A CONTEXT-AWARE INFERENCE SYSTEM, TO CAPTURE DESIGN RATIONALE FROM LEGACY MCAD

by

GANESHRAM RAMJI IYER

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2007

ACKNOWLEDGEMENTS

I would like to first and foremost thank my wife for all her support and understanding during the entire duration of my doctoral degree. Without her support and encouragement this degree would not have been possible.

I would also like to specially thank both Dr. Venkat Devarajan and Dr. John Mills for their guidance, advice and support through the course of research work. I also need to express gratitude towards my esteemed dissertation committee Dr. Lawrence, Dr. Subbarao, Dr. Agonafer and Dr. Dogan for their inputs. Gratitude is also due to Dr. Ram Sriram of NIST for his inputs on the dissertation documentation.

I would also like to thank Aditya Deshmukh and Dr. Sunil Belligundu for the help they provided with the design rationale capture analysis process.

October 2, 2007

ABSTRACT

A CONTEXT-AWARE INFERENCE SYSTEM,

TO CAPTURE DESIGN RATIONALE

FROM LEGACY CAD

Publication No. _____

Ganeshram Ramji Iyer, PhD.

The University of Texas at Arlington, 2007

Supervising Professor: Dr. Venkat Devarajan

There exist numerous design rationale systems that convert captured information into structured design rationale while providing rationale representation and retrieval. These systems woefully neglect the design rationale that is present in legacy CAD such as 2D drawings and 3D models. The dissertation addresses the issues that arise when dealing with the capture, representation and retrieval of design rationale from the 2D legacy CAD data, specifically the non-form related data (e.g. text and symbols). A definition for design rationale in the CAD domain is presented which forms the basis of the proposed approach. The approach uses a unique context-aware inference system to capture design rationale from legacy CAD data. A brief explanation

of context is provided along with the advantages of using context for this task. The need and use of an inference system is detailed. Additionally a prototype system is implemented to address these issues from a software system point of view. A verification process is suggested that will validate the design rationale captured by the system to that captured by human re-designers.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background

The use of CAD/CAE in design documentation and modelling is becoming ubiquitous [1]. Feature-based CAD systems have demonstrated clear potential for creating attractive design environments and facilitating geometric reasoning related to design function, performance evaluation, manufacturing process planning, NC programming and other engineering tasks. In the last decade, interest in design rationale systems has grown. Design rationale systems are important tools because they can include not only the reasons behind a design decision but also the justification for it, the other alternatives considered, the tradeoffs evaluated, and the argumentation that led to the decision. The use of a design rationale system - a tool for capturing and making design rationale easily accessible - can thus improve dependency management, collaboration, reuse, maintenance, learning, and documentation. On the down side, while it was expected that solid modelling or design rationale systems would replace drafting systems in design, this turned out to not be the case. Even today, most CAD applications are based on two-dimensional drafting. Shah [2] states that the reason for this failure is the deficiency of the geometric modelling tools. As the design and the manufacturing process evolve around the geometric shape of the product, the current

generation of CAD systems is based on geometric modelling techniques. These techniques have proved to be deficient as their usefulness is limited to recording the embodiment detail of the product. Unfortunately designers no longer merely exchange geometric data but need to share more general information about the product such as the design rationale, constraints, specifications and manufacturing knowledge. As design becomes increasingly knowledge intensive, the need for computational frameworks to effectively support the formal representation, capture, retrieval and reuse of product knowledge/design rationale, becomes more critical [3]. Commercial and governmental entities looking to use design rationale systems to improve their product development process, have to deal with the bulk of the design rationale that resides in their current design data, such as the 2-dimensional drawings. This design rationale needs to be propagated to a more reusable, intelligent and structured format such as those used by design rationale or knowledge-based systems.

1.2 What is MCAD and legacy MCAD?

Before stating the overall problem section provides the necessary background on Mechanical Computer-aided Design (MCAD) and legacy MCAD. MCAD normally refers to geometry authoring tools primarily used to create detailed designs in numerous mechanical domains such as automotive, aerospace, ship building etc. Legacy MCAD refers to the two main formats viz. 2-dimensional (2D) drawings and 3-dimensional (3D) models used by MCAD tools to store detailed designs. 2D drawings contain CAD entities such as points, lines, arcs, circles, splines etc. in addition to text and symbols. 3D models on the other hand are composed of CAD entities such as edges, surfaces,

solids etc. If the 3D model is parametric then it also contains features, parts and sub-assemblies which are composed of the lower level CAD entities mentioned previously. The text contained in the legacy CAD indicate notes, materials, dimensions, tolerances, company-, project-, design-, designer info, surface finish etc. The geometry indicates shape, alternatives, dimensions, tolerances etc.

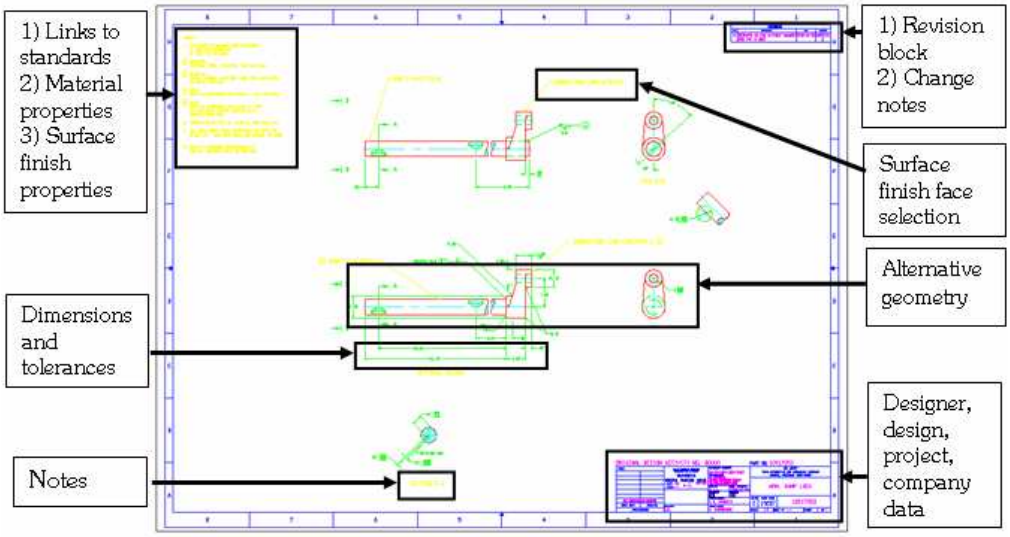

Figure 1.1: Representative 2D drawing with relevant elements highlighted

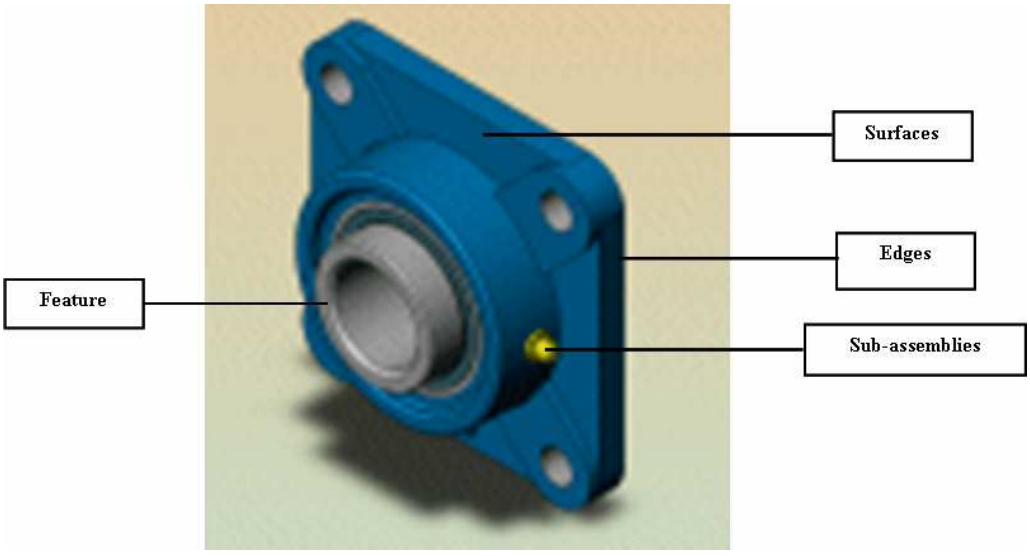

Figure 1.2: Representative 3D model with relevant elements highlighted

There are numerous CAD systems that support these two formats. Autodesk's AutoCAD software is probably the most used 2D drawing product in the market with Autodesk's proprietary DWG the most common storage file format. Autodesk has another file format termed the Drawing Exchange Format (DXF) that is ASCII based. The 3D market has many players such as Dassault Systemes' CATIA and SolidWorks, UGS, PTC's Pro/Engineer, Autodesk's Inventor etc.

This dissertation addresses 2D formats, although the methods and hypothesis stated could easily apply to 3D formats too. With regards to file formats the proposed approach should work with all the proprietary formats of the commercial systems though intermediate translators may have to be written to allow entity extraction. The two file formats directly targeted are the DXF and the DWG. The DXF file primarily stores 2D drawings and although 3D is also possible in DXF the current dissertation does not address that specific format. To handle the DWG file format, existing software provided by Autodesk to convert the DWG file to DXF is used. There is no loss of necessary information in this conversion process and hence deemed acceptable. In addition to serving as a storage format for geometry, text and symbols the DXF file also provides manageability objects such as groups and layers. To provide a brief explanation of these manageability objects consider the need to separate entities based on their type i.e. by placing the geometry and text on different layers the user can use the properties of the layers such as visibility to view either the layer containing geometry or the layer containing text or both simultaneously.

1.3 Overall Problem

The origin of the problem addressed in this dissertation comes from the U.S Army the Tank Automotive Research, Development and Engineering Centre's (TARDEC) Tank Automotive and Armaments Command (TACOM) department. It was claimed that the manufacturing team from TACOM were reluctant to move to 3-dimensional (3D) solid models from 2-dimensional (2D) drawings. One primary reason stated was the lack of design rationale in the 3D models that was available in 2D drawings. The lack of this information in the 3D models may have something to do with the manner in which these 3D models were created compared with 2D drawings.

2D CAD systems such as AutoCAD primarily provide the drafters with tools to replicate in digital format the paper and Mylar based blueprints that used to store design artifact information. These paper and Mylar based blueprints were considered as storehouses of all design decisions and information with regards to the various artifacts that were designed. Using a variety of techniques such as scanning of paper blueprints to digital raster formats and then converting the raster formats to vector formats using software such as VPHybridCAD©. These vector formats were usually available in commonly available file formats such as AutoCAD's proprietary DWG or the DXF format, which were both primarily 2D. Since the newly created vector-based, digital 2D drawings were facsimiles of the original paper blueprints these new 2D drawings were now the storehouses of the design decisions that were contained in the paper blueprints. But 2D drawings and the CAD tools used to create them e.g. AutoCAD had many problems viz. possibility of creating ambiguous geometry, non-parametric geometry,

non-associative dimensions etc. To address these problems 3D modeling MCAD (Mechanical Computer-aided design/drafting) systems were introduced that created parametric, unambiguous, feature-based models. To take advantage of these new functionalities designers and corporations began converting their 2D drawings to 3D models. But 3D MCAD systems use a very different layout of the artifacts as compared to the 2D drawings. The 3D CAD systems focus primarily on the geometry while incorporating the non-geometric elements in arbitrary manners. Additionally the process of converting 2D drawings to 3D models largely ignored the non-geometric information stored in the 2D drawings.

These two reasons account for the lack of the non-geometric information in the 3D models that was present in the 2D drawings but the claim made by TACOM manufacturers was that they could capture design rationale from 2D drawings that they could not from 3D models. To validate this claim and address the need of a method to move design rationale from 2D drawings to 3D models, a Small Business Innovation Research (SBIR) proposal was submitted in collaboration with Imagecom Inc, which was awarded to Imagecom Inc in 2004.

The overall problem that this dissertation addresses stems from this SBIR proposal i.e. how can we capture design rationale from legacy drawings assuming that a valid rationale exists behind any information that is included on legacy MCAD ?

1.4 Key Issues and Hypothesis

The previous section (section 1.2) briefly provided the overall problem. From this overall problem statement we can identify a few questions that need to be answered which are:

- What is design rationale?
- What is design rationale in the domain of legacy MCAD?
- Why is it important to capture design rationale from legacy MCAD?

These questions are answered in Chapter 2 (Related Research) in section 2.3. The key issues that are addressed in this dissertation are:

- Existing design rationale capture methods do not address legacy CAD.
- Addressing the primary key issue requires us to address secondary issues regarding scope when dealing with legacy CAD viz.
    - a. What percent of all legacy MCAD can we address?
    - b. What percent of design rationale on legacy CAD can we capture?
    - c. Does using standard legacy CAD formats (ASME, ISO) aid in addressing scope?

These key issues are addressed systematically in the subsequent chapters. In an effort to address these key issues the dissertation describes a new method by proving the following hypothesis:

"LEGACY CAD DESIGN RATIONALE CAN BE CAPTURED BY IDENTIFYING THE SURROUNDING CONTEXT."

The subsequent chapters also detail the steps needed to prove the hypothesis. Chapter 2 provides the related research regarding the key ideas such as “legacy CAD design rationale” and “context”.

1.5 Scope of Research

The idea of capturing design rationale from legacy MCAD is not new but is treated quite differently from the current state of the research in legacy MCAD. Chapter 2 provides the required related research on the definition of design rationale in legacy MCAD and its relationship to the definition of design rationale in general and why it is more important to treat design rationale in legacy MCAD in this manner rather than treat it in a manner consistent with current state of research. The scope can be addressed by providing the answers to the following questions:

- What percent of all legacy CAD can we address?

This dissertation is limited to Mechanical Engineering and more specifically to machined piece parts and sub-assemblies. System level CAD is ignored.

- What percent of design rationale on legacy CAD can we capture?

The answer to this lies not in the proposed approach but in comparing the rationale captured by the suggested method to that captured by an experienced re-designer. The system that is built with this dissertation cannot capture more rationale than an experienced re-designer as the system will then have to be rather extensive with unlimited scope to match the experience and knowledge that the re-designer has access to. But in general the system will perform better than an inexperienced re-designer if it has access to well defined rule-bases. Additionally, as detailed later the percent of

design rationale that can be captured will increase with continued input from experienced re-designers.

- Does using standard legacy CAD formats (ASME, ISO) aid in addressing scope?

It is generally easier to extract context from well defined CAD formats, rather than random representation formats. While theoretically the system should be able to parse CAD candidates at the same level as that possible by a human, in practice the maturity of the system will decide its accuracy. A well defined format aids in limiting the scope of applicable legacy CAD to the following:

- Groups of related (by company, project, design) legacy CAD increases percent of successful context extraction.
- Related, formal legacy CAD increases probability of inference: The system that is proposed in this dissertation has been developed to address legacy CAD files belonging to a single project or company that have standardized layout formats rather than dealing with individual unrelated CAD files.

1.6 Outline of dissertation

The following is a brief outline of the rest of the chapters in this dissertation.

Chapter 2 provides the required related research. Starting with the need for design rationale chapter 2 provides the state of the research in design rationale, its capture, representation and retrieval. With an understanding of design rationale in general, the chapter then details legacy MCAD design rationale, its definition,

importance and applicability of current suggested capture, representation and retrieval methods. Chapter 2 also provides the state of the research on the idea of context and specifically the concepts of context in design and context in legacy MCAD. To better understand context chapter 2 provides the detail of the nature, structure, importance and impact of context in both design and legacy MCAD.

Chapter 3 details the approach proposed to address the hypothesis stated in section 1.4. The first step of the approach is an analysis process to identify the process and rationale that can be captured by human candidates. The goal of the analysis is to identify the nature of a software system that is developed as a part of this dissertation to address the hypothesis in as automated a manner as possible.

Chapter 4 details the software architecture proposed to automate the capture of design rationale from legacy MCAD. This chapter also briefly describes the system implementation details such as the programming languages, file formats and databases used to achieve automation.

Finally, in chapter 5, a validation approach is proposed to prove that the design rationale that is captured by such a system is the same as that can be captured by human candidates. During this validation step the design rationale captured by the two means is compared to prove the quality of the design rationale captured by the software system.

CHAPTER 2

RELATED RESEARCH

This chapter provides the related research and background required for the rest of the dissertation. The topics covered in this chapter are design documentation, design rationale and more specifically design rationale in the domain of legacy MCAD, context in design and context in legacy MCAD.

2.1 Design Documentation

With any design there is a need for design documentation. Design documentation is primarily used to store the data/information that is generated during the design process. Design documentation is a storehouse of the final design or can be seen as a snapshot of the final decisions of the design process. Re-designers use design documentation for any required design changes and/or improvements while manufacturers use the documentation for production purposes, manufacturing instructions, material selection etc. There are many advantages and disadvantages to using design documentation methods and tools.

2.1.1 Advantages of Design Documentation

Design documentation and tools that support documentation are rather ubiquitous and fairly easy to use. Existing, specialized tools such as computer-aided design (CAD), computer-aided manufacturing (CAM), product data management

(PDM) and product lifecycle management (PLM) in addition to general productivity tools such as Microsoft Word or Excel support design documentation very well. The specialized tools focus on advanced geometry representation and manufacturing simulation and planning while the general tools serve as storehouses of standards, design tables and calculations etc. Due to the nature and availability of documentation tools almost all product design tasks are accompanied by the outputs of these tools providing ad-hoc standards to store design decisions e.g. using spreadsheets for design calculations.

2.1.2 Disadvantages of Design Documentation

While design documentation is considered very valuable it has some notable disadvantages. Design documentation tends to get very voluminous and most times has an unstructured format. The completeness of design documentation relies on the designer. If the designer does not do a thorough job of providing necessary details the documentation remains incomplete. If the department or the company does not provide a formal, standard method for documentation then the documentation format tends to be rather informal, once again dependent on the designers to define completeness and consistency. All these reasons make the process of maintaining and querying the documentation rather expensive both economically and temporally. But the primary reasons why design documentation is considered insufficient are:

- It does not store the reasons or justifications for a particular design decision
- It does not also store the alternatives that were explored during the design process and the reasons for their rejection. Even if the designer does state

the justifications for a design decision over an alternative, this information is not easily available and very rarely stored with the original design itself.

2.2 Design Rationale

To address these disadvantages of design documentation researchers suggest storing the rationale behind a design decision in addition to the data or information pertaining to the design decision. This stored rationale is commonly referred to as design rationale, design intent or design history. Design rationale stores design decisions along with their reasons and justifications in addition to the alternatives explored and the reasons and justifications for their rejection. Thus design rationale provides both argumentation (a way to query for the reason behind a particular decision) and communication (a way to store the design discourse viz. the design space explored) in addition to documentation (storing design data/information). Various design rationale systems have been developed since the early 1980's. The research has ranged from basic observations about the design process to different approaches to capturing design rationale [4]. The results of the research in design rationale have been to suggest definitions for design rationale in addition to developing, approaches to design rationale systems, representation schema for design rationale, approaches to capture design rationale and, design rationale retrieval strategies. Some suggested definitions include:

"Design rationale expresses elements of the reasoning which has been invested behind the design of an artifact" [5].

"Design rationale is the reasoning and argument that leads to the final decision of how the design intent is achieved." "Design intent is the `expected' effect or behavior that the designer intended the design object should achieve to fulfill the required function." [6]

"Design rationale means statements of reasoning underlying the design process that explain, derive, and justify design decisions." [7]

Design rationale means "information that explains why an artifact is structured the way that it is and has the behavior that it has" [8].

"Design rationales include not only the reasons behind a design decision but also the justification for it, the other alternatives considered, the tradeoffs evaluated, and the argumentation that led to the decision" [9].

Lee's [9] definition is used as the basis for this entire dissertation.

2.2.1 Advantages of Design Rationale

It is well accepted within the design community that the availability of explicit, declaratively represented design rationale would be a tremendous asset. Design rationale would serve as a record of the basic structure of a design, codifying how the design satisfies specified requirements, as well as key decisions that were made during the design process. This information would facilitate collaboration among multiple distributed designers – a tremendous benefit for large-scale design efforts. Rationale would also provide guidance in exploring alternative designs, whether as part of the natural evolution of a design or in response to changing requirements. Finally, design rationale would enable easier maintenance of artifacts over their life cycles and more

effective reuse of designs by making it easier for downstream engineers to understand how a design works [10].

2.2.2 Disadvantages of Design Rationale

While considerable effort has been put into developing design rationale systems, none of these systems has been adopted for widespread industrial use [4]. Molavi and his colleagues state that this failure is above all due to the costs and disruptions associated with capturing and structuring of the design rationale [33]. They contend that although there have been some individual success stories of applying design rationale systems in practice, almost every one of these has been associated with heroic efforts by a solitary champion of design rationale within the successful project. There are few or no published indications that significant design rationale projects have been continued past the pilot project stage [11]. Most design rationale systems are still in the laboratory stage because further research and development is needed to focus on the advancements needed to take the science to the level at which it can be effectively deployed in industry [4].

2.2.3 Overview of Design Rationale Systems

Figure 2.1 [4] shows the flow of data through a general design rationale system. The next few sub-sections provide a summary of the background on the commonly used terms with respect to design rationale viz. approaches, capture, representation and retrieval.

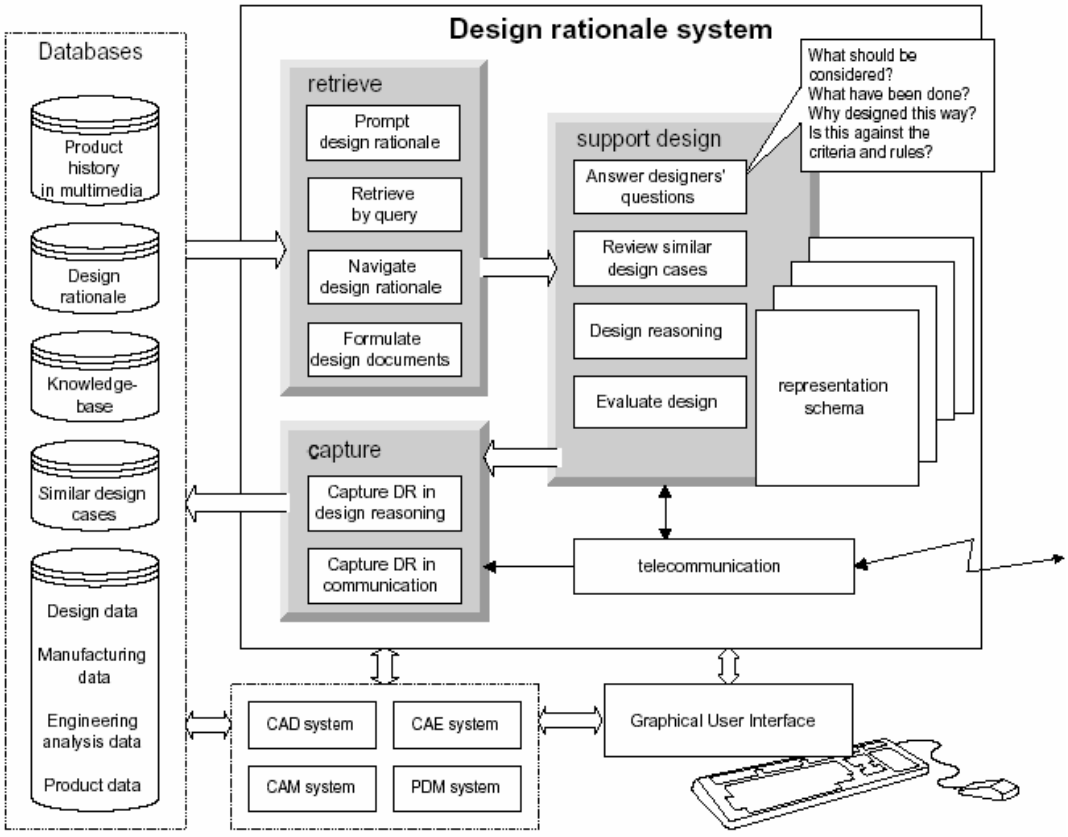

Figure 2.1: General architecture of Design Rationale systems [4]

2.2.3.1 Approaches to building design rationale systems

The main approaches to design rationale systems are *process-oriented* and *feature-oriented*. In dynamic design domains the process-oriented approach is used to give historical representation of artifacts while in fields with a relatively high degree of standardization, the feature-oriented approach is used to give logical representation of artifacts, to follow the rigorous and logical rules of the design process.

- Process-oriented approaches emphasize the design rationale as a history of the design process. Most design rationale approaches are process-oriented. The representation schema of process-oriented rationale system is generally

graph-based using nodes and links, with nodes indicating possible issues and links indicating relationships among the nodes.

- Feature-oriented design rationale systems contain domain knowledge-bases, which can be used to support automated reasoning and the generation of design rationale. So representations of design rationale are usually more formal than in a process-oriented design rationale system. In some systems, the design rationale is represented with links to the existing knowledge-base. The retrieval and reuse of design rationale seems very natural in the design process of later artifacts [4].

2.2.3.2 Capture of Design Rationale

In a design process, capturing design rationale involves recording the reasonings, decisions, oppositions, trade-offs, etc and constructing a formal or semi-formal structure so that the design rationale can be used in the decision-making process during design [4]. There are two main methods to capture design rationale viz. automatic and user-intervention [12]. The automated method does not require the designer to input or record design discussions, decisions and reasoning themselves while the user-intervention method does. These two methods are used to capture design rationale using either process-oriented or feature-oriented approach. In the process-oriented approach design rationale is seen as a history of the design process while in the feature-oriented approach design rationale has a formal, logical structure and is supported by domain knowledge-bases. Thus in fields with relatively high degree of standardization the feature-oriented approach is used while the process-oriented

approach is used in dynamic design domains. Lee [13] offers the following classification for the rationale capture systems:

- Reconstruction [9]
    - Captured outside the design process, usually after it has been performed using information recorded during design.
- Automatic generation [9]
    - Generated from an execution history
- Methodological byproduct [9]
    - Emerges during the design process.
    - Methodology aids design and captures rationale
- Apprentice [9]
    - System monitors designers actions and compares with pre-generated rationale
- Historian [14]
    - Similar to Apprentice but does not make suggestions.

2.2.3.3 Representation of Design Rationale

The choice of a representation schema is a critical issue [4] because it determines how to organize this enormous amount of diverse material and build in into a usable structure. It also determines how to capture and retrieve the design rationale [4]. The following are some of the commonly referred representation schema.

- Argumentation-based design rationale representation is the most common format. With argumentation, designers can easily maintain consistency in

decision-making, keep track of decisions and communicate about design reasonings. The most common argument structures for selecting and organizing information are IBIS (Issue-based information system) [15], PHI (Procedural Hierarchy of Issues) [16], QOC (Questions Options and Criteria) [17] and DRL (Decision Rationale Language) [18].

- In IBIS the key issues are usually articulated as questions, with each issue followed by one or more positions that respond to the issue. Each position can potentially resolve or be rejected from the issue. Arguments either support or object to a position.
- The Procedural Hierarchy of Issues (PHI) extends IBIS by broadening the scope of the concept “issue” and by altering the structure that relates issues, answers and arguments. First, it simplifies relations among issues by using the “serve” relationship only. Second, it provides two methods to deal with design issues: deliberation and decomposition i.e., to give answers to the issue or to break down the issue into a variety of sub-issues which in turn could be deliberated or decomposed.
- QOC represents the design space using three components viz. questions identify key issues for structuring the space of alternative; options provide possible answers to the questions; criteria are the bases for evaluating and choosing among the options. The QOC representation emphasizes the systematic development of a space of design options structured by questions, and the rationale representation in QOC is

created along with the descriptive representation (specification) or the artifact itself (prototype).

