst load at the casing surface–

- Internal pressure: Gas inside with gas pressure at the surface
- External pressure: Zero

$$\Delta P_{I\_b\_surface} = P_{fracture} e^{\frac{gM(h_2 - h_1)}{ZRT_{avg}}}$$ Eq. (A-7)

Intermediate casing burst load at the casing shoe–

- Internal pressure: Fracture pressure at the shoe
- External pressure: Freshwater gradient behind the casing

$$\Delta P_{I\_b\_shoe} = (\gamma_{fracture} - \gamma_{water})h$$ Eq. (A-8)

Production casing collapse load at the casing surface–

- Internal pressure: Zero
- External pressure: Zero

$$\Delta P_{P\_c\_surface} = 0$$ Eq. (A-9)

Production casing collapse load at the casing shoe–

- Internal pressure: Empty on the inside
- External pressure: Mud pressure when run

$$\Delta P_{P\_c\_shoe} = \gamma_{mud} h - 0$$ Eq. (A-10)

Production casing burst load at the casing surface–

- Internal pressure: Gas inside with gas pressure at the surface
- External pressure: Zero

$$\Delta P_{P\_b\_surface} = P_{fracture} e^{\frac{gM(h_2 - h_1)}{ZRT_{avg}}}$$ Eq. (A-11)

Production casing burst load at the casing shoe–

- Internal pressure: Fracture pressure at the shoe
- External pressure: Freshwater gradient behind the casing

$$\Delta P_{P\_b\_shoe} = (\gamma_{pore} - \gamma_{water})h$$ Eq. (A-12)

Temperature Calculation:

It is assumed that the temperature gradient is linear. So T$_{\text{average}}$ can be calculated as –

$$T_{average} = \frac{T_{surface} + T_{shoe}}{2} + 420^0 R$$ Eq. (A-13)

$$T_{shoe} = T_{surface} + (\frac{D_{shoe}}{TVD})(T_{TVD} - T_{surface})$$ Eq. (A-14)

A.2 Drill String Design Calculation (API RP 7G, 1998):

Length of collar,

$$L_c = \frac{Bit_{wm}}{Cos\alpha \times NP \times k_b \times W_c}$$ Eq. (A-15)

Length of drill pipe

$$L_{dp} = \frac{P_t \times 0.9}{SF \times k_b \times W_{dp}} - \frac{W_c \times L_c}{W_{dp}}$$ Eq. (A-16)

Or

$$L_{dp} = \frac{P_t \times 0.9 - MOP}{k_b \times W_{dp}} - \frac{W_c \times L_c}{W_{dp}}$$ Eq. (A-17)

Allowable collapse factor

$$P_{ac} = \frac{P_p}{S_F}$$ Eq. (A-18)

If no fluid inside the pipe, the actual collapse pressure may be calculated by –

$$P_c = \frac{LW_g}{19.251}$$ Eq. (A-19)

Or

$$P_c = \frac{LW_f}{144}$$ Eq. (A-20)

Appendix B Programming Codes

B.1 NX Code for Drill String Design:

Header File

#Drill_string.h

```
int Drill_string_save_close();

int Drill_string_assembly(tag_t *drill_pipe,tag_t *drill_collar,tag_t *hwdp,tag_t *drill_bit,tag_t *bit_sub);
int Drill_pipe_sub_assembly(tag_t *drill_pipe_body,tag_t *dp_Tj_pin,tag_t *dp_Tj_box);
int Drill_string_comp_creation(tag_t *drill_pipe, tag_t *drill_collar, tag_t *hwdp, tag_t *drill_bit, tag_t *bit_sub);
int Drill_pipe_comp_creation(tag_t *drill_pipe_body, tag_t *dp_Tj_pin, tag_t *dp_Tj_box);
#define UF_CALL(X) (report_error( __FILE__, __LINE__, #X, (X)))
int report_error( char *file, int line, char *call, int irc); // for error checking
int drill_pipe_expr(void);
//int Drill_string_comp_dp_ieu(tag_t *new_part);
//int Drill_pipe_subassembly(tag_t *dp_body,tag_t *dp_Tj_pin,tag_t *dp_Tj_box);

int Drill_pipe_body(void);
int Drill_pipe_tool_joint_pin(void);
int Drill_pipe_tool_joint_box(void);
int Drill_collar(void);
int Bit_sub_A(void);
int Drill_bit(void);
int Drill_string_drill_pipe_array(int dptotal, double dst_btn_dp, tag_t *parent, tag_t *comp);
int Drill_string_drill_collar_array(int dctotal, double dst_btn_dc, tag_t *parent, tag_t *comp);
```

Drill_string_save_close   //Save and close existing file

```
#include <string.h>
#include <uf_defs.h>
#include <uf.h>
#include <uf_modl.h>
#include <uf_part.h>
#include <uf_ui.h>
#include "Drill_string.h"



int Drill_string_save_close()
{
/********************* Routine variable declaration  ********************/

    char option[2][38],      /* menu options */
        estr[132];          /* general purpose string */

    int numparts,           /* number of currently loaded parts */
       deflt,              /* menu default */
       errcount,           /* count of errors from UF_PART_save_all */
       *errcodes,          /* error codes from UF_PART_save_all */
       resp;               /* user response */

    tag_t parent,           /* tag of parent part */
         *errtags;          /* tags of parts that failed to save */

    UF_PART_load_status_t estat;  /* structure for load part */

/********************* Beginning of Executable code **********************/


/* Determine if any parts are open */
    numparts = UF_PART_ask_num_parts();
    if(numparts > 0)
    {
                /* Put up 2 options;  Save and close or close without saving.
                Use the 'Save' option as the default. */
                uc1601("Input save options", 1);
                strcpy(estr,"Select Save Options");
                strcpy(option[0],"Save & Close");
                strcpy(option[1], "Close");
                deflt=1;            //changed
                resp=uc1603(estr, deflt, option, 2);                    //changed

                /* Return a 1 if Back or 2 if Cancel was picked, or perform selected          option. */

                if(resp==1) return(0);
                        if(resp==2) return (0);            //changed
                if (resp==5)
                        {           /* save & close */
                        //  printf("Save & close\n");
                          UF_PART_save_all(&errcount, &errtags, &errcodes);
                          UF_PART_close_all();
                        }
                else if(resp==6)
                        {           /* close without save */
                        //  printf("close\n");
                          UF_PART_close_all();
                        }
        }

/* Retrieve Drill_string_assembly.prt using UF_PART_open.  Display an error
   dialog and return an non-zero if an error occurs. */
               strcpy(estr,
"D:\\Amar_Gobesona\\Nx_Open\\Drill_String_Modeling\\Drill_string_assembly\\Drill_string_model\\Assembly.prt");
              //changed

               resp=UF_PART_open(estr, &parent, &estat);
               if(resp!=0)
               {            uc1601("Error while opening drill_string.prt",1);
                            if(estat.n_parts !=0)
                            {  UF_free(estat.statuses);
                               UF_free_string_array(estat.n_parts, estat.file_names);
                            }
               }
      return (resp);
}
```

Utility_fucntion   //To check error in the NX functions

```
#include <uf_part.h>
#include <uf_ui.h>
#include <uf.h>
#include <stdio.h>


int report_error( char *file, int line, char *call, int irc)
{
   if (irc)
   {
      char err[133],
           msg[133];

      sprintf(msg, "*** ERROR code %d at line %d in %s:\n+++ ",
         irc, line, file);
      UF_get_fail_message(irc, err);

      UF_print_syslog(msg, FALSE);
      UF_print_syslog(err, FALSE);
      UF_print_syslog("\n", FALSE);
      UF_print_syslog(call, FALSE);
      UF_print_syslog(";\n", FALSE);

      if (!UF_UI_open_listing_window())
      {
         UF_UI_write_listing_window(msg);
```

```
            UF_UI_write_listing_window(err);
            UF_UI_write_listing_window("\n");
            UF_UI_write_listing_window(call);
            UF_UI_write_listing_window(";\n");
        }
    }

    return(irc);
}

int save_close_part(void)
{               /* Determine if any parts are open */

                int numparts = UF_PART_ask_num_parts();
                if(numparts > 0)
                {

                }
                  return (0);
}
```

Drill_string_model   // Main function of the progam that calls other sub routines

```
/* Include files */
#if ! defined ( __hp9000s800 ) && ! defined ( __sgi ) && ! defined ( __sun )
#   include <strstream>
#   include <iostream>
    using std::ostrstream;
    using std::endl;
    using std::ends;
    using std::cerr;
#else
#   include <strstream.h>
#   include <iostream.h>
#endif
#include <uf.h>
#include <uf_ui.h>
#include <uf_exit.h>

#include "Drill_string.h"


#define UF_CALL(X) (report_error( __FILE__, __LINE__, #X, (X)))

static int report_error( char *file, int line, char *call, int irc)
{
    if (irc)
    {
        char err[133],
             msg[133];

        sprintf(msg, "*** ERROR code %d at line %d in %s:\n+++ ",
            irc, line, file);
        UF_get_fail_message(irc, err);

        UF_print_syslog(msg, FALSE);
        UF_print_syslog(err, FALSE);
        UF_print_syslog("\n", FALSE);
        UF_print_syslog(call, FALSE);
        UF_print_syslog(";\n", FALSE);

        if (!UF_UI_open_listing_window())
        {
            UF_UI_write_listing_window(msg);
            UF_UI_write_listing_window(err);
            UF_UI_write_listing_window("\n");
            UF_UI_write_listing_window(call);
            UF_UI_write_listing_window(";\n");
        }
    }

    return(irc);
}


extern DllExport void ufusr( char *parm, int *returnCode, int rlen )

{
        //Routine variable declaration

        tag_t drill_pipe,
                drill_collar,
                hwdp,
                drill_bit,
                bit_sub;
        tag_t drill_pipe_body,
                dp_Tj_pin,
                dp_Tj_box;

        int flag;

    /* Initialize the NX environment */
    if( UF_CALL(UF_initialize()) )
    {
        /* Failed to initialize */
        return;
    }


        flag=Drill_string_save_close();
        if(flag!=0) return;

        flag=Drill_string_assembly(&drill_pipe,&drill_collar,&hwdp,&drill_bit,&bit_sub);
        if(flag!=0) return;
        flag=Drill_string_comp_creation(&drill_pipe, &drill_collar,
          &hwdp, &drill_bit, &bit_sub);
        if(flag!=0) return;


    /* Terminate the NX environment */
    UF_CALL(UF_terminate());
}

extern int ufusr_ask_unload( void )
{
    return( UF_UNLOAD_IMMEDIATELY  );
}
```

**Drill_string_assembly		// Creates the drill string assembly in top down appraoch**

```
# Drill_string_assembly

#include <string.h>
#include <uf.h>
#include <uf_defs.h>
#include <uf_part.h>
#include <uf_layer.h>
#include <uf_param.h>
#include <uf_ui.h>
#include <uf_assem.h>
#include <uf_modl.h>
#include <stdio.h>
#include "Drill_string.h"

#define UF_CALL(X) (report_error( __FILE__, __LINE__, #X, (X)))
```

```
static int report_error( char *file, int line, char *call, int irc)
{
   if (irc)
   {
      char err[133],
          msg[133];

      sprintf(msg, "*** ERROR code %d at line %d in %s:\n+++ ",
         irc, line, file);
      UF_get_fail_message(irc, err);

      UF_print_syslog(msg, FALSE);
      UF_print_syslog(err, FALSE);
      UF_print_syslog("\n", FALSE);
      UF_print_syslog(call, FALSE);
      UF_print_syslog(";\n", FALSE);

      if (!UF_UI_open_listing_window())
      {
         UF_UI_write_listing_window(msg);
         UF_UI_write_listing_window(err);
         UF_UI_write_listing_window("\n");
         UF_UI_write_listing_window(call);
         UF_UI_write_listing_window(";\n");
      }
   }

   return(irc);
}


int Drill_string_assembly(tag_t *drill_pipe,tag_t *drill_collar,tag_t *hwdp,tag_t *drill_bit,tag_t *bit_sub)


{
/******************** Routine variable declaration  ******************/
     char    pname[MAX_FSPEC_SIZE+1],          /* new part name */
           refset[MAX_ENTITY_NAME_SIZE+1],   /* reference set name */
           iname[MAX_ENTITY_NAME_SIZE+1];    /* name of comp inst */

     tag_t   workp,        /* current work part */
           nullt;

     int     units,        /* 1 = millimeters, 2 = inches */
          /*i, */               /* loop variable */                                                //dont understand
           flag;         /* results from create component */

     int     zero=0, one=1, two=2, three=3, output=0;
     int layer;
     double  imat[]={1.0, 0.0, 0.0,
                                                  0.0, 1.0, 0.0 },  /* orientation for components */
                               imat_dc[]={1.0, 0.0, 0.0,
                                                  0.0, -1.0, 0.0 }, /* Orientation of drill collar*/
                               imat_bs[]={1.0, 0.0, 0.0,
                                                  0.0, 1.0, 0.0 }, /* Orientation of bit sub*/
                               imat_db[]={1.0, 0.0, 0.0,
                                                  0.0, -1.0, 0.0 }; /* Orientation of drill bit*/


              double origin_1[]={0.0, 0.0, dp_origin};/* drill pipe position */
              double origin_2[]={0.0,0.0,dc_origin};/* drill collar position*/
              double origin_3[]={0.0,0.0,bit_sub_origin };/* bit sub position*/
              double origin_4[]={0.0,0.0,0.0};/* drill bit  position*/
              double origin_5[]={0.0, 0.0, 0.0}; /* heavy weight drill pipe position */


               workp = UF_ASSEM_ask_work_part();
 units = 2;                                                                                             //changed

     nullt = NULL_TAG;
```

```
    strcpy(pname,"drill_pipe");
    refset[0] = '\0';
    strcpy(iname,"DRILL_PIPE");
    flag=UF_CALL(UF_ASSEM_create_component_part(workp,pname,refset,iname,
                        units,zero,origin_1,imat,zero,&nullt,drill_pipe));

   flag=UF_LAYER_set_status(2,2);
    strcpy(pname,"drill_collar");
    strcpy(iname,"DRILL_COLLAR");
    flag=UF_ASSEM_create_component_part(workp,pname,refset,iname,
                        units,two,origin_2,imat_dc,zero,&nullt,drill_collar);


  flag=UF_LAYER_set_status(3,2);
    strcpy(pname,"hwdp");
    strcpy(iname,"HWDP");
    flag=UF_ASSEM_create_component_part(workp,pname,refset,iname,
                        units,three,origin_5,imat,zero,&nullt,hwdp);

    strcpy(pname,"bit_sub");
    refset[0] = '\0';                                   //dont understand
    strcpy(iname,"BIT_SUB");
    flag=UF_CALL(UF_ASSEM_create_component_part(workp,pname,refset,iname,
                        units,zero,origin_3,imat_bs,zero,&nullt,bit_sub));

    strcpy(pname,"drill_bit");
    refset[0] = '\0';                                   //dont understand
    strcpy(iname,"DRILL_BIT");
    flag=UF_CALL(UF_ASSEM_create_component_part(workp,pname,refset,iname,
                        units,zero,origin_4,imat_db,zero,&nullt,drill_bit));


             return (0);
}
```