  - Decision Rationale Language (DRL) uses design rationale as an account of how the designed artifact serves or satisfies expected functionalities. DRL is an expressive language, which represents the space around decisions. The DRL is used to represent and manage the qualitative elements of decision making: for example, the alternatives being considered, their current evaluations, the arguments responsible for these evaluations, and evaluation criteria.

- Functional representation (FR) is a modified form of argumentation-based representation. Like DRL, FR uses design rationale as an account of how the designed artifact serves or satisfies expected functionality. FR takes a top-down approach to represent a device; the overall function is described first and the behavior of each component is described in the context of this function. FR encodes the designer's account of the causal processes in the device that culminate in achieving its functions. The Structure-Behavior-Function (SBF) [19] and KRITIK [20] are examples of FR.
- Augmenting Design Documentation (ADD) [21] represents design rationale by documenting the complete design decision path associated with the artifact as well as the rationale behind each decision presented by the user. Recently a system called ADD+ was proposed that uses the same basic model as ADD but

stores the wealth of knowledge by organizing it into high-level rhetorical structures.

- The Core Product Model (CPM) [22] is an object-oriented framework for representing products. The CPM stores a complete view of a product and hence contains a Rationale class. The Rationale class along with its sub-classes stores the justifications and evolution of the product including functions and constraints.

2.2.3.4 Retrieval of Design Rationale

At different design stages there are various purposes for accessing design rationale. The reuse of design rationale is made possible by successful retrieval strategies. The integration of design rationale systems with other design support systems can greatly improve the retrieval of design rationale. The following are some of methods proposed for design rationale retrieval:

- Design rationale retrieval shares much in common with case-based reasoning and retrieval. The goal of most design rationale systems is to store rationale so that relevant past experiences can be retrieved to aid in current problem solving. Case-based retrieval methods are thus the most common retrieval methods in design rationale systems.
- Design rationale Navigators permit the designers to investigate stored rationale using a graphical interface. The designers traverse between nodes connected by links to facilitate investigation. Navigators are commonly implemented in most design rationale systems to provide a graphical interface to the designer.

- Retrieval strategies that retrieve answers to designers' queries are generally considered more efficient than navigators. Questions of the type of "what-if" or "why" provide the designer with ways of exploring alternatives and justifications of the argumentation or reasoning behind a decision.
- Automatic triggering is another common retrieval method that monitors the designer's actions as it checks the design process, compares the decisions made with the constraints, rules or criteria in a design rationale library or knowledge-base. If differences are detected the new decision will be stored in the design rationale library. This type of retrieval is ideal for use during the design process.

2.3 Legacy MCAD Design Rationale

With a general understanding of design rationale and the state of the research in the areas of approach, capture, representation and retrieval, the following section deals with design rationale in the limited domain of legacy MCAD. Current attempts to capture design rationale from legacy MCAD fall largely into two categories 2D drawings to 3D model conversions and drawing interpretation. The next two sub-sections present the state of the research and art on design rationale in legacy MCAD.

2.3.1 2D drawings to 3D model conversions

There exists much research to capture the information present on the 2D drawings and propagate it to an intelligent, parametric, feature-based 3D model. Weiss and Dori [23] propose an approach that automates the 3D object reconstruction from 2D engineering drawings by mimicking human intelligence. Dori and Wenyin [24] have described a complete system that realizes the entire process of understanding

mechanical engineering drawings from scanning to 3D reconstruction. The system described has the capability of separating geometric entities from non-geometric, such as text, arrowheads, leaders, dashed lines and hatch lines etc. Tanaka et al [25] describe a method to automatically convert 2D assembly drawings to 3D part models, generating a unique solution for designers regardless of the complexity of the original 2D assembly drawings. They use the dimension lines, part numbers and part lists, usually drawn on the 2D, to create the 3D assemblies.

On the commercial side too there are numerous proposed solutions. Imagecom Inc. [26] has a product named FlexiDesign that converts 2D drawings in AutoCAD DWG and DXF formats to parametric, feature-based 3D models in a variety of target MCAD systems e.g. PTC's Pro/Engineer, SolidWorks, Autodesk's Inventor etc. FlexiDesign, as it stands, currently handles piece-part drawings but is being extended to handle assembly drawings. Various MCAD systems such as SolidWorks and Pro/Engineer also provide the user with basic 2D to 3D conversion tools built into their system, though these tools are largely manual and are primarily facilitate drawing reuse rather than conversion.

2.3.2 Drawing interpretation

There are many drawing interpretation solutions also suggested that allow a designer to query CAD files for required information. Joseph [27] has presented a methodology for the interpretation of engineering drawings based on a combination of schemata describing drawing constructs with a library of low-level image analysis routines and a set of explicit control rules applied by a parser. The resulting system

integrates bottom-up and top-down processing strategies within a single, flexible framework modelled on the human perception cycle. The system, termed Anon, is a knowledge based image analysis system intended to extract 2D graphical elements and symbols from a grey level image of a mechanical engineering drawing. The system classifies the information on the drawing into appropriate schematic classes such as solid, dashed and chained lines, solid and dashed curves, cross hatching, text, witness and leader lines and certain forms of dimensioning. Cheng and Yang [28] propose a knowledge-based graphic description tool that is used to recognize and understand engineering drawings. The graphic description tool basically consists of a concept description network, a graphic description language, a physical description framework, a set of image processing modules, a matcher, a rule-based inference engine, an interpreter and blackboard control architecture. The concept description framework, graphic description language, and physical description framework are designed to represent domain knowledge, graphic semantic knowledge and physical properties of engineering drawings in different fields. The matcher recognizes all graphic symbols and characters that are extracted by the low-level image processing routines. The rule-based inference engine is built to infer possible relations among graphic symbols and generate a relational graph. The interpreter is used to generate an acceptable explanation in terms of traversal of the relational graph. This framework does not attempt to create a solid model from the captured information but instead builds an engineering drawing understanding system that could be queried as necessary. Vaxiviere and Tombre [29] present a knowledge based system named CELESSTIN that extracts technologically

significant entities and analyzes the whole setup with respect to disassembling and kinematics knowledge. These technologically significant entities allow CELESSTIN to start using rules referring to the semantics of the represented object itself. The paper illustrates how to assign a simple syntax on the basic structures to recognize simple mechanical entities such as shafts or screws.

Almost all solutions, except Cheng's and Yang's [28], and Vaxiviere's and Tombre's [29] address the problem of 2D drawings to 3D model conversion as mostly geometric with possible input from the supporting symbols and text that may be present on the 2D drawing sheet. Research to identify and separate product geometry from dimension sets, arrowheads, hatching lines, text and symbols fall short in failing to recognize that the non-geometric information on the drawing sheet contributes to engineering knowledge, design intelligence and some design rationale. Tanaka's [25] solution is further limited, in that one major requirement for their algorithm to work is that the original assembly drawings consist of standard parts such as bars and plates. While Cheng and Yang's [28] paper describes a rule based system that recognizes, examines and classifies the graphic symbols in the engineering drawings, their graphic description language diverges from the current practice of using vectorized geometric information. The specific domain knowledge of the drawing that their system extracts is mostly used to examine and classify the graphical symbols in a field. Very little semantic knowledge is attached to the graphical symbols using the domain knowledge. Vaxiviere's and Tombre's CELESSTIN [29], while able to recognize simple mechanical entities, will face difficulties when the complexity of the mechanical entities

grows beyond symmetric blocks. CELESSTIN also lacks support for non-geometric entities, which could pose problems when information is missing from the geometry of the artefact.

Most importantly it must be noted that none of these solutions capture design rationale as the reason for the artefact or a part of it. No methods exist that capture the functions, justifications or alternatives in the same way as traditional design rationale systems. This is one of the biggest failures of legacy MCAD design rationale systems.

2.3.3 Definition of legacy MCAD Design Rationale

From the summary of related research and background presented in preceding sub-sections it is clear that there exists no clear understanding of what is meant by design rationale in legacy MCAD. Based on an extensive literature survey it is concluded that no researcher has even provided a definition for design rationale in the domain of legacy MCAD. The definitions stated for generic design rationale in sub-section 2.2 are not applicable in the domain of legacy MCAD. In an effort to better understand legacy MCAD design rationale the following definition is suggested:

"Design rationale contained in legacy CAD is the insight into the design variables implicit in the structural, semantic and practical relationships between the geometric and textual entities present in the CAD representation" [30].

The design variables are the functions, flows, objectives, constraints, principles, guidelines and manufacturing that are considered during the design process. It should be noted, however, that the design rationale that can be captured from legacy CAD will be limited due to the nature of the information present on it. 2D drawings contain only

unstructured graphic entities such as lines, texts and symbols while 3D models, although they contain a well structured geometric view, do not provide a comprehensive view of the product beyond geometry. To better illustrate the definition for legacy MCAD design rationale refer to Figure 2.2, which shows that design rationale is implicit from the entities present in the legacy CAD and also from the relationships that exist between the entities. This dissertation contends that these are the aspects that play a major role in design rationale capture.

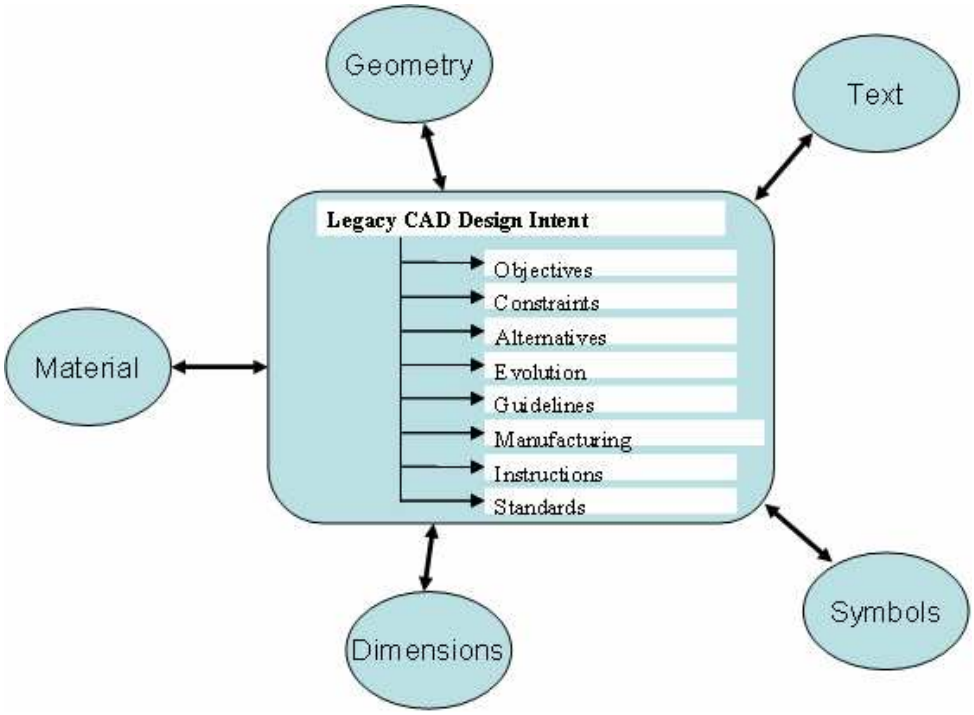

Figure 2.2: Legacy CAD Design Rationale

2.3.4 Importance of legacy MCAD design rationale

Design rationale is considered important for various reasons. Pena-Mora et al. [31], while they do not attempt to define design rationale, state that the Architecture/

Engineering/Construction industries can benefit from the explicit representation of the design process rationale in many ways;

- Large and lengthy projects change over time and require certain design decisions to be modified during the design-construction process. Reasons or justifications used during the initial design stages can be lost resulting in the need to define them over and over resulting in increased project costs and delays. The ability to store and recall these reasons will improve productivity.
- The quality of the project increases as the project rationale is represented explicitly and is readily accessible for review.
- A model that allows the rationale to be explicitly stated and easily manipulated leads to a more intelligent use of knowledge and resources.
- Understanding design rationale of designers is also important to achieve coherent integration of design solutions and transfer design knowledge [32].

2.3.5 Lack of Design Rationale system deployments

While industry increasingly uses more intelligent engineering frameworks to improve their product development process there is still a lack of design rationale system use. Hu et al provide a list of reasons for this in [12]. They state that for design rationale systems to be adopted for widespread industrial use, the systems have to be advanced enough to be effectively deployed. They claim that there are still open issues with regards to capture, representation, retrieval and approach that need to be addressed before any effective deployment. Molavi et al [33] state the reason for the failure of design rationale systems is due to the costs and disruptions associated with capturing

and structuring of the rationale. The following reasons can be added to the list of failures. Legacy CAD is still considered as the storehouse of current design data. There are still a large number of legacy CAD files in use in government and industry and their use is not decreasing any time soon. While it is generally considered better to use design rationale systems we cannot ignore the bulk of the design rationale (product knowledge) that resides in legacy CAD. Some method is needed that captures the design rationale from legacy CAD and propagates it to the design rationale system. The next few sub-sections evaluate the applicability and limitations of the capture, representation and retrieval methods available for generic design rationale systems to the domain of legacy MCAD.

2.3.5.1 Applicability of Capture methods

When considering legacy MCAD the design process is nearly complete since at this stage we have a detailed representation of the product. This implies that any design rationale capture method must be after the completion of the design process. This means that the methods mentioned previously in section 2.2.3.2 for design rationale capture viz. Methodological byproduct, Apprentice, Automatic Generation and Historian are not applicable as they capture design rationale during the design process. The Reconstruction method is the only one that seems applicable but based on its definition the Reconstruction method depends on process data captured during design e.g. video, email discussions, design documents etc. to infer rationale. In the limited domain of legacy CAD there is little or no access to this data.

2.3.5.2 Applicability of Representation methods

Legacy CAD design rationale representation faces a similar problem as capture (section 2.3.5.1). The current representation methods suggested in section 2.2.3.3 do not address legacy CAD. Those representation methods require knowledge of issues, questions, options, alternatives, functions, evaluation criteria etc., most of which are unknown in the domain of legacy CAD, where only details of the end product is known. If we were to use one of the aforementioned representation schemas, it may result in incomplete representation due to lack of data. As a side note, one representation method that seems viable is the Core Product Model's [22] Rationale class. Since the Rationale class is object-oriented we may be able to modify this class and limit it to legacy CAD.

2.3.5.3 Applicability of Retrieval methods

Unlike capture and representation methods, current retrieval methods mentioned in section 2.2.3.4 may still be relevant in the domain of legacy CAD. Since retrieval methods are dependent on capture and representation methods and on the particular use of design rationale, some modifications may be required but largely still applicable.

2.4 Context

This section presents a survey of literature related to context [34] [35]. This is primarily to help understand what is context, what is its structure, is it important in design, does it influence design, how would one use context in design, how does designing using context differ from other approaches to design modeling and finally what is the impact of context in design? An excellent survey of context in a wide variety of fields (Artificial Intelligence, Natural Language Processing, Architecture, Software

Engineering) is presented by Brezillon [36]. The survey presented here is intended to look at the use of context and to help determine the importance of context in domains other than those covered by Brezillon [36].

2.4.1 What is Context?

To answer this question the following paragraphs present a survey of the use of context and definitions provided for it in domains such as patterns, design, user access control, software etc.

2.4.1.1 Context in Patterns

Alexander [37] defines a pattern (in architectural design) as the description of a problem that occurs repeatedly in our environment and the core, reusable abstracted solution in a context. This explicitly means that when creating patterns such as those described by Alexander one must take into account the context that is relevant to the design pattern. The introductory paragraph of a pattern explains how the current pattern fits in or completes the larger patterns and sets the context of the pattern. The context delineates the situation under which the pattern is applicable. Context often includes background, discussions of why the current pattern exists, and evidence for generality [38]. The importance of context in patterns is thus to tie the stated problem to the core solution described in the pattern. The lack of a well defined context will hamper the re-usability of the pattern taking away from the basic purpose of patterns. Unfortunately, Alexander does not define context very clearly. Despite this, his idea of patterns and contexts in design is finding increasing use in software engineering.

2.4.1.2 Context in Search

Many aspects of design involve search: search for catalogue items such as bearings and gears, search through a multi-parameter space for an optimum set of values, etc. Web search engines generally treat search requests in isolation. This is yet another area where the context will gain importance in next generation search engines [39]. Glover et al. [40] present a meta-search engine that operates as a layer above regular search engines. The engine, Inquirus 2, takes the query plus context information and attempts to use the context information to find relevant documents via regular web search engines. Budzik [41] present a system that automatically infers the context of the search request. The system, Watson, does this based on the contents of the document that the user is editing. Popular, commercial web search engines such as Google [42, 43] and Yahoo [44] also support user context such as the user's location and personalization to improve search results. When a user logs into these commercial web search engines the search systems store the context surrounding the search terms as well the results. The system may also track the results that the user views. Using an intelligent ranking system on the captured context of the users search behaviour will provide search results that are more relevant to the user. No definition is given, however. If such context based search engines become prevelant and we can define the context surrounding a specific design problem, design will become easier.

2.4.1.3 Context in Access Control

Mostefaoui and Brezillon [45] propose a conceptual model for context-based authorizations that offers a fine grained control over access on protected resources. The

context, in this case, is made up of the users and environment's state and information. As opposed to a static access policy based on the user's identity, Mostefaoui and Brezillon state that a context based policy would respond in a flexible manner in highly dynamic computing environments due to the dynamic nature of context itself. Context thus becomes the key to the approach to specify the policies to grant or deny access to resources. Within global design teams access to protected resources is becoming important. Such research results will make it easier for the member of such teams to access the design data and information.

2.4.1.4 Context-aware computing

Dey [46] presents an operational definition of context-aware computing and discuss the different ways in which context can be used by context-aware applications such as possibly those used by global design teams. His definition is: a system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task. He states that context is a poorly used source of information in computing environments. As a result we have an impoverished understanding of what context is and how it can be used. In [46] Dey presents the Context Toolkit, an architecture that supports the building of context-aware applications. Dey contends that a new definition of context is required as the existing definitions do not provide any easy way to determine whether a type of information listed in the definition is context or not. His new definition, Dey states, makes it easier for developers to enumerate the context for a given applications scenario. Dey uses this

definition of context as the basis for his Context Toolkit which makes it easy to add the use of context to existing non-context-aware applications.

2.4.1.5 Context in Software

Software applications used by designers among others need to be deployed on a variety of platforms and within a variety of contexts in general. Currently platform independent modelling techniques such as the Unified Modelling Language are used to model the software applications and these models are then transformed to a refined model. That means that for each new target platform at least one new model transformation is needed. Wagelaar [47] proposes a context-driven modelling framework that models each target context in a context model. The framework can automatically select appropriate transformation rules for a concrete context and configure them into a context-optimized transformation. While he does not define context, this research could have application in design where design models need to be viewed from different contexts (design, manufacturing, procurement, analysis etc). If we can define the various design contexts properly, then such a system could automatically transform the CAD model into the appropriate design context.

2.4.1.6 Context in Communication

Communication is yet another domain where context plays an important role. The context surrounding a particular communication helps clarify the ambiguity that may exist with words that may have varying meanings. Fogarty et al [48] present a study on the usage of a context-aware communication client. The results of the study show that the users of the client use the context of their colleagues as an indication of

presence rather than the status of availability. Since communication clients that integrate chat, video and voice is becoming an important tool in communication, especially among global design teams, the inclusion of the awareness of the users' context is increasingly found in many commercial clients. No commonly agreed on definition in this domain exists, however. This research also has possible application in global design teams since not only do the members of such teams come from different cultures they also come from different industrial contexts.

2.4.1.7 Context in Databases

Goh et al [49] present an approach for database interoperability, in which the notion of context is the key to circumvent the problems that arise when dealing with schematic and semantic incompatibilities of underlying, heterogeneous and autonomous databases. By context they refer to the implicit assumptions underlying the way in which an interoperating agent routinely represents or interprets data. Since more and more companies are using Product Data Managers (PDMs) to manage the data and information created during design and often different members of the same team use different PDMs, this is an increasing problem in design.

2.4.1.8 Context in Artificial Intelligence

Turner treats context for intelligent agents as any identifiable configuration of environmental, mission-related, and agent-related features that has predictive power for behavior [50]. Bremond and Thonnat deal with contextual information of a process as the information whose value remains constant during processing and changes when the process is used for another application [51]. Bigolin and Brezillon state that context

delimits a domain allowing the designer to restrict the possible solution space of a problem [52]. In their survey of context in problem solving, Pomerol and Brezillon treat context as a constraint in problem solving that does not intervene in it explicitly [53].

From the above short review, it appears that a clear definition of context has not yet appeared in the field of context in general although the following definitions for context have been suggested by various researchers:

"Turner treats context for intelligent agents as any identifiable configuration of environmental, mission-related, and agent-related features that has predictive power for behavior [54]."

In addition to this definition, Turner also states that "context is a distinguished (e.g., named) collection of possible world features that has predictive worth to the agent [50]."

Bigolin and Brezillon state that "context delimits a domain allowing the designer to restrict the possible solution space of a problem [55]."

Pomerol and Brezillon state that "context is what constrains a problem solving without intervening in it explicitly [53]."

Bremond and Thonnat define "contextual information of a process as information whose value remains constant during processing and changes when the process is used for another application [51]."

Bigolin and Brezillon define context "as the delimitation of a domain, that allows to restrict the possible solution-space of a problem [52]."

Dey defines context “as any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves [46].”

Mills and Goossenaerts state that “a context surrounding an entity of interest is a set of properties (with values), that are (a) provided by a set of entities in the same symbolic or physical space as the entity of interest, (b) relevant to the entity of interest in that situation of interest during some time interval and (c) added to the properties of that entity only within that context [56]. Properties can be parameters, rules, behaviors, principles, filters, objects with their own properties, attributes, etc.”

The definition by Mills and Goossenaerts, which is used in this work, can be elaborated upon to understand it better. At any point in the design, the focus is on some entity of interest which exists in a symbolic or physical design space. The surrounding situation (i.e. the context) adds to that entity of interest a set of relevant properties which are in the same design space. A change in either the entity of interest or the surrounding situation would change the context that is applicable. The entity of interest is suggested by Dey [46] to be a person, place or abstract object. In engineering design the entity of interest could be the design problem, a design variable (e.g. design function, solutions etc), or a designer etc.

2.4.2 What is the structure of context?

Sowa has discussed a structure for contexts, based on the linguistics domain where the idea of context has been studied the most [57]. Sowa has proposed three levels:

- Syntax: Partial basic meaning of a word or phrase is extracted from the position of the word in the sentence.
- Semantics: Further meaning of the word or phrase is extracted from its location in the paragraph(s) surrounding it.
- Pragmatics: The final level of meaning is extracted from the rest of the surrounding situation in which the document was created: author(s), milieu, time of day etc.

We apply this idea of a context structure to the context surrounding key aspects of design; the function to be provided by the artefact and the problem solution. Drawing a parallel between Sowa's structure in linguistics and engineering design we have for design:

- Syntax: Key design constraints (e.g. spatial)
- Semantics: Weighted objectives, other constraints (e.g. safety factors, weight, stiffness)
- Pragmatics: Design rules, guidelines, standards, domain, environment

The syntax level consists of key constraints, which can identify an expected behavior. In several domains that we have looked at, such key constraints are often implicit and need to be made explicit. For e.g. the key constraints for the domain of

Mechanical Structure design are the allowance of external power sources, the presence or absence of a ground plane and a set of spatial constraints on the force. Consider the problem of supporting an object in a gravitational field. The function is that of providing one or more forces which bring the object, acting under the force of gravity, into equilibrium. This function can be provided by several behaviors including (a) posts anchored to the ground,(b) ties anchored to a ceiling or to a helicopter, (c) beams anchored to side supports, (d) arches also anchored to side supports, (e) a rocket motor pushing up on the weight, etc. Each of these behaviors can be supplied by numerous shapes, dimensions and materials. In the case of our example, the first key constraint would be constraint on the application of external physical power sources. This constrains the use of actively providing the force through the means of artifacts such as a helicopter or rocket motor and restricts us to passive means of supplying the force such as reaction forces from a ground plane (i.e. using the earth). If power sources were allowed, then these solutions would remain active but may be eliminated at another level (e.g. due to cost constraints). Using similar key constraints we should be able to identify a few relevant solutions which can then be analyzed for suitability.

The next is the semantic level. The idea here follows that first suggested by Pahl and Beitz [58] and further amplified by Dym and Little [59] and Dieter [60]. They use constraints and weighted objectives in general engineering design to evaluate design concepts. In the semantic level we place the other constraints on the function and the metrics of the weighted objectives to be fulfilled. In the Mechanical Structures domain context, other constraints typically include constraints on weight, cost, safety,

durability, etc. The metrics for the weighted objectives also include the same parameters as the constraints. No parameter however can be both a constraint and an objective [59]. Constraints are statements about the desired product that limit the design space. Objectives are statements that enlarge the design space but can be used later to rank possible solutions. The major difficulty at this contextual level is that although we know what the metrics for the objectives and the constraints should be, to filter out unwanted candidates we need to calculate the actual values for equivalent properties of the physical entities for various solution contexts that could provide a solution. For these computations we need to know the shape, the dimensions and the material properties as well as equations for calculating cross sectional areas, volumes, costs, moments of inertia, stress and deflection. The domain restricts the large possible set of equations relating the metrics to the properties. The equations for stress and deflection, etc are further restricted by the syntax level to those associated with particular structural behavior that the syntax level has identified. That is, as soon as the syntax level has been built and surrounds the function, the particular equations have been identified.

The third is the pragmatic level. In the pragmatics level are the design principles, design guidelines, Governmental and industry regulations. These are applied to the solution to add refinement. Application of this principle to the solution would require modifications to the overall design but would most likely not require major modifications.

2.4.3 Is context important in design and does it influence design?

These questions are treated together since they are interdependent. To be important, context has to have some use and influence design in some way. The view of the research conducted during this dissertation is that context contributes implicitly but strongly to design decisions. Most experienced designers reach a solution to a design problem much more easily than inexperienced ones. This is because they make use of context implicitly as discussed below and that making context explicit will enable any designer to reach an appropriate solution/decision faster, easier and requiring less experience. Context has an influence on design and the design process primarily because any design problem exists in a surrounding context. By understanding the elements of context and their individual influence on design, context can be made explicit.

Every design problem exists in an overall pragmatics such as the domain of the design problem, the domain of the design organization, the type of design and other environmental aspects such as the experience of the designers, their training and the hierarchy that the designer is a part of (department, design team, projects etc). The following is a brief explanation of the influence of this pragmatic context.

- Designers attempting to solve design problems implicitly define the domain that the problem lies in thereby limiting the class of appropriate solutions. Even when solving multi-domain problems one of the steps is to break down the problem into its individual domains before design and analysis. This implies that if the properties of the domain were defined, thereby defining the domain

context, we would essentially limit the list of functions, constraints, objectives and solutions relevant to the design problem. In my view, one of the reasons inexperienced designers take longer to create a design is that they spend considerable time implicitly defining the context surrounding their particular design problem – the entity of interest.

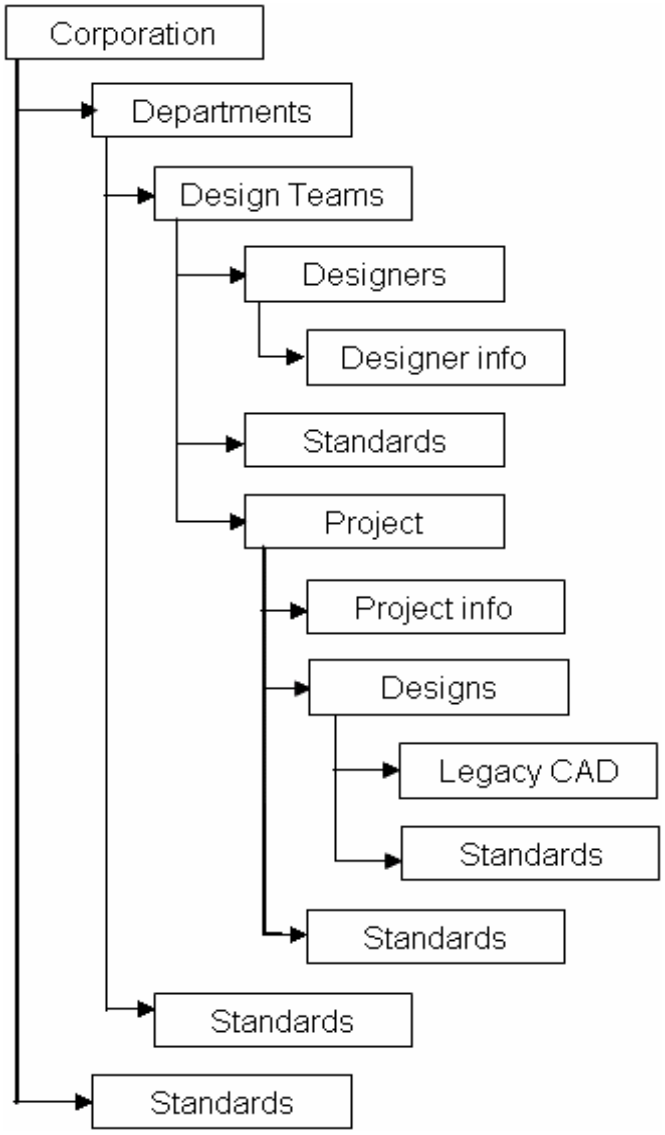

Figure 2.3: Hierarchy within corporation

- The type of design is an important contextual parameter. The design variables are dependent to some extent on the type of the design. Well understood design types such as selection of bearing from a bearing catalogue or choosing appropriate dimensions in parametric design are easier to characterize than say novel design where the designer may have to generate new concepts to arrive at a solution. Thus design types that can be considered as routine may be easier to automate and may require less experience to solve.
- To understand the influence of the domain of the design organization, consider the designers and the information related to them. It can be seen from Figure 2.3 that the designer does not exist independently but instead forms an important part of a design environment. Depending on the company hierarchy the designer would be a part of one or more design teams, and the design teams would be a part of departments and so on. Design teams follow standards, guidelines or good practices. These form a part of the context that influences the designer's decision while solving a problem. Similarly if the departments or the parent company have a different set of standards or guidelines then these too would form a part of the designer's context. (Of course it should be noted that in a stable hierarchy such as this we do not account for conflicting properties being added to the designer's context from the different standards. The "standards" is an example of the context that is external to the designer. Factors internal to the designer are also equally applicable. For example as already mentioned, the experience level of the designer suggests the breadth of knowledge possessed by

the designer. This includes knowledge related to design in general such as the related domains and solutions and more specifically the surrounding context related to the operational, financial and other goals of the company. To properly aid the designer or to successfully automate the design process any implemented knowledge-driven design system needs access to this contextual knowledge that surrounds the designer (and hence the design problem) to arrive at a solution more efficiently.

Just as the context that surrounds the design problem influences it, the context that surrounds and influences design variables viz. function, objectives, constraints, solutions etc are identified. The following sub-sections detail the identified context, its influence and its use to simplify design problems.

2.4.4 How would a designer use context in design?

In the course of research a design model was developed that is context-aware and uses the structure for context suggested by Sowa [57]. The design model is based on the process model suggested by Pahl and Beitz [61] and furthered by Dym and Little [62], Fenves et al [63] and Gero and Kannengiesser [64]. The design model uses the context structure to progressively narrow the relevant list of solutions in a step by step manner using the three contextual levels (syntax, semantics, pragmatics) suggested above. The basic premise of this model is: a function in a context defines a solution. This model is termed the Function-Context-Structure (FCS) model of design and is shown in Figure 2.4. Like its parent models the FCS model is also function-based but essentially adds a filter in the form of a structured context to provide a mechanism to

map design function to artifact solution. The FCS design model follows the stages prescribed by Pahl and Beitz [58] and expanded by Dym and Little [59], Fenves et al [63], Gero and Kannengiesser [64] and Dieter [60] etc. which are:

i. Problem definition

ii. Conceptual design

iii. Preliminary design

iv. Detailed design

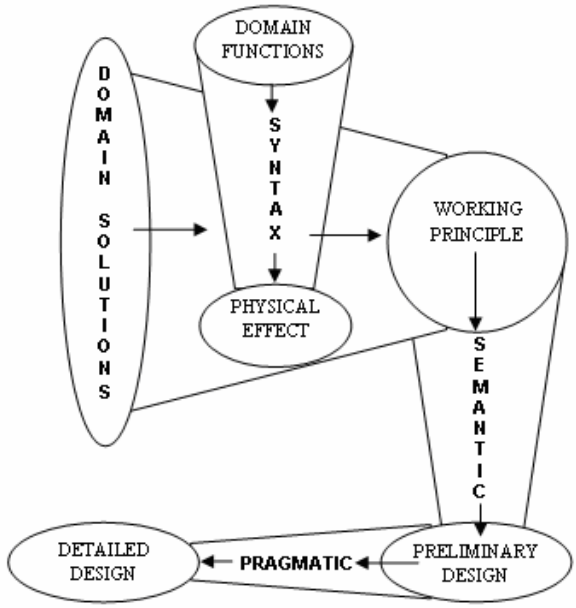

Figure 2.4: Function-Context-Structure Model of Design

All design begins with a problem statement which may merely be a client's statement. The first step in any design process is to define the problem clearly. This step is not unique to the FCS model but is detailed here as it is during this step that the domain and inner context are defined. The following are the prescribed steps:

- Identify the flows (input and output) both implicit and explicit in the problem.
- Determine the domains/sub-domains that the design problem lies in. Some of the domains/sub-domains can be inferred from the flows identified in the previous step. The domains / sub-domains contribute to the domain context.
- Identify the overall design goal. This goal can be organized into sub-objectives.
- Identify any design constraints from the problem statement.
- Determine the possible list of design functions that relate the input flow to the output flow in this domain.
- Recall the designer's context e.g. design guidelines, principles and standards pertaining to the corporation (commercial or governmental) etc.
- Having identified these design variables organize them into appropriate context levels. Constraints can be split into key e.g. spatial and other constraints e.g. safety etc. Key constraints primarily form syntactic context. Objectives and other constraints form a part of semantic context. Knowledge of domain context allows the designer to filter out functions and solution artefacts that are not relevant in the current design domain. The principles, standards and guidelines and domain context form the pragmatic context.

Step 2, conceptual design, involves the generation of concepts or schemes of candidate designs. The main task in this step is to generate design alternatives viz. identify solutions that can achieve the objective. In typical function based design, the

goal of the designer would be to identify solutions that realize the functions while satisfying constraints. For this stage Fenves et al's model [63] relies on a function-to-form transformation to populate the initial structural solution. Since it is not always possible to infer functional information from a structural description or vice-versa [65] the FCS model follows an adaptation of Gero's and Kannengiesser's [64] approach of determining conceptual solutions using the syntactical context surrounding the function. The syntax is made up of the key constraints that were identified in the 'Problem Definition' step. The design function is filtered through the syntax to arrive at physical effects. The classes of solutions belonging to the domain identified in the 'Problem Definition' step are then filtered through the combination of the syntax and physical effects to arrive at the conceptual designs.

Step 3 involves refining the conceptual schemes into preliminary designs. The semantic context that was identified during 'Problem Definition' forms the primary decision variable. The semantics mainly contain the weighted objectives and other constraints (not considered key constraints) such as safety, weight, price etc. The semantics also consist of the equations determined by the relevant physical effects of the conceptual designs. The designer uses the semantics to decide attributes such as material and geometry. Ashby [66] provides a detailed methodology to select an appropriate material along with a way to identify an efficient geometric shape and we are looking to use his methodology in the FCS model. The semantic filtering requires the computation of values for equivalent properties of the physical entities for various solution contexts that could provide a solution.

The fourth step adds an additional level of filter viz. the pragmatic context. The design principles, design guidelines, governmental and industrial regulations, domain knowledge, company and designer information form the pragmatic context. The result of this stage would be a design artefact solution with the level of detail required for production.

2.4.5 How does using context differ from other approaches to design modelling?

Design models, the results of much design research, are a symbolic representation of real entities/processes for design, analysis, simulation etc. A model captures essential parts of a system. Most design models viz. [67, 63, 68, 69, 64]) use well-understood object representation schemes such as UML, taxonomies, semantic meta/networks, ontologies, process-oriented schemes, etc. Some of the newer methods are using knowledge-based frameworks, patterns or context-aware models etc. The following is a brief description of the various schemes:

- Taxonomies: - Simple hierarchical tree/graph like structures that specify relationships between the nodes. The nodes represent real world entities/processes.
- UML: - In UML real world entities or processes are represented using classes, class properties, use-cases and objects (specific instantiations of classes).
- Semantic networks: - A semantic network or net is a graphic notation for representing knowledge in patterns of interconnected nodes and arcs. What is common to all semantic networks is a declarative graphic representation

that can be used to either represent knowledge or support automated systems for reasoning about knowledge. Some versions are highly informal, but other versions are formally defined systems of logic. Following are six of the most common kinds of formal semantic networks; definitional networks, assertional networks, implicational networks, executable networks, learning networks, hybrid networks.

1. Ontologies: - An ontology has been defined as a specification of a conceptualization [70]. That is, an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. An ontology is represented as classes with slots (properties with/out values and limits), but is not considered complete without instances (like objects these are specific instantiations of classes). Unlike UML an ontology describes concepts of a domain specifically designed for sharing and reuse. On a side note, a shared context is represented using ontologies [45].

- Patterns: - A pattern describes a problem, which occurs over and over again in an environment and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice. A pattern is not an isolated entity (problem-solution-description). It should merely specify the field of relationships and only the essential parameters to implement (solve) the problem [71]. The following is the format of a pattern:

- A picture (symbolic representation) showing an archetypal example of that pattern.
- Introductory paragraph which sets the context of the pattern. This paragraph explains how the current pattern fits/completes the larger pattern.
- Headline giving the essence of the problem (problem definition).
- Body of the problem – empirical background, evidence for validity, range of methods of manifestation (application).
- Solution to problem – contains required field of physical and social relationships, instructions to build the pattern.
- Parent and child patterns.

Based on the preceding sections and our research on context we can state the following differences among context-based models and the other schemes:

- Unlike UML or taxonomies, context is not a representation language/method. One can use these languages/methods to represent context.
- While UML classes are usually static (though flexible) representations of real world entities/objects/processes, context adds relevance and dynamism to the entity of interest.
- A context provides the structure, the relationship and the situation (domain/sub-domain) to define the problem, the solution or the approach to the solution. It does not merely serve to represent real world entities/processes like the other methods.

- If considered according to Alexander's pattern, a context represents relationships among the current design problems (or solutions) and the parent (or child) problems (or solutions).
- Unlike ontologies which specify concepts, contexts are dynamic filters that narrow down the relevant information that apply to the current entity of interest (e.g. to specify the field (domain) of interest of the design problem/solution).

2.4.6 What is the impact of context in design?

Bigolin and Brezillon [72] have stated that context constrains the problem space. That statement can be extended to mean that it must also delimit the design domain allowing the designer to restrict the possible solution space of a problem. The impact of the domain context has been recognized in research but no conclusive results exist that show how a designer would use the domain context. This section provides the details on the nature of domain context, its usage and impact on the design process and the solution. The term "domain" refers essentially to the discplines that are in existence in society today (e.g. engineering in general. mechanical engineering, electrical engineering). This dissertation has focussed mainly on the engineering domain and specifically on mechanical engineering. Table 2.1 illustrates the proposed taxonomy of domains. The classification of domains is based on the flow that is contained in the domain. For example if we are talking about money then we are in the finance domain. Similarly if we are talking forces and torques then we are in the mechanical sub-domain (sub-domain of engineering). The presence of fluids would either indicate fluid or thermal sub-domain which can be further broken based on the context of the fluid e.g.

temperature considerations, relevant flow of fluid or heat transfer etc. Since we are concerned only with the engineering domain, other domains like Finance, Philosophy etc. have been left out to enhance readability. Further we only sub-classify the Mechanical domain (as that is our domain of interest) further treating Mechanisms, Thermal, Fluids, Civil, Aerospace and Hydraulics etc. as sub-domains of the Mechanical domain.

Table 2.1: Taxonomy of Engineering Domains

Social
- Engineering
  - Mechanical
    - *Structures*
    - *Mechanisms*
      - Motion Without acceleration
      - Motion With acceleration
    - *Thermal*
    - *Fluids*
    - *Civil*
      - Structural
      - Transportation
    - *Materials*
      - Ceramics
      - Polymer
      - Metallurgical
    - *Aerospace*
  - Electronics
  - Hydraulics
  - Electrical
  - Software
  - Industrial
  - Manufacturing
  - Biomedical
  - Chemical
  - Nuclear
  - Petroleum

Next presented is the taxonomy of functions delimited by the Mechanical Engineering domain. This is not a new taxonomy but is based on the taxonomy proposed by Szykman et al [73]. This parent taxonomy is used as it is one of the most complete taxonomy of functions available in research while being sufficiently concise as they have removed synonyms and specialized functions. Szykman et al's taxonomy is not domain-centric. To arrive at the domain-centric taxonomies start with their taxonomy and eliminate those functions that do not apply to the "Mechanical" domain. The result of this elimination is all the functions in Szykman's taxonomy, except for the division "Assembly-function" and their sub-divisions. The "Assembly-function" and their sub-divisions are relevant in the "Manufacturing" domain. This condensed taxonomy is shown in Table 2.2.

The taxonomy shown in Table 2.2 is constrained further by the sub-domains of "Mechanisms". To do so identify those functions that are relevant in the "Structures" or "No motion" sub-domain and those that are relevant in the sub-domain with "Motion" and further for "Acceleration" and "No acceleration". For the "Structures" or "No motion" sub-domain this implies that the "Conveyance-function" and "Signal/Control-function" sub-divisions are irrelevant. The reason for this is because "Conveyance-functions" primarily cause motion while "Signal/Control-functions" are applied to mechanical control elements e.g. mechanical valves.

Table 2.2: Taxonomy of Functions in the Mechanical Domain

Mechanical Functions

- Usage-function
  - *Sink*
    - Absorb
    - Consume
    - Dissipate
  - *Source*
    - Extract
    - Generate
    - Supply
  - *Storage*
    - Store
- Combination/Distribution-function
  - Connect
  - Couple
  - Distribute
  - Link
- Transformation-function
  - Amplify
  - Attenuate
  - Convert
  - Decrease
  - Modify
- Conveyance-function
  - Rotate
  - Transfer
  - Translate
  - Transmit
- Signal/Control-function
  - Actuate
  - Adjust
  - Decrease
  - Increase
  - Indicate
  - Inhibit
  - Maintain
  - Measure
  - Resist

Other examples such as “Convert” are generally applicable in conjunction with motion e.g. conversion of electrical energy to mechanical energy is accompanied by rotational motion. Functions from other sub-divisions such as the “couple” function are usually used in reference to the coupling of elements for transfer of motion or energy etc. The taxonomy of functions in the “Structures” domain is shown in Table 2.3.

Table 2.3: Taxonomy of Functions in the Structures (no motion) domain

Structures Functions (No motion)
- Usage-function
  - *Sink*
    - Absorb
  - *Source*
    - Supply
  - *Storage*
    - Store
- Combination/Distribution-function
  - Distribute
  - Link
- Transformation-function
  - Amplify
  - Decrease

Similarly, to prepare a taxonomy of functions for the “Motions” domain all functions shown in Table 2.2 are relevant but for the “Signal/Control-function” division and its sub-divisions. While only the Mechanisms domain is discussed the same principles of classification apply to other sub-domains also. Consider for example the “Manufacturing” sub-domain. The functions from Szykman et al’s taxonomy that are applicable in the “Manufacturing” domain are the sub-types of “Assembly-function”.

In a similar manner the taxonomy of flows in the Mechanical Engineering domain can be prepared. Once again Szykman et al’s taxonomy of flows [73] is used as

a starting point, but do not follow their criteria of classification as is. It has been modified to ensure uncommon flows such as “Generic-Energy” or “Generic-Power” or even those that we consider properties of flows such as “Acceleration” or “Jerk” are not part of my taxonomy of flows. This was done because flows like “Generic-Energy” or “Generic-Power” cannot be used directly in design problems. These have to be converted to more useful types such as active “Forces” or “Kinetic-Energy” before they can be applied. The resulting flow taxonomy is shown in Table 2.4.

The next step is to delimit the flows shown in Table 2.4 further for the sub-domains of Structures and Mechanisms. This is a similar process as performed the functions where the irrelevant flows are discarded when considering the “No motion” sub-domain or the “Motion” sub-domain and its sub-domains. The result of this action leads us to the taxonomies shown in Table 2.5 and Table 2.6.

Care should be taken in how the taxonomy of flows in “Motion” domain is used. The sub-domains traditionally dealing with mechanisms in motion such as kinetics, kinematics and dynamics differ in the properties of the flows such as velocity, acceleration and their time derivatives but deal with the same flows as that found in the “No motion” (“Structures”) domain. To better explain this, consider the flow “force” in the “No motion” domain vs. in the “Motion” domain. The difference would be the property of the force for e.g. in the “No motion” domain the force does not cause motion whereas in “Motion” domain the force could be accelerated or could be a jerk, causing a similar motion.

Table 2.4: Taxonomy of Flows in the Mechanisms Domain

- **Mechanical Flows**
  - Energy
    - Motion
      - *Translational*
        - Impedance
        - Oscillatory
        - Relative
      - *Rotational*
        - Impedance
        - Oscillatory
        - Relative
    - Force
      - Friction
      - Weight
      - Spring
    - Torque
    - Generic
      - Kinetic
      - Potential
        - Gravity
        - Spring

Table 2.5: Taxonomy of Flows in Structures (No motion) domain

- **Structures (No Motion) Flows**
  - *Force*
    - Friction
    - Weight
    - Spring
  - Torque
  - Kinetic
  - Potential
    - Spring
    - Weight

Table 2.6: Taxonomy of Flows in Motion domain

Motion Flows
- Translational
  - Impedance
  - Oscillatory
  - Relative
- Rotational
  - Impedance
  - Oscillatory
  - Relative
- Force
  - Friction
  - Weight
  - Spring
- Torque
- Kinetic
- Potential
  - Spring
  - Weight

With the taxonomies of domains, functions and flows, it has been shown that identifying the domain allows designers to constrain the design variables such as functions and flows to a manageable few. But these taxonomies have an implicit relationship embedded in them: only certain functions are relevant when considering certain flows in a particular domain [74]. So essentially identifying the domain and the flows of the design problem allows us to constrain the taxonomy of functions. This is important because clearly defining the variables applicable to the design problem makes the process of design more efficient and additionally the fewer choices of design variables relevant to the current design problem allow for easier design decisions. This sub-section presents an additional set of taxonomies that define the relationships between functions and the relevant flows. The taxonomies for the “Structures” (“No

motion") domain are presented but the same principle may be applied to other domains in order to prepare complete taxonomies ready for use by design automation systems.

Table 2.7: Relationship among Functions and Flows in Structures (No motion) domain

Absorb *Kinetic* Potential Spring	**Supply** Force Friction Spring Torque Kinetic Potential Spring	**Store** Potential Spring
Distribute Force Friction Weight Spring Torque Potential Gravity Spring	**Link** Force Torque	**Amplify** Force Friction Spring Torque
Decrease Force Friction Spring Torque		

The taxonomies presented have shown that the impact of the domain on the design functions and flows is to limit them. The function and flow taxonomies presented here show that identifying the domain constrains the functions and flows to a manageable few. The focus is mainly on the "Mechanical" domain for this dissertation but Function and flow taxonomies for other domains exist and are suggested as future research. Such delimited taxonomies when used in conjunction with the macroscopic taxonomies presented by Ullman [75], Dixon et al [76] and Pahl and Beitz [58] etc.

serve to provide a complete view of design. One use of our taxonomies could be as a tool to educate novice designers. Current design curricula concentrate on results of numerous analyses of design artefacts as the primary selection criteria. Our delimited taxonomies could be used to teach designers to select the most appropriate solution artefact based on design variables that can be identified from the client's statement such the design domains, functions and flows. Similarly our taxonomies could be used by design automation systems to simplify the path to a solution.

To be truly useful to the designer or design automation systems they must be able to map the function domain to the physical domain. To do so effectively numerous product matrices are proposed based on the following basic idea:

- There exist various artifacts viz. gears, springs, structures etc which provide varying functions while meeting varying objectives and constraints. Designers implicitly group artifacts into classes based on the functions that the artifacts realize.
- Each artifact class has numerous sub-types, (e.g. (i) worm, helical, spur and others are sub-types of gears, (ii) flat and V are sub-types of belts and (iii) torque, conical, barrel, hourglass, torsion are sub-types of springs) that have further constraints that must be considered when selecting that particular sub-type.
- Designers select an artifact class based on the function that they can provide but selecting from the sub-types that exist is usually done

based on certain key constraints. By identifying these key constraints (or syntactic context) the selection process is formalized.

Using that basic idea the following product matrices are contributed. The original product matrix was developed by Dr. John Mills who is using this matrix in his course work to teach design students on using the idea of context based design selection. The following tables provide the product matrices for gears, springs, structures and motors. It should be pointed out that these matrices are in an early draft and further research is required to consider them truly useful. To use the product matrices presented in Figure 2.5, Figure 2.6 and Figure 2.7 identify the constraints in your problem and eliminate any selections that have an “X” in the row. The sub-type(s) that are remaining are the artifacts that will most closely provide your function while meeting the design constraints.

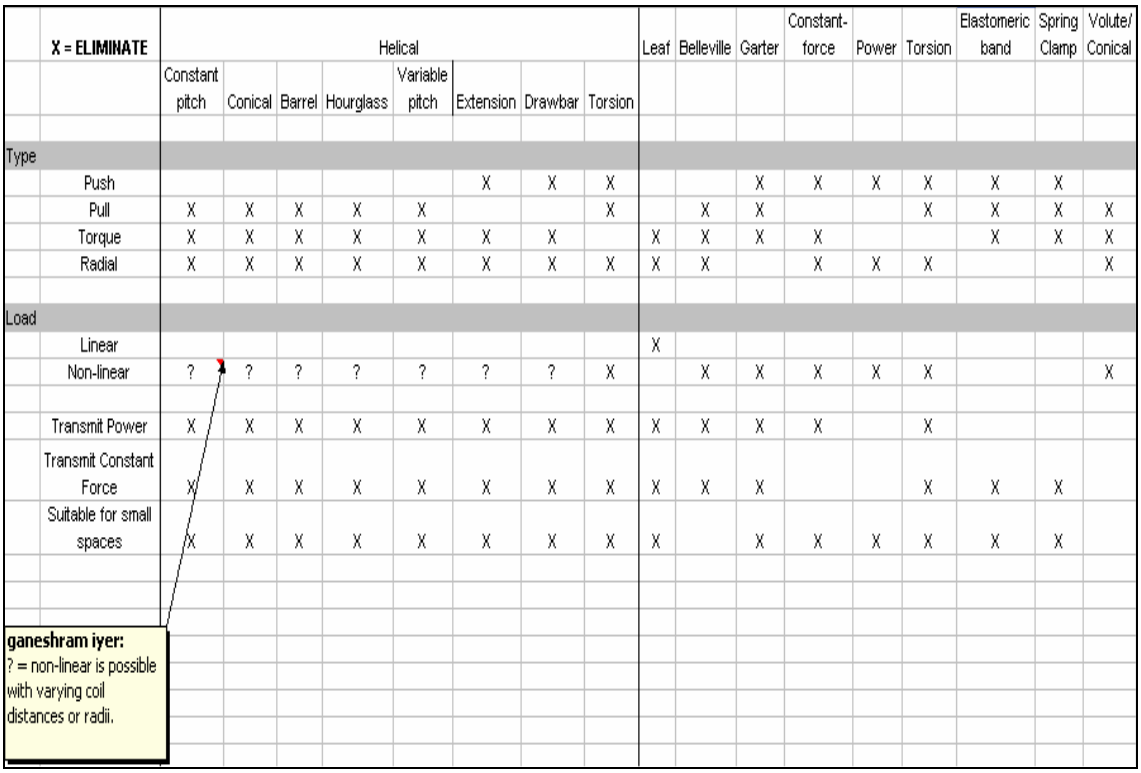

Figure 2.5: Product matrix for springs

2.5 Context in legacy MCAD

The definition for design rationale in legacy MCAD, stated in section 2.3.3, gives us the impact of context in legacy CAD. The definition organizes design rationale into the three levels for context suggested by Sowa. The definition for design rationale mentions three levels of relationships viz. structural, semantic and practical. The structural relationship is a synonym for Sowa's syntactical context, while the practical relationship is synonymous to Sowa's pragmatic context. Iyer et al [77] have also suggested a context-based inference approach to capture design intent from legacy CAD. They use the three levels of context to classify the raw data that is extracted from the legacy CAD into syntax, semantics and pragmatics.