Drill_string_comp_creation   //Creates the drill string components

#Drill_string_comp_creation

```
#include <stdio.h>
#include <uf_obj.h>
#include <uf_defs.h>
#include <uf_part.h>
#include <uf_assem.h>
#include <uf_modl.h>
#include <math.h>
#include <uf.h>
#include <uf_defs.h>
#include <uf_csys.h>
#include "Drill_string.h"

/* utilities */


//#define UF_CALL(X) (report( #X, __FILE__, __LINE__, (X)))
static int report( char *call, char *file, int line, int irc)
{
 if (irc)
 {
  char messg[133];
  printf("%s\n%s, line %d:   ", call, file, line);
  (UF_get_fail_message(irc, messg)) ?
     printf("returned %d\n", irc):
     printf("error %d: %s\n", irc, messg);
 }
 return(irc);
}
```

```
int Drill_string_comp_creation(tag_t *drill_pipe, tag_t *drill_collar,
                 tag_t *hwdp, tag_t *drill_bit, tag_t *bit_sub)


{
/******************** Routine variable declaration  *******************/
     int color,     /* default entity color */
         layer,     /* default entity layer */
         density,   /* default entity width */
         font;      /* default line font */

     int flag;      /* results from functions */

     tag_t wpart,   /* part associated with component */
          parent;  /* current work part;  root of the assembly */
                 tag_t abc,
                                drill_pipe_body,
                                dp_Tj_pin,
                                dp_Tj_box;

/******************** Beginning of Executable code *********************/

/* Set parent to the current work part */
                 parent = UF_ASSEM_ask_work_part();

     wpart=UF_ASSEM_ask_child_of_instance(*drill_pipe);
     if(wpart==NULL_TAG)return(1);
     flag=UF_CALL(UF_PART_set_display_part(wpart));
     if (flag != 0) return(flag);

/* Set the defaults for object creation color, layer, density and font */
     color = 7;
     layer = 1;
     density = 2;
     font = 1;
     FTN(uf5025)(&color,&layer,&density,&font);


/* Call the routine that creates the drill pipe sub assembly. */


                flag=Drill_pipe_sub_assembly(&drill_pipe_body,&dp_Tj_pin,&dp_Tj_box);
                flag=Drill_pipe_comp_creation(&drill_pipe_body,&dp_Tj_pin,&dp_Tj_box);
                  /* Array of drill pipe*/

     wpart=UF_ASSEM_ask_child_of_instance(*drill_collar);
     if(wpart==NULL_TAG)return(1);
     flag=UF_CALL(UF_PART_set_display_part(wpart));
     if (flag != 0) return(flag);

/* Set the defaults for object creation color, layer, density and font */
     color = 7;
     layer = 1;
     density = 2;
     font = 1;
     FTN(uf5025)(&color,&layer,&density,&font);


/* Call the routine that creates drill collar. */
     flag = Drill_collar();
     if(flag != 0) return (flag);


     wpart=UF_ASSEM_ask_child_of_instance(*bit_sub);
     if(wpart==NULL_TAG)return(1);
     flag=UF_CALL(UF_PART_set_display_part(wpart));
     if (flag != 0) return(flag);
/* Set the defaults for object creation color, layer, density and font */
     color = 7;
     layer = 1;
     density = 2;
     font = 1;
     FTN(uf5025)(&color,&layer,&density,&font);


/* Call the routine that creates drill collar. */
     flag = Bit_sub_A();
     if(flag != 0) return (flag);

/* Set the display part to the parent (original work/display part) */
   flag=UF_CALL(UF_PART_set_display_part(parent));
     if (flag != 0) return(flag);

     wpart=UF_ASSEM_ask_child_of_instance(*drill_bit);
     if(wpart==NULL_TAG)return(1);
     flag=UF_CALL(UF_PART_set_display_part(wpart));
     if (flag != 0) return(flag);

/* Set the defaults for object creation color, layer, density and font */
     color = 7;
     layer = 1;
     density = 2;
     font = 1;
     FTN(uf5025)(&color,&layer,&density,&font);


/* Call the routine that creates drill collar. */
     flag = Drill_bit ();
     if(flag != 0) return (flag);


/* Set the display part to the parent (original work/display part) */
   flag=UF_CALL(UF_PART_set_display_part(parent));
     if (flag != 0) return(flag);


/*Array Drill Pipe*/
double dptotal;
double dst_btn_dp;
double dctotal=5;
double dst_btn_dc=36;
//tag_t *parent;

flag = UF_CALL(UF_MODL_import_exp("D:\\Amar_Gobesona\\Nx_Open\\Journal_1\\Array.exp",0));
            if (flag) return flag;
            flag=UF_CALL(UF_MODL_eval_exp("dptotal", &dptotal));
            if (flag!=0) return(flag);
            flag=UF_CALL(UF_MODL_eval_exp("dst_btn_dp", &dst_btn_dp));
            if (flag!=0) return(flag);

            flag=Drill_string_drill_pipe_array(dptotal,dst_btn_dp, &parent, drill_pipe);

            flag=Drill_string_drill_collar_array(dctotal, dst_btn_dc, &parent,drill_collar);

     return (0);
}
```

Drill_pipe_body/ / Code for drill pipe body

# Drill_pipe_body

```
#include <stdio.h>
#include <uf_part.h>
#include <uf_modl.h>
#include <uf_obj.h>
#include <uf_ui.h>
#include <string.h>
#include <math.h>
```

```
#include <uf_attr.h>
#include <NXOpen/Session.hxx>


using namespace NXOpen;

#include "Drill_string.h"


#define PART_PATH "D:\\Amar_Gobesona\\Nx_Open\\Journal_1\\Drill_pipe_internal_external_upset_expression.exp"

int Drill_pipe_body(void)
{
        char svalue[30];      /* string for attribute value */
        UF_ATTR_value_t att_value;   /* attribute data structure */
        int flag = 0;

        flag = UF_CALL(UF_MODL_import_exp(PART_PATH,0));
        if (flag) return flag;

        flag = UF_CALL (UF_MODL_update());
        if (flag) return flag;


        double Dp_ieu_origin[3],
                Dp_ieu_direction[3];
        tag_t Dp_ieu_body,
                cyl_feature_Dp_ieu_1;

        Dp_ieu_origin[0] = 0.0;
        Dp_ieu_origin[1] = 0.0;
        Dp_ieu_origin[2] = 0.0;

        Dp_ieu_direction[0] = 0.0;
        Dp_ieu_direction[1] = 0.0;
        Dp_ieu_direction[2] = 1.0;



        // Create the cylinder and check the return code
 flag = UF_CALL(UF_MODL_create_cyl1(UF_NULLSIGN, Dp_ieu_origin, "Dp_ieu_leu", "Dp_ieu_uod",
Dp_ieu_direction, &cyl_feature_Dp_ieu_1));
  if(flag!=0)
                uc1601("Error in upset cylinder bottom creation",1);

        // Obtain the body tag from the feature tag.
        // Assign the name LONG_CYL to the body.    Check the return code.
   flag = UF_CALL(UF_MODL_ask_feat_body(cyl_feature_Dp_ieu_1, &Dp_ieu_body) );
        flag = UF_CALL(UF_OBJ_set_name(Dp_ieu_body, "Dp_ieu_LONG_CYL"));

        //Creating cone at the bottom

        double Dp_ieu_origin_3[3];
        double Dp_ieu_direction_3[3];
        char * Dp_ieu_diameter_3[2]={"Dp_ieu_uod","Dp_ieu_od"};

        tag_t Dp_ieu_cone1;
        flag=UF_CALL(UF_MODL_eval_exp("Dp_ieu_leu", &Dp_ieu_origin_3[2]));
        if(flag!=0) return (flag);
        Dp_ieu_origin_3[0] = 0.0;
        Dp_ieu_origin_3[1] = 0.0;
        Dp_ieu_direction_3[0] = 0.0;
        Dp_ieu_direction_3[1] = 0.0;
        Dp_ieu_direction_3[2] = 1.0;

        flag=UF_CALL(UF_MODL_create_cone1(UF_POSITIVE,Dp_ieu_origin_3,"Dp_ieu_lit_meu",Dp_ieu_diamet
er_3,Dp_ieu_direction_3, &Dp_ieu_cone1));
        if (flag!=0)
        uc1601("Vhul hoise mamu cone in Dp_ieu",1);

        // creating drill pipe body
        double Dp_ieu_origin_4[3],
                Dp_ieu_direction_4[3];
        tag_t cyl_feature_Dp_ieu_2;

        flag=UF_CALL(UF_MODL_eval_exp("Dp_ieu_sum_leu_meu", &Dp_ieu_origin_4[2]));
        if(flag!=0) return (flag);
        Dp_ieu_origin_4[0] = 0.0;
        Dp_ieu_origin_4[1] = 0.0;

        Dp_ieu_direction_4[0] = 0.0;
        Dp_ieu_direction_4[1] = 0.0;
        Dp_ieu_direction_4[2] = 1.0;

        // Create the cylinder and check the return code
 flag = UF_CALL(UF_MODL_create_cyl1(UF_POSITIVE, Dp_ieu_origin_4, "Dp_ieu_subtract_length", "Dp_ieu_od",
Dp_ieu_direction_4, &cyl_feature_Dp_ieu_2));
  if(flag!=0)
                uc1601("Error in drill pipe body creation",1);

//Creating cone at the top
        double Dp_ieu_origin_5[3];
        double Dp_ieu_direction_5[3];
        char * Dp_ieu_diameter_5[2]={"Dp_ieu_od","Dp_ieu_uod"};

        tag_t Dp_ieu_cone2;
        flag=UF_CALL(UF_MODL_eval_exp("Dp_ieu_subtract_length_2", &Dp_ieu_origin_5[2]));
        if(flag!=0) return (flag);
        Dp_ieu_origin_5[0] = 0.0;
        Dp_ieu_origin_5[1] = 0.0;
        Dp_ieu_direction_5[0] = 0.0;
        Dp_ieu_direction_5[1] = 0.0;
        Dp_ieu_direction_5[2] = 1.0;

        flag=UF_CALL(UF_MODL_create_cone1(UF_POSITIVE,Dp_ieu_origin_5,"Dp_ieu_lit_meu",Dp_ieu_diamet
er_5,Dp_ieu_direction_5, &Dp_ieu_cone2));
        if (flag!=0)
        uc1601("Vhul hoise mamu cone in Dp_eu",1);

                // creating a upset cylinder feature in the top of the part(External upset).

        double Dp_ieu_origin_6[3],
                Dp_ieu_direction_6[3];
        tag_t cyl_feature_Dp_ieu_3;

        flag=UF_CALL(UF_MODL_eval_exp("Dp_ieu_subtract_length_2_meu", &Dp_ieu_origin_6[2]));
        if(flag!=0) return (flag);
        Dp_ieu_origin_6[0] = 0.0;
        Dp_ieu_origin_6[1] = 0.0;

        Dp_ieu_direction_6[0] = 0.0;
        Dp_ieu_direction_6[1] = 0.0;
        Dp_ieu_direction_6[2] = 1.0;


 flag = UF_CALL(UF_MODL_create_cyl1(UF_POSITIVE, Dp_ieu_origin_6, "Dp_ieu_leu", "Dp_ieu_uod",
Dp_ieu_direction_6, &cyl_feature_Dp_ieu_3));
  if(flag!=0)
                uc1601("Error in external upset at the top creation",1);

        double Dp_ieu_point_7[3]={0.0,0,0};
        double Dp_ieu_direc_7[3]={0,0,1};
        tag_t Dp_ieu_dplane_4_external_upset_bottom;
        flag=UF_MODL_create_fixed_dplane(Dp_ieu_point_7,Dp_ieu_direc_7,&Dp_ieu_dplane_4_external_upset_bo
ttom);

        double Dp_ieu_location_7[3]={0,0,0};

        tag_t Dp_ieu_external_upset_bottom;
        flag=UF_CALL(UF_MODL_create_simple_hole(Dp_ieu_location_7,Dp_ieu_direc_7,"Dp_ieu_uid","Dp_ieu_li
u","Dp_ieu_tip_angle_0",Dp_ieu_dplane_4_external_upset_bottom,NULL_TAG, &Dp_ieu_external_upset_bottom));

        if(flag!=0)
                uc1601("error in external upset bottom hole creation",1);



double Dp_ieu_direc_8[3]={0,0,-1};
double Dp_ieu_point_8[3];
flag=UF_CALL(UF_MODL_eval_exp("Dp_ieu_length", &Dp_ieu_point_8[2]));
        if(flag!=0) return (flag);
        Dp_ieu_point_8[0] = 0.0;
        Dp_ieu_point_8[1] = 0.0;

        tag_t Dp_ieu_dplane_4_external_upset_top;
        flag=UF_MODL_create_fixed_dplane(Dp_ieu_point_8,Dp_ieu_direc_8,&Dp_ieu_dplane_4_external_upset_to
p);

        double Dp_ieu_location_8[3];
        flag=UF_CALL(UF_MODL_eval_exp("Dp_ieu_length", &Dp_ieu_location_8[2]));
        if(flag!=0) return (flag);
        Dp_ieu_location_8[0] = 0.0;
        Dp_ieu_location_8[1] = 0.0;

        tag_t Dp_ieu_external_upset_top;

        flag=UF_CALL(UF_MODL_create_simple_hole(Dp_ieu_location_8,Dp_ieu_direc_8,"Dp_ieu_uid","Dp_ieu_li
u","Dp_ieu_tip_angle_0",Dp_ieu_dplane_4_external_upset_top,NULL_TAG, &Dp_ieu_external_upset_top));
        if(flag!=0)
                uc1601("error in external upset top hole creation",1);

        double Dp_ieu_point_9[3];
        flag=UF_CALL(UF_MODL_eval_exp("Dp_ieu_half_length", &Dp_ieu_point_9[2]));
        if(flag!=0) return (flag);
        Dp_ieu_point_9[0] = 0.0;
        Dp_ieu_point_9[1] = 0.0;
        tag_t Dp_ieu_bore_hole_1,Dp_ieu_plane_0;
        flag=UF_MODL_create_fixed_dplane(Dp_ieu_point_9,Dp_ieu_direc_8,&Dp_ieu_bore_hole_1);
        double Dp_ieu_location_9[3];
        flag=UF_CALL(UF_MODL_eval_exp("Dp_ieu_half_length", &Dp_ieu_location_9[2]));
        if(flag!=0) return (flag);
        Dp_ieu_location_9[0]=0;
        Dp_ieu_location_9[1]=0;
        char Dp_ieu_tip_angle_0[10];
        tag_t Dp_ieu_bore_hole_0;

        flag=UF_CALL(UF_MODL_create_simple_hole(Dp_ieu_location_9,Dp_ieu_direc_8,"Dp_ieu_id","Dp_ieu_su
btract_length_3","Dp_ieu_tip_angle_0",Dp_ieu_bore_hole_1,NULL_TAG, &Dp_ieu_bore_hole_0));
        //uc1601("Drill pipe bore hole 1 is created",1);
        if(flag!=0)
                uc1601("error in Drill pipe bore hole 1 creation",1);

        double Dp_ieu_location_10[3];
        flag=UF_CALL(UF_MODL_eval_exp("Dp_ieu_half_length", &Dp_ieu_location_10[2]));
        if(flag!=0) return (flag);
        Dp_ieu_location_10[0]=0;
        Dp_ieu_location_10[1]=0;
        tag_t Dp_ieu_bore_hole_2;
        flag=UF_CALL(UF_MODL_create_simple_hole(Dp_ieu_location_10,Dp_ieu_direc_7,"Dp_ieu_id","Dp_ieu_s
ubtract_length_3","Dp_ieu_tip_angle_0",Dp_ieu_bore_hole_1,NULL_TAG, &Dp_ieu_bore_hole_2));

        if(flag!=0)
                uc1601("error in bore hole2 creation",1);

    att_value.type = UF_ATTR_string;
    strcpy(svalue,"001");           /* First call for the part number */
    att_value.value.string = svalue;

    strcpy(svalue,"DRILL_PIPE_BODY");   /* Second call for the description. */
```

```
   return (0);


}
```