	Gears												Others				
	Spur	Helical		Bevel					Worm	Gear trains			Belts and Pulleys			Chain and Sprocket	Flexible cable
Constraints		Parallel	Crossed	Straight	Helical	Hypoid	Spiroid	Zerol		Simple	Compound,r everted	Compound, non-reverted	Flat	V	Timing		
Geometric Constraints																	
Axis orientation																	
Parallel			X	X	X	X	X	X	X								
Center distance can vary																	
>0.5 addendum	X	X	X	X	X				X								
Very large parallel axis :	X	X	X	X	X	X	X	X	X		X			X	X		
Non-parallel																	
intersecting																	
90	X	X	X						X								
>< 90	X	X	X	X		X	X	X	X								
non-intersecting																	
90	X	X															
No offset	X	X				X	X	X	X								
offset >0.5 gear	X	X	X	X	X	X			X								
offset <0.3 gear	X	X	X	X	X		X	X	X								
<> 90	X	X		X													
No offset	X	X		X		X	X	X	X								
offset >0.5 gear	X	X	X	X	X	X			X								
offset <0.3 gear	X	X	X	X	X		X	X	X								
Space limitation										X		X					
Power transmitted constraint																	
high power	X			X											X		X
medium power																	X
Speed constraint																	
Constant angular velocity																X	
No slip allowed													X	X			
Rotational speed/gear ratio constraint																	
~1								X	X								
>3									X								X
>10																X	X
>50	X	X	X	X	X		X			X			X	X	X	X	X
Surface velocity »																	
>5000 ft/min	X		X	X	X									X	X		
>3000 fr/min																X	
>1000 ft/min				X													
<1000 ft/min														X	X		
Torsional variations possible	X	X	X	X	X		X	X	X					X			
Efficiency constraint																	
>99%			X					X	X	X	X	X	X	X			
>90%			X						X					SOME			
>70%									SOME								
No lubrication possible	X	X	X	X	X		X	X	X	X	X	X				X	X
Low noise	X			X						X			X	X		X	
Cooling not possible									X								

Figure 2.6: Product matrix for gears, belts and pulleys, chains and sprockets and cables (courtesy Dr. John Mills)

		AC											DC			Torque	Servo	Stepper	Linear
		Asynchronous									Sync	Universal	Shunt	Series	Compound				
		Single Phase			Three Phase														
		Capacitor	Permanent-Split Capacitor	Shaded Pole	NEMA A	NEMA B	NEMA C	NEMA D	NEMA F	Wound Rotor									
Motion																			
	Rotary																		X
	Linear										X		X	X	X	X	X	X	
Power																			
	High																		
	Medium																		
	Low																		
Accuracy																			
	High																		
	Low																		
Slip																			
	None			X	X	X	X	X	X	X									
	Low			X				X											
	Medium			X	X	X	X	X	X	X									
	High				X	X	X		X	X									
	%					0.5-5	1-5.0	5-8,8-13, 15-											
Starting Torque																			
	Low				X	X	X	X					X		X				
	Normal		X	X			X	X	X						X				
	Medium		X	X	X	X	X	X	X						X				
	High		X	X	X	X			X										
Starting Current																			
	Low					X	X												
	Normal				X			X	X										
	Medium				X	X	X	X	X										
	High				X	X	X	X	X										
Efficiency																			
	Low					X	X	X		X									
	Medium									X									
	High						X	X											
Speed																			
	Low																		
	Normal																		
	Medium																		
	High																		
	Varying										X								
	Constant												X						
Load																			
	Light													X					
	Medium																		
	Heavy																		

Figure 2.7: Product matrix for motors

By comparing Sowa's levels of context with legacy MCAD we derive the following context levels:

Table 2.8: Context in legacy MCAD

Syntax	Notes, titleblock, shape, symbols
Semantics	Standards, manufacturing, material (with characteristics), parts list, application, milieu, alternatives, part name, inspection

Table 2.8 - continued

Pragmatics	Function, flow, domain, objectives, constraints, design rules, guidelines, designer, project, company info, domain

The following is a sample of the information that is found:

- *Syntax:*
    - The geometry of the drawing to determine overall shape of the part.
    - The tolerance values, including number of decimal places, as an indicator of tight vs. loose tolerances that may be critical for assemblies and part function.
    - The use of firm and relative dimensions to infer the datum plane and feature dependencies in the resulting 3D model. Use of dimensions, table data for validating geometry.
    - Importance of precise placement of holes for connectivity between related parts to form assemblies.
- *Semantic:*
    - The part name as an indicator of the part's function (e.g. bracket, pump, gear etc).
    - The material as an indicator of cost, strength, part thickness, manufacturing process, etc. of the part.
    - The surface finish as an indication of possible exposure to elements, mating connections, etc.
    - Any notes referencing markings and etchings as an indicator that the part may be a replacement part in the field.
    - Correlation of elements in notes and geometry to infer the treatment and/or manufacturing operations.

    - Symbols for manufacturing processes such as finishing, welding, assembling, etc
    - Geometry and attributes of part as an indicator feature objective (i.e. ribs in a part add strength while pockets make the part lighter.)
- *Pragmatic:*
    - Company and department name along with appropriate knowledge as a method to infer information such as applicable standards, specifications, applicable disciplines etc,
    - Designer information to gauge project information,
    - Date/time information to gauge document revision/completion information.

2.6 Conclusion

With the required background the next chapters will address how a context-based approach will address the problem, stated in chapter 1, of capturing design rationale from legacy MCAD.

CHAPTER 3

APPROACH TO SOLVING PROBLEM

The following chapter details the approach proposed to address the problem dealt with in this dissertation. In order to do so the first requirement is to understand the nature of legacy MCAD design rationale. The second requirement is to identify the process of legacy MCAD design rationale capture itself. Only after meeting those requirements can we discuss a software architecture that will incorporate the process identified to satisfying the second requirement. The software architecture is discussed in the next chapter.

3.1 Design Rationale Analysis of Legacy MCAD

From Chapter 1 we know the claim that the manufacturers were making i.e. that they are able to capture design rationale from 2D drawings that they are unable to capture from parametric 3D models. This claim required further investigation and this section details the approach and the results of the investigation. Legacy MCAD, specifically 2D drawings, contains only unstructured graphical entities such as geometry, text and symbols. In order to properly interpret the meaning and significance of the contents of the legacy file, human re-designers implicitly create associations between the unstructured entities to identify more meaningful classes of structured entities. For example by creating associations between relevant geometric entities, the

re-designer identifies the shape of the part. By creating associations between geometry and text the re-designer could identify dimensions and tolerances or manufacturing symbols (e.g. surface finish) depending on the properties of the entities and the associations created. When comparing the contents of a 2D drawing with that of a 3D model it can be seen that the 3D model has more structured geometric entities but uses a different layout as compared to 2D drawings to display their content. Most 3D models do not contain the same text and symbol entities that re-designers expect to see in the same manner as 2D drawings. This additional text and symbols may be the key to understanding the validity of the claim that the re-designers could capture design rationale from the 2D drawings and not 3D models. This reiterates the contention made in section 2.3.3 on the importance of the information other than geometry that is contained in the legacy MCAD file.

3.1.1 Design Rationale Capture Analysis Process

To better understand the importance of all the entities to capture design rationale from legacy MCAD and the process that re-designers use to capture design rationale we used the method suggested by Stauffer, Ullman and Dietterich published in their paper titled “Protocol Analysis of Mechanical Engineering Design” [78] and performed “design rationale capture analysis of legacy MCAD”. We worked with two candidates for the first stage of analysis, the first an experienced re-designer and the second an experienced modeler. We identified sample drawings from a repository available with Imagecom Inc [79]. The two candidates were presented with the same set of sample drawings to ensure consistency between their findings. A sample drawing is shown in

Figure 3.1. The analysis process required the candidates to verbalize their thought process when identifying the entities, any relevant associations between the entities and inferring design rationale.

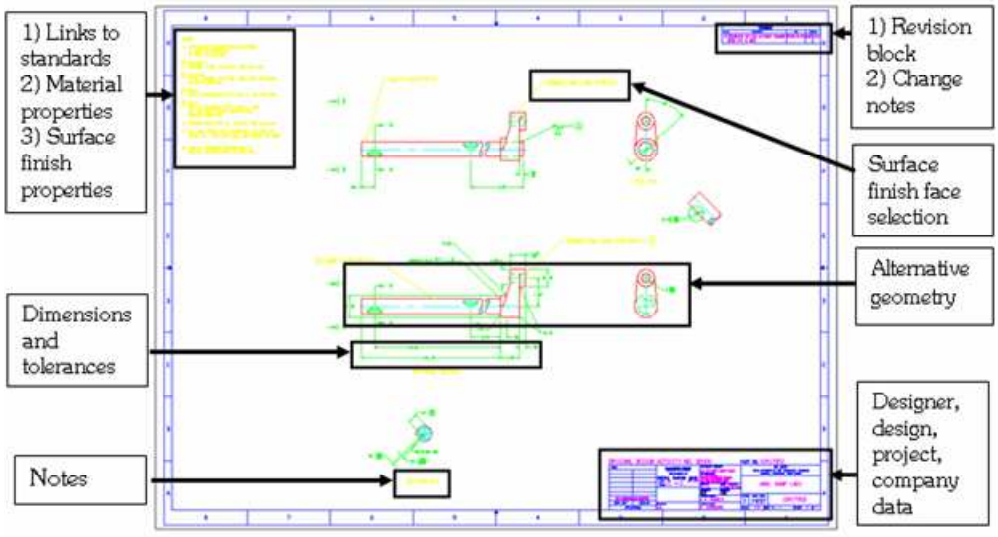

Figure 3.1: Sample 2D drawing

To provide support to the candidates and to impart a formal order to the verbalization, the candidates were presented with a list of questions that they needed to answer. The following is a sample list of questions posed to the interviewees.

- What steps do you take when presented with a MCAD artifact for redesign?
- How do you rate the importance of entities contained in the legacy MCAD – geometric, textual etc?
- Please infer the following classes from the drawing:
    - Function of part
    - Flow related to part

    - Domain of part or assembly
    - Manufacturing instructions and symbols
    - Material
    - Standards, principles and guidelines
    - Alternatives presented (geometry, material, manufacturing etc)
    - Constraints (dimensional, assembly etc)
    - Possible objectives of part
- Please state importance of above identified classes for purpose of redesign of part or assembly contained in drawing.
- Do you use sources external to the MCAD representation to identify the classes mentioned above? If yes, what are these sources – design databases, engineering dictionaries, previous knowledge or other?
- How do you deal with assembly representations? Is there a need for individual part drawings to determine assembly constraints?
- Do assembly drawings provide more information about the function, flow, domain, alternatives, objectives or constraints?
- Do manufacturing instructions included in the legacy CAD provide any clues to the possible objectives or constraints of the part or assembly?
- How do you deal with alternatives (geometry or otherwise) shown on the legacy CAD?
- Do the change notes included in the CAD representation provide any insight into the design evolution of the part or assembly?

3.1.2 Results of Design Rationale Analysis

One result of the design rationale analysis was that the candidates were inferring a lot of meaning and significance from the legacy file, but each individual assigned different importance to the identified classes. The experienced re-designer assigned higher importance to the text and other attributes contained on the legacy MCAD file to infer design rationale such as function, objectives or constraints. The experienced modeler on the other hand focused more on the geometry, parts list and assembly constraints as possible design rationale concerned with the use and application of the part, represented in the MCAD file. The modeler used the inferred application of the part as possible reasons for the decisions contained on the MCAD file. On the whole the candidates could infer information about costs, application, mating surfaces, function, flow and domain by creating associations between the various classes identified in chapter 2. This validates the claims made in Chapter 1 by the manufacturers. The subjects were able to infer design rationale from the legacy CAD beyond the data that is contained on the legacy CAD.

Additionally with this analysis we are able to establish that design rationale on legacy MCAD is not explicitly stated. The entities on the MCAD file and the relationships between those entities i.e. context in legacy MCAD, provide useful information that someone with the required domain knowledge can use to infer design rationale about the represented part.

Finally the process employed by the subjects to capture design rationale from the legacy CAD files was documented. The following are the broad steps that the subjects were using to infer design rationale.

- Brief review of the legacy MCAD file to identify relevant entities and their properties e.g. location, dimensions etc.
    - This step is comparable to the parsing of the files where the human candidates spent some time to identify the types of entities contained on the legacy MCAD file along with their location and relation to other entities based on certain associated properties such as location, type, color, annotations etc. The result of this step is that the human candidates have essentially identified essential entities that are present on the legacy file.
- Having identified the essential entities the next step the human candidates take is to identify the relationships that exist between the identified entities. The key to creating these relationships is their knowledge of the domain. Three types of relationships are created viz. the context levels – syntax, semantics, pragmatics. The first relationship to be created is the syntax, which the candidates create by identifying the properties and attributes of the entities such as location, type, color etc. By doing so they have created groups of entities that would aid in identifying further relationships. Examples of such groups are shape, notes, titleblock, symbols etc. The second relationship created is the semantics. Starting with the groups of entities identified at the syntax stage the candidates

try and identify common keywords and patterns that exist within the groups. These identified keywords and patterns provide meaning to the created groups. Examples of identified keywords would be iron, aluminum, steel (which would indicate materials), gear, yoke, cover (which would indicate artifact names) and ASME, DOD, STD (which would indicate standards).

- The final relationship that the candidates infer is the pragmatics. This involves retrieving valuable information based on their experience and knowledge of the concerned domain of the legacy MCAD file. The meaning they inferred in the second relationship step allows the candidates to create links between a particular meaning and the significance the meaning implies. For example knowing a particular material would allow us to infer certain significance such as metal or non-metal, strength, hardness etc. Knowing the part name allows us to infer for example possible functions and flows. If standards were identified then possible guidelines and principles could be inferred.
- Having inferred the pragmatics the re-designers use these links to infer possible design variables and possible reasons in the form of objectives and constraints, possible alternatives and insights about material choice. All these collectively form design rationale as we have seen in chapter 2. For example by identifying the name of the part they could infer the possible function of the part. From the material, the re-designers could infer possible objectives such as low weight, high strength or corrosion resistance. From the manufacturing process specified

the re-designers could infer if the part was meant for large or small batch products.

3.2 Identify context levels in legacy MCAD

Section 2.5 presented an overview of context in legacy MCAD. With the results of the design rationale analysis presented in section 3.1.2, the following section provides the details of context in legacy MCAD. The three levels of context that were identified in section 2.5 are expanded upon to include the entities contained on the legacy MCAD and numerous other classes that can be inferred based on the relationships between the entities which are also modeled.

The models were created using Microsoft's Visio software in UML. The models are standard UML Class diagrams with Generalization or Dependency relationships. The Generalization relationship can be simply described as "part-of" where all child classes are subparts of the parent class. Similarly the Dependency relationship can be thought of as "depends on" where the properties of the parent class are inherited by the child classes.

3.2.1 Syntax in legacy MCAD

Since legacy MCAD contains only unstructured geometric and textual entities the first step taken by the human candidates during the design rationale analysis was to perform a detailed survey of the entities, their locations, properties, layers and any structural relationships. Using this data the candidates were able to identify the syntax level of the context in legacy MCAD which are the unstructured entities in various related groups. This class typically consists of the shape of the part, dimensions,

tolerances, notes or title-block information. Figure 3.2 shows the syntax level with its child classes.

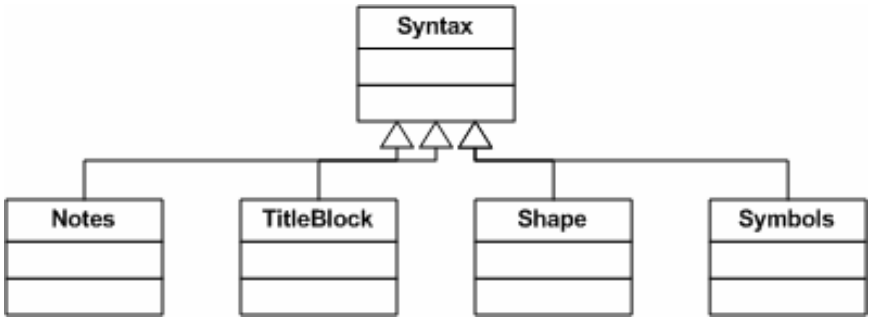

Figure 3.2: Syntax level with child classes

The following is a brief explanation of the child classes of syntax, the entities they contain and the relationships between the entities:

3.2.1.1 Notes

These are typically text with some symbols included. The text can be single-lined or multi-lined, while the symbols are usually used to indicate surface finish, tolerances or dimensions. The notes class is made up of numerous single- or multi- line independent notes that may be numbered or bulleted.

3.2.1.2 Titleblock

Titleblock is typically formed of tables, text and symbols. The tables could be composed of numerous lines arranged to contains rows and columns forming individual cells that contain the text and symbols. The text may be single- or multi- line and could indicate parameter variables with their values, while symbols could be used to indicate global tolerances and milieu information such as the company name, designer, design and project information.

3.2.1.3 Shape

The shape is normally only composed of geometry (point, lines, arcs, circles, splines etc) but could use textual descriptions to indicate details or repeating arrays of entities or groups of entities.

3.2.1.4 Symbols

Symbols are normally made up of both geometric and textual entities. Symbols indicate surface finish, manufacturing, dimensions, tolerances, markers etc. Based on the shape and contents the appropriate symbol is inferred.

3.2.2 Semantics in legacy MCAD

Having identified the syntax classes the next step that the human candidates take is to identify the semantics level from the syntax classes using a variety of techniques. By looking for certain patterns and keywords that can be found in the syntax classes the candidates identify the semantics classes that are shown in Figure 3.3.

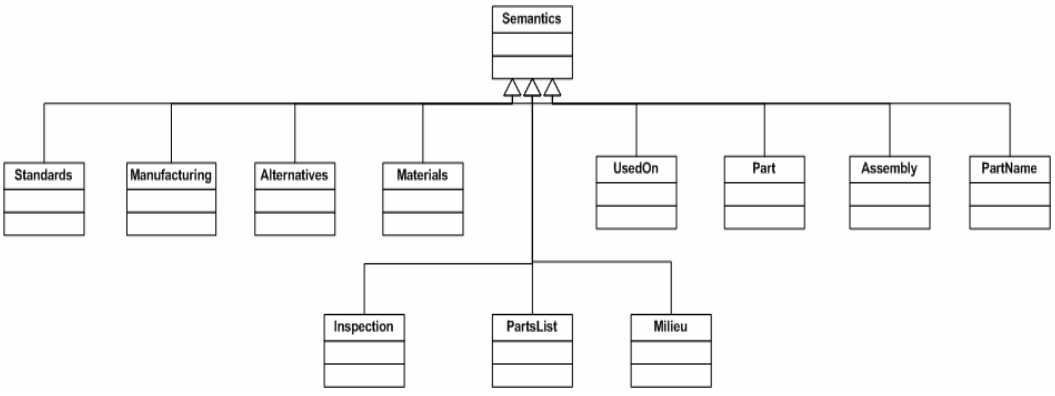

Figure 3.3: Semantics level with child classes

The following is a brief explanation of the child classes of semantics, the entities they contain and the relationships between the entities:

3.2.2.1 Standards

This class, shown in Figure 3.4, typically contains the standards, guidelines and principles that are composed of text and explicitly stated in the notes or title-block classes. Some examples of standards are DOD-STD-00100D (AR), ANSI Y14.5M-1982 etc. Standards are company, project or design specific.

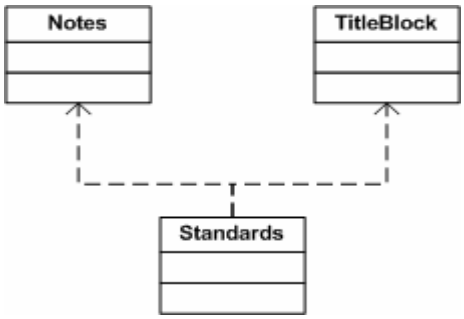

Figure 3.4: Standards class dependent on Notes and Titleblock classes

3.2.2.2 Manufacturing

The manufacturing class is normally composed of text or symbols and contains instructions for the production of the part represented in the legacy MCAD. Some examples of text instructions are casting, forging, quench, temper etc. Surface finish symbols and weld symbols are examples of the manufacturing symbols that contain text to indicate required values.

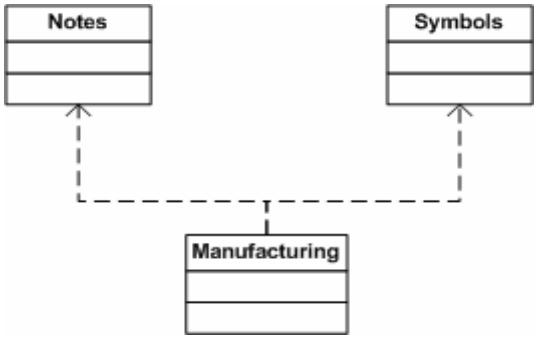

Figure 3.5: Manufacturing class dependent on Notes and Symbols classes

3.2.2.3 Alternatives

Alternative shapes, materials or manufacturing instructions are usually explicitly represented and appropriately labeled as such. They could contain geometry or text and can be inferred from shape, notes and symbols.

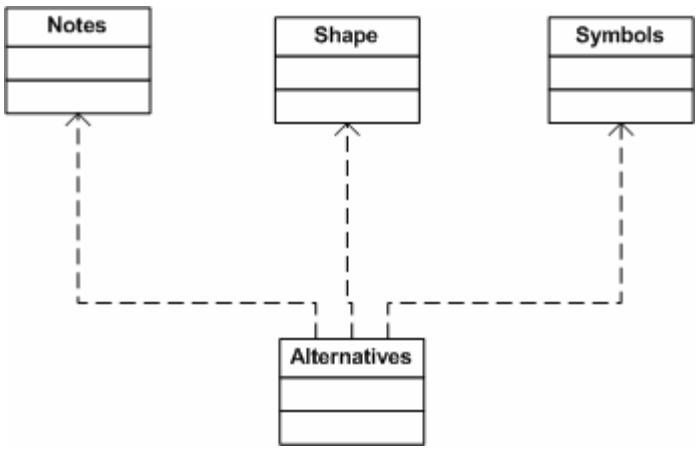

Figure 3.6: Alternatives class dependent on Notes, Shape and Symbols classes

3.2.2.4 Materials

The Materials class is typically composed of text entities and is derived from Notes class. Some examples are Steel, Aluminum, Al, Bronze, Cu, Sn etc.

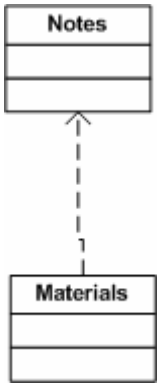

Figure 3.7: Materials class dependent on Notes class

3.2.2.5 UsedOn

The UsedOn class is composed of text and depends on the Titleblock class. It indicates the assembly or the system level artifact of which the currently represented artifact is a part.

Figure 3.8: UsedOn class dependent on Titleblock class

3.2.2.6 Part and Assembly

The Part and Assembly classes are composed of text entities and depend on the Titleblock, Shape and Symbol classes. They help identify whether the currently represented artifact is a piece part of an assembly.

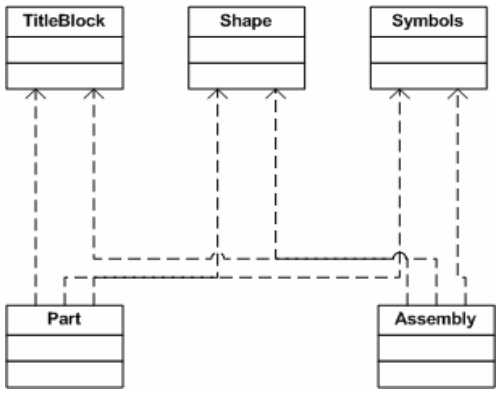

Figure 3.9: Part and Assembly classes depend on Titleblock, Shape and Symbols class

3.2.2.7 PartName

The PartName class is composed of text entities and depends on the Titleblock class.

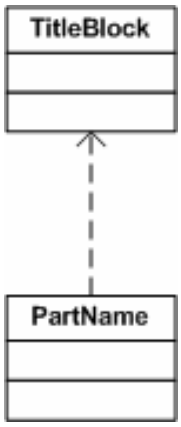

Figure 3.10: PartName class depends on Titleblock class

3.2.2.8 Inspection

The Inspection class is composed typically of symbols and text and depends on the Symbols and Notes class.

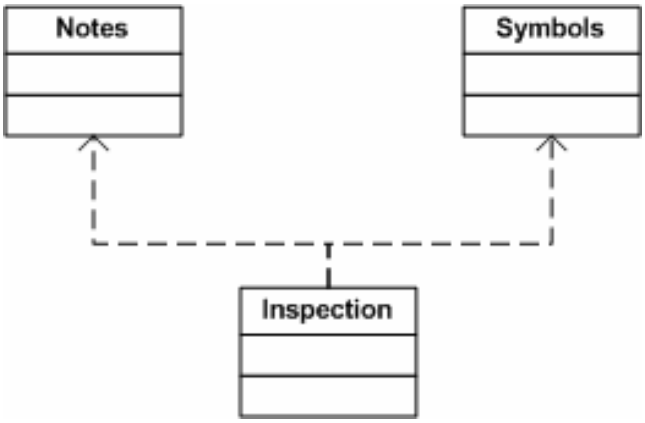

Figure 3.11: Inspection class depends on Notes and Symbols classes

3.2.2.9 PartsList and Milieu

The PartsList and Milieu classes are composed of text and depend on the Titleblock class.

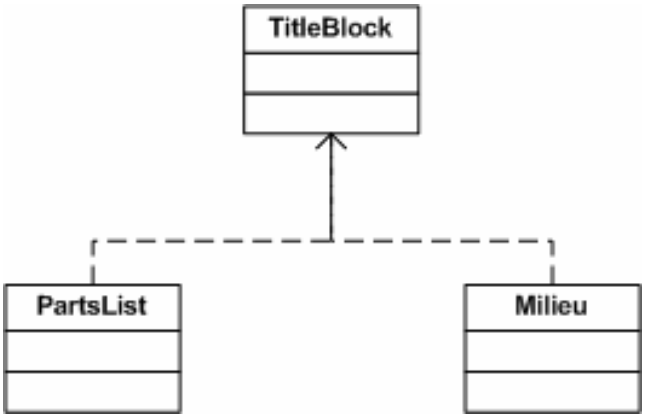

Figure 3.12: PartsList and Milieu classes depend on Titleblock class

3.2.3 Pragmatics in legacy MCAD

Having identified the semantics classes the human candidates inferred the final level i.e. the pragmatics level by using their expertise and knowledge of the domain. By considering each semantics class individually the candidates infer the appropriate pragmatics class. Figure 3.13 shows the child classes at the pragmatics level.

The following is a brief explanation of the child classes of pragmatics, the entities they contain and the relationships between the entities:

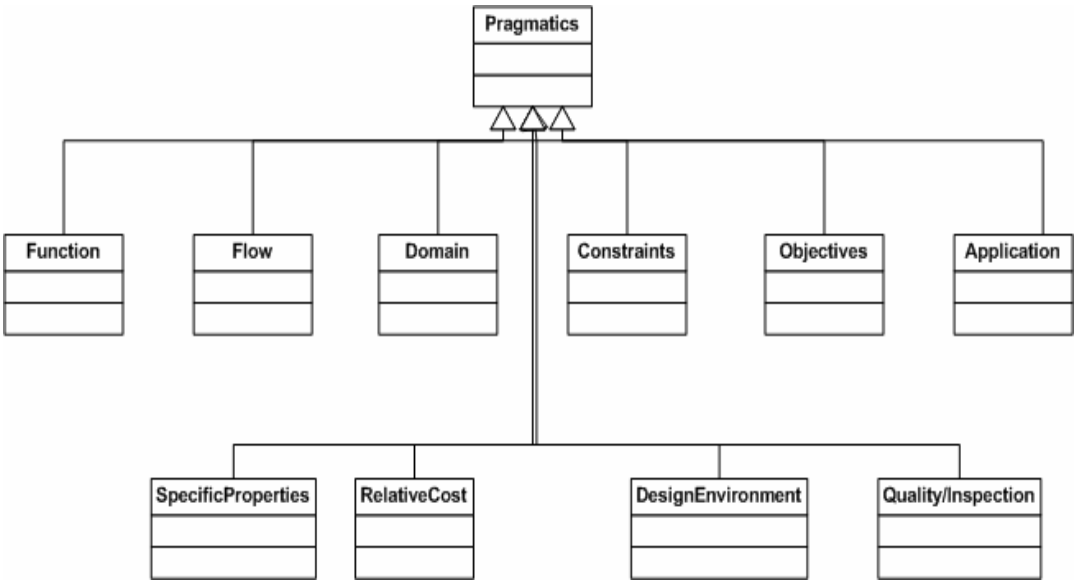

Figure 3.13: Pragmatics level with child classes in legacy MCAD

3.2.3.1 Function, Flow, Domain and Application

Information about the Function, Flow, Domain and Application can be inferred from the PartName semantics class.

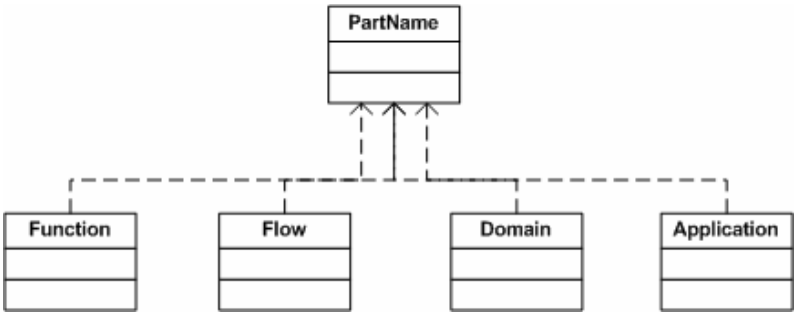

Figure 3.14: Function, Flow, Domain and Application inferred from PartName

3.2.3.2 DesignEnvironment, QualityInspection, Objectives and Constraints

Information about the DesignEnvironment, QualityInspection, Objectives and Constraints can be inferred from the Standards semantics class.

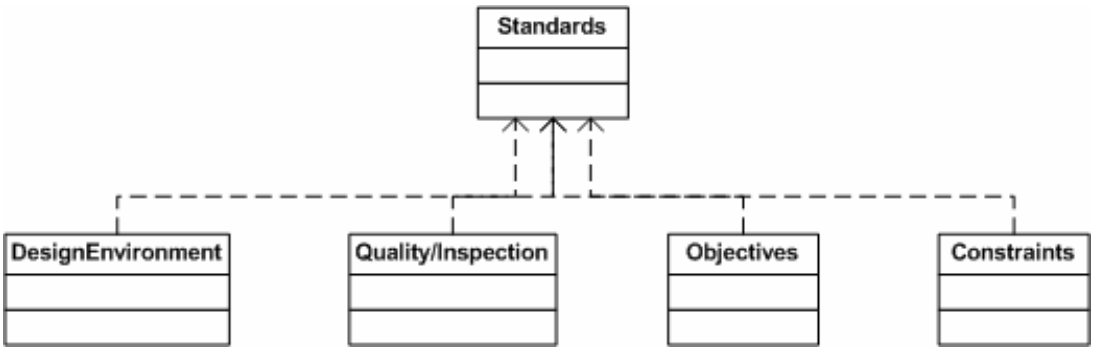

Figure 3.15: DesignEnvironment, QualityInspection, Objectives and Constraints inferred from Standards

3.2.3.3 Constraints, RelativeCost, QualityInspection

We can infer information about Constraints, RelativeCost and QualityInspection from the Manufacturing semantics class.

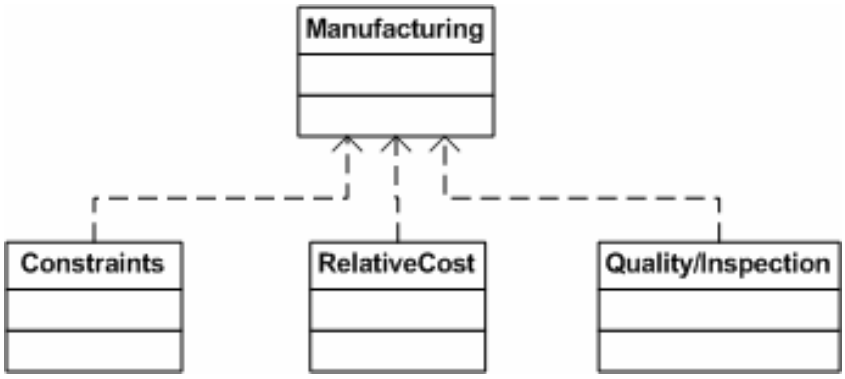

Figure 3.16: Constraints, RelativeCost and QualityInspection inferred from Manufacturing

3.2.3.4 Objectives and Constraints

Information about Objectives and Constraints can be inferred from the Alternatives semantics class.

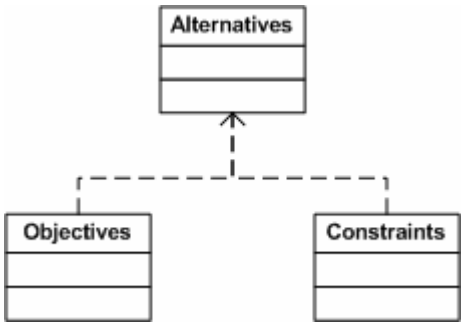

Figure 3.17: Objectives and Constraints inferred from Alternatives

3.2.3.5 Objectives, SpecificProperties, Application and RelativeCost

We can infer information about Objectives, SpecificProperties, Application and RelativeCost from the Materials semantics class.

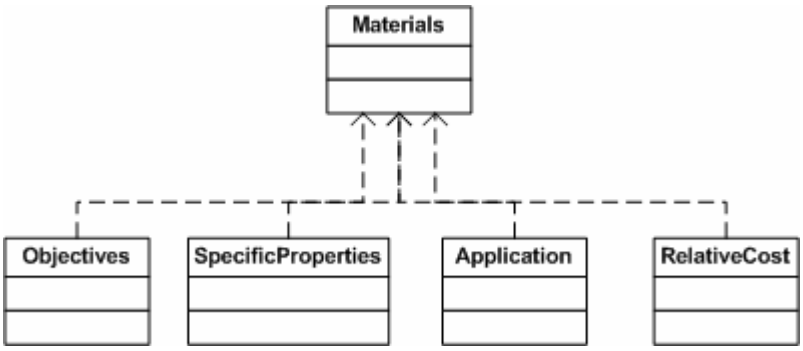

Figure 3.18: Objectives, SpecificProperties, Application and RelativeCost inferred from Materials

3.2.3.6 Application and DesignEnvironment

Information about the Application and DesignEnvironment can be inferred from the UsedOn semantics class.

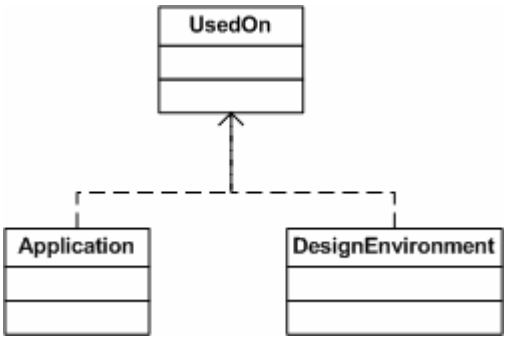

Figure 3.19: Application and DesignEnvironment inferred from UsedOn

3.2.3.7 Constraints and Application

We can infer information about Constraints and Application from the Part and Assembly semantics classes.

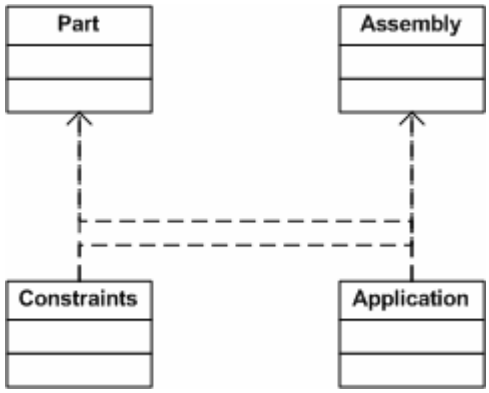

Figure 3.20: Constraints and Application inferred from Part and Assembly

3.2.3.8 RelativeCost and QualityInspection

Information about the RelativeCost and QualityInspection can be inferred from the Inspection semantics class.

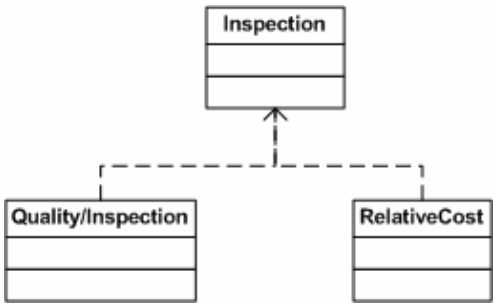

Figure 3.21: QualityInspection and RelativeCost inferred from Inspection

3.2.3.9 Function and Constraints

Information about the Function and Constraints can be inferred from the PartsList semantics class.

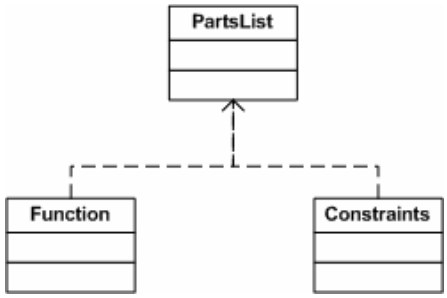

Figure 3.22: Function and Constraints inferred from PartsList

3.2.3.10 Application, DesignEnvironment and Domain

We can infer information about Application, DesignEnvironment and Domain from the Milieu semantics class.

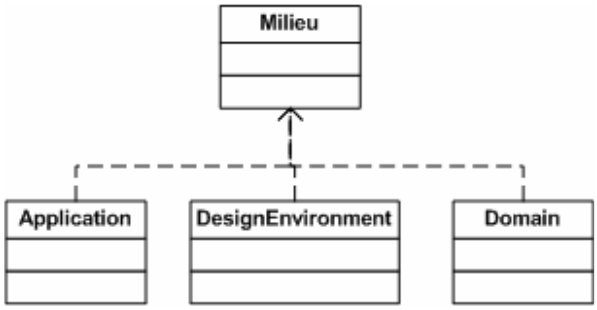

Figure 3.23: Application, DesignEnvironment and Domain inferred from Milieu

3.3 Automated capture of context from legacy MCAD

Having provided an understanding of the context that can be captured from legacy MCAD and the relationships that exist between the context classes at the three levels, this section describes the need and process of capturing the context from legacy MCAD using an automated software system.

3.3.1 Need for automated capture of design rationale

One of the results of the design rationale capture analysis stated in section 3.1.2 was the process that the re-designers used to extract context from legacy MCAD and capture design rationale from the extracted context. If design rationale that is contained is legacy MCAD is to prove useful i.e. that re-designers looking to modify existing designs use the rationale contained in these legacy design storehouses, then the design rationale capture process needs to be simplified by providing an automated and standard capture method.

3.3.2 Process of automated design rationale capture

The automated capture design rationale from legacy MCAD has two distinct steps viz.

- Extraction of context from the legacy MCAD
- Inferring design rationale from the extracted context

An illustration of this process is shown in Figure 3.24. The following sub-sections provide the details on the process emulated by the automated software system. To be truly useful, the proposed automated system must closely emulate the process followed by the re-designer and modeler described in section 3.1.2.

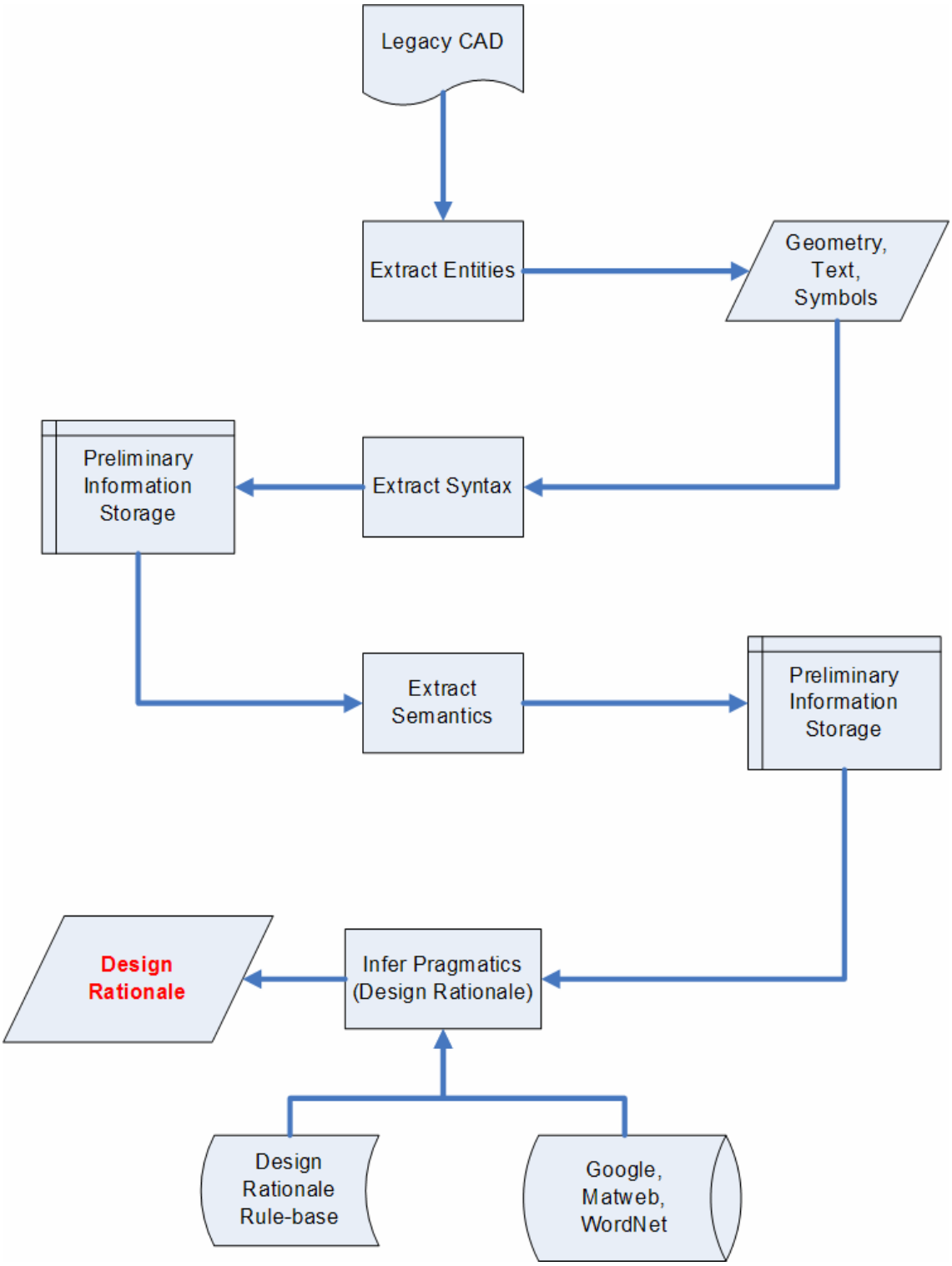

Figure 3.24: Overall approach to capture Design Rationale from Legacy MCAD

It should be noted that some steps of the process followed by re-designers may prove impossible to emulate using software but the number of such steps can be reduced and the quality of the output of the system can be improved by building a system that will grow based on input from experienced re-designers. In short by making the system extensible we can ensure that the output is increasingly closer to that captured by the re-designer and modeler.

3.3.2.1 Extract context from legacy MCAD

To extract context from legacy MCAD the re-designer and modeler followed a process that is described in section 3.1.2. For an automated system to capture context in a similar manner the following are the steps. These steps are analogous to the steps of the process described in section 3.1.2.

- Read all the raw data contained in the legacy MCAD: At this stage all entities contained on the legacy CAD are extracted, and, for simplification, sorted into either geometric or textual entity types. Basic pattern recognition techniques are used to informally group and categorize the entities into these types. A temporary data storage is used for the output of this step. By using an open ASCII based MCAD file such as DXF the system will be able to read the raw data in addition to their properties.
- Extract the syntax classes: By identifying the relationships between the raw data as described in section 3.2.1 the system will extract the syntax classes. From the Design Rationale Capture Analysis it was observed

that the human candidates used the layout of the raw data to extract relationships. Using comprehensive pattern search and analysis methods we categorize the extracted entities. To achieve this in a repeatable manner the patterns could be pre-determined and stored. By identifying the common patterns that exist for different situations e.g. company, project or industry we can create templates that describe the syntax relationships that exist. Using these templates we then focus on the extracted entity along with its context (e.g. surrounding entities, entity type, entity properties) as defined by the template.

- Extract the semantics classes: By using the relationships between the syntax classes the system can extract the semantics classes as stated in section 3.2.2. In a manner similar to the process documented during the design rationale capture analysis the system uses numerous keywords files and pattern matching algorithms to extract the semantics from the syntax classes.
- Infer the pragmatics classes: Using the relationships between the semantics classes as stated in section 3.2.3 the system can infer the pragmatics classes.

3.3.2.2 Infer Design Rationale

The final step is to infer design rationale using the extracted contextual relationships. To do so, an inference engine was implemented as a part of this dissertation. An inference engine is a software component that is implemented

specifically to separate the data from the logic. Logic is language of reasoning. It consists of a collection of rules that is used when reasoning. The next section describes the various types of logic that exist, the type of logic that is implemented through the proposed inference engine and the reason for choosing that type of logic.

1. Propositional logic: Propositional logic is logic at the sentential level. The smallest unit of information that has to be dealt with is the sentence which is called a proposition. The reasoning process assumes each statement as either true or false. No analysis is performed of the individual statements. The primary goal of propositional logic is to identify truth or falsehood of sentences based on preceding sentences. The following example illustrates propositional logic. Consider the following statements.

   All men are mortals

   Socrates is a man

   An inference engine treats these statements as true and based on these propositions can assert the following proposition which is treated as true for any succeeding propositions that may follow.

   Therefore Socrates is a mortal

   Propositional logic is not powerful enough to represent all types of assertions that are used in computer science and mathematics or to express certain types of relationships between propositions such as equivalence. For example, the assertion "x is greater than 1", where x is a

variable, is not a proposition because you can not tell whether it is true or false unless you know the value of x. Thus propositional logic can not deal with such sentences. However, such assertions appear quite often in mathematics and we want to perform inference on those assertions. Thus we need more powerful logic to deal with these and other problems. To address this need other types of logic exist.

2. Predicate logic: Predicate logic allows us to represent fairly complex facts about the world, and to derive new facts in a way that guarantees that, if the initial facts were true then so are the conclusions. It is a well understood formal language, with well-defined syntax, semantics and rules of inference [80].
3. Syllogistic logic: Syllogistic logic contains the analysis of the judgments into propositions consisting of two terms that are related by one of a fixed number of relations and the expression of inferences by means of syllogisms that consists of two propositions sharing a common term as premise, and a conclusion which was a proposition involving the two unrelated terms from the premises.
4. Modal logic: Modal logic deals with the phenomenon that subparts of a sentence may have their semantics modified by special verbs or modal particles.
5. Mathematical logic: Mathematical logic is a subfield of mathematics that is concerned with formal systems in relation to the way that they encode

intuitive concepts of mathematical objects such as sets and numbers, proofs, and computation.

6. Philosophical logic: Philosophical logic is the study of the more specifically philosophical aspects of logic. It deals with formal descriptions of natural language. It is concerned only with those entities — thoughts, sentences, or propositions — that are capable of being true and false [81].

The inference engine implemented with this dissertation follows propositional logic. The reason for selecting propositional logic over other types of logic is the format and nature of the data available in the limited domain of legacy MCAD. As has been detailed previously, legacy MCAD is typically composed of unstructured graphical entities such as geometry, text and symbols. This results in the captured contextual relationships containing identifiable propositions such as the material selected, manufacturing instructions, part name, individual assembly parts, dimensions, tolerances and design environment such company name, project name, and designer name. The resultant propositions are assumed as true by default without further analysis. This goes back to one of the assumptions of the dissertation that a valid rationale exists behind any information that is included on legacy MCAD. In the case of legacy MCAD the individual propositions do not need additional analysis to assert facts. Since the primary goal of the advanced types of logic is to analyze the propositions further they would be inapplicable in the specific context of legacy MCAD.

The overall architecture of the implemented inference engine is shown in Figure 3.24. The input to the system is the information extracted from the legacy MCAD viz. Assembly, Artifact, Material, Manufacturing conditions and parameters and Geometry. The inferencing itself is done in numerous sub-modules, also shown in Figure 3.25 viz. Infer Domain, Infer Function, Infer Flow, Infer Objectives, Infer Constraints, Infer Material Properties. The details on each of these sub-modules are provided in following sections. For the implemented inference engine to perform the task of asserting succeeding propositions based on the input (preceding) propositions it has a collections of known propositions or rules that is stored in a rule-base, also shown in Figure 3.25 viz. Domains, Functions, Flows and Production Rules. The next section details the rule-base, its need and format and the difficulties that exist to build and extend the base.

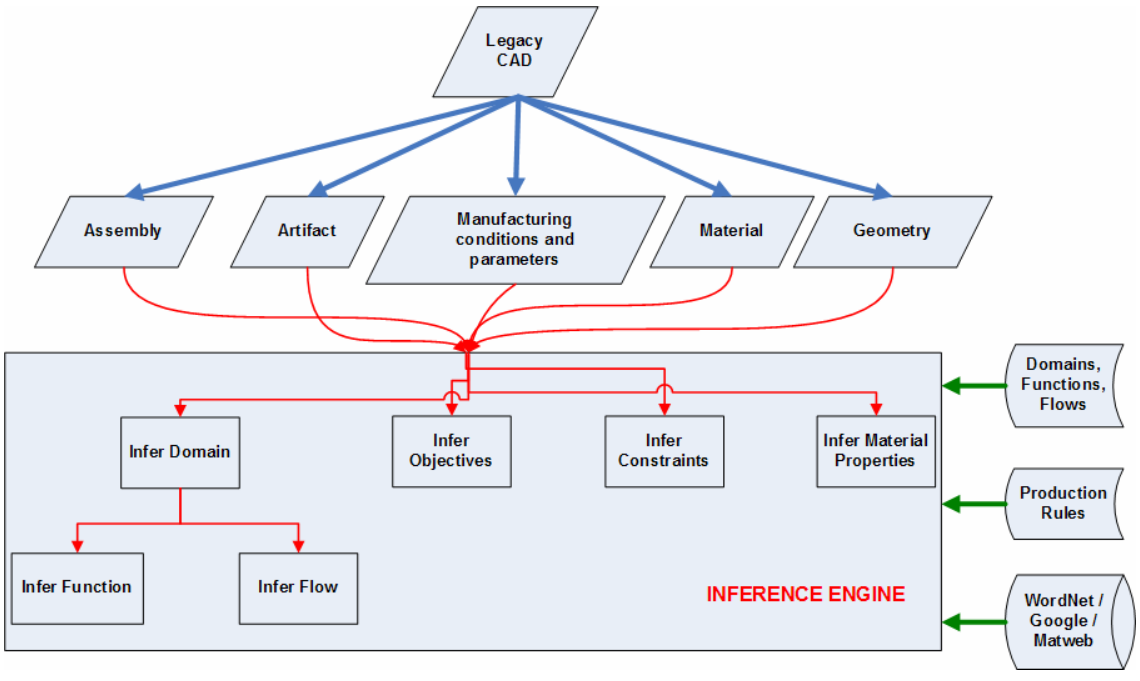

Figure 3.25: Overall architecture of Inference Engine

The rule-base is where the collection of known propositions is stored. An inference engine needs a rule-base as this helps separate the data from the logic implemented by the engine to select appropriate rules based on the input. The rule-base also provides an easy to understand explanation of why a particular rule was selected as no knowledge of the inference engine code is needed. By implementing the rule-base using XML we achieve easy editing of existing rules and addition of new rules. Other formats for rule-bases exist such as the Resource Description Framework which is a World Wide Web Consortium (W3C) specification for modeling information. The RuleML open specification also exists that provides a markup language to publish and share rules on the World Wide Web [82]. The rule-base implemented in this dissertation implements a format simpler than that suggested by RuleML. The reason for this is the lack for information that is needed to completely populate the individual rules as specified by RuleML.

Samples of the rules that were generated with this dissertation are presented in Figure 3.26 and Figure 3.27. The following is the syntax that was used to create the individual rules.

- *<rule>:* Each individual rule is enclosed within this node. The start and end tags separate individual rules.
- *<name>:* The name of the individual rule. The value of this entry needs to be unique for each rule. This entry is required.

- *<symbol>:* The symbol is usually reserved for materials and is usually specified as the symbol of the material that the <name> node states. This entry is optional but must be included when materials are involved.
- *<objective>:* The objectives that are associated with the rule. This entry is optional. If provided then the inference engine selects the objectives for the matched rule. The objective node has two sub-nodes.
    - *<property>:* This entry states the specific objective at the current sub-node. If the <objective> node is included then at least one node containing the property is required.
    - *<value>:* This entry states the value for the property at the current node. If the <objective> node is included then at least one node containing the value for the stated property is required.
- *<constraint>:* The constraints that are associated with the rule. This entry is optional. If provided then the inference engine selects the constraints for the matched rule. The constraint node has two sub-nodes:
    - *<property>:* This entry states the specific constraint at the current sub-node. If the <constraint> node is included then at least one node containing the property is required.
    - *<value>:* This entry states the value for the property at the current node. If the <constraint> node is included then at least one node containing the value for the stated property is required.

- *<additionalrule>:* Specifies any additional rules that are associated to the current rule. This entry is optional. If provided then the inference engine processes the rule stated in this node to match and select additional specific rules. For example if the additional rule stated points to a second rule-base then the inference engine processes this second rule-base and matches and selects additional rules from this rule-base. This is important in cases such as materials that require additional processing to select all appropriate rules.

With the above description of the need and format of the rule-base this next section describes the population of the rule-base, the method, the collected propositions and the problems encountered.

For the inference engine to function the rule-base needs appropriate propositions to be present in the rule-base. The inference engine uses these initial propositions to match and select the appropriate rules based on the current input. These propositions were created manually from various sources such as Tool Manufacturer's Engineering Handbook [83], Mechanical Engineering Design [84], Machine Design: An Integrated Approach [85], Machine Elements in Mechanical Design [86], Mechanics of Materials [87], and Materials Selection in Mechanical Design [88] among others. A large portion of the rules have their origins in the output of the Design Rationale Analysis [Section 3.1]. Sample of the rule-base are shown in Figure 3.26 and Figure 3.27. The rule-base does not differentiate among the various types of proposition being stored. The rule-

base stores propositions on material, manufacturing, company information, functions, flows, domains etc.

```
<rule>
  <name>tacom</name>
  <symbol></symbol>
  <objective>
    <property>reliability</property>
    <value>high</value>
  </objective>
  <objective>
    <property>cost</property>
    <value>low priority</value>
  </objective>
  <constraint>
    <property>shelf life</property>
    <value>long</value>
  </constraint>
  <additionalrule></additionalrule>
</rule>
<rule>
  <name>forging</name>
  <symbol></symbol>
  <objective>
    <property>number of parts manufactured</property>
    <value>typically large</value>
  </objective>
  <objective>
    <property>dies</property>
    <value>single set of reusable</value>
  </objective>
  <objective>
    <property>turnaround time</property>
    <value>once dies have been manufactured, low</value>
  </objective>
  <objective>
    <property>material properties at surface</property>
    <value>good</value>
  </objective>
  <objective>
    <property>repeatability of dimensions</property>
    <value>good</value>
  </objective>
  <constraint>
    <property>geometric dimension</property>
    <value>restricted</value>
```

Figure 3.26: Sample rules from rule-base used by inference engine

```
<rules>
  <rule>
    <name>aluminum</name>
    <symbol>al</symbol>
    <objective>
      <property>weight</property>
      <value>low</value>
    </objective>
    <objective>
      <property>strength</property>
      <value>medium to high</value>
    </objective>
    <objective>
      <property>corrosion resistance</property>
      <value>high</value>
    </objective>
    <objective>
      <property>workability</property>
      <value>good</value>
    </objective>
    <constraint>
      <property></property>
      <value></value>
    </constraint>
<additionalrule>C:\cygwin\home\ganesh\cad\dissertation\dev\2005
\read_dxf\data\aluminum.xml</additionalrule>
  </rule>
  <rule>
    <name>steel</name>
    <symbol></symbol>
    <objective>
      <property>strength</property>
      <value>low to high</value>
    </objective>
    <objective>
      <property>strength to weight ratio</property>
      <value>medium to high</value>
    </objective>
```

Figure 3.27: Sample rules from rule-base used by inference engine

The rule-base stores propositions by identifying the object about which the proposition is predicated. For example consider the rule named “aluminum” in Figure 3.27. This rule asserts the following propositions:

The material is aluminum

Aluminum has symbol Al

Aluminum has objective of low weight

Aluminum has objective of medium to high strength

Aluminum has objective of high corrosion resistance

Aluminum has objective of good workability

Aluminum has additional rule-base located in “aluminum.xml” on disk

This rule is sufficient to handle input that provides very little information to a lot. For example if the input states just aluminum, the input proposition can be read as:

The Material is Aluminum

Since this is assumed true the following propositions can be asserted about the subject “Material” by the above described rule:

Material has symbol Al

Material has objective of low weight

Material has objective of medium to high strength

Material has objective of high corrosion resistance

Material has objective of good workability

Material has additional rule-base located in “aluminum.xml” on disk

Thus by using the rule-base we are able to assert the objectives for the input material (viz. Aluminum) which constitutes design rationale based on the proposed definition [30].