Drill_pipe_tool_joint_box   //Code for drill pipe tool joint box

```
# Drill_pipe_tool_joint_box

#include <stdio.h>

#include <uf_part.h>
#include <uf_modl.h>
#include <uf_obj.h>
#include <uf_ui.h>
#include <string.h>
#include <math.h>
#include <uf_attr.h>
#include <NXOpen/Session.hxx>
using namespace NXOpen;
#include "Drill_string.h"
#define PART_PATH "D:\\Amar_Gobesona\\Nx_Open\\Journal_1\\Tool_joint_pin_expression.exp"

int Drill_pipe_tool_joint_box()
{

        char svalue[10];     /* string for attribute value */
        UF_ATTR_value_t att_value;   /* attribute data structure */
        int flag=0;

        flag = UF_CALL(UF_MODL_import_exp(PART_PATH,0));
        if (flag) return flag;

        flag = UF_CALL (UF_MODL_update());
        if (flag) return flag;

        double Tjb_origin[3],
                Tjb_direction[3];
        tag_t Tjb_body,
                cyl_feature_Tjb_1;

        Tjb_origin[0] = 0.0;
        Tjb_origin[1] = 0.0;
        Tjb_origin[2] = 0.0;

        Tjb_direction[0] = 0.0;
        Tjb_direction[1] = 0.0;
        Tjb_direction[2] = 1.0;


 flag   =   UF_CALL(UF_MODL_create_cyl1(UF_NULLSIGN,   Tjb_origin,   "Tjb_a_4",   "Tj_dia_box_upset_D_te",
Tjb_direction, &cyl_feature_Tjb_1));
  if(flag!=0)
                uc1601("Error in tool joint_pin creation",1);

   flag = UF_CALL(UF_MODL_ask_feat_body(cyl_feature_Tjb_1, &Tjb_body) );
        flag = UF_CALL(UF_OBJ_set_name(Tjb_body, "LONG_CYL"));

        char msg[20], str[20];
        sprintf(str,"%d", Tjb_body);
        sprintf(msg,"%d", cyl_feature_Tjb_1);

        double Tjb_origin_2[3];
        double Tjb_direction_2[3];
        char * Tjb_diameter_2[2]={"Tj_dia_box_upset_D_te","Tj_od_p_b_D"};

        tag_t Tjb_cone;
        flag=UF_CALL(UF_MODL_eval_exp("Tjb_a_3", &Tjb_origin_2[2]));
```

```
    if(flag!=0) return (flag);
    Tjb_origin_2[0] = 0.0;
    Tjb_origin_2[1] = 0.0;
    Tjb_direction_2[0] = 0.0;
    Tjb_direction_2[1] = 0.0;
    Tjb_direction_2[2] = 1.0;

    flag=UF_CALL(UF_MODL_create_cone1(UF_POSITIVE,Tjb_origin_2,"Tjb_a_3",Tjb_diameter_2,Tjb_direction_2, &Tjb_cone));
    if (flag!=0)
    uc1601("Vhul hoise mamu cone in tjp",1)

    double Tjb_origin_3[3],
        Tjb_direction_3[3];
    tag_t cyl_feature_Tjb_3;

    flag=UF_CALL(UF_MODL_eval_exp("sum_a_3_a_4", &Tjb_origin_3[2]));
    if(flag!=0) return (flag);
    Tjb_origin_3[0] = 0.0;
    Tjb_origin_3[1] = 0.0;


    Tjb_direction_3[0] = 0.0;
    Tjb_direction_3[1] = 0.0;
    Tjb_direction_3[2] = 1.0;

 flag  =  UF_CALL(UF_MODL_create_cyl1(UF_POSITIVE,Tjb_origin_3,  "Tj_box_tong_space_L_b",  "Tj_od_p_b_D", Tjb_direction_3, &cyl_feature_Tjb_3));
  if(flag!=0)
        uc1601("Error in tool joint_box creation",1);

    tag_t Tjb_face1, Tjb_face2;
    uf_list_p_t Tjb_list1, Tjb_list2;
    int Tjb_i,Tjb_count=0,Tjb_ftype,Tjb_dsense;
    double    Tjb_pt1[3],
            Tjb_box[6],
            Tjb_rad1,
            Tjb_rad2;
    double Tjb_dir[3]={0,0,1};
    tag_t Tjb_edge, Tjb_cham_feature1;
    flag=UF_CALL(UF_MODL_ask_body_faces(Tjb_body,&Tjb_list1));
    if(flag) return (flag);
    flag=UF_CALL(UF_MODL_ask_list_count(Tjb_list1,&Tjb_count));
    if(flag) return(flag);

    double Tjb_cyl_len;
    flag=UF_CALL(UF_MODL_eval_exp("Tjb_total_length",&Tjb_cyl_len));
    if(flag!=0) return(flag);
    for(Tjb_i=0; Tjb_i<Tjb_count;Tjb_i++)
    {

        flag=UF_MODL_ask_list_item(Tjb_list1,Tjb_i,&Tjb_face1);
        if(flag!=0) return(flag);
    flag=UF_CALL(UF_MODL_ask_face_data (Tjb_face1,&Tjb_ftype,Tjb_pt1,Tjb_dir,Tjb_box,&Tjb_rad1,&Tjb_rad2,&Tjb_dsense));
    if(flag) return (flag);
    if(Tjb_ftype==UF_bounded_plane_type)
        {

        if(fabs(Tjb_pt1[2]-(Tjb_cyl_len))<0.001)

            {
                Tjb_face2=Tjb_face1;
                break;
            }
        }
    }
    flag=UF_MODL_ask_face_edges(Tjb_face2, &Tjb_list1);
    flag=UF_CALL(UF_MODL_create_chamfer(3,"Tj_chamfer_offset_1","Tj_chamfer_offset_2","Tj_chamfer_theta",Tjb_list1,&Tjb_cham_feature1));
```

```
        flag=UF_CALL(UF_MODL_delete_list(&Tjb_list1));

        double Tjb_point_0[3];
        flag=UF_CALL(UF_MODL_eval_exp("Tjb_total_length", &Tjb_point_0[2]));
        if(flag!=0) return (flag);
        Tjb_point_0[0] = 0.0;
        Tjb_point_0[1] = 0.0;
        double Tjb_direc_0[3]={0,0,-1};
        tag_t Tjb_fixed_dplane_4_box_hole,Tjb_plane_0;
        flag=UF_MODL_create_fixed_dplane(Tjb_point_0,Tjb_direc_0,&Tjb_fixed_dplane_4_box_hole);
        double Tjb_location_0[3];
        flag=UF_CALL(UF_MODL_eval_exp("Tjb_total_length", &Tjb_location_0[2]));
        if(flag!=0) return (flag);
        Tjb_location_0[0]=0;
        Tjb_location_0[1]=0;
        char Tjb_tip_angle_1[10];
        tag_t Tjb_box_hole;

        flag=UF_CALL(UF_MODL_create_simple_hole(Tjb_location_0,Tjb_direc_0,"Tj_dia_box_upset_D_te","Tjp_pin_length","Tjb_tip_angle_1",Tjb_fixed_dplane_4_box_hole,NULL_TAG, &Tjb_box_hole));
        if(flag!=0)
                uc1601("error in collar hole creation",1);

        double Tjb_point[3]={0.0,0,0};
        double Tjb_direc[3]={0,0,1};
        tag_t Tjb_fixed_dplane_4_hole,Tjb_plane;
        flag=UF_MODL_create_fixed_dplane(Tjb_point,Tjb_direc,&Tjb_fixed_dplane_4_hole);
        double Tjb_location_5[3]={0,0,0};
        double Tjb_direc_5[3];
        Tjb_direc_5[0]=0;
        Tjb_direc_5[1]=0;
        Tjb_direc_5[2]=1;
        char Tjb_tip_angle[10];
        tag_t Tjb_bore_hole;

        flag=UF_CALL(UF_MODL_create_simple_hole(Tjb_location_5,Tjb_direc_5,"Tj_id_pin","sum_a3_a4_l_b_pin_length","Tjp_tip_angle",Tjb_fixed_dplane_4_hole,NULL_TAG, &Tjb_bore_hole));
        if(flag!=0)
                uc1601("error in collar hole creation",1);


 att_value.type = UF_ATTR_string;
    strcpy(svalue,"003");           /* First call for the part number */
    att_value.value.string = svalue;

    strcpy(svalue,"DP_TJ_BOX");            /* Second call for the description. */
    return (0);






}
```

Drill_pipe_tool_joint_pin   //Code for drill pipe tool joint pin

```
# Drill_pipe_tool_joint_pin

#include <stdio.h>
#include <uf_part.h>
#include <uf_modl.h>
#include <uf_obj.h>
#include <uf_ui.h>
#include <string.h>
#include <uf_attr.h>
#include <math.h>
#include <NXOpen/Session.hxx>
using namespace NXOpen;
```

```
#include "Drill_string.h"
#define PART_PATH "D:\\Amar_Gobesona\\Nx_Open\\Journal_1\\Tool_joint_pin_expression.exp"
int Drill_pipe_tool_joint_pin()
{
        char svalue[30];      /* string for attribute value */
        UF_ATTR_value_t att_value;   /* attribute data structure */
        int flag=0;


        flag = UF_CALL(UF_MODL_import_exp(PART_PATH,0));
        if (flag) return flag;

        flag = UF_CALL (UF_MODL_update());
        if (flag) return flag;

        double origin[3],
                direction[3];
        tag_t Tjp_body,
                cyl_feature_Tjp_1;

        origin[0] = 0.0;
        origin[1] = 0.0;
        origin[2] = 0.0;

        direction[0] = 0.0;
        direction[1] = 0.0;
        direction[2] = 1.0;

 flag = UF_CALL(UF_MODL_create_cyl1(UF_NULLSIGN, origin, "Tj_a_2", "Tj_dia_pin_upset_D_pe", direction,
&cyl_feature_Tjp_1));
  if(flag!=0)
                uc1601("Error in tool joint_pin creation",1);

   flag = UF_CALL(UF_MODL_ask_feat_body(cyl_feature_Tjp_1, &Tjp_body) );
        flag = UF_CALL(UF_OBJ_set_name(Tjp_body, "LONG_CYL"));

        char msg[20], str[20];
        sprintf(str,"%d", Tjp_body);
        sprintf(msg,"%d", cyl_feature_Tjp_1);

        double Tjp_origin_2[3];
        double Tjp_direction_2[3];
        char * Tjp_diameter_2[2]={"Tj_dia_pin_upset_D_pe","Tj_od_p_b_D"};

        tag_t Tjp_cone;
        flag=UF_CALL(UF_MODL_eval_exp("Tj_a_2", &Tjp_origin_2[2]));
        if(flag!=0) return (flag);
        Tjp_origin_2[0] = 0.0;
        Tjp_origin_2[1] = 0.0;
        Tjp_direction_2[0] = 0.0;
        Tjp_direction_2[1] = 0.0;
        Tjp_direction_2[2] = 1.0;

        flag=UF_CALL(UF_MODL_create_cone1(UF_POSITIVE,Tjp_origin_2,"Tj_a_2",Tjp_diameter_2,Tjp_directi
on_2, &Tjp_cone));
        if (flag!=0)
        uc1601("Vhul hoise mamu cone in tjp",1);

        double Tjp_origin_3[3],
                Tjp_direction_3[3];
        tag_t cyl_feature_Tjp_3;

        flag=UF_CALL(UF_MODL_eval_exp("sum_a_1_a_2", &Tjp_origin_3[2]));
        if(flag!=0) return (flag);
        Tjp_origin_3[0] = 0.0;
        Tjp_origin_3[1] = 0.0;


        Tjp_direction_3[0] = 0.0;
        Tjp_direction_3[1] = 0.0;
        Tjp_direction_3[2] = 1.0;

flag  =  UF_CALL(UF_MODL_create_cyl1(UF_POSITIVE,Tjp_origin_3,  "Tj_pin_tong_space_L_pb",  "Tj_od_p_b_D",
Tjp_direction_3, &cyl_feature_Tjp_3));
  if(flag!=0)
                uc1601("Error in tool joint_pin creation",1);

        tag_t Tjp_face1, Tjp_face2;
        uf_list_p_t Tjp_list1, Tjp_list2;
        int Tjp_i,Tjp_count=0,Tjp_ftype,Tjp_dsense;
        double  Tjp_pt1[3],
                        Tjp_box[6],
                        Tjp_rad1,
                        Tjp_rad2;
        double Tjp_dir[3]={0,0,1};
        tag_t Tjp_edge, Tjp_cham_feature1;
        flag=UF_CALL(UF_MODL_ask_body_faces(Tjp_body,&Tjp_list1));
        if(flag) return (flag);
        flag=UF_CALL(UF_MODL_ask_list_count(Tjp_list1,&Tjp_count));
        if(flag) return(flag);
        double Tjp_cyl_len;
        flag=UF_CALL(UF_MODL_eval_exp("sum_a_1_a_2_L_pb",&Tjp_cyl_len));
        if(flag!=0) return(flag);
        for(Tjp_i=0; Tjp_i<Tjp_count;Tjp_i++)
        {
                flag=UF_MODL_ask_list_item(Tjp_list1,Tjp_i,&Tjp_face1);
                if(flag!=0) return(flag);
        flag=UF_CALL(UF_MODL_ask_face_data
(Tjp_face1,&Tjp_ftype,Tjp_pt1,Tjp_dir,Tjp_box,&Tjp_rad1,&Tjp_rad2,&Tjp_dsense));
        if(flag) return (flag);
        if(Tjp_ftype==UF_bounded_plane_type)
                {

                if(fabs(Tjp_pt1[2]-(Tjp_cyl_len))<0.001)

                        {
                                Tjp_face2=Tjp_face1;
                                break;
                        }
                }
        }
        flag=UF_MODL_ask_face_edges(Tjp_face2, &Tjp_list1);
        flag=UF_CALL(UF_MODL_create_chamfer(3,"Tj_chamfer_offset_1","Tj_chamfer_offset_1","Tj_chamfer_th
eta",Tjp_list1,&Tjp_cham_feature1));
        flag=UF_CALL(UF_MODL_delete_list(&Tjp_list1));
        double Tjp_origin_4[3],
                Tjp_direction_4[3];
        tag_t cyl_feature_Tjp_4;

        flag=UF_CALL(UF_MODL_eval_exp("sum_a_1_a_2_L_pb", &Tjp_origin_4[2]));
        if(flag!=0) return (flag);
        Tjp_origin_4[0] = 0.0;
        Tjp_origin_4[1] = 0.0;


        Tjp_direction_4[0] = 0.0;
        Tjp_direction_4[1] = 0.0;
        Tjp_direction_4[2] = 1.0;

 flag  =  UF_CALL(UF_MODL_create_cyl1(UF_POSITIVE,Tjp_origin_4,  "Tjp_pin_length",  "Tj_dia_pin_upset_D_pe",
Tjp_direction_4, &cyl_feature_Tjp_4));
  if(flag!=0)
                uc1601("Error in tool joint_pin creation",1);
        double Tjp_point[3]={0.0,0,0};
        double Tjp_direc[3]={0,0,1};
        tag_t Tjp_fixed_dplane_4_hole,Tjp_plane;
        flag=UF_MODL_create_fixed_dplane(Tjp_point,Tjp_direc,&Tjp_fixed_dplane_4_hole);
        double Tjp_location_5[3]={0,0,0};
        double Tjp_direc_5[3];
        Tjp_direc_5[0]=0;
        Tjp_direc_5[1]=0;
        Tjp_direc_5[2]=1;
        char Tjp_tip_angle[10];
        tag_t Tjp_bore_hole;

        flag=UF_CALL(UF_MODL_create_simple_hole(Tjp_location_5,Tjp_direc_5,"Tj_id_pin","Tjp_total_length","
Tjp_tip_angle",Tjp_fixed_dplane_4_hole,NULL_TAG, &Tjp_bore_hole));
        if(flag!=0)
                uc1601("error in collar hole creation",1);


 att_value.type = UF_ATTR_string;
   strcpy(svalue,"002");           /* First call for the part number */
   att_value.value.string = svalue;

   strcpy(svalue,"DP_TJ_PIN");/* Second call for the description. */
   return (0);
}
```