If in case the input states detailed information such as Aluminum 4140 the input proposition can be read as follows:

The Material is Aluminum 4140

This allows us to assert the propositions that are asserted above when the proposition is simply “The Material is Aluminum”. But to assert additional propositions based on the detailed input “4140” we need to refer to the additional rule-base located in “aluminum.xml”. A sample of this rule-base is shown in Figure 3.28. From the detailed input it can be asserted known that the material belongs to the 4000 series of wrought Aluminum.

This assertion depends on the numbering formats that are employed for designating the individual alloys. This designation system is completely set forth in the American National Standard ANSI H35.1 and in various publications issued by The Aluminum Association [83]. Appropriately the following propositions can be asserted by selecting the <4000> series node for <wrought> aluminum from the additional rule-base [Figure 3.28]:

Material has Typical Applications in welding wire, low melting point brazing alloys and architecture

Material has Objective of low weight

Material has Objective of high electrical conductivity

Material has Objective of high machinability

Material has Objective of low melting point

```
<rules>
  <rule>
    <name>wrought</name>
    <symbol>1000</symbol>
    <objective>
      <property>for electrical and chemical fields</property>
      <value>good</value>
    </objective>
    <objective>
      <property>corrosion resistance</property>
      <value>excellent</value>
    </objective>
    <objective>
      <property>workability</property>
      <value>excellent</value>
    </objective>
    <objective>
      <property>weight</property>
      <value>low</value>
    </objective>
    <objective>
      <property>formability</property>
      <value>high</value>
    </objective>
    <objective>
      <property>machinability</property>
      <value>high</value>
    </objective>
    <objective>
      <property>electrical conductivity</property>
      <value>high</value>
    </objective>
    <constraint>
      <property>strength</property>
      <value>low, but can be increased by strength hardening</value>
    </constraint>
    <constraint>
      <property>hardness</property>
      <value>low</value>
    </constraint>
  </rule>
```

Figure 3.28: Rule-base for Aluminum

Any asserted propositions that are repeating between the simple and detailed input can be safely ignored. Thus having detailed input allows for additional propositions being asserted in terms of typical applications and additional objectives, which provides design rationale as previously defined. As can be noticed from the description above the following propositions were also used in addition to the input proposition to assert the design rationale:

Wrought Aluminum 4000 Series starts with the digit 4

Wrought Aluminum 4000 Series is a 4 digit system

Using these above propositions the following proposition was asserted:

4140 is part of Wrought Aluminum 4000 Series

These propositions are part of rule-base as the material designation systems are very specific to the material i.e. there is one designation for Aluminum, another for Copper (the 5 digit Unified Numbering System for Metals and Alloys [83]), another for Steel (4 digit AISI, ASTM or SAE) and so on for other materials. The <additionalrule> node specifies the location of the rule-base that stores these propositions for the individual materials.

But it should be noted that while some designations assist in identifying the detailed proposition from the additional rule-base this is not the case for all materials. This next section describes some of the problems that were faced in matching and selecting the additional rule from the material's designation. The prime example in this case is Steel. Steels are available in numerous types viz. carbon, alloy, high-strength low-alloy, stainless, maraging and cast. The most common type of steel is carbon steel,

which comes in three main groups – low, medium and high. A four-numerical series adopted by the AISI and the SAO is used to designate standard carbon steels specified to chemical decomposition ranges [87]. Designers select from among carbon steels based on the carbon content which provides information regarding the steel properties. The designations do not provide any way to differentiate between the various groups of carbon steels if only the 4 digit designation number is known. So no rule can be used to identify the carbon steel group i.e. no proposition exists similar to the ones stated above for Aluminum. For example consider the following proposition:

Low carbon steels have low strength

Consider for example that the input provides detailed material information as:

The material is Steel 1018

In order to assert propositions regarding the material then we would need the following proposition:

Steel 1018 is a low carbon steel

But this proposition is not available and cannot be deduced from the designation system for Steels. In order to circumvent this problem the proposed and implemented method is to use open materials database available in digital format such as efunda [89] and MatWeb [90]. There are no known researches or applications that utilize this idea to identify propositions for use in an inference engine but the results from the current implementation suggest that this is a viable method. The basic idea is similar to that suggested by Google's Custom Search Engine which is a tailored search engine, which prioritizes or restricts search results based on websites and pages that are specified, and

which can be tailored to reflect a specific point of view or area of expertise [91]. In the case of materials the aforementioned websites [89], [90] are the specified websites that provide materials related information required for generating necessary propositions.

The next section provides a detailed understanding of the implemented inference engine. The inference engine consists of three distinct parts:

1. Production Rules: The Production Rules are stored in the rule-base. These describe the condition that has to be matched and also specify the action that the inference engine must take when the condition is triggered.
2. Working Memory: Consists of the input that is then treated as a proposition.
3. Pattern Matcher: This is the main component of the inference engine. The primary goal of the Matcher is to identify production rules that closely match the input, select one or at the most few rules by resolving conflicts occurring due to the selected rules and execute the rules.

The inference engine has three basic processes: Match rules, Select rules and Execute rules – which it performs to infer the appropriate rule. In Match Rule all the rules that are similar to the input are retrieved. In Select Rule the rule that is exactly like the input is selected by eliminating the other rules based on additional processing. Finally in Execute Rule the rule is executed which may only be displaying the properties of the selected rule or may be finding additional rules that match the input thereby repeating the process from Match Rule. Also there are two basic types of

inference engines – forward chaining and backward chaining. Forward chaining is data based while backward chaining is goal driven. In the delimited context of legacy MCAD only forward chaining is relevant as there is no appropriate goal that needs to be achieved which is the basic function of backward chaining.

3.4 Necessary Validation of Automatically Captured Design Rationale

Once the automated system has inferred design rationale the final step is to test the validity and quality of the captured design rationale. This dissertation proposes to perform this validation step using human re-designers with varying experience and knowledge levels. In a manner similar to the design rationale capture analysis described in section 3.1.1 we conduct validation interviews where human re-designers record the design rationale that they were able to capture from a sample set of drawings. The software system is also fed the same sample set and its output is recorded. The two sets of outputs are compared for validating and checking quality of the output from the software system. This procedure is dealt with in more detail in chapter 5.

3.5 Conclusion

With this chapter we have seen the approach followed to address the specific details of design rationale, context in design rationale and capture of design rationale from legacy MCAD. The next chapter details the software architecture of the system proposed to perform the automated capture of design rationale from legacy MCAD.

CHAPTER 4

SOFTWARE ARCHITECTURE

The following chapter provides the details of the architecture of the software system proposed to automate the capture of design rationale from legacy MCAD. Chapter 3 briefly described the two primary steps that have to be taken by a software system viz. (i) extract context from legacy MCAD and (ii) use extracted context to infer design rationale. The following sections expand on these two steps starting with an overall architecture and then drill deeper into each identified sub-step.

4.1 Overall Architecture

The overall architecture is based on the approach stated in section 3.3.2 in Chapter 3. An illustration of this overall architecture is shown in Figure 4.1. The following is a brief overview of the steps in the process:

4.1.1 Entity Parsing

The first step in the legacy MCAD design rationale capture process is the parsing of entities. In this step we read and parse three basic classes of entities viz. geometry, text and symbols.

4.1.2 Extract Syntax

Having parsed the three basic entity classes the next step is to extract the syntax. This is achieved by categorizing the entities parsed in the previous step into the

appropriate syntax class viz. notes, title-block, shape and symbol. This is done by using templates that describe the class including their location and other properties.

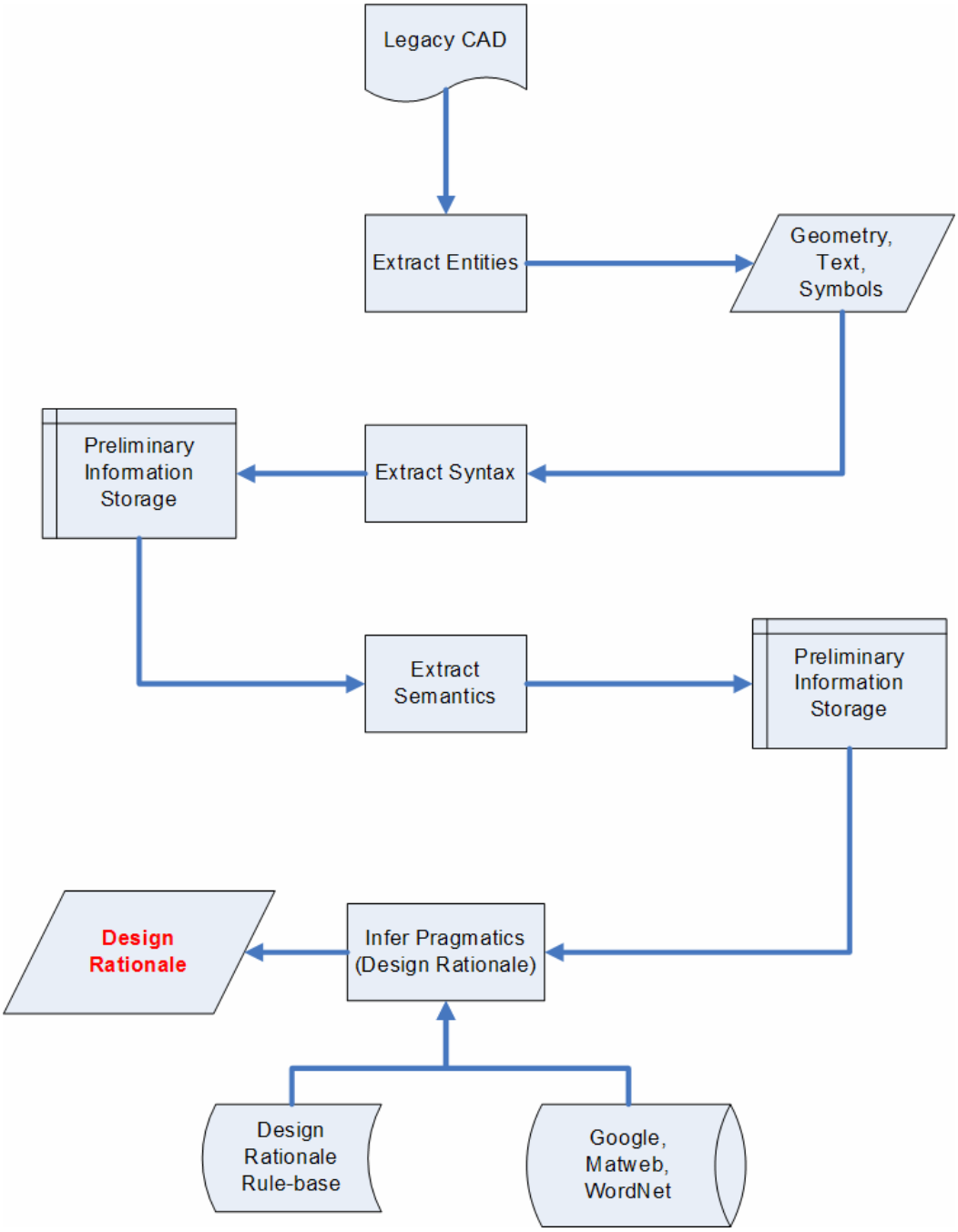

Figure 4.1: Overall Architecture

4.1.3 Extract Semantics

With the syntax available from the previous step, the third step is to extract semantics from the syntax. This requires the identification of numerous patterns and keywords contained in the syntax to extract the appropriate semantics class.

4.1.4 Infer Pragmatics

The penultimate step in the process is the inference of the pragmatics classes which provide the first level of design rationale in the form of values for design variables viz. function, flow, domain, objectives, constraints, application, relative cost etc.

4.1.5 Infer Design Rationale

The final step in the process is the combination of the pragmatics classes to infer the second level of design rationale as specified in section 3.3.2.2.

4.2 Detailed Architecture

With that brief overview of the architecture the following section provides the details on the methods, algorithms and dictionaries that were used to build the software system.

4.2.1 Entity parsing

As mentioned in section 1.2 this dissertation primarily deals with the DXF file format. To read the contents of the DXF file that contains the legacy MCAD the system uses a set of libraries developed by Imagecom Inc [26]. These libraries were developed using the Visual C++ language and are deployed as shared dynamic linked libraries (DLL). This allowed the current software system to call methods and use properties

implemented by Imagecom's Tools as black box calls in a manner that no implementation details were known or necessary to read the DXF file. The only method that was called to read the DXF file was the "Read_DXF" method. This method accepts as input the path to the DXF file that needs to be processed. The method outputs an array of GE2D_Entity objects that is contained in the DXF file. The GE2D_Entity is a custom class implemented by Imagecom Tools to store the geometry, text and symbols contained in the DXF file. GE2D_Entity stores the entities, their types, location and other properties and mirrors the entities in the DXF file. The advantage of using the GE2D_Entity class is that it is object oriented. Each entity sub-type viz. line, arc, circle, text, symbol is stored as exclusive objects allowing protected access to its properties.

Figure 4.2: Read DXF File and store array of GE2D_Entity objects

4.2.2 Extract Syntax

With all the raw data available from the previous step in the form of an array of GE2D_Entity objects, the next step is to extract the syntax from this raw data. DXF files provide little formal structure to the drawings that are generated as a part of design projects. The syntax that the drawings contain does not have pre-defined locations or other attributes thus making it difficult to extract. As briefly described in section 3.3.2.1 the syntax extraction can follow a method similar to that employed by human re-designers i.e. it can be done using situation specific templates. This is advantageous because with a change in the situation, which may change the syntax attributes, we can

specify a different, more relevant template to use. In this implementation phase a sample template was implemented as a test case. The template design was performed with a care to easy extensibility and derivability i.e. if needed the template can be extended for the current situation and also for new situations we can derive common elements from an existing template to create the new template for the new situation. The templates primarily describe the syntax and semantics classes and their attributes. The software system reads the templates and extracts the syntax from the DXF based on the template attributes.

To simplify the implementation of the templates it was designed in two specific layers. The first and topmost layer is a general layer that is specific to the company. The second layer is specific to each syntax class that is described in the company specific template.

4.2.2.1 First layer template

The first layer template consists of information that is specific for a single company (or even project). Figure 4.3 shows the “company.xml” template for a sample company.

```
<?xml version="1.0" encoding="utf-8" ?>
<company name="sample">
    <template_dir
    location="C:\users\gri\cad\dissertation\read_dxf\sample"></template_dir>
    <sheets location=""></sheets>
    <materials location=""></materials>
    <manufacturing location=""></manufacturing>
    <standards location=""></standards>
    <titleblock location=""></titleblock>
    <notes location=""></notes>
</company>
```

Figure 4.3: Sample Company template

The template is implemented in an XML format. The advantage of doing so is that it provides an open, easily editable format. If needed users can open the templates in a text editor and modify the contents. The root node is <company> that has one attribute “name” that specifies the name of the company that this template belongs to. The <template_dir> tag has one attributes “location” that points to the location of the second layer of templates viz. the sheets, materials, manufacturing, standards, titleblock and notes templates. Different companies or projects may use different values for the attributes based on their specific situations. The nodes, for the second layer of templates, have a single attribute “location”. If this attribute does not have any specific value then it is assumed that they are located in the same directory as that specified in the <template_dir> node.

The second template in this layer is the sheets.xml template shown in Figure 4.4. This template is drawing sheet specific but could also vary from company (or project) to company.

The sheets.xml template describes the sizes of the standard drawing sheet sizes typically identified by alphabets viz. C, D, E, K etc. The sheets.xml template describes the dimensions of the individual sheet itself along with the location and dimensions of the notes and title-block syntax classes. The location and dimensions of the shape classes can be inferred from the location and dimensions of the notes and title-block classes and hence has been ignored in the template. If deemed necessary in the future the sheets.xml template can be extended to include that information.

```
<?xml version="1.0" encoding="utf-8" ?>
<sheets>
        <sheet size="c" llx="0.0" lly="0.0" urx="44" ury="34" titleblockllx="14.95"
titleblocklly="1.5" titleblockurx="43" titleblockury="7.5" notesllx="1.25" noteslly="7.5"
notesurx="12.5" notesury="32.45"></sheet>
        <sheet size="e" llx="0.0" lly="0.0" urx="44" ury="34" titleblockllx="29.5"
titleblocklly="1" titleblockurx="43.5" titleblockury="4" notesllx="0.6" noteslly="4"
notesurx="10" notesury="32.5"></sheet>
        <sheet size="k" llx="0.0" lly="0.0" urx="77" ury="40" titleblockllx="62.45"
titleblocklly="0.5" titleblockurx="76.5" titleblockury="3.5" notesllx="0.6" noteslly="3.5"
notesurx="10" notesury="39.25"></sheet>
        <sheet size="k" llx="0.0" lly="0.0" urx="66" ury="40" titleblockllx="51.45"
titleblocklly="0.5" titleblockurx="66.5" titleblockury="3.5" notesllx="0.6" noteslly="3.5"
notesurx="10" notesury="39.25"></sheet>
        <sheet size="d" llx="0.0" lly="0.0" urx="17" ury="11" titleblockllx="9.45"
titleblocklly="0.25" titleblockurx="16.5" titleblockury="1.75" notesllx="0.6"
noteslly="1.75" notesurx="4.5" notesury="10.70"></sheet>
        <sheet size="d" llx="0.0" lly="0.0" urx="34" ury="22" titleblockllx="18.5"
titleblocklly="0.5" titleblockurx="33" titleblockury="3.5" notesllx="1.1" noteslly="3.5"
notesurx="9" notesury="21"></sheet>
</sheets>
```

Figure 4.4: Sample Sheets template

The sheets.xml template is useful to the software system as it provides the system with data necessary to locate and extract the various syntax classes. To use the sheets template for other situations it can be extended to describe the location and dimensions of the syntax classes.

4.2.2.2 Second layer template

The second layer templates are similarly stored in XML format. There are four second layer templates that describe the syntax classes viz. notes, title-block, shape, symbols.

Figure 4.5 shows the “notes.xml” template. The Notes class is extracted by the software system using the “sheets.xml” template that specifies the location and dimensions of the notes. But having extract the Notes class, the system still needs to recognize any patterns that exist to process the Notes. This is where the “notes.xml” template helps. The notes template has four child nodes. The <layer> node specifies on which drawing sheet layer the Notes are located. If this is a blank value then the system assumes the default drawing sheet layer as location of Notes. It is common to provide a title for the Notes section on the drawing file. If one is provide then the value for that title is specified in the <title> node of the “notes.xml” template. If each individual Note is numbered then their presence and the style of number of each Note and each sub-Note is indicated using attributes “value” and “style” of the <numbering> node. The first value of the “style” attribute indicates the numbering style of each Note while the second value indicates the numbering style of each sub-Note. It is also common to find Notes that are broken into multiple sentences. The <lines> node describes whether the Notes are “single” or “multi” sentence and if they are multi- sentence the “ydist” attribute specifies the distance between two multi- sentence Note.

```
<notes>
  <layer value="40"></layer>
  <title value="notes"></title>
  <numbering value="yes" style="numeric,
  alpha"></numbering>
  <lines value="multi" ydist="0.27"></lines>
</notes>
```

Figure 4.5: Sample Notes template

Next is the title-block.xml template. Figure 4.6 shows the title-block.xml template. The title-block on a drawing sheet is typically laid out in a grid format with each cell of the grid containing parameters with the respective values. The title-block template describes the contents of the title-block by providing an identifier that qualifies the content of each cell. For example a standard drawing sheet title-block may contain the name of the company to which the drawing belongs. In order to name the company the title-block would contain either just the name or may contain an identifier such as "Name of Company" with a value in front of that identifier that names the company.

Other possible parameters may be the name of the part that is contained in the drawings, a part number, designer information e.g. name, group etc, design information e.g. date etc. The title-block.xml file contains numerous <cell> nodes that identify the various cells of the grid along with the parameter that is located at that particular cell number. The numbering of the cell is dynamic and can be changed based on various criteria. Currently the numbering is done based on the relative horizontal and vertical locations of the cell in the grid. For a particular situation e.g. drawings belonging to the same company, the title-block may be pre-defined and the cell numbers may remain constant. When the situation changes i.e. a different company or project, a different title-block template may be needed.

The third template in this layer is the shape template. This dissertation does not delve into extracting the shape of the artifact contained in the legacy MCAD.

```
<title_block>
    <cell nos="1" content="company_info"
    value=""></cell>
    <cell nos="2" content="part_name" value=""></cell>
    <cell nos="3" content="part_number"
    value=""></cell>
    <cell nos="4" content="cage_code" value=""></cell>
    <cell nos="5" content="sheet_size" value=""></cell>
    <cell nos="6" content="sheet_nos_of_nos"
    value=""></cell>
    <cell nos="7" content="unit_wt" value=""></cell>
    <cell nos="8" content="scale" value=""></cell>
    <cell nos="9" content="contractor_info"
    value=""></cell>
    <cell nos="10" content="date" value=""></cell>
    <cell nos="11" content="drawn_by" value=""></cell>
    <cell nos="12" content="engineer" value=""></cell>
    <cell nos="13" content="checked_by"
    value=""></cell>
    <cell nos="14" content="dwg_approval"
    value=""></cell>
    <cell nos="15" content="design_approval"
    value=""></cell>
    <cell nos="16" content="tolerance" value=""></cell>
    <cell nos="17" content="unknown" value=""></cell>
    <cell nos="18" content="matl_engineer_date"
    value=""></cell>
    <cell nos="19" content="pmic" value=""></cell>
    <cell nos="20" content="used_on" used_on_nos="1"
    value=""></cell>
    <cell nos="21" content="next_assy"
    next_assy_nos="1" value=""></cell>
    <cell nos="34" content="used_on_title"
    value=""></cell>
    <cell nos="35" content="next_assy_title"
    value=""></cell>
    <cell nos="36" content="application" value=""></cell>
</title_block>
```

Figure 4.6: Sample title-block template

This task is passed to Imagecom's FlexiDesign technology and is not addressed in this current dissertation. The shape template is primarily geared toward identifying the information required by FlexiDesign such as the location of the geometry describing the shape. At this time the shape template is inferred from the sheets.xml template based on the location of the Notes and Title-block. Since overlapping elements are not considered good design the area of the drawing sheet not occupied by the Notes and Title-block is considered to belong to the Shape.

The last template in this layer is the symbols template. Currently this template does not exist in an open format. The system uses Imagecom's toolset to identify the symbols that are contained on the drawing file. Current drawing sample files contain exploded types of symbols that are composed of simpler geometric and textual entities. Numerous algorithms have been developed to extract these related geometric and textual entities to group them to identify the symbol based on certain commonly known patterns. For example to identify the surface finish symbol the system tries to identify two angular lines that intersect each other at one endpoint with an appropriate angle between them (the angle between them is a pattern value depending on different companies or projects or designers) with one line shorter than the other. Other shapes and variation of the surface finish symbol can also be extracted based on the standards of surface finish representation. A sample illustration of this symbol is shown in Figure 4.7.

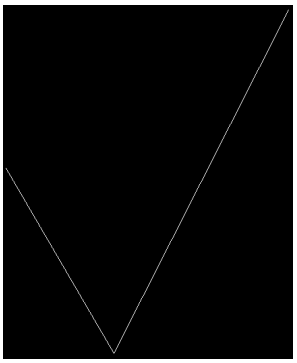

Figure 4.7: Surface Finish symbol

Another example of symbols that are extracted by the system is the geometric dimensions and tolerance (or commonly known as GDT). These are typically represented as shown in Figure 4.8. The GDT is also extracted using Imagecom's toolset and libraries with the extraction algorithms designed and implemented in the system currently being described. The GDT is primarily used for inspection after the artifact has been manufactured to confirm the quality of the part.

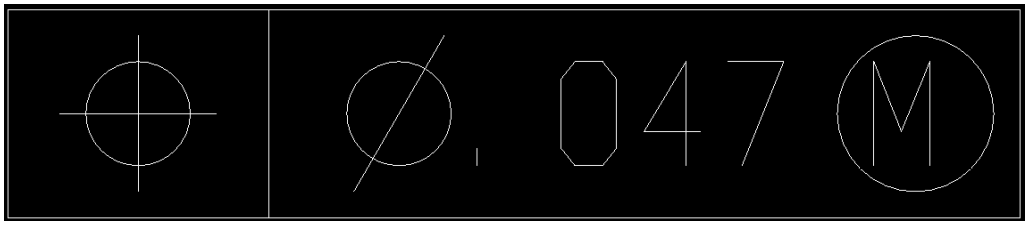

Figure 4.8: Geometric Dimension and Tolerance symbol

4.2.3 Extract Semantics

After the system has extracted the syntax from the legacy MCAD file the next step is to extract semantics from the syntax classes based on the relationships between the syntax and semantics classes as shown in section 3.2.2. To find the semantics classes from the syntax the system uses numerous keywords files and pattern matching

algorithms. These keywords store the vocabularies used by re-designers that are used in various specific situations viz. specific domain, sub-domain or company. The current implementation of the keywords files uses flat text files with keywords separated by a new line characters. The patterns that the system uses is stored in terms of the identifiers stored in the template files described in sections 4.2.2.1 and 4.2.2.2.

4.2.3.1 Keywords

Based on the type of information that needs to be accessed, the corresponding keywords are stored with the rules database. Various portions of the rules are used as keywords e.g. the rule name or symbol are commonly used as keywords to match and select the appropriate rules. This allows a common location for the keywords that are identified throughout the data set and additionally provides a common store to relate the keywords with the information inferred about the keyword.

To understand how the system uses the keywords to extract the semantics classes consider the materials keywords. The Materials semantics class depends on the Notes syntax class as shown in section 3.2.2. Once the system has extracted the Notes syntax class along with the text that is contained within the Notes, the next step is to run a comprehensive search for the material keywords within that text. All search results are stored and marked as possible materials that were stated for the particular drawing. Once a material keyword is found the system also reads the text that surround this material to identify more specific detail about the material. A similar process is performed for the manufacturing and standards keywords.

4.2.3.2 Common Patterns

The templates described in sections 4.2.2.1 and 4.2.2.2 are used by the system to not only extract syntax but to also extract semantics.

The title-block template for example stores the identifiers that help the system identify specific semantics classes. The complete listing of the semantics classes and the syntax classes from which they are extracted is recorded in section 3.2.2. The following is a description of how the system uses the title-block template to identify the following semantics classes – Standards, UsedOn, Part, Assembly, PartName, PartsList and Milieu. The title-block on a drawing is normally used as the placeholder for this kind of information. But unlike a human re-designer the system would need additional information to identify these classes merely from the text that is contained in the title-block cells. To aid the system the title-block template stores identifiers that provide meaning for the content of the title-block cell. For example, cell number 1 from the title-block template contains the company information; cell number 2 contains the part name etc. Once the system has extracted the TitleBlock syntax class by using these identifiers from the title-block template the system extracts the semantics.