Drill_collar **// Code for Drill collar**

# Drill_collar

```
#include <stdio.h>
#include <uf_part.h>
#include <uf_modl.h>
#include <uf_obj.h>
#include <uf_ui.h>
#include <string.h>
#include <math.h>
#include <uf_attr.h>
#include <NXOpen/Session.hxx>


using namespace NXOpen;

#include "Drill_string.h"

#define PART_PATH "D:\\Amar_Gobesona\\Nx_Open\\Journal_1\\Drill_collar.exp"

int Drill_collar()
{
        int flag=0;
        char svalue[10];      /* string for attribute value */
        UF_ATTR_value_t att_value;   /* attribute data structure */

        flag = UF_CALL(UF_MODL_import_exp(PART_PATH,0));
        if (flag) return flag;

        flag = UF_CALL (UF_MODL_update());
        if (flag) return flag;

        double origin[3],
                direction[3];
        char height[10],
                diameter[10];
        tag_t body,
                cyl_feature_Dc;

        origin[0] = 0.0;
        origin[1] = 0.0;
        origin[2] = 0.0;

        direction[0] = 0.0;
        direction[1] = 0.0;
        direction[2] = 1.0;
flag = UF_CALL(UF_MODL_create_cyl1(UF_NULLSIGN, origin, "Value_Dc_L", "Size_Dc_OD", direction,
&cyl_feature_Dc));
  if(flag!=0)
```

```
                uc1601("Error in drill collar creation",1);

  flag = UF_CALL(UF_MODL_ask_feat_body(cyl_feature_Dc, &body) );
        flag = UF_CALL(UF_OBJ_set_name(body, "LONG_CYL"));
        tag_t face, face1;
        uf_list_p_t list1, list2;
        int i,count=0,ftype,dsense;
        double  pt1[3],
                        dir[3],
                        box[6],
                        rad1,
                        rad2;
        tag_t edge, cham_feature;
        flag=UF_CALL(UF_MODL_ask_body_faces(body,&list1));
        if(flag) return (flag);
        flag=UF_CALL(UF_MODL_ask_list_count(list1,&count));
        if(flag) return(flag);
        double cyl_len;
        flag=UF_CALL(UF_MODL_eval_exp("Value_Dc_L",&cyl_len));
        if(flag!=0) return(flag);
        for(i=0; i<count;i++)
        {
                flag=UF_MODL_ask_list_item(list1,i,&face);
                if(flag!=0) return(flag);
        flag=UF_CALL(UF_MODL_ask_face_data (face,&ftype,pt1,dir,box,&rad1,&rad2,&dsense));
        if(flag) return (flag);
        if(ftype==UF_bounded_plane_type)
                {
                        if(fabs(pt1[2]-(cyl_len))<0.001)
                        {
                                face1=face;
                                break;
                        }
                }
        }
        flag=UF_MODL_ask_face_edges(face1, &list1);

        flag=UF_CALL(UF_MODL_create_chamfer(3,"Value_Dc_chamfer_offset_1","Value_Dc_chamfer_offset_2",
"Value_Dc_chamfer_theta",list1,&cham_feature));
        flag=UF_CALL(UF_MODL_delete_list(&list1));

        double point[3]={0.0,0,0};
        double direc[3]={0,0,1};
        tag_t fixed_dplane_4_hole,plane;
        flag=UF_MODL_create_fixed_dplane(point,direc,&fixed_dplane_4_hole);
        double location[3]={0,0,0};
        double direc_2[3];
        direc_2[0]=0;
        direc_2[1]=0;
        direc_2[2]=1;
        char tip_angle[10];
        tag_t collar_hole,collar_box_hole;

        flag=UF_CALL(UF_MODL_create_simple_hole(location,direc_2,"Value_Dc_ID","Value_Dc_L","Collar_hole
_tip_angle",fixed_dplane_4_hole,NULL_TAG, &collar_hole));
        if(flag!=0)
                uc1601("error in collar hole creation",1);
        flag=UF_CALL(UF_MODL_create_simple_hole(location,direc_2,"Value_Dc_dia_box","Value_Dc_T_B_L","
Collar_hole_tip_angle",fixed_dplane_4_hole,NULL_TAG,&collar_box_hole));
        flag=UF_CALL(UF_MODL_eval_exp("Value_Dc_L", &origin[2]));
        if(flag!=0) return(flag);

        tag_t collar_box;

        flag=UF_CALL(UF_MODL_create_cyl1(UF_POSITIVE,origin,"Value_Dc_T_P_L","Value_Dc_dia_pin",dire
ction, &collar_box));
        if (flag!=0)
        uc1601("Vhul hoise mamu",1);
        tag_t plane_3;
        flag=UF_CALL(UF_MODL_eval_exp("Sum_of_L_and_T_P_L", &point[2]));
		if(flag!=0) return (flag);
		flag=UF_MODL_create_fixed_dplane(point,direc,&plane_3);
		double location_3[3];
		double direc_3[3];
		direc_3[0]=0;
		direc_3[1]=0;
		direc_3[2]=-1;
		location_3[0]=0;
		location_3[1]=0;
		flag=UF_CALL(UF_MODL_eval_exp("Sum_of_L_and_T_P_L", &location_3[2]));
		if(flag!=0) return(flag);
		char tip_angle_3[10];
		tag_t collar_pin_hole;
		flag=UF_CALL(UF_MODL_create_simple_hole(location_3,direc_3,"Value_Dc_ID","Value_Dc_T_P_L","Collar_hole_tip_angle",plane_3,NULL_TAG, &collar_pin_hole));

   att_value.type = UF_ATTR_string;
   strcpy(svalue,"004");          /* First call for the part number */
   att_value.value.string = svalue;
          strcpy(svalue,"DRILL");          /* Second call for the description. */
   return (0);
}
```

Bit_sub_A // Code for bit sub

```
#define PART_PATH "D:\\Amar_Gobesona\\Nx_Open\\Journal_1\\Bit_sub_A.exp"

int Bit_sub_A(void)
{
		char svalue[10];      /* string for attribute value */
		UF_ATTR_value_t att_value;   /* attribute data structure */
		int flag=0;

		flag = UF_CALL(UF_MODL_import_exp(PART_PATH,0));
		if (flag) return flag;

		flag = UF_CALL (UF_MODL_update());
		if (flag) return flag;

				// creating a cylinder feature (pin) in the part. Set up data for cylinder creation.

		double Bit_sub_A_origin[3],
				Bit_sub_A_direction[3];
		tag_t Bit_sub_A_body,
				cyl_feature_Bit_sub_A_1;

		Bit_sub_A_origin[0] = 0.0;
		Bit_sub_A_origin[1] = 0.0;
		Bit_sub_A_origin[2] = 0.0;

		Bit_sub_A_direction[0] = 0.0;
		Bit_sub_A_direction[1] = 0.0;
		Bit_sub_A_direction[2] = 1.0;

		// Create the cylinder and check the return code
 flag = UF_CALL(UF_MODL_create_cyl1(UF_NULLSIGN, Bit_sub_A_origin, "Bit_sub_A_total_height",
"Bit_sub_A_OD", Bit_sub_A_direction, &cyl_feature_Bit_sub_A_1));
  if(flag!=0)
				uc1601("Error in drill_bit_pin creation",1);

		   flag = UF_CALL(UF_MODL_ask_feat_body(cyl_feature_Bit_sub_A_1, &Bit_sub_A_body) );
		flag = UF_CALL(UF_OBJ_set_name(Bit_sub_A_body, "LONG_CYL"));

				char msg[20], str[20];
		sprintf(str,"%d", Bit_sub_A_body);
		sprintf(msg,"%d", cyl_feature_Bit_sub_A_1);


		tag_t Bit_sub_A_face1, Bit_sub_A_face2;
```

```
        uf_list_p_t Bit_sub_A_list1, Bit_sub_A_list2;
        int Bit_sub_A_i,Bit_sub_A_count=0,Bit_sub_A_ftype,Bit_sub_A_dsense;
        double  Bit_sub_A_pt1[3],
                        Bit_sub_A_box[6],
                        Bit_sub_A_rad1,
                        Bit_sub_A_rad2;
        double Bit_sub_A_dir[3]={0,0,1};
        tag_t Bit_sub_A_edge, Bit_sub_A_cham_feature1;
        flag=UF_CALL(UF_MODL_ask_body_faces(Bit_sub_A_body,&Bit_sub_A_list1));
        if(flag) return (flag);
        flag=UF_CALL(UF_MODL_ask_list_count(Bit_sub_A_list1,&Bit_sub_A_count));
        if(flag) return(flag);
        double Bit_sub_A_cyl_len;
        flag=UF_CALL(UF_MODL_eval_exp("Bit_sub_A_total_height",&Bit_sub_A_cyl_len));
        if(flag!=0) return(flag);
        for(Bit_sub_A_i=0; Bit_sub_A_i<Bit_sub_A_count;Bit_sub_A_i++)
        {
                flag=UF_MODL_ask_list_item(Bit_sub_A_list1,Bit_sub_A_i,&Bit_sub_A_face1);
                if(flag!=0) return(flag);
        flag=UF_CALL(UF_MODL_ask_face_data
(Bit_sub_A_face1,&Bit_sub_A_ftype,Bit_sub_A_pt1,Bit_sub_A_dir,Bit_sub_A_box,&Bit_sub_A_rad1,&Bit_sub_A_rad
2,&Bit_sub_A_dsense));
        if(flag) return (flag);
        if(Bit_sub_A_ftype==UF_bounded_plane_type)
                {

                if(fabs(Bit_sub_A_pt1[2]-(Bit_sub_A_cyl_len))<0.001)

                        {
                                Bit_sub_A_face2=Bit_sub_A_face1;
                                break;
                        }
                }
        }
        flag=UF_MODL_ask_face_edges(Bit_sub_A_face2, &Bit_sub_A_list1);
        flag=UF_CALL(UF_MODL_create_chamfer(3,"Bit_sub_A_chamfer_offset_1","Bit_sub_A_chamfer_offset_2"
,"Bit_sub_A_chamfer_theta",Bit_sub_A_list1,&Bit_sub_A_cham_feature1));
        flag=UF_CALL(UF_MODL_delete_list(&Bit_sub_A_list1));

        tag_t Bit_sub_A_face3, Bit_sub_A_face4;
        tag_t Bit_sub_A_edge_2, Bit_sub_A_cham_feature2;
        flag=UF_CALL(UF_MODL_ask_body_faces(Bit_sub_A_body,&Bit_sub_A_list2));
        if(flag) return (flag);
        flag=UF_CALL(UF_MODL_ask_list_count(Bit_sub_A_list2,&Bit_sub_A_count));
        if(flag) return(flag);
        if(flag!=0) return(flag);
        for(Bit_sub_A_i=0; Bit_sub_A_i<Bit_sub_A_count;Bit_sub_A_i++)
        {

                flag=UF_MODL_ask_list_item(Bit_sub_A_list2,Bit_sub_A_i,&Bit_sub_A_face3);
                if(flag!=0) return(flag);
        flag=UF_CALL(UF_MODL_ask_face_data
(Bit_sub_A_face3,&Bit_sub_A_ftype,Bit_sub_A_pt1,Bit_sub_A_dir,Bit_sub_A_box,&Bit_sub_A_rad1,&Bit_sub_A_rad
2,&Bit_sub_A_dsense));
        if(flag) return (flag);
        if(Bit_sub_A_ftype==UF_bounded_plane_type)
                {

                if(fabs(Bit_sub_A_pt1[2])<0.001)

                        {
                                Bit_sub_A_face4=Bit_sub_A_face3;
                                break;
                        }
                }
        }
        flag=UF_MODL_ask_face_edges(Bit_sub_A_face4, &Bit_sub_A_list2);

        flag=UF_CALL(UF_MODL_create_chamfer(3,"Bit_sub_A_chamfer_offset_1","Bit_sub_A_chamfer_offset_2"
,"Bit_sub_A_chamfer_theta",Bit_sub_A_list2,&Bit_sub_A_cham_feature2));
```

```
            flag=UF_CALL(UF_MODL_delete_list(&Bit_sub_A_list2));
            double Bit_sub_A_point[3]={0.0,0,0};
            double Bit_sub_A_direc[3]={0,0,1};
            tag_t Bit_sub_A_fixed_dplane_4_hole,Bit_sub_A_plane;
            flag=UF_MODL_create_fixed_dplane(Bit_sub_A_point,Bit_sub_A_direc,&Bit_sub_A_fixed_dplane_4_hole);
            double Bit_sub_A_location_3[3]={0,0,0};
            double Bit_sub_A_direc_3[3];
            Bit_sub_A_direc_3[0]=0;
            Bit_sub_A_direc_3[1]=0;
            Bit_sub_A_direc_3[2]=1;
            char Bit_sub_A_tip_angle[10];
            tag_t Bit_sub_A_bore_hole;
            flag=UF_CALL(UF_MODL_create_simple_hole(Bit_sub_A_location_3,Bit_sub_A_direc_3,"Bit_sub_A_ID","
Bit_sub_A_total_height","Bit_sub_A_tip_angle",Bit_sub_A_fixed_dplane_4_hole,NULL_TAG,
&Bit_sub_A_bore_hole));
            double Bit_sub_A_point_4[3];
            flag=UF_CALL(UF_MODL_eval_exp("Bit_sub_A_total_height", &Bit_sub_A_point_4[2]));
            if(flag!=0) return (flag);
            Bit_sub_A_point_4[0]=0;
            Bit_sub_A_point_4[1]=0;
            double Bit_sub_A_direc_4[3];
            Bit_sub_A_direc_4[0]=0;
            Bit_sub_A_direc_4[1]=0;
            Bit_sub_A_direc_4[2]=-1;
            tag_t Bit_sub_A_fixed_dplane_4_box,Bit_sub_A_plane_4;
            flag=UF_MODL_create_fixed_dplane(Bit_sub_A_point_4,Bit_sub_A_direc_4,&Bit_sub_A_fixed_dplane_4_b
ox);

            double Bit_sub_A_location_4[3]={0,0,0};
            flag=UF_CALL(UF_MODL_eval_exp("Bit_sub_A_total_height", &Bit_sub_A_location_4[2]));
            if(flag!=0) return (flag);
            Bit_sub_A_location_4[0]=0;
            Bit_sub_A_location_4[1]=0;
            tag_t Bit_sub_A_bore_hole_4;

            flag=UF_CALL(UF_MODL_create_simple_hole(Bit_sub_A_location_4,Bit_sub_A_direc_4,"Bit_sub_A_box_
dia","Bit_sub_A_box_height","Bit_sub_A_tip_angle",Bit_sub_A_fixed_dplane_4_box,NULL_TAG,
&Bit_sub_A_bore_hole_4));

            double Bit_sub_A_point_5[3];
            Bit_sub_A_point_5[0]=0;
            Bit_sub_A_point_5[1]=0;
            Bit_sub_A_point_5[2]=0;
            double Bit_sub_A_direc_5[3];
            Bit_sub_A_direc_5[0]=0;
            Bit_sub_A_direc_5[1]=0;
            Bit_sub_A_direc_5[2]=1;
            tag_t Bit_sub_A_fixed_dplane_4_box_l,Bit_sub_A_plane_5;
            flag=UF_MODL_create_fixed_dplane(Bit_sub_A_point_5,Bit_sub_A_direc_5,&Bit_sub_A_fixed_dplane_4_b
ox_l)


            double Bit_sub_A_location_5[3]={0,0,0};
            tag_t Bit_sub_A_bore_hole_5;

            flag=UF_CALL(UF_MODL_create_simple_hole(Bit_sub_A_location_5,Bit_sub_A_direc_5,"Bit_sub_A_box_
dia","Bit_sub_A_box_height","Bit_sub_A_tip_angle",Bit_sub_A_fixed_dplane_4_box_l,NULL_TAG,
&Bit_sub_A_bore_hole_5)

            att_value.type = UF_ATTR_string;
    strcpy(svalue,"005");          /* First call for the part number */
    att_value.value.string = svalue;

    strcpy(svalue,"BIT_SUB_A");            /* Second call for the description. */
    return (0);


}
```