In a similar manner the system also extracts the Inspection class by identifying meaning that surround the Symbols class. The system searches for texts that accompany the symbols that were extracted as part of the syntax. This is done based on a neighborhood algorithm depending on the orientation of the symbol. Currently the system can extract the texts that are oriented in the four regular, orthogonal quadrants and are present in a specific location near the symbol. For example consider the surface

finish symbol. The simplest form of the symbol is represented as shown in Figure 4.7 but may also have a text value contained near the shorter leg. Figure 4.9 and Figure 4.10 show two sample orientations of the surface finish symbols in the 1st and 3rd quadrant that the system can extract along with the surface finish value.

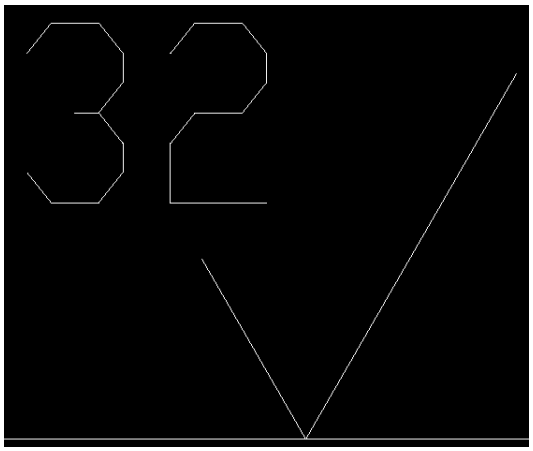

Figure 4.9: Surface Finish symbol in the 1st quadrant

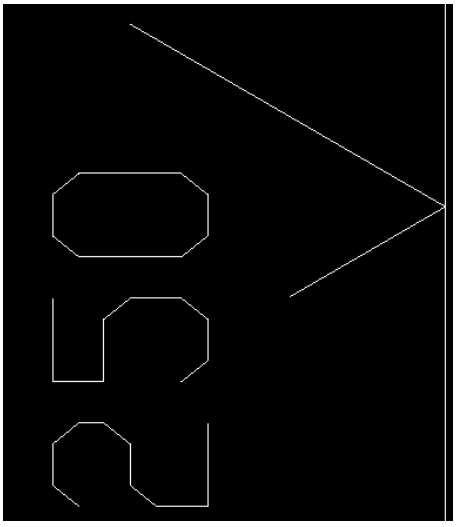

Figure 4.10: Surface Finish symbol in the 3rd quadrant

4.2.4 Infer Pragmatics

Having extracted the semantics classes the penultimate step that the system takes is to infer the pragmatics. The system employs numerous heuristic and retrieval methods to infer the pragmatics classes. These methods are described in the following sub-sections. The system uses heuristic algorithms to infer the Function, Flow, Application and Constraints classes and it uses retrieval methods to find the Domain, RelativeCost, QualityInspection, DesignEnvironment, Objectives, Constraints, SpecificProperties and Application classes. In the following sub-sections the method used to infer the specific pragmatics class is detailed.

4.2.4.1 Infer Domain

The system infers the domain to which the current drawing belongs to. It was documented in the design rationale capture analysis process that human re-designers use their existing knowledge and expertise to make such determinations. For the software system to make this determination in a similar manner requires access to similar knowledge. At the time of current implementation a data file, shown in Figure 4.11, is developed that allows the system to infer the Domain based on the PartName. The data file is implemented as a flat XML file. By using the definitions for the PartName from resources such as WordNet [92] and even common search engines such as Google [91], the system identifies the domain keywords within the search results. This overall methodology is shown in Figure 4.12. Using these identified keywords, the system then calculates probabilities for each domain and selects the domain with higher probability for further processing as the domain of the Part contained in the legacy CAD file.

Currently this data file is built manually but methods exist to automate building such a data file in an automated manner. This requires access to a dictionary such as the Dictionary of Mechanical Engineering by G. H. F. Nayler in digital form. The system could run heuristic searches on such a digital dictionary to determine the specific PartNames that belong in the Mechanical Engineering domain and use that to build the data file. Similar techniques with the relevant dictionaries for other domains could be used to build data files for all domains. This suggested automated method is out of scope of this dissertation due to lack of data needed.

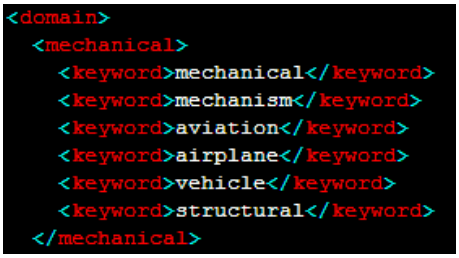

Figure 4.11: Sample from the Domain keywords data file

4.2.4.2 Infer Function

As detailed in section 3.2.3 the function can be inferred from the PartName and PartsList class. The system accomplishes this by applying heuristics on definitions retrieved from a lexical reference system. The lexical reference systems that are currently being used is the WordNet [92] database developed by Cognitive Science Laboratory at the Princeton University and the Google [91] search engine. The system accesses the WordNet database through COM (Component Object Model) interfaces, while access to the Google search engine is through simple internet access using an idealized web client. The methodology used by the system to infer the function of the

artifact contained in the drawing file is shown in Figure 4.13. It should be noted that the "Domain Function Taxonomy" data set that is shown in the Figure 4.13 is filtered to only include those functions that are relevant to the Domain that was retrieved in step 4.2.4.1.

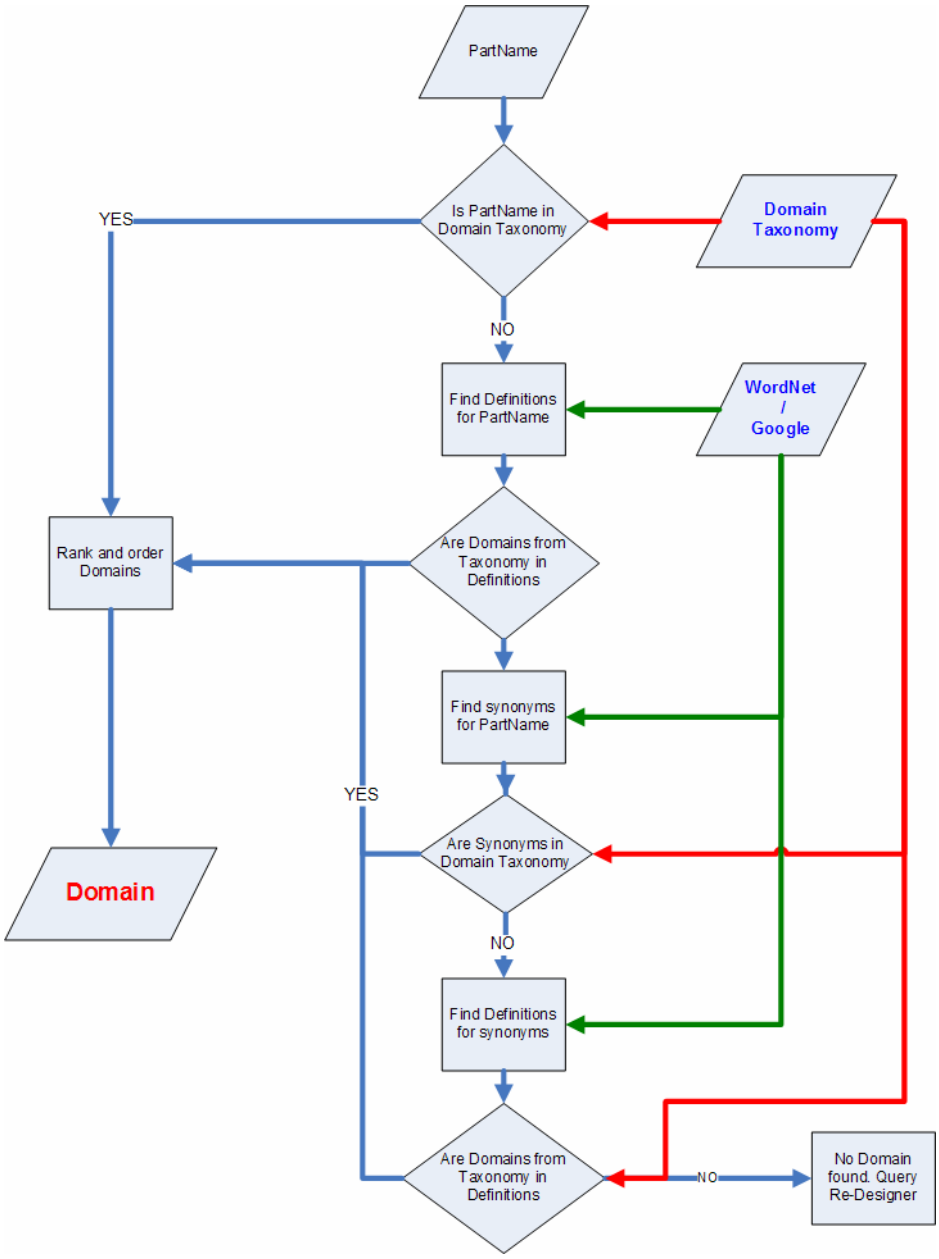

Figure 4.12: Infer Domain

4.2.4.3 Infer Flow

Similar to inferring function the system infers Flow from the PartName. The only difference for inferring Flow is that the system depends more on synonyms defined in the Domain Flow Taxonomy. This is because of the limited occurrence of the flows from the taxonomy in the PartName definition. This inference module is shown in Figure 4.14.

4.2.4.4 Retrieve SpecificProperties

By using the Materials class the system can retrieve SpecificProperties for the specific material. To retrieve the SpecificProperties the system uses a materials handbook that is available in digital format [93]. The system uses a wrapper around the data provided over the internet by [93] to extract the required properties. For example if the Material extract is Steel FS1025 then from [93] the system retrieves the web page that provides the material properties for FS1025. The implemented wrapper then extracts the SpecificProperties from this retrieved web page viz. Low Carbon Steel, Density, Modulus of Elasticity, Tensile Strength both Yield and Ultimate and Brinell Hardness.

4.2.4.5 Infer Objectives and Constraints

The system infers Objectives from the Materials class, the Company Name and the Manufacturing instructions. By using information from a Materials handbook provided in print format [94] copied over to a digital representation for easy access the system infers the Objective for the extracted Materials class. Inferences from the Company name are available from the Design Rationale Analysis process that was

detailed earlier, whereas inferences from Manufacturing instructions were gleaned from previously stated sources [88, 87, 86, 94].

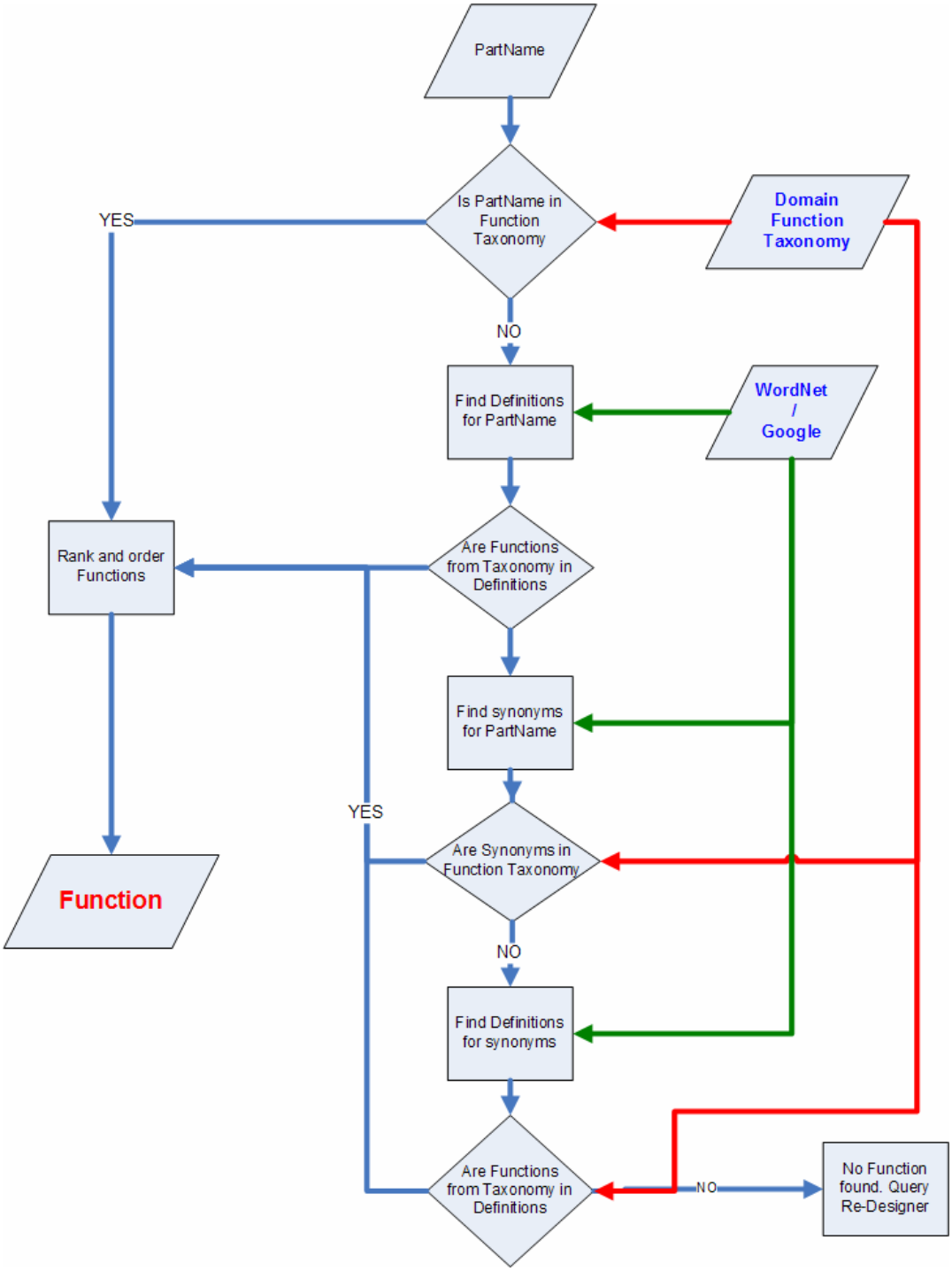

Figure 4.13: Infer Function

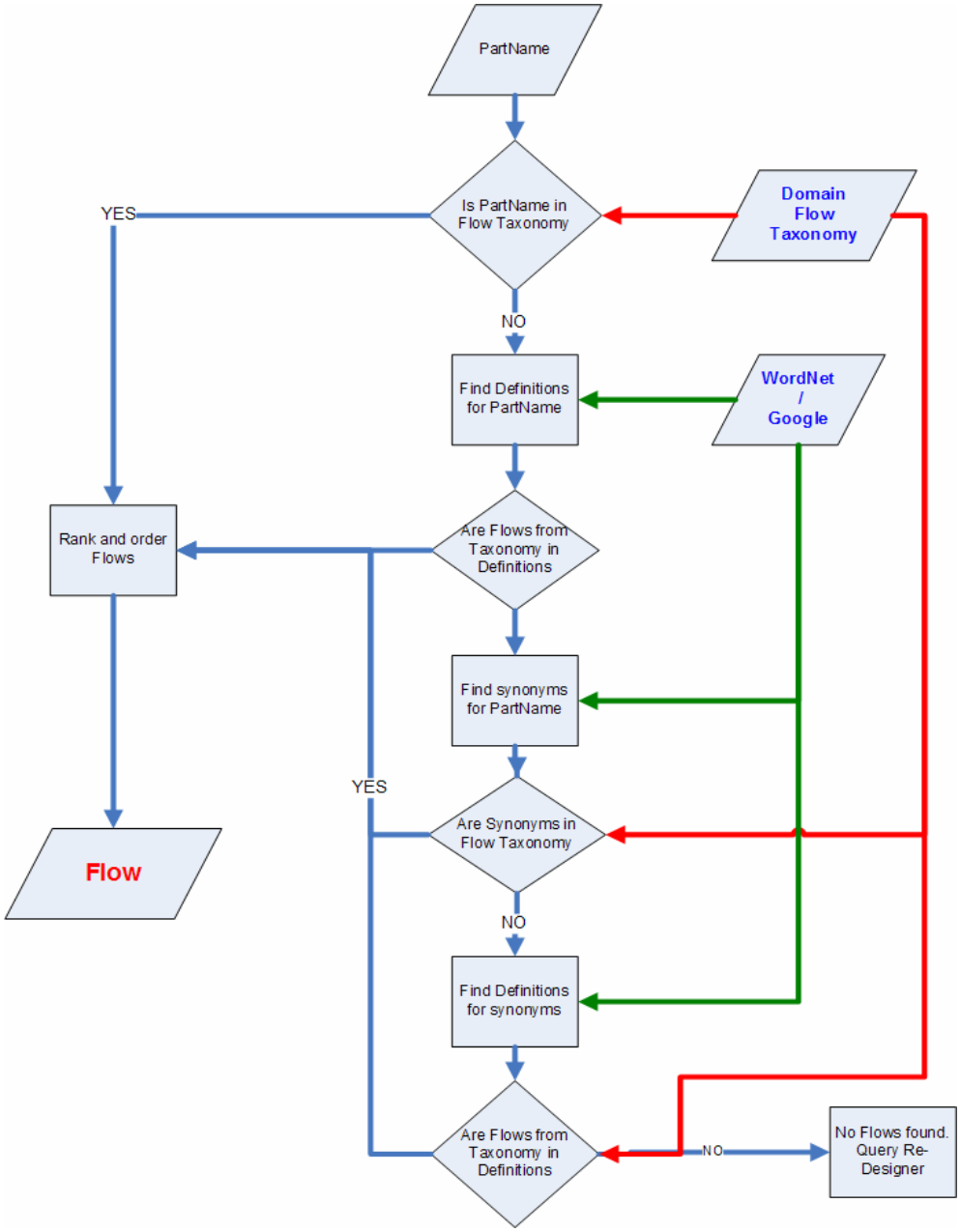

Figure 4.14: Infer Flow

The inferred Objectives typically describe key material properties such as weldability, machinability, formability, strength and hardness which are the selection

criteria for selecting the specific Material. To deal with the print format of data a simplified version is created in a template format illustrated in Figure 4.15. From the Materials class the system retrieves the SpecificProperties from section 4.2.4.4 and using the information from the SpecificProperties the system infers the Objectives from the template file. This inference module is shown in Figure 4.16.

```
<rules>
  <rule>
    <name>carbon</name>
    <symbol>low</symbol>
    <objective>
      <property>weldability</property>
      <value>good</value>
    </objective>
    <objective>
      <property>machinability</property>
      <value>good</value>
    </objective>
    <objective>
      <property>formability</property>
      <value>good</value>
    </objective>
    <constraint>
      <property>strength</property>
      <value>low</value>
    </constraint>
    <constraint>
      <property>hardness</property>
      <value>low</value>
    </constraint>
  </rule>
  <rule>
    <name>carbon</name>
    <symbol>medium</symbol>
    <objective>
      <property>weldability</property>
      <value>better than low carbon steel</value>
    </objective>
    <objective>
      <property>machinability</property>
      <value>better than low carbon steel</value>
    </objective>
    <objective>
      <property>formability</property>
      <value>better for hot than cold</value>
    </objective>
```

Figure 4.15: Materials Objectives and Constraints Template

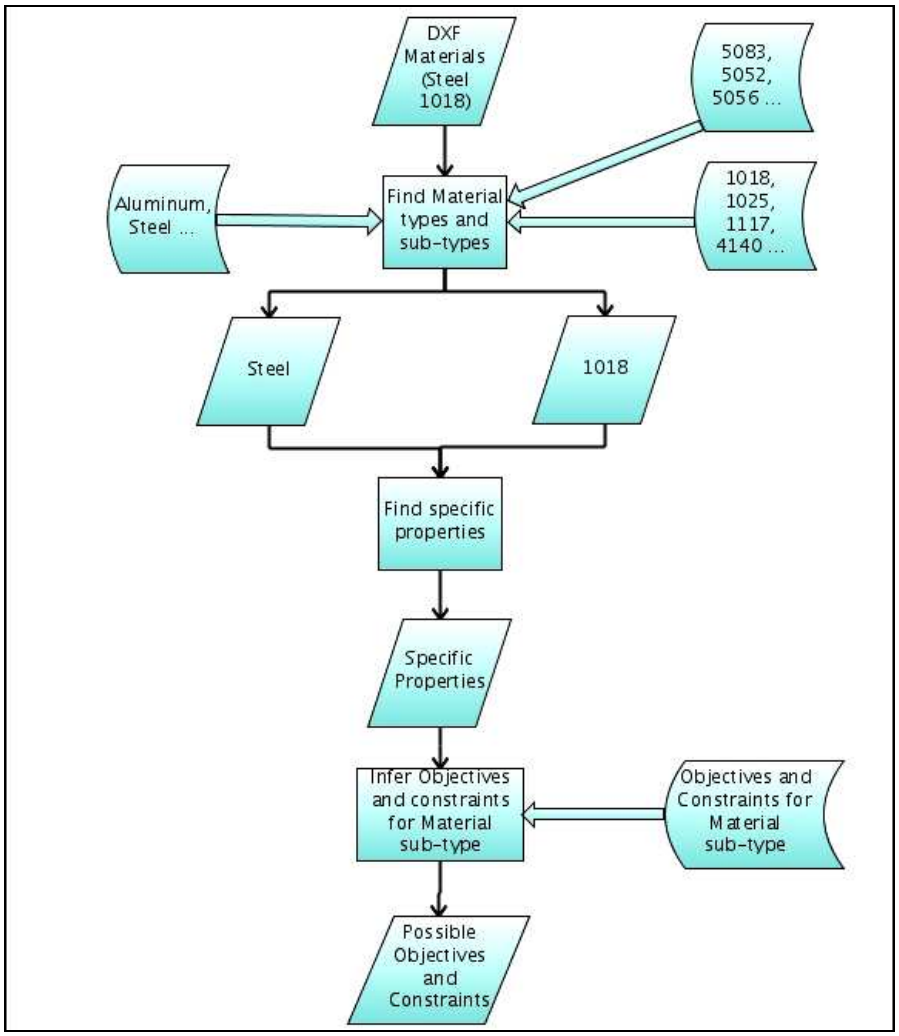

Figure 4.16: Infer Objectives and Constraints from Materials (example)

4.3 System Implementation

The system was implemented using the Visual C++ programming language. The extraction, inference and retrieval modules were written as components, primarily libraries. Additionally in an order to take advantage of the digital dictionaries and online materials handbook the Visual C# language was used. Visual C# allows easy access to COM objects and is portable across operating systems though the current system was developed, deployed and tested only on the Windows Operating System.

To ensure that the data generated and used by the system is re-usable in other systems and projects most of the data was saved as text files or XML templates. The design of this data store was done using the Visual Studio Integrated Development Environment (IDE) 2005. Since the data store are open formats any text editor can be used to modify these files.

Additionally the extraction process depended on the libraries and toolset provided by Imagecom. Imagecom also provided the functionality to read the DXF files. Imagecom's toolset are implemented as libraries and the current system calls the methods and uses the properties by dynamically linking to the libraries.

4.4 Conclusion

This chapter has provided the algorithms required to extract the context and infer the design rationale from the entities on the drawing file, specifically the DXF file. The system is implemented as libraries allowing multiple client access. Currently the design rationale is stored as objects but output as text so the results need to be collected for the validation that is detailed in chapter 5.

CHAPTER 5

RESULTS AND VALIDATION

Having provided in detail the approach and the software architecture of the system developed to automate the capture process, this chapter provides the output of the system and also the output recorded during the design rationale capture analysis process described in section 3.1.1. The two sets of outputs are used to validate the stated approach and also the effectiveness of the system. Also described is the rationale behind choosing this proposed validation method as opposed to other possible validation techniques.

5.1 Selecting a Validation Method

Two relevant validation methods were suggested during the course of the development of the software system. Of these one method was finalized upon based on the advantages it offered. The following sub-sections details these two methods, their respective advantages and disadvantages and finally selects one of these methods.

5.1.1 Validation Method 1

The first validation method suggested requires a large sample set of more than a 100 drawing files. This sample set needs to be analyzed by human re-designers and the software system. The results of both analyses are collected using the output format detailed later. These results are then compared for quality and accuracy to validate the

effectiveness of the proposed approach and developed software system. The following are the advantages and disadvantages of using this validation method.

5.1.1.1 Advantages

The large sample set will prove that the approach and developed software system do indeed work as proposed. By ensuring random selection process of the drawings it can be shown that the software system is capable of handling parts of varying complexity and is as effective as a human re-designer in capturing design rationale.

5.1.1.2 Disadvantages

Since the system needs to handle a large sample set of drawing files with varying complexity, it needs to be an extensive system with numerous templates and keywords and large amounts of inference information. Such monolithic systems are difficult to develop as part of an individual's dissertation as this normally requires access to large amounts of resources. Additionally such an approach would be a hit or miss one, where the results of the human re-designers are used to merely validate the software system. The system does not have access to design rationale captured by the human re-designers and analyzing larger number of parts will not serve to improve the effectiveness of the system.

5.1.2 Validation Method 2

The second validation method suggested requires an initial categorization of the drawing files based on their complexity. The development of software system should take this categorization into account. The proposed system should be developed by

starting with the lowest level of complexity and become progressively able to handle more complex drawings. This requires that the system be extensible.

5.1.2.1 Advantages

The focus is not analyzing a large sample set of drawing files but instead on the quality and effectiveness of the design rationale captured by the software system. This provides an easier method to determine success of the system, as by developing a system that can address a majority of the drawings at the lowest level of complexity we can ensure that the output of the system can rival that of a human re-designer. There is no need to develop a monolith system as by developing an extensible system we can ensure that the system can progressively handle higher levels of complexity. Additionally by developing a system that can take advantage of the design rationale captured by the re-designers the quality of the system will continue to grow.

5.1.2.2 Disadvantages

The disadvantage of selecting the second validation method lies in the development of an extensible system. To take advantage of the design rationale captured by the re-designers the system must be able to take advantage of both the increasing database that the re-designer has access in terms of experience and knowledge as well as the inference techniques that the re-designer employs.

5.1.3 Selecting a Validation method

This dissertation suggest selecting the second validation method over the first and effectively addressing the issues related to the second validation method as follows. By using open database formats and an open, modular architecture extensibility can be

added to the system. The open formats allow the number of templates and keywords that the system recognizes to be easily increased thereby increasing the amount and quality of design rationale that it captures. Additionally the second validation method necessitates clearly defining the various complexity levels of the drawing files. In the following sub-sections these issues are addressed.

5.1.3.1 Definition of drawing file's complexity

Four levels of complexity are suggested as follows:

- Simple: The Simple drawing files contain standard or at least consistent sheet sizes. The entities are all of known types with known attributes and are well laid out on the sheet. The drawing file uses layers appropriately by cleanly separating the various syntax groups on different layers. There are no keywords on the simple drawing file that the system is unaware of. Finally the file is less busy with as few entities as needed.
- Medium: The Medium drawing files contain more entities than the Simple and are busier. There are keywords that the system does not recognize. Multiple views are added for the shape of the part and may contain additional inferences. Bill of Materials or Part Tables may be present that provides rationale regarding Assembly or Application. Other attributes remain the same as the Simple drawing files.
- Medium-complex: The Medium-complex drawing files contain numerous entities and the entities are laid out more poorly than the Medium level. This would necessitate additional capture methods viz.

semi-automatic or interactive methods to extract the context. Other attributes remain the same as the Medium drawing files.