Drill_bit   //Code for drill bit

```
#define PART_PATH "D:\\Amar_Gobesona\\Nx_Open\\Journal_1\\Drill_bit.exp"
int Drill_bit(void)
{
        char svalue[10];      /* string for attribute value */
        UF_ATTR_value_t att_value;   /* attribute data structure */
        int flag=0;

        flag = UF_CALL(UF_MODL_import_exp(PART_PATH,0));
        if (flag) return flag;

        flag = UF_CALL (UF_MODL_update());
        if (flag) return flag;

        double drill_bit_origin[3],
                drill_bit_direction[3];
        tag_t drill_bit_body,
                cyl_feature_drill_bit_1;

        drill_bit_origin[0] = 0.0;
        drill_bit_origin[1] = 0.0;
        drill_bit_origin[2] = 0.0;

        drill_bit_direction[0] = 0.0;
        drill_bit_direction[1] = 0.0;
        drill_bit_direction[2] = 1.0;
        flag = UF_CALL(UF_MODL_create_cyl1(UF_NULLSIGN, drill_bit_origin, "Drill_bit_pin_height",
"Drill_bit_pin_dia", drill_bit_direction, &cyl_feature_drill_bit_1));
  if(flag!=0)
                uc1601("Error in drill_bit_pin creation",1);

   flag = UF_CALL(UF_MODL_ask_feat_body(cyl_feature_drill_bit_1, &drill_bit_body) );
        flag = UF_CALL(UF_OBJ_set_name(drill_bit_body, "LONG_CYL"));

        char msg[20], str[20];
        sprintf(str,"%d", drill_bit_body);
        sprintf(msg,"%d", cyl_feature_drill_bit_1);

        double Drill_bit_origin_2[3],
                Drill_bit_direction_2[3];
        tag_t cyl_feature_Drill_bit_2;

        flag=UF_CALL(UF_MODL_eval_exp("Drill_bit_pin_height", &Drill_bit_origin_2[2]));
        if(flag!=0) return (flag);
        Drill_bit_origin_2[0] = 0.0;
        Drill_bit_origin_2[1] = 0.0;


        Drill_bit_direction_2[0] = 0.0;
        Drill_bit_direction_2[1] = 0.0;
        Drill_bit_direction_2[2] = 1.0;

flag = UF_CALL(UF_MODL_create_cyl1(UF_POSITIVE,Drill_bit_origin_2, "Drill_bit_body_height",
"Drill_bit_body_dia", Drill_bit_direction_2, &cyl_feature_Drill_bit_2));
  if(flag!=0)
                uc1601("Error in drill bit body creation",1);

        double drill_bit_origin_3[3];
        double drill_bit_direction_3[3];
        char * drill_bit_diameter_3[2]={"Drill_bit_body_dia","Drill_bit_size"};

        tag_t drill_bit_cone;
        flag=UF_CALL(UF_MODL_eval_exp("Drill_bit_sum_pin_body", &drill_bit_origin_3[2]));
        if(flag!=0) return (flag);
        drill_bit_origin_3[0] = 0.0;
        drill_bit_origin_3[1] = 0.0;
        drill_bit_direction_3[0] = 0.0;
        drill_bit_direction_3[1] = 0.0;
        drill_bit_direction_3[2] = 1.0;
```

```
            flag=UF_CALL(UF_MODL_create_cone1(UF_POSITIVE,drill_bit_origin_3,"Drill_bit_cone_height",drill_bit
_diameter_3,drill_bit_direction_3, &drill_bit_cone));
            if (flag!=0)
            uc1601("Vhul hoise mamu in drill bit creation",1);

            double drill_bit_point[3]={0.0,0,0};
            double drill_bit_direc[3]={0,0,1};
            tag_t drill_bit_fixed_dplane_4_hole,drill_bit_plane;
            flag=UF_MODL_create_fixed_dplane(drill_bit_point,drill_bit_direc,&drill_bit_fixed_dplane_4_hole);


            double drill_bit_location_4[3]={0,0,0};
            double drill_bit_direc_4[3];
            drill_bit_direc_4[0]=0;
            drill_bit_direc_4[1]=0;
            drill_bit_direc_4[2]=1;
            char drill_bit_tip_angle[10];
            tag_t drill_bit_bore_hole;

            flag=UF_CALL(UF_MODL_create_simple_hole(drill_bit_location_4,drill_bit_direc_4,"Drill_bit_inner_dia","
Drill_bit_total_length","Drill_bit_tip_angle",drill_bit_fixed_dplane_4_hole,NULL_TAG, &drill_bit_bore_hole))

            att_value.type = UF_ATTR_string;
    strcpy(svalue,"006");           /* First call for the part number */
    att_value.value.string = svalue;

    strcpy(svalue,"DRILL_BIT");             /* Second call for the description. */
    return (0);
}
```

Drill_string_drill_pipe_array   // Code for drill pipe array

```
int Drill_string_drill_pipe_array(int dptotal, double dst_btn_dp, tag_t *parent, tag_t *comp)
//int Drill_string_drill_pipe_array(void)

{
            uc1601("Array is working",1);

char cname[132],                    /* component name */
 refset[132],                       /* reference set name*/
             iname[132],                        /* instance name*/
             pname[132];                        /* part name*/

int layer=0,
            i,k,
            flag,comp_count;
tag_t newinst, displayed_part, root_occ, *child_comp;
double pos[3],
              origin[3],
              matrix[9],
              transform[4][4];
UF_PART_load_status_t status;

displayed_part= UF_PART_ask_display_part();
root_occ=UF_ASSEM_ask_root_part_occ(displayed_part);
comp_count=UF_ASSEM_ask_part_occ_children(root_occ, &child_comp);

for (k=0; k<comp_count;k++)
{

flag=UF_CALL(UF_ASSEM_ask_component_data(child_comp[k],pname,refset,cname,pos,matrix,transform));
if(flag!=0) return (flag);
if(!(strcmp("DRILL_PIPE",cname))) break;/*Search the part by instance name*/
}

if(strcmp(refset,"None")==0) refset[0]='\0';

origin[0]=pos[0];
origin[1]=pos[1];
for(i=0;i<dptotal;i++)
{               if(i == 0) continue;
                origin[2]=pos[2]+i*dst_btn_dp;
                strcpy(iname,cname);
                sprintf(iname+strlen(cname),"_%d",i);
                flag=UF_ASSEM_add_part_to_assembly(*parent,pname,refset,cname,origin,matrix,layer,&newinst,&status);
                if(flag!=0) return(flag);
}

return(0);
}
```

B.2. Excel Codes:

Casing Setting Depth:

```
Sub casing_depth()


      Dim inputNo As Integer
      Dim Depth(100) As Integer, PorePressure(100) As Double ', SerialNo(100) As Integer
      Dim TripMargin As Double
      Dim KickMargin As Double
      Dim nue As Double
      Dim sigma_v As Double
      Dim FractureGradient As Double

    Sheets("Casing_cal").Select

      'Worksheets("Casing_cal").Range("b301").Value = InputBox("Enter Trip margin")
      'Worksheets("Casing_cal").Range("b302").Value = InputBox("Enter Kick margin")
      'Worksheets("Casing_cal").Range("b303").Value = InputBox("Enter Poisson's ratio")
      'Worksheets("Casing_cal").Range("b304").Value = InputBox("Enter Overburde in psi/ft")


      TripMargin = Worksheets("Casing_cal").Range("b301").Value
      KickMargin = Worksheets("Casing_cal").Range("b302").Value
      nue = Worksheets("Casing_cal").Range("b303").Value
      sigma_v = Worksheets("Casing_cal").Range("b304").Value
      inputNo = Worksheets("Casing_cal").Range("b305").Value


      'inputNo = InputBox("Enter number of input")

Inconsistant_data_1:

      Worksheets("Casing_cal").Range("b3").Value = InputBox("Enter True Vertical Depth")
      For i = 1 To inputNo

      Depth(i) = InputBox("Please enter Depth in ft: Inuput number - " & i)
      Cells(i + 310, 2) = Depth(i)
      Cells(i + 310, 9) = Depth(i) 'For chart creation

      '--------------------------------------------------------------------
      'Inconsistant data handling (If the entered data is not in ascending order the system will give _
      and error message - "Please enter data in ascending order" and the input window will start again _
      from the very begining, that is start from the first data input..i=1)

      If i > 1 Then

         If Cells(i - 1 + 310, 2) > Cells(i + 310, 2) Then
         MsgBox ("Please enter data in ascending order")
         GoTo Inconsistant_data_1
         Else
         End If
      Else
      End If
```

```
        PorePressure(i) = InputBox("Enter corresponding Pore pressure: Inuput number - " & i)
        Cells(i + 310, 3) = PorePressure(i)

        'Pore pressure gradient

        Cells(i + 310, 4) = PorePressure(i) / Depth(i)

        'comment: PorePressure = Cells(i + 310, 4)


        ' Specific gravity of Pore pressure

       Cells(i + 310, 5) = Cells(i + 310, 4) / (8.33 * 0.052)

        'Mud pressure Specific gravity

        Cells(i + 310, 6) = Cells(i + 310, 5) + TripMargin

        '/comments'Fracture pressure gradient
        'Eaton's method
        'FG=(nue/1-nue)((sigma_v/D-Pf/D)+Pf/D; here, nue=poison's ratio; sigma_v=overburden psi/ft, Pf=Pore pressure
        ' sigma_v/D=Over_burden_stress, Pf/D=Pore_pressure_grad//

        FractureGradient = (nue / (1 - nue)) * ((sigma_v - Cells(i + 310, 4))) + Cells(i + 310, 4)
         Cells(i + 310, 7) = FractureGradient / (8.33 * 0.052)

        'Adding kick margin of fracture pressure gradient

        Cells(i + 310, 8) = Cells(i + 310, 7) - KickMargin

        Next


        For j = inputNo + 1 To 100
        Cells(j + 310, 1).Clear
        Cells(j + 310, 2).Clear
        Cells(j + 310, 3).Clear
        Cells(j + 310, 4).Clear
        Cells(j + 310, 5).Clear
        Cells(j + 310, 6).Clear
        Cells(j + 310, 7).Clear
        Cells(j + 310, 8).Clear
        Cells(j + 310, 9).Clear
        Cells(j + 310, 10).Clear
        Cells(j + 310, 11).Clear


        Next
'-----------------------------------------------------------------------------------------------------------------------------


Dim minm_depth_diff_btn_surf_intermediate_casing As Double

'------------------------------------------------------------------------
'Default value for minimum depth difference between surface and intermediate casing

minm_depth_diff_btn_surf_intermediate_casing = 100
'------------------------------------------------------------------------

If Cells(inputNo + 310, 2) = Worksheets("Casing_cal").Range("b3").Value Then
Temp_mud_sg_1 = Cells(inputNo + 310, 6)
Temp_kick_margin_sg_1 = Temp_mud_sg_1

'----------------------------------------------------------------------------------
'Check whether the value of tepm_mud_Sg_1 is less than the minimum kick margin sg????

  If Temp_mud_sg_1 < Cells(1 + 310, 8) Then
  minm_surface_casing_depth = InputBox("Please enter the minimum surface casing depth")
  Worksheets("Casing_cal").Range("b24").Value = minm_surface_casing_depth
    If Worksheets("Casing_cal").Range("b24").Value < Worksheets("Casing_cal").Range("b3").Value Then
```

```
       If Worksheets("Casing_cal").Range("b3").Value - minm_surface_casing_depth <
minm_depth_diff_btn_surf_intermediate_casing Then
       surface_casing_depth = Worksheets("Casing_cal").Range("b3").Value
       Worksheets("Casing_cal").Range("p327").Value = surface_casing_depth
       Worksheets("Casing_cal").Range("p328").Value = Temp_mud_sg_1
       Worksheets("Casing_cal").Range("p312").Value = 0
       Worksheets("Casing_cal").Range("p313").Value = 0
       Worksheets("Casing_cal").Range("p317").Value = 0
       Worksheets("Casing_cal").Range("p318").Value = 0
       Worksheets("Casing_cal").Range("p322").Value = 0
       Worksheets("Casing_cal").Range("p323").Value = 0

       '------------------------------------------------------------------
       'Finish the design job

       GoTo final_design

       '------------------------------------------------------------------

       Else

       surface_casing_depth = minm_surface_casing_depth
       Worksheets("Casing_cal").Range("p327").Value = surface_casing_depth
       Worksheets("Casing_cal").Range("p328").Value = Temp_mud_sg_1
       'Producation casing
  'Casing shoe depth (TVD)
       Worksheets("Casing_cal").Range("p312").Value = Worksheets("Casing_cal").Range("b3").Value

  'Required mud specific gravity at production casing

       Worksheets("Casing_cal").Range("p313").Value = Temp_mud_sg_1
      '------------------------------------------------------------------
      'Other sections of the casing is assigned zero value

       Worksheets("Casing_cal").Range("p317").Value = 0
       Worksheets("Casing_cal").Range("p318").Value = 0
       Worksheets("Casing_cal").Range("p322").Value = 0
       Worksheets("Casing_cal").Range("p323").Value = 0
       '------------------------------------------------------------------
       'Finish the design job

       GoTo final_design

       '------------------------------------------------------------------
       End If


    Else
       If Worksheets("Casing_cal").Range("b24").Value = Worksheets("Casing_cal").Range("b3").Value Then
       surface_casing_depth = Worksheets("Casing_cal").Range("b3").Value
       Worksheets("Casing_cal").Range("p327").Value = surface_casing_depth
       Worksheets("Casing_cal").Range("p328").Value = Temp_mud_sg_1
       Worksheets("Casing_cal").Range("p312").Value = 0
       Worksheets("Casing_cal").Range("p313").Value = 0
       '-------------------------------------------------------------------
      'Other sections of the casing is assigned zero value

       Worksheets("Casing_cal").Range("p317").Value = 0
       Worksheets("Casing_cal").Range("p318").Value = 0
       Worksheets("Casing_cal").Range("p322").Value = 0
       Worksheets("Casing_cal").Range("p323").Value = 0
       '------------------------------------------------------------------
       '------------------------------------------------------------------
       'Finish the design job

       GoTo final_design

       '------------------------------------------------------------------
       Else
```

```
         MsgBox ("Minimum surface casing depth is greater than the TVD")
         Worksheets("Casing_cal").Range("p327").Value = Worksheets("Casing_cal").Range("b3").Value
         Worksheets("Casing_cal").Range("p328").Value = Temp_mud_sg_1

         '-----------------------------------------------------------------
        'Other sections of the casing is assigned zero value
         Worksheets("Casing_cal").Range("p312").Value = 0
         Worksheets("Casing_cal").Range("p313").Value = 0
         Worksheets("Casing_cal").Range("p317").Value = 0
         Worksheets("Casing_cal").Range("p318").Value = 0
         Worksheets("Casing_cal").Range("p322").Value = 0
         Worksheets("Casing_cal").Range("p323").Value = 0
         '----------------------------------------------------------------
         '----------------------------------------------------------------
         'Finish the design job

         GoTo final_design

         '----------------------------------------------------------------
         End If
      End If
   Else

   End If

'---------------------------------------------------------------------------------
'Producation casing
'Casing shoe depth (TVD)
   Worksheets("Casing_cal").Range("p312").Value = Worksheets("Casing_cal").Range("b3").Value

'Required mud specific gravity at production casing

   Worksheets("Casing_cal").Range("p313").Value = Temp_mud_sg_1
'-------------------------------------------------------------------------------------------------------------------
'If the required specific gravity is out of range the system will show an error message. the code is as follows(No need,
alredy protected)

  ' If Temp_kick_margin_sg_1 < Cells(1 + 310, 8) Or Temp_kick_margin_sg_1 > Cells(inputNo + 310, 8) Then
   'MsgBox ("Kick marging Specific gravity is out of range at production casing ")
  ' Else
   'End If
'--------------------------------------------------------------------------------------------------------------------

   For i = 1 To inputNo

   If Cells(i + 310, 8) = Temp_kick_margin_sg_1 Then
   Depthe_2 = Cells(i + 310, 2)

   Else

      If Cells(i + 310, 8) < Temp_kick_margin_sg_1 And Cells(i + 1 + 310, 8) > Temp_kick_margin_sg_1 Then

      Temp_value_1_1 = Cells(i + 310, 8)
      Temp_value_1_2 = Cells(i + 1 + 310, 8)
      Temp_value_1_3 = Cells(i + 310, 2)
      Temp_value_1_4 = Cells(i + 1 + 310, 2)
      depth_2 = Int(((Temp_kick_margin_sg_1 - Temp_value_1_1) * (Temp_value_1_4 - Temp_value_1_3) /
(Temp_value_1_2 - Temp_value_1_1)) + Temp_value_1_3)
      Worksheets("Casing_cal").Range("p317").Value = depth_2

   '------------------------------------------------------------------------------------------------------------------
     ' Worksheets("Casing_cal").Range("n310").Value = Temp_value_1_1
      'Worksheets("Casing_cal").Range("n311").Value = Temp_value_1_2
      'Worksheets("Casing_cal").Range("n312").Value = Temp_value_1_3
      'Worksheets("Casing_cal").Range("n313").Value = Temp_value_1_4
   '-------------------------------------------------------------------------------------------------------------------

      Else

      End If
   End If

   Next

'Casing shoe depth
'Comment: Worksheets("Casing_cal").Range("p317").Value = depth_2

   For i = 1 To inputNo

   If Cells(i + 310, 2) = depth_2 Then
   Temp_mud_sg_2 = Cells(i + 310, 6)

   Else

      If Cells(i + 310, 2) < depth_2 And Cells(i + 1 + 310, 2) > depth_2 Then

      Temp_value_2_1 = Cells(i + 310, 2)
      Temp_value_2_2 = Cells(i + 1 + 310, 2)
      Temp_value_2_3 = Cells(i + 310, 6)
      Temp_value_2_4 = Cells(i + 1 + 310, 6)
      Temp_mud_sg_2 = ((depth_2 - Temp_value_2_1) * (Temp_value_2_4 - Temp_value_2_3) / (Temp_value_2_2 -
Temp_value_2_1)) + Temp_value_2_3

      'Required mud specific gravity at production casing

      Worksheets("Casing_cal").Range("p318").Value = Temp_mud_sg_2

      '-------------------------------------------------------------------

      '[Worksheets("Casing_cal").Range("n314").Value = Temp_value_2_1
     ' Worksheets("Casing_cal").Range("n315").Value = Temp_value_2_2
      'Worksheets("Casing_cal").Range("n316").Value = Temp_value_2_3
      'Worksheets("Casing_cal").Range("n317").Value = Temp_value_2_4]
     '----------------------------------------------------------------------

      Else

      End If
   End If

   Next

'------------------------------------------------------------------------------------
'Check whether the value of tepm_mud_Sg_2 is less than the minimum kick margin sg????

   If Temp_mud_sg_2 < Cells(1 + 310, 8) Then
   minm_surface_casing_depth = InputBox("Please enter the minimum surface casing depth")
   Worksheets("Casing_cal").Range("b24").Value = minm_surface_casing_depth
      If Worksheets("Casing_cal").Range("b24").Value < Worksheets("Casing_cal").Range("p317").Value Then
         If depth_2 - Worksheets("Casing_cal").Range("b24").Value < minm_depth_diff_btn_surf_intermediate_casing
Then
         surface_casing_depth = Worksheets("Casing_cal").Range("p317").Value
         Worksheets("Casing_cal").Range("p327").Value = surface_casing_depth
         Worksheets("Casing_cal").Range("p328").Value = Temp_mud_sg_2
         '-----------------------------------------------------------------
         'Other sections of the casing is assigned to zero value

         Worksheets("Casing_cal").Range("p317").Value = 0
         Worksheets("Casing_cal").Range("p318").Value = 0
         Worksheets("Casing_cal").Range("p322").Value = 0
         Worksheets("Casing_cal").Range("p323").Value = 0
         '----------------------------------------------------------------
         'Finish the design job

         GoTo final_design

         '----------------------------------------------------------------


         Else
```

```
            surface_casing_depth = Worksheets("Casing_cal").Range("b24").Value
            Worksheets("Casing_cal").Range("p327").Value = surface_casing_depth
            Worksheets("Casing_cal").Range("p328").Value = Temp_mud_sg_2
            Worksheets("Casing_cal").Range("p317").Value = depth_2
            Worksheets("Casing_cal").Range("p318").Value = Temp_mud_sg_2
            '------------------------------------------------------------------
            'Other sections of the casing is assigned to zero value

            Worksheets("Casing_cal").Range("p322").Value = 0
            Worksheets("Casing_cal").Range("p323").Value = 0
            '-----------------------------------------------------------------
            'Finish the design job

            GoTo final_design

            '-----------------------------------------------------------------
            End If


        Else
            If Worksheets("Casing_cal").Range("b24").Value = Worksheets("Casing_cal").Range("p317").Value Then
            'minm_surface_casing_depth=depth_2

            surface_casing_depth = depth_2
            Worksheets("Casing_cal").Range("p327").Value = surface_casing_depth
            Worksheets("Casing_cal").Range("p328").Value = Temp_mud_sg_2

            '------------------------------------------------------------------
            'Other sections of the casing is assigned to zero value

            Worksheets("Casing_cal").Range("p317").Value = 0
            Worksheets("Casing_cal").Range("p318").Value = 0
            Worksheets("Casing_cal").Range("p322").Value = 0
            Worksheets("Casing_cal").Range("p323").Value = 0
            '-----------------------------------------------------------------
            'Finish the design job

            GoTo final_design

            '-----------------------------------------------------------------
            Else
            MsgBox ("You have a problem, the minimum surface casing depth is greater than the intermediate section")

            'Option should be provided

            '-----------------------------------------------------------------
            'Finish the design job

            GoTo final_design

            '-----------------------------------------------------------------

            End If
        End If
    Else

    End If

'Kick margin

Temp_kick_margin_sg_2 = Temp_mud_sg_2


'-----------------------------------------------------------------------------------------------------------------------------
'If the required specific gravity is out of range, the system will show an error message. the code is as follows
    If Temp_kick_margin_sg_2 < Cells(1 + 310, 8) Or Temp_kick_margin_sg_2 > Cells(inputNo + 310, 8) Then
    MsgBox ("Please check the input")
    Else
    End If
'--------------------------------------------------------------------------------------------------------------------------------


    For i = 1 To inputNo

    If Cells(i + 310, 8) = Temp_kick_margin_sg_2 Then
    Depthe_3 = Cells(i + 310, 2)

    Else

        If Cells(i + 310, 8) < Temp_kick_margin_sg_2 And Cells(i + 1 + 310, 8) > Temp_kick_margin_sg_2 Then

        Temp_value_3_1 = Cells(i + 310, 8)
        Temp_value_3_2 = Cells(i + 1 + 310, 8)
        Temp_value_3_3 = Cells(i + 310, 2)
        Temp_value_3_4 = Cells(i + 1 + 310, 2)
        depth_3 = Int(((Temp_kick_margin_sg_2 - Temp_value_3_1) * (Temp_value_3_4 - Temp_value_3_3) /
(Temp_value_3_2 - Temp_value_3_1)) + Temp_value_3_3)
        Worksheets("Casing_cal").Range("p322").Value = depth_3

    '-----------------------------------------------------------------------------------------------------------------------------
       ' Worksheets("Casing_cal").Range("n318").Value = Temp_value_1_1
        'Worksheets("Casing_cal").Range("n319").Value = Temp_value_1_2
        'Worksheets("Casing_cal").Range("n320").Value = Temp_value_1_3
        'Worksheets("Casing_cal").Range("n321").Value = Temp_value_1_4
    '------------------------------------------------------------------------------------------------------------------------------

        Else

        End If
    End If

    Next

'Casing shoe depth
'Comment: Worksheets("Casing_cal").Range("p322").Value = depth_3

    For i = 1 To inputNo

    If Cells(i + 310, 2) = depth_3 Then
    Temp_mud_sg_3 = Cells(i + 310, 6)

    Else

        If Cells(i + 310, 2) < depth_3 And Cells(i + 1 + 310, 2) > depth_3 Then

        Temp_value_4_1 = Cells(i + 310, 2)
        Temp_value_4_2 = Cells(i + 1 + 310, 2)
        Temp_value_4_3 = Cells(i + 310, 6)
        Temp_value_4_4 = Cells(i + 1 + 310, 6)
        Temp_mud_sg_3 = ((depth_3 - Temp_value_4_1) * (Temp_value_4_4 - Temp_value_4_3) / (Temp_value_4_2 -
Temp_value_4_1)) + Temp_value_4_3

        'Required mud specific gravity at production casing

        Worksheets("Casing_cal").Range("p323").Value = Temp_mud_sg_3

        '-------------------------------------------------------------------------

        'Worksheets("Casing_cal").Range("n322").Value = Temp_value_2_1
        'Worksheets("Casing_cal").Range("n323").Value = Temp_value_2_2
        'Worksheets("Casing_cal").Range("n324").Value = Temp_value_2_3
        'Worksheets("Casing_cal").Range("n325").Value = Temp_value_2_4
       '---------------------------------------------------------------------------

        Else

        End If
    End If
    Next

'-----------------------------------------------------------------------------------
'Check whether the value of tepm_mud_Sg_3 is less than the minimum kick margin sg????

    If Temp_mud_sg_3 < Cells(1 + 310, 8) Then
    minm_surface_casing_depth = InputBox("Please enter the minimum surface casing depth")
        If Worksheets("Casing_cal").Range("b24").Value < Worksheets("Casing_cal").Range("p322").Value Then
        'minm_surface_casing_depth < depth_3

            If Worksheets("Casing_cal").Range("p322").Value - Worksheets("Casing_cal").Range("b24").Value < minm_depth_diff_btn_surf_intermediate_casing Then
            'epth_3 - minm_surface_casing_depth < minm_depth_diff_btn_surf_intermediate_casing

            surface_casing_depth = depth_3
            Worksheets("Casing_cal").Range("p327").Value = surface_casing_depth
            Worksheets("Casing_cal").Range("p328").Value = Temp_mud_sg_3

            '-------------------------------------------------------------------
            'Other sections of the casing is assigned to zero value

            Worksheets("Casing_cal").Range("p322").Value = 0
            Worksheets("Casing_cal").Range("p323").Value = 0
            '-------------------------------------------------------------------
            'Finish the design job

            GoTo final_design

            '-------------------------------------------------------------------
            Else
            surface_casing_depth = minm_surface_casing_depth
            Worksheets("Casing_cal").Range("p327").Value = surface_casing_depth
            Worksheets("Casing_cal").Range("p328").Value = Temp_mud_sg_3

            Worksheets("Casing_cal").Range("p322").Value = depth_3
            Worksheets("Casing_cal").Range("p323").Value = Temp_mud_sg_3

            '-------------------------------------------------------------------
            'Finish the design job

            GoTo final_design

            '-------------------------------------------------------------------
            End If


        Else
            If Worksheets("Casing_cal").Range("b24").Value = Worksheets("Casing_cal").Range("p322").Value Then
            'minm_surface_casing_depth = depth_3
            surface_casing_depth = depth_3
            Worksheets("Casing_cal").Range("p327").Value = surface_casing_depth
            Worksheets("Casing_cal").Range("p328").Value = Temp_mud_sg_3

             '-------------------------------------------------------------------
            'Other sections of the casing is assigned to zero value

            Worksheets("Casing_cal").Range("p322").Value = 0
            Worksheets("Casing_cal").Range("p323").Value = 0
            '-------------------------------------------------------------------
            'Finish the design job

            GoTo final_design

            '-------------------------------------------------------------------

            Else
            'the minimum casing setting depth is greater than the depth_3, so _
            we havto set casing setting depth=minimum casing setting depth and will find out the corresponding _
            required mud specific gravity
```

```
      Worksheets("Casing_cal").Range("p327").Value = minm_surface_casing_depth
      'surface casing depth=minimum surface casing depth
      'have to find out the corresponding mud sg

      For i = 1 To inputNo

        If Cells(i + 310, 2) = minm_surface_casing_depth Then
        Temp_mud_sg_4 = Cells(i + 310, 6)

         Else

          If Cells(i + 310, 2) < minm_surface_casing_depth And Cells(i + 1 + 310, 2) > minm_surface_casing_depth
Then

          Temp_value_5_1 = Cells(i + 310, 2)
          Temp_value_5_2 = Cells(i + 1 + 310, 2)
          Temp_value_5_3 = Cells(i + 310, 6)
          Temp_value_5_4 = Cells(i + 1 + 310, 6)
          Temp_mud_sg_4 = ((minm_surface_casing_depth - Temp_value_5_1) * (Temp_value_5_4 -
Temp_value_5_3) / (Temp_value_5_2 - Temp_value_5_1)) + Temp_value_5_3

          'Required mud specific gravity at production casing

          Worksheets("Casing_cal").Range("p328").Value = Temp_mud_sg_4

          Else

          End If
        End If

       Next


      '-------------------------------------------------------------------
      'Other sections of the casing is assigned to zero value

      Worksheets("Casing_cal").Range("p322").Value = 0
      Worksheets("Casing_cal").Range("p323").Value = 0
      '-------------------------------------------------------------------
      'Finish the design job

      GoTo final_design

      '-------------------------------------------------------------------
      End If
    End If
  Else

  End If

Else

MsgBox ("Please enter the pore pressure at the True Vertical Depth (TVD)")

'If the pore pressure is not given in the TVD then the system will return again to the _
input window.

GoTo Inconsistant_data_1

'Please note that, this go to option will be changed when I will generate an independent input module
final_design:

End If

'Input for surface casing programs-
'Sheets("Casing_cal").Activate

'From Casing_cal:
'Section Depth
Sheets("Casing_cal").Range("b4").Formula = Sheets("Casing_cal").Range("p327").Formula
```

```
'Mud SG
Sheets("Casing_cal").Range("b5").Formula = Sheets("Casing_cal").Range("p328").Formula

'Input for Intermediate casing 1 programs-
'Sheets("Casing_Cal_Intermediate_2").Activate
'From Casing_cal:
'TVD
Sheets("Casing_Cal_Intermediate_2").Range("b3").Formula = Sheets("Casing_cal").Range("b3").Formula
'Section Depth
Sheets("Casing_Cal_Intermediate_2").Range("b4").Formula = Sheets("Casing_cal").Range("p322").Formula
'Mud SG
Sheets("Casing_Cal_Intermediate_2").Range("b5").Formula = Sheets("Casing_cal").Range("p323").Formula
'Input fof Intermediate casing 2 programs-
'From Casing_cal:
'TVD
Sheets("Casing_Cal_Intermediate_1").Range("b3").Formula = Sheets("Casing_cal").Range("b3").Formula
'Section Depth
Sheets("Casing_Cal_Intermediate_1").Range("b4").Formula = Sheets("Casing_cal").Range("p317").Formula
'Mud SG
Sheets("Casing_Cal_Intermediate_1").Range("b5").Formula = Sheets("Casing_cal").Range("p318").Formula

'Input for Production programs-
'From Casing_cal:
'TVD
Sheets("Casing_Cal_Production").Range("b3").Formula = Sheets("Casing_cal").Range("b3").Formula
'Section Depth
Sheets("Casing_Cal_Production").Range("b4").Formula = Sheets("Casing_cal").Range("p312").Formula
'Mud SG
Sheets("Casing_Cal_Production").Range("b5").Formula = Sheets("Casing_cal").Range("p313").Formula
'Formation pressure at the casing shoe
Sheets("Casing_Cal_Production").Range("b8").Formula =
'Input for surface hole -
'From Casing_cal:
'Hole Size
Sheets("D_String_Design_Surface").Range("b13").Formula = Sheets("Casing_cal").Range("p330").Formula
'Casing Size
Sheets("D_String_Design_Surface").Range("b14").Formula = Sheets("Casing_cal").Range("p329").Formula
'Depth of surface section
Sheets("D_String_Design_Surface").Range("b15").Formula = Sheets("Casing_cal").Range("p327").Formula
'Mud specific gravity
Sheets("D_String_Design_Surface").Range("b16").Formula = Sheets("Casing_cal").Range("p328").Formula

'Input for Intermediate 1 hole-
'From Casing_cal:
'Hole Size
Sheets("D_String_Design_Intermediate_1").Range("b13").Formula = Sheets("Casing_cal").Range("p320").Formula
'Casing Size
Sheets("D_String_Design_Intermediate_1").Range("b14").Formula = Sheets("Casing_cal").Range("p319").Formula
'Depth of Intermediate_1 section
Sheets("D_String_Design_Intermediate_1").Range("b15").Formula = Sheets("Casing_cal").Range("p317").Formula
'Mud specific gravity
Sheets("D_String_Design_Intermediate_1").Range("b16").Formula = Sheets("Casing_cal").Range("p318").Formula

'From Casing_cal:
'Hole Size
Sheets("D_String_Design_Intermediate_2").Range("b13").Formula = Sheets("Casing_cal").Range("p325").Formula
'Casing Size
Sheets("D_String_Design_Intermediate_2").Range("b14").Formula = Sheets("Casing_cal").Range("p324").Formula
'Depth of Intermediate_2 section
Sheets("D_String_Design_Intermediate_2").Range("b15").Formula = Sheets("Casing_cal").Range("p322").Formula
'Mud specific gravity
Sheets("D_String_Design_Intermediate_2").Range("b16").Formula = Sheets("Casing_cal").Range("p323").Formula

'From Casing_cal:
'Hole Size
Sheets("D_String_Design_Production").Range("b13").Formula = Sheets("Casing_cal").Range("p315").Formula
'Casing Size
Sheets("D_String_Design_Production").Range("b14").Formula = Sheets("Casing_cal").Range("p314").Formula
'Depth of Intermediate_1 section
Sheets("D_String_Design_Production").Range("b15").Formula = Sheets("Casing_cal").Range("p312").Formula
```

```
'Mud specific gravity
Sheets("D_String_Design_Production").Range("b16").Formula = Sheets("Casing_cal").Range("p313").Formula
End Sub
```

Casing Design:

```
Sub Surface_Casing()
Sheets("Casing_cal").Select

'Input from previous programs-
'From Casing_cal:

'Section Depth
'Sheets("Casing_cal").Range("b4").Formula = Sheets("Casing_cal").Range("p327").Formula
'Mud SG
'Sheets("Casing_cal").Range("b5").Formula = Sheets("Casing_cal").Range("p328").Formula
'------------------------------------------------------------------------------------
'Available surface casing input
Dim Available_surface_csg As Integer
Available_surface_csg = Cells(19, 2)
For avl_casg = 0 To Available_surface_csg
Cells(101 + avl_casg, 2) = Cells(101 + avl_casg, 42)
Cells(101 + avl_casg, 3) = Cells(101 + avl_casg, 43)
Cells(101 + avl_casg, 4) = Cells(101 + avl_casg, 44)
Cells(101 + avl_casg, 5) = Cells(101 + avl_casg, 45)
Cells(101 + avl_casg, 6) = Cells(101 + avl_casg, 46)
Cells(101 + avl_casg, 7) = Cells(101 + avl_casg, 47)
Cells(101 + avl_casg, 8) = Cells(101 + avl_casg, 48)
Cells(101 + avl_casg, 16) = Cells(101 + avl_casg, 49)
Next
'--------------------------------------------------------------------------------------
'Suface:
'Collapse Load Calculation
'Variables
Dim s_cg_clps_load_shoe As Double
Dim s_cg_clps_load_srfce As Double
Dim s_cg_dsgn_clps_load_shoe As Double
Dim s_cg_dsgn_clps_load_srfce As Double
Dim s_cg_slope_clps_load As Double
Dim s_cg_df_clps As Double

' Buoyancy Factor Calculations
Cells(8, 10) = 1 - (Cells(5, 2) * 8.34) / 65.5
'Assign design factor:
s_cg_df_clps = Worksheets("Casing_cal").Range("b9").Value
' collapse pressure surface 0 lbf/in^2
s_cg_clps_load_srfce = 0 'lbf/in^2
'Designed collapse pressure (after multiplying the Design Factor)
'Surface
Worksheets("Casing_cal").Range("j3").Value = s_cg_clps_load_srfce * s_cg_df_clps
'Collapse Pressure at casing shoe (lbf/in^2)=external pressure_at shoe - internal pressure at shoe
'external pressure_at shoe =0.052*SG*8.33*h lbf/in^2
'internal pressure at shoe = 0 lbf/in^2
s_cg_clps_load_shoe = 0.052 * Worksheets("Casing_cal").Range("b5").Value *
Worksheets("Casing_cal").Range("b4").Value * 8.33
'Designed collapse pressure (after multiplying the Design Factor)
'Casing Shoe
s_cg_dsgn_clps_load_shoe = s_cg_clps_load_shoe * s_cg_df_clps
'Make the designed collapse pressure value to nearest tenth to round off
If (Int(s_cg_dsgn_clps_load_shoe) Mod 10) > 5 Then
s_cg_dsgn_clps_load_shoe = (Int(s_cg_dsgn_clps_load_shoe / 10) + 1) * 10
Else: s_cg_dsgn_clps_load_shoe = (Int(s_cg_dsgn_clps_load_shoe / 10)) * 10
End If
Worksheets("Casing_cal").Range("j4").Value = s_cg_dsgn_clps_load_shoe
MsgBox "Collapse load is calculated"
'Casing selection
'slope(m) of the designed collapse line
'Equation of straight line going through the origin y=mx, m=(y_2-y_1)/(x_2-x_1)
'Here, x_1,y_1=0
'so, m=y_2/x_2; y_2=Depth of casing shoe, x_2= Collapse pressure at the casing shoe
s_cg_slope_clps_load = Worksheets("Casing_cal").Range("b4").Value / Worksheets("Casing_cal").Range("j4").Value
```

```
'Depth up to which the casing is capable to sustain safely, y=mx
'have to change the condition
For i = 1 To Cells(19, 2)
Cells(100 + i, 9) = 0

If Cells(100 + i, 6) = 0 Then
Cells(100 + i, 10) = 0
Else
Cells(100 + i, 10) = Int(s_cg_slope_clps_load * Cells(100 + i, 6))
End If
Next
' Burst load
'fracture pressure=0.052*SG*8.33*depth
'please note that, another option should be provided for fracture pressure calculations, like if the provided input is
'fracture gradient than the system can calculate the pressure
'Burst load at the casing shoe
'variables
Dim s_cg_intr_brst_load_shoe As Double
Dim s_cg_extr_brst_load_shoe As Double
Dim s_cg_brst_load_shoe As Double
Dim s_cg_dsgn_brst_load_shoe As Double
Dim s_cg_df_brst As Double
'Internal pressure at casing shoe = Fracture pressure+Injection pressure
s_cg_Intr__brst_load_shoe = 0.052 * Worksheets("Casing_cal").Range("b8").Value * 8.33 *
Worksheets("Casing_cal").Range("b4").Value _
+ Worksheets("Casing_cal").Range("b14").Value
'External pressure at casing shoe = Fresh water pressure at casing shoe
s_cg_extr_brst_load_shoe = 0.052 * 8.33 * Worksheets("Casing_cal").Range("b4").Value
'Burst load at casing shoe
s_cg_brst_load_shoe = s_cg_Intr__brst_load_shoe - s_cg_extr_brst_load_shoe
'Assignment of design factor for burst
s_cg_df_brst = Worksheets("Casing_cal").Range("b10").Value

'Designed Burst load at casing shoe
s_cg_dsgn_brst_load_shoe = s_cg_brst_load_shoe * s_cg_df_brst

'Make the designed collapse pressure value to nearest tenth to round off
   If (Int(s_cg_dsgn_brst_load_shoe) Mod 10) > 5 Then
   s_cg_dsgn_brst_load_shoe = (Int(s_cg_dsgn_brst_load_shoe / 10) + 1) * 10
   Else: s_cg_dsgn_brst_load_shoe = (Int(s_cg_dsgn_brst_load_shoe / 10)) * 10
   End If
'Designed Burst load at casing shoe
Worksheets("Casing_cal").Range("j7").Value = s_cg_dsgn_brst_load_shoe
'Burst load at the casing surface
'Variables
Dim s_cg_tmp_at_casing_shoe As Double
Dim s_cg_tmp_average As Double
Dim s_cg_brst_load_surface As Double
Dim s_cg_dsgn_brst_load_surface As Double
'Temperature at casing shoe=Tempt at surface + (casing shoe depth/total vertical depth)*(Temp at TVD - Temp at surface)
s_cg_tmp_at_casing_shoe = Worksheets("Casing_cal").Range("b6").Value +
(Worksheets("Casing_cal").Range("b4").Value / _
Worksheets("Casing_cal").Range("b3").Value) * (Worksheets("Casing_cal").Range("b7").Value -
Worksheets("Casing_cal").Range("b6").Value)
'Average temperature and conversion of temperature to Rankin scale to farenhite scale = (Tempt at surface+Temp at casing
shoe)/2 +460
s_cg_tmp_average = (Worksheets("Casing_cal").Range("b6").Value + s_cg_tmp_at_casing_shoe) / 2 + 460 'temperature at
Rankin scale
s_cg_brst_load_surface = s_cg_Intr__brst_load_shoe * Exp((Worksheets("Casing_cal").Range("b17").Value * (0 -
Worksheets("Casing_cal").Range("b4").Value)) _
/ (Worksheets("Casing_cal").Range("b18").Value * s_cg_tmp_average * Worksheets("Casing_cal").Range("b16").Value))

s_cg_dsgn_brst_load_surface = s_cg_brst_load_surface * s_cg_df_brst

'Make the designed collapse pressure value to nearest tenth to round off

   If (Int(s_cg_dsgn_brst_load_surface) Mod 10) > 5 Then
   s_cg_dsgn_brst_load_surface = (Int(s_cg_dsgn_brst_load_surface / 10) + 1) * 10
   Else: s_cg_dsgn_brst_load_surface = (Int(s_cg_dsgn_brst_load_surface / 10)) * 10
   End If
```

```
Worksheets("Casing_cal").Range("j6").Value = s_cg_dsgn_brst_load_surface


'Casing selection
'Length upto which the casing is capable to sustain the burst load
'Evaluate the  casing

For i = 1 To Cells(19, 2)
If Cells(100 + i, 7) > Cells(6, 10) Then
Cells(100 + i, 12) = 0
Cells(100 + i, 13) = Cells(4, 2)
Else
Cells(100 + i, 12) = Int((s_cg_dsgn_brst_load_surface - Cells(100 + i, 7)) * (Cells(4, 2)) / (s_cg_dsgn_brst_load_surface -
s_cg_dsgn_brst_load_shoe))
Cells(100 + i, 13) = Cells(4, 2)
End If
Next

Dim casing_1_start_depth As Double
Dim casing_1_end_depth As Double
For i = 1 To 10
Cells(100 + i, 14) = maximum(Cells(100 + i, 9), Cells(100 + i, 12))
Cells(100 + i, 15) = minimum(Cells(100 + i, 10), Cells(100 + i, 13))
Next

'-------------------------------------------------------------------------------------------------------------------
'Maximum depth that can be cased by using the available casings - Part 1
Dim nuber_of_available_casing As Double
Dim maximum_depth_can_b_cased As Double

nuber_of_available_casing = Worksheets("Casing_cal").Range("b19").Value
maximum_depth_can_b_cased = Cells(100 + Worksheets("Casing_cal").Range("b19").Value, 15)
Cells(118, 16) = maximum_depth_can_b_cased
If Cells(118, 16) > Worksheets("Casing_cal").Range("b4").Value Or _
Cells(118, 16) = Worksheets("Casing_cal").Range("b4").Value Then
GoTo Casing_selection

'Clear the cells:
For i = 1 To 20
Cells(100 + i, 21).Clear
Cells(100 + i, 22).Clear
   For j = 1 To 20
   Cells(120 + i, j + 1).Clear
   Next
Next
'------------------------
'Break Points
For i = 1 To Worksheets("Casing_cal").Range("b19").Value
   If Cells(100 + i, 15) - Cells(100 + i, 14) > Worksheets("Casing_cal").Range("b12").Value Or _
   Cells(100 + i, 15) - Cells(100 + i, 14) = Worksheets("Casing_cal").Range("b12").Value Then
   Cells(120 + i, 2) = Cells(100 + i, 14) ' First Break points
    Cells(120 + i, 3) = Cells(100 + i, 15) ' second break points
   End If
Next
'--------------------------
For i = 1 To Worksheets("Casing_cal").Range("b19").Value * 2

Cells(120 + 2 * i - 1, 4) = Cells(120 + i, 2)
Cells(120 + 2 * i, 4) = Cells(120 + i, 3)
Next

'----------------------------------------------
' Sorting the Break points
Dim b_itr As Integer ' Number of break points
b_itr = Worksheets("Casing_cal").Range("b19").Value * 2


   ActiveWorkbook.Worksheets("Casing_cal").Sort.SortFields.Clear
   ActiveWorkbook.Worksheets("Casing_cal").Sort.SortFields.Add Key:=Range(Cells(121, 4), Cells(b_itr + 120, 4)), _
```

```
    SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
  With ActiveWorkbook.Worksheets("Casing_cal").Sort
    .SetRange Range(Cells(121, 4), Cells(b_itr + 120, 4))
    .Header = xlGuess
    .MatchCase = False
    .Orientation = xlTopToBottom
    .SortMethod = xlPinYin
    .Apply
 End With
'-------------------------------------
'Available Candidates
For i = 1 To b_itr
  k = 1
  For j = 1 To Worksheets("Casing_cal").Range("b19").Value
  If (Cells(100 + j, 14) < Cells(120 + i, 4) Or Cells(100 + j, 14) = Cells(120 + i, 4)) And _
  Cells(100 + j, 15) > Cells(120 + i, 4) Or Cells(100 + j, 15) = Cells(120 + i, 4) And _
  (Cells(100 + j, 15) - Cells(100 + j, 14) > Worksheets("Casing_cal").Range("b12").Value Or _
  Cells(100 + j, 15) - Cells(100 + j, 14) = Worksheets("Casing_cal").Range("b12").Value) Then
  Cells(120 + i, 4 + k) = j
  k = k + 1
  End If
  Next

Cells(120 + i, 15) = k - 1 'Number of candidates
Next

'--------------------------------------------
'Selection of casing

'Number of casing used, m
'Casing_covered=0,cells(119,17)
'r=break_point[]
'Total length =
Dim b2b As Integer
Dim m As Integer
Dim r As Double
Dim casing_covered As Double

m = 0
casing_covered = 0
b2b = 1
r = Cells(121, 4)
casing_covered = Cells(119, 17)
        '(Check whether the casing setting depth is less than the minimum casing section then the program will show a
message)
        If Worksheets("Casing_cal").Range("b4").Value < Worksheets("Casing_cal").Range("b12").Value Then
        MsgBox ("The casing setting depth is less than the minimum casing section")
        GoTo out_of_program
        Else
        GoTo start_the_program
        End If

start_the_program:

Do While casing_covered < Worksheets("Casing_cal").Range("b4").Value
  If Worksheets("Casing_cal").Range("b4").Value - Cells(100 + m, 22) > Worksheets("Casing_cal").Range("b12").Value
Or _
  Worksheets("Casing_cal").Range("b4").Value - Cells(100 + m, 22) = Worksheets("Casing_cal").Range("b12").Value
Then
        If m > 10 Then ' Limit the value of m to less then 10
        GoTo out_of_program 'system will out from the program
        Else
        GoTo inside_the_program 'system will run inside the program
        End If

inside_the_program:

 For i = 1 To Cells(120 + b2b, 15)

    If Cells(100 + Cells(120 + b2b, i + 4), 15) - r > Worksheets("Casing_cal").Range("b12").Value Or _
```

```
        (Cells(100 + (Cells(120 + b2b, i + 4)), 15) - r) = Worksheets("Casing_cal").Range("b12").Value Then
        Cells(100 + m + 1, 21) = Cells(120 + b2b, i + 4)
        Cells(100 + m + 1, 22) = Cells(120 + b2b, 4)


            If Cells(120 + b2b + 1, 4) > r + Worksheets("Casing_cal").Range("b12").Value Then
            Cells(100 + m + 1, 22) = Cells(120 + b2b + 1, 4) 'C_starts
            r = Cells(100 + m + 1, 22)
            Else
            Cells(100 + m + 1, 22) = r + Worksheets("Casing_cal").Range("b12").Value
            r = Cells(100 + m + 1, 22)
            End If
            GoTo abc

        End If

            If Cells(100 + m, 21) = Cells(120 + b2b, i + 4) Then
            Cells(100 + m + 1, 21) = Cells(100 + m, 21)
            Cells(100 + m + 1, 22) = Cells(120 + b2b, 4)
            End If

    Next
abc:

itr = Cells(119, 19)

        itr = b2b

        Do While Cells(120 + itr, 4) < Cells(100 + m + 1, 22) Or _
        Cells(120 + itr, 4) = Cells(100 + m + 1, 22) And _
        (itr < (b_itr - 1) Or itr = b_itr - 1)
        itr = itr + 1
        Loop


    Else
    Cells(100 + m, 22) = Worksheets("Casing_cal").Range("b4").Value

        If Cells(120 + b2b + 1, 4) = Worksheets("Casing_cal").Range("b4").Value Then
        Cells(100 + m, 21) = Cells(120 + b2b + 1, 1 + 4)
        Else
            If Cells(120 + b2b + 2, 4) = Worksheets("Casing_cal").Range("b4").Value Then
            Cells(100 + m, 21) = Cells(120 + b2b + 2, 1 + 4)
            Else
                If Cells(120 + b2b + 3, 4) = Worksheets("Casing_cal").Range("b4").Value Then
                Cells(100 + m, 21) = Cells(120 + b2b + 3, 1 + 4)
                Else
                    If Cells(120 + b2b + 4, 4) = Worksheets("Casing_cal").Range("b4").Value Then
                    Cells(100 + m, 21) = Cells(120 + b2b + 4, 1 + 4)
                    Else
                    MsgBox ("Please check again")
                    End If
                End If
            End If
        End If



    End If
    b2b = itr - 1
    casing_covered = Cells(100 + m + 1, 22)
    m = m + 1


Loop

out_of_program:

Else
MsgBox ("The available casing is not capable to case total depth")
```

```
For i = 1 To 20
Cells(100 + i, 21).Clear
Cells(100 + i, 22).Clear
    For j = 1 To 20
    Cells(120 + i, j + 1).Clear
    Next


Next
End If
'-------------------------------------------------------------------


'Axial load calculation

                        '--------------------------------
                        Dim counter_1 As Double
                        counter_1 = 0
                        For i = 1 To 20
                        If Cells(100 + i, 21) > 0 Then
                        counter_1 = counter_1 + 1
                        Cells(98, 22) = counter_1
                        Else
                        End If
                        Next

For i = 1 To 10
    For j = 1 To 36
    Cells(50 + i, 6 + j).Clear
    Next
Next
'----------------------------

Dim counter_2 As Double
counter_2 = 0
For i = 1 To counter_1
    If Cells(100 + i, 21) = Cells(100 + 1 + i, 21) Then
    counter_2 = counter_2 + 1
    Cells(97, 22) = counter_2 'keep the counter_2 value in the correspondin cells
    Else
    Cells(50 + i - counter_2, 9) = Cells(100 + i, 21)
    Cells(50 + i - counter_2, 15) = Cells(100 + i, 22)
    End If
Next

Cells(51, 14) = 0 'Define the start value

                        'Define another counter to count the number of sections
                        Dim counter_3 As Double
                        counter_3 = 0
                        For i = 1 To 20
                        If Cells(50 + i, 9) > 0 Then
                        counter_3 = counter_3 + 1
                        Cells(96, 22) = counter_3
                        Else
                        End If
                        Next

'---------------------------------------
'Start Depth
Cells(51, 14) = 0 'Define the start value
For i = 1 To counter_3 - 1
Cells(51 + i, 14) = Cells(50 + i, 15)
Next
'--------------------------------------
'Length:
For i = 1 To counter_3
Cells(50 + i, 16) = Cells(50 + i, 15) - Cells(50 + i, 14)

'Weight
```

```
Cells(50 + i, 10) = Cells(100 + Cells(50 + i, 9), 2)

'Grade
Cells(50 + i, 11) = Cells(100 + Cells(50 + i, 9), 3)

'Connection
Cells(50 + i, 12) = Cells(100 + Cells(50 + i, 9), 4)

'Inner Diameter
Cells(50 + i, 13) = Cells(100 + Cells(50 + i, 9), 5)

'Air weight:
Cells(50 + i, 17) = Cells(50 + i, 10) * Cells(50 + i, 16)

'Buoyed weight
Cells(50 + i, 18) = Int(Cells(50 + i, 17) * Cells(8, 10))

'Collapse rating
Cells(50 + i, 23) = Cells(100 + Cells(50 + i, 9), 6)

'Burst rating
Cells(50 + i, 24) = Cells(100 + Cells(50 + i, 9), 7)

'Tensile rating
Cells(50 + i, 25) = Cells(100 + Cells(50 + i, 9), 8)

'Unit cost

Cells(50 + i, 33) = Cells(100 + Cells(50 + i, 9), 16)

'Total cost

Cells(50 + i, 34) = Cells(50 + i, 33) * Cells(50 + i, 16)
Next

'Total Cost of the surface casing

'Cumulative weight

If counter_3 > 1 Then
For i = 1 To (counter_3 - 1)
'Cumulative air weight

Cells(50 + counter_3, 19) = Cells(50 + counter_3, 17)
Cells(50 + counter_3 - i, 19) = Cells(50 + counter_3 - i + 1, 19) + Cells(50 + counter_3 - i, 17)

'Cumulative buoyed weight

Cells(50 + counter_3, 20) = Cells(50 + counter_3, 18)
Cells(50 + counter_3 - i, 20) = Cells(50 + counter_3 - i + 1, 20) + Cells(50 + counter_3 - i, 18)
Next
Else
Cells(50 + counter_3, 19) = Cells(50 + counter_3, 17)
Cells(50 + counter_3, 20) = Cells(50 + counter_3, 18)
End If

' Design Safety factor Calculations
For i = 1 To counter_3

'Clear the color shade
Cells(50 + i, 32).Select
   With Selection.Interior
       .Pattern = xlSolid
       .PatternColorIndex = xlAutomatic
       .ThemeColor = xlThemeColorDark1
       .TintAndShade = 0
       .PatternTintAndShade = 0
    End With

Cells(50 + i, 32) = Cells(50 + i, 25) / Cells(50 + i, 19)
```

```
If Cells(50 + i, 32) < Cells(11, 2) Then

Cells(50 + i, 32).Select
    With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .Color = 255
        .TintAndShade = 0
        .PatternTintAndShade = 0
    End With
    MsgBox ("Axial load is not satisfied")
Else
End If
Next

End Sub
```

Code for expression file creation

```
Sub Expression_creatoin()

Sheets("D_pipe_internal_external_upset").Select
ActiveWorkbook.SaveAs Filename:= _
"C:\Documents and
Settings\shahmd\Desktop\Thesis\Draft_version\Draft_version_2\Drill_pipe_internal_external_upset_expression.exp",
FileFormat:= _
        xlTextMSDOS, CreateBackup:=False
    ActiveWorkbook.Save

 Sheets("Tool_joint_pin").Select
ActiveWorkbook.SaveAs Filename:= _
"C:\Documents and Settings\shahmd\Desktop\Thesis\Draft_version\Draft_version_2\Tool_joint_pin", FileFormat:= _
        xlTextMSDOS, CreateBackup:=False
    ActiveWorkbook.Save

 Sheets("Drill_collar").Select
ActiveWorkbook.SaveAs Filename:= _
"C:\Documents and Settings\shahmd\Desktop\Thesis\Draft_version\Draft_version_2\Drill_collar", FileFormat:= _
        xlTextMSDOS, CreateBackup:=False
    ActiveWorkbook.Save

 Sheets("Bit_sub_A").Select
ActiveWorkbook.SaveAs Filename:= _
"C:\Documents and Settings\shahmd\Desktop\Thesis\Draft_version\Draft_version_2\Bit_sub_A", FileFormat:= _
        xlTextMSDOS, CreateBackup:=False
    ActiveWorkbook.Save

 Sheets("Drill_bit").Select
ActiveWorkbook.SaveAs Filename:= _
"C:\Documents and Settings\shahmd\Desktop\Thesis\Draft_version\Draft_version_2\Drill_bit", FileFormat:= _
        xlTextMSDOS, CreateBackup:=False
    ActiveWorkbook.Save

Sheets("Array").Select
ActiveWorkbook.SaveAs Filename:= _
"C:\Documents and Settings\shahmd\Desktop\Thesis\Draft_version\Draft_version_2\Array.exp", FileFormat:= _
        xlTextMSDOS, CreateBackup:=False
    ActiveWorkbook.Save

End Sub
```