- Complex: The complex drawing files are those that contain very little context. They have no standard sheet sizes, few layers to separate the different entity types and may contain little or no keywords. The files primarily contain geometry which provides little design rationale.

5.2 Output Format

This section details the output format chosen to display the design rationale. The format was chosen for the ease of use by the re-designers. It also simplifies the process of comparison required for validation. To fill out the form the re-designer needs to simplify answer the question based on the instructions provided for each individual question. If multiple answers are required the re-designer merely needs to repeat the answer format as many times as necessary. To answer the questions the re-designers were provided with a sample set of drawings and this questionnaire to answer. This questionnaire can be filled out with an editor such as Microsoft Office Word or OpenOffice Writer.

Table 5.1: Design Rationale Capture Analysis Output Format

1. What function, flow and domain can you ascertain for the part that is contained in the drawing? These can be identified from the name of the part, if appropriately named. i. State part name =

Table 5.1 - continued

ii. Infer function =

iii. Infer flow =

iv. Infer domain =

2. Can you determine if the represented part is a part of an assembly? If yes, identify the assembly constraints. These would be typically identified from the geometric dimensions and tolerance that are present on the drawing. In addition state if possible the type of assembly constraint that is represented.

    a. State dimension with tolerance =

        i. Infer constraint =

3. Identify the specified material for the part. This can be found in the notes section of the drawing. Infer the reason this material was chosen viz. the Objectives and Constraints for selecting the specified material.

    a. State material =

        i. Infer objective =

        ii. Infer constraint =

4. Retrieve specific properties for the material

    a. Modulus of Elasticity =

    b. Tensile Strength, Yield =

    c. Tensile Strength, Ultimate =

    d. Hardness, Brinell =

Table 5.1 - continued

e. Relative Cost =

f. Machinability (Based on AISI 1212 steel. as 100% Machinability) =

g. Density =

5. Identify the company name that is stated on the drawing. From the stated company name infer the design environment (viz. commercial, governmental, US Army etc)

   a. State company name =

      i. Infer design environment =

      ii. Infer possible goals (e.g.)

         1. Reliability =

         2. Shelf life =

         3. Cost priority =

6. Identify all manufacturing instructions stated on the drawing. Infer possible objectives and constraints for the manufacturing instructions (e.g. quench and temper to increase surface hardness)

   a. Manufacturing =

      i. Objective =

      ii. Constraint =

5.3 Design Rationale captured by re-designers

With that output format available this section provides the design rationale captured by the re-designers. Only a sample set of results are presented here. The initial capture analysis process was done with two candidates with six drawings. The second analysis process was done with two candidates with three drawings. The results that are presented here are from the second analysis process. It should be noted that the re-designers who were interviewed for the second analysis may be considered inexperienced to intermediately experienced re-designers. Candidate 1 holds a Doctoral degree and has a little more than a year experience as a junior designer at Siemens. Candidate 2 holds a Masters' degree with more than 3 years experience.

5.3.1 Sample drawing 1

Table 5.2: Design Rationale captured from sample drawing 1 by re-designer

1. What function, flow and domain can you ascertain for the part that is contained in the drawing? These can be identified from the name of the part, if appropriately named.
    b. State part name = Yoke
        i. Infer function = Connect
        ii. Infer flow = Force
        iii. Infer domain = Mechanical
        iv. Can you determine if the represented part is a part of an assembly? If yes, identify the assembly constraints.

Table 5.2 - continued

These would be typically identified from the geometric dimensions and tolerance that are present on the drawing. In addition state if possible the type of assembly constraint that is represented.

c. State dimension with tolerance = -0.005, +0.010

i. Infer constraint = Assembly mate

2. Identify the specified material for the part. This can be found in the notes section of the drawing. Infer the reason this material was chosen viz. the Objectives and Constraints for selecting the specified material.

d. State material = Forging Steel, FS 1018

i. Infer objective = Good Machinability

ii. Infer objective = High tensile strength

iii. Infer constraint =

3. Retrieve specific properties for the material

e. Modulus of Elasticity = 29700 psi

f. Tensile Strength, Yield = 39900 psi

g. Tensile Strength, Ultimate = 39900 psi

h. Hardness, Brinell = 126

i. Relative Cost = 0.8

j. Machinability (Based on AISI 1212 steel. as 100% Machinability) = 52%

Table 5.2 - continued

k. Density = 0.284 lb/in³

4. Identify the company name that is stated on the drawing. From the stated company name infer the design environment (viz. commercial, governmental, US Army etc)

    l. State company name = US Army Tank-Automotive and Armaments Command

        i. Infer design environment = US Army

        ii. Infer possible goals (e.g.)

            1. Reliability = High

            2. Shelf life = Long

            3. Cost priority = Low

5. Identify all manufacturing instructions stated on the drawing. Infer possible objectives and constraints for the manufacturing instructions (e.g. quench and temper to increase surface hardness)

    m. Manufacturing =

        i. Objective =

            Constraint =

5.3.2 Sample drawing 2

Table 5.3: Design Rationale captured from sample drawing 2 by re-designer

1. What function, flow and domain can you ascertain for the part that is contained in the drawing? These can be identified from the name of the part, if appropriately named.
    a. State part name = Arm, Support
        i. Infer function = Support
        ii. Infer flow = Force
        iii. Infer domain = Mechanical
2. Can you determine if the represented part is a part of an assembly? If yes, identify the assembly constraints. These would be typically identified from the geometric dimensions and tolerance that are present on the drawing. In addition state if possible the type of assembly constraint that is represented.
    a. State dimension with tolerance = Depth +/- 1/64
        i. Infer constraint = Assembly mate
3. Identify the specified material for the part. This can be found in the notes section of the drawing. Infer the reason this material was chosen viz. the Objectives and Constraints for selecting the specified material.
    a. State material = Forged Steel 4145H
        i. Infer objective = Good Machinability

Table 5.3 - continued

ii. Infer objective = High tensile strength

iii. Infer objective = Rockwell hardness C32

iv. Infer constraint =

4. Retrieve specific properties for the material

   a. Modulus of Elasticity = 29700 psi

   b. Tensile Strength, Yield = 39900 psi

   b. Tensile Strength, Ultimate = 39900 psi

   c. Hardness, Brinell = 126

   d. Relative Cost = 0.8

   e. Machinability (Based on AISI 1212 steel. as 100% Machinability) = 52%

   f. Density = 0.284 lb/in^3

5. Identify the company name that is stated on the drawing. From the stated company name infer the design environment (viz. commercial, governmental, US Army etc)

   a. State company name = US Army Tank-Automotive and Armaments Command

      i. Infer design environment = US Army

      ii. Infer possible goals (e.g.)

         1. Reliability = High

         2. Shelf life = Long

Table 5.3 - continued

3. Cost priority = Low 6. Identify all manufacturing instructions stated on the drawing. Infer possible objectives and constraints for the manufacturing instructions (e.g. quench and temper to increase surface hardness) a. Manufacturing = Quench and temper v. Objective = Increase hardness

5.3.3 Sample drawing 3

Table 5.4: Design Rationale captured from sample drawing 3 by re-designer

1. What function, flow and domain can you ascertain for the part that is contained in the drawing? These can be identified from the name of the part, if appropriately named. a. State part name = Arm, Support, Suspension i. Infer function = Support ii. Infer flow = Force iii. Infer domain = Mechanical

Table 5.4 - continued

2. Can you determine if the represented part is a part of an assembly? If yes, identify the assembly constraints. These would be typically identified from the geometric dimensions and tolerance that are present on the drawing. In addition state if possible the type of assembly constraint that is represented.
    a. State dimension with tolerance = Depth +/- 1/32
        i. Infer constraint = Assembly mate
3. Identify the specified material for the part. This can be found in the notes section of the drawing. Infer the reason this material was chosen viz. the Objectives and Constraints for selecting the specified material.
    a. State material = Forged Steel 4145H
        i. Infer objective = Good machinability
        ii. Infer constraint =
    b. State material = Forged Steel 4337H
        i. Infer objective = Good machinability
        ii. Infer constraint =
4. Retrieve specific properties for the material
    a. 4145H
        i. Modulus of Elasticity = 29700 ksi
        ii. Tensile Strength, Yield =

Table 5.4 - continued

iii. Tensile Strength, Ultimate =

iv. Hardness, Brinell = 208

v. Relative Cost =

vi. Machinability (Based on AISI 1212 steel. as 100% Machinability) = 60%

vii. Density = 0.284 lb/in^3

b. 4337H

i. Modulus of Elasticity =

ii. Tensile Strength, Yield =

iii. Tensile Strength, Ultimate =

iv. Hardness, Brinell =

v. Relative Cost =

5. Machinability (Based on AISI 1212 steel. as 100% Machinability) =

vi. Density =

6. Identify the company name that is stated on the drawing. From the stated company name infer the design environment (viz. commercial, governmental, US Army etc)

a. State company name = US Army Tank-Automotive and Armaments Command

i. Infer design environment = US Army

Table 5.4 - continued

ii. Infer possible goals (e.g.)

1. Reliability = High
2. Shelf life = Long
3. Cost priority = Low

7. Identify all manufacturing instructions stated on the drawing. Infer possible objectives and constraints for the manufacturing instructions (e.g. quench and temper to increase surface hardness)

a. Manufacturing =

i. Objective =

viii. Constraint =

5.4 Design Rationale captured by software system

The following section presents the design rationale captured by the software system. The output shown here is formatted to match the results of the capture analysis process performed with the re-designers. This is to simplify the comparison process.

Table 5.5: Design Rationale captured from sample drawing 1 by software

1. What function, flow and domain can you ascertain for the part that is contained in the drawing? These can be identified from the name of the part, if appropriately named.

a. State part name = Yoke

Table 5.5 - continued

i. Infer function = Join, Support, Connect

ii. Infer flow = Motion

iii. Infer domain = Mechanical

2. Can you determine if the represented part is a part of an assembly? If yes, identify the assembly constraints. These would be typically identified from the geometric dimensions and tolerance that are present on the drawing. In addition state if possible the type of assembly constraint that is represented.

   a. State dimension with tolerance = Diameter -0.005

      i. Infer constraint = Assembly mate (Shaft)

3. Identify the specified material for the part. Infer the reason this material was chosen viz. the Objectives and Constraints for selecting the specified material.

   a. State material = FS 1018

      i. Infer objective = Good weldability, good machinability, good formability

      ii. Infer constraint = low strength, low hardness

Table 5.5 - continued

4. Retrieve specific properties for the material

   a. FS 1018

      i. Modulus of Elasticity = 29700 ksi

      ii. Tensile Strength, Yield = 39900 psi

      iii. Tensile Strength, Ultimate = 63800 psi

      iv. Hardness, Brinell = 126

      v. Relative Cost = 1.6

      vi. Machinability (Based on AISI 1212 steel. as 100% Machinability) = 52%

      vii. Density = 0.284 lb/in^3

   b. FS 1025

      i. Modulus of Elasticity = 29700 ksi

      ii. Tensile Strength, Yield = 39900 psi

      iii. Tensile Strength, Ultimate = 63800 psi

      iv. Hardness, Brinell = 126

      v. Relative Cost = 1.6

      vi. Machinability (Based on AISI 1212 steel. as 100% Machinability) = 52%

      vii. Density = 0.284 lb/in^3

Table 5.5 - continued

5. Identify the company name that is stated on the drawing. From the stated company name infer the design environment (viz. commercial, governmental, US Army etc)
    a. State company name = US Army Tank-Automotive and Armaments Command
        i. Infer design environment = US Army
        ii. Infer possible goals (e.g.)
            4. Reliability = High
            5. Shelf life = Long
            6. Cost priority = Low
6. Identify all manufacturing instructions stated on the drawing. Infer possible objectives and constraints for the manufacturing instructions (e.g. quench and temper to increase surface hardness)
    a. Manufacturing =
        i. Objective =
7. Constraint =

Table 5.6: Design Rationale captured from sample drawing 2 by software

1. What function, flow and domain can you ascertain for the part that is contained in the drawing? These can be identified from the name of the part, if appropriately named.

Table 5.6 - continued

a. State part name = Arm, Support

i. Infer function = Support, Cover

ii. Infer flow =

iii. Infer domain = Mechanical

2. Can you determine if the represented part is a part of an assembly? If yes, identify the assembly constraints. These would be typically identified from the geometric dimensions and tolerance that are present on the drawing. In addition state if possible the type of assembly constraint that is represented.

a. State dimension with tolerance = Diameter 0.173-0.203, 0.44-0.56

i. Infer constraint = Assembly mate

3. Identify the specified material for the part. This can be found in the notes section of the drawing. Infer the reason this material was chosen viz. the Objectives and Constraints for selecting the specified material.

a. State material = 4145H, 4147H, 86B45H

i. Infer objective = Weldability and Machinability better than low carbon steel, good for hot formed

Table 5.6 - continued

ii. Infer constraint = medium strength, and hardness

4. Retrieve specific properties for the material
   a. 4145H, 4147H, 86B45H
      i. Modulus of Elasticity = 29700 ksi
      ii. Tensile Strength, Yield =
      iii. Tensile Strength, Ultimate =
      iv. Hardness, Brinell = 208
      v. Relative Cost =
      vi. Machinability (Based on AISI 1212 steel. as 100% Machinability) = 60%
      vii. Density = 0.284 lb/in^3
5. Identify the company name that is stated on the drawing. From the stated company name infer the design environment (viz. commercial, governmental, US Army etc)
   a. State company name = US Army Tank-Automotive and Armaments Command
      i. Infer design environment = US Army
      ii. Infer possible goals (e.g.)
         7. Reliability = High
         8. Shelf life = Long

Table 5.6 - continued

9. Cost priority = Low

6. Identify all manufacturing instructions stated on the drawing. Infer possible objectives and constraints for the manufacturing instructions (e.g. quench and temper to increase surface hardness)
   a. Manufacturing =
      1. Objective =
      2. Constraint =
7. Identify the Application of the part. This can be retrieved from the UsedOn information that is stated on the drawings.
   a. State UsedOn variable = M113A1
      iii. Application = Armed personal carrier

Table 5.7: Design Rationale captured from sample drawing 3 by software

1. What function, flow and domain can you ascertain for the part that is contained in the drawing? These can be identified from the name of the part, if appropriately named.
   a. State part name = Arm, Support, Suspension
      i. Infer function = Support, Cover
      ii. Infer flow =
      iii. Infer domain = Mechanical

Table 5.7 - continued

2. Can you determine if the represented part is a part of an assembly? If yes, identify the assembly constraints. These would be typically identified from the geometric dimensions and tolerance that are present on the drawing. In addition state if possible the type of assembly constraint that is represented.
    a. State dimension with tolerance = +/- 1/64, +/- 1/32
        i. Infer constraint = Assembly mate
3. Identify the specified material for the part. This can be found in the notes section of the drawing. Infer the reason this material was chosen viz. the Objectives and Constraints for selecting the specified material.
    a. State material = 4145H, 4147H, 86B45H
        i. Infer objective = Weldability and Machinability better than low carbon steel, good for hot formed,
        ii. Infer constraint = medium strength, medium hardness
4. Retrieve specific properties for the material
    a. 4145H, 4147H, 86B45H
        i. Modulus of Elasticity = 29700 ksi

Table 5.7 - continued

ii. Tensile Strength, Yield =

iii. Tensile Strength, Ultimate =

iv. Hardness, Brinell = 208

v. Relative Cost =

vi. Machinability (Based on AISI 1212 steel. as 100% Machinability) = 60%

vii. Density = 0.284 lb/in^3

5. Identify the company name that is stated on the drawing. From the stated company name infer the design environment (viz. commercial, governmental, US Army etc)

   a. State company name = US Army Tank-Automotive and Armaments Command

      i. Infer design environment = US Army

      ii. Infer possible goals (e.g.)

         1. Reliability = High

         2. Shelf life = Long

         3. Cost priority = Low

6. Identify all manufacturing instructions stated on the drawing. Infer possible objectives and constraints for the manufacturing instructions (e.g. quench and temper to increase surface hardness)

   a. Manufacturing =

Table 5.7 - continued

1. Objective =
2. Constraint =

7. Identify the Application of the part. This can be retrieved from the UsedOn information that is stated on the drawings.
   a. State UsedOn variable =
   viii. Application =

5.5 Results of comparison

Comparing the design rationale captured by the software system with the re-designers' output it can be seen that the design rationale is of the same quality. It should be noted that the re-designers whose results are presented in this dissertation are primarily inexperienced to medium experienced re-designers. This implies that the system is mature enough to capture design rationale comparable to inexperienced to medium experienced re-designers.

REFERENCES

[1] M.A. Rosenman, and J.S. Gero, Purpose and function in a collaborative CAD environment, Reliability Engineering and System Safety, 1999(64), pp 164 – 179.

[2] J. J. Shah, M. Mäntylä, Parametric and Feature-based CAD/CAM: Concepts, Techniques and Applications, ISBN 0-471-00214-3, 1995.

[3] Szykman, et. al., The Role of Knowledge in Next-generation Product Development Systems, ASME Journal of Computing and Information Science in Engineering, 2001.

[4] Xiaochun Hu Jun Pang, Yan Pang, Michael Atwood, Wei Sun, William C. Regli, A Survey on design rationale: Representation, Capture and Retrieval, Proceedings of DETC'00: 2000 ASME Design Engineering Technical Conferences, September 10-13, 2000, Baltimore, Maryland.

[5] Shum, S., Hammond, N. (1993), Argumentation-Based Design Rationale: From Conceptual Roots to Current Use, Tech. Report EPC-1993-106, Rank Xerox Research Centre, Cambridge.

[6] Sim, S., Duffy, A. (1994), A New Perspective to Design Intent and design Rationale, in Artificial Intelligence in Design Workshop Notes for Representing and Using Design Rationale, 15-18 August, pp. 4-12.

[7] Fischer, G., Lemke, A., McCall, R., Morch, A. (1995), Making Argumentation Serve Design, in Design Rationale Concepts, Techniques, and Use, T. Moran and J. Carroll, eds., Lawrence Erlbaum Associates, pp. 267-294.

[8] Conklin, J. and Burgess-Yakamovic, K. (1995), A Process-Oriented Approach to Design Rationale, in Design Rationale Concepts, Techniques, and Use, T. Moran and J. Carroll, eds., Lawrence Erlbaum Associates, Mahwah, NJ, pp. 293-428.

[9] Lee, J. (1997), Design Rationale Systems: Understanding the Issues, IEEE Expert, Vol. 12, No. 3, pp. 78-85.

[10] Karen L. Myers, Nina B. Zumel, Pablo Garcia, Acquiring design rationale automatically, AIEDAM Vol. 14 Nos. 2, 1999, pp 115–135.

[11] Jeffrey M. Molavi, Raymond McCall, and Anthony Songer, A new approach to effective use of design rationale in practice, Journal of Architectural Engineering, 2003, pp 62 – 69.

[12] Hu, X., Pang, J., Pang, Y., Atwood, M., Sun, W. and Regli, W. C., "A Survey on Design Rationale: Representation, Capture and Retrieval," Proceedings of the 2000 ASME DETC, Paper No. DETC2000/DFM-14008.

[13] Lee, J., Design rationale systems: understanding the issues, Expert, IEEE, Volume 12, Issue 3, May-June 1997 Page(s):78 - 85

[14] Chen, A., McGinnis, B., Ullman, D., Dietterich, T. (1990), Design History Knowledge Representation and Its Basic Computer Implementation, The 2nd

International Conference on Design Theory and Methodology, ASME, Chicago, IL, pp. 175-185.

[15] W. Kunz and W. Rittel, Issues as elements of information systems, Center for Planning and Development Research, University of California, Berkeley, August 1970.

[16] R. J. McCall. PHI: A conceptual foundation for design hypermedia, Design Studies, 12(1):30–41, January 1991.

[17] A. MacLean, R. Young, V. Belloti, and T. Moran. Questions, options, and criteria: Elements of design space analysis, Human-Computer Interaction, 6(3-4):201–250, 1991.

[18] J. Lee and K. Lai. What's in design rationale. Human-Computer Interaction, 6(3-4):251–280, 1991.

[19] A. Goel. Model revision: A theory of incremental model learning. In 8th International Conference on Machine Learning, pages 605–609, Chicago, USA, June 1991. AAAI.

[20] A. Goel. A model-based approach to case adaptation. In 13th Annual Conference of the Cognitive Science Society, pages 143–148, Hillsdale, NJ, 1991. Cognitive Science Society, Lawrence Erlbaum.

[21] A. C. B. Garcia and C. S. de Souza. Add+: Including rhetorical structures in active documents. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 11(2):109–124, April 1997.

[22] R. Sudarsan, S.J. Fenves, R.D. Sriram and F. Wang, A product information modeling framework for product lifecycle management, Computer-Aided Design, Volume 37, Issue 13, November 2005, Pages 1399-1411.

[23] M. Weiss and D. Dori, A Scheme for 3D Object Reconstruction from Dimensioned Orthographic Views. IEEE, 1995: p. 335-338.

[24] D. Dori and L. Wenyin, Automated CAD Conversion with the Machine Drawing Understanding System: Concepts, Algorithms and Performance. IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans, 1999. 1(4): p. 411 - 416.

[25] M. Tanaka, A single solution method for converting 2D assembly drawings to 3D part drawings, Journal of Computer Aided Design, 2003.

[26] FlexiDesign [http://www.aspire3d.com/FlexiDesign.aspx] © Imagecom Inc.

[27] S. H. Joseph, T. P. Pridmore, Knowledge-directed interpretation of mechanical engineering drawings, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1992. 14(9): p. 928-940.

[28] Y. Q. Cheng and J. Y. Yang, A Knowledge-based Graphic Description Tool for Understanding Engineering Drawings, Proceedings of the 2nd International IEEE Conference. 1990.

[29] P. Vaxiviere, K. Tombre, CELESSTIN IV: Knowledge-based analysis of mechanical engineering drawings, IEEE International Conference, 1992.

[30] Ganeshram R. Iyer, John J. Mills, Design Intent in 2D CAD: Definition and Survey, Computer-Aided Design and Applications, Vol. 3, Nos. 1-4, 2006, p 259-267.

[31] F. Pena-Mora, D. Sriram, R. Logcher, SHARED-DRIMS: SHARED design recommendation-intent management system, Enabling Technologies: Infrastructure for Collaborative Enterprises, 1993. Proceedings., Second Workshop on 20-22 April 1993 Page(s):213 – 221.

[32] Y. Ishino, (IMPACT Laboratory, Dept of Aerospace Engineering, Univ. of Southern California), Y. Jin, Source: Advanced Engineering Informatics, v 16, n 2, April, 2002, p 107-125.

[33] Jeffrey M. Molavi, Raymond McCall, and Anthony Songer, A new approach to effective use of design rationale in practice, Journal of Architectural Engineering, 2003, pp 62 – 69.

[34] G. R. Iyer, J. J. Mills, A survey of the importance of Context in Mechanical Engineering Design, accepted at the 16th CIRP International Design Seminar, Canada - July 16-19, 2006.

[35] G. R. Iyer, J. J. Mills, Impact of context in Mechanical Engineering Domain, accepted at the 16th CIRP International Design Seminar, Canada - July 16-19, 2006.

[36] Brézillon P., Context in problem solving: A survey, The Knowledge Engineering Review, 1999, 14(1): 1-34.

[37] Alexander C, A Pattern Language: Towns, Buildings, Construction, Oxford University Press, New York, 1977.

[38] Doug Lea, Christopher Alexander: An introduction for object-oriented designers, ACM SIGSOFT Software Engineering Notes, 19(1):39–46, January 1994.

[39] S. Lawrence, “Context in Web Search,” IEEE Data Engineering Bulletin, vol. 23, pp. 25-32, 2000.

[40] Eric Glover, Steve Lawrence, William Birmingham, and C. Lee Giles. Architecture of a metasearch engine that supports user information needs, Eighth International Conference on Information and Knowledge Management, CIKM 99, pages 210–216, Kansas City, Missouri, November 1999.

[41] J. Budzik, K.J. Hammond, C. Marlow, A. Scheinkman, Anticipating information needs: Everyday applications as interfaces to Internet information servers, Proceedings of the 1998 World Conference of the WWW, Internet and Intranet, Orlando, Florida, 1998. AACE Press.

[42] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In Seventh International World Wide Web Conference, Brisbane, Australia, 1998.

[43] Website link : http://www.google.com

[44] Website link : http://myweb2.search.yahoo.com/

[45] Mostefaoui, G. K. and Brezillon, P., A Generic Framework for Context-Based Distributed Authorizations, Springer, LNAI2680, pp. 204–217, 2003.

[46] Dey, A. K., Understanding & Using Context, Personal and Ubiquitous Computing, Vol. 5, No. 1, 2001, pp. 4-7.

[47] Wagelaar, D. Context-Driven Model Refinement. Proceedings of the MDAFA 2004 workshop, Linköping, Sweden, June 2004 - LNCS 3599, August 2005, pp. 189-203.

[48] Fogarty, J., Lai, J., Christensen J., Presence versus availability: the design and evaluation of a context-aware communication client, Int. J. Human-Computer Studies 61 (2004) 299–317

[49] Goh, Cheng Hian, Madnick, Stuart E, Siegel, Michael D. International Conference on Information and Knowledge Management, Proceedings, 1994, p 337.

[50] Turner, R. M., Context-mediated behavior for intelligent agents, Int. J. Human–Computer Studies, V 48, pp. 307-330, 1998.

[51] Bremond, F. And Thonnat, M, Issues of representing context illustrated by video-surveillance applications, Int. J. Human-Computer Studies Vol. 48, pp 375—391, 1998.

[52] Bingolin and Brezillon, An Experience using Context in Translation from System's Requirements to Conceptual Model, Journal of Human computer studies 48(3).

[53] Pomerol and Brezillon, Context in Problem Solving: A Survey, The Knowledge Engg Review, 41(1) 47-80, 1999.

[54] Turner, R. M., Context-mediated behavior for intelligent agents, Int. J. Human–Computer Studies, V 48, pp. 307-330, 1998.

[55] Bingolin and Brezillon, An Experience using Context in Translation from System's Requirements to Conceptual Model, Journal of Human computer studies 48(3).

[56] J. J. Mills, J. Goossenaerts, H.J. Pels, The Role of Context in the Product Realization Process, Proc. CIRP Design Seminar, Stockholm, June (2001) 175-180.

[57] Sowa, J., 2000, Knowledge Representation: Logical, Philosophical and Computational Foundations, F. Brooks/Cole Thompson Learning, New York.

[58] Pahl, G. and Beitz, W, Engineering Design: A Systematic Approach, ISBN 3-540-19917-9.

[59] Dym, C. L., Little, P., Engineering Design: A Project-Based Introduction, ISBN: 0-471-28296-0.

[60] Dieter, G. E., Engineering Design: A Materials and Processing Approach, ISBN 0-07-366136-8.

[61] Pahl, G. and Beitz, W, Engineering Design: A Systematic Approach, ISBN 3-540-19917-9.

[62] Dym, C. L., Little, P., Engineering Design: A Project-Based Introduction, ISBN: 0-471-28296-0.

[63] Fenves, S., Choi, Y., Gurumurthy, B., Mocko, G., Sriram, R., Master Product Model for the Support of Tighter Integration of Spatial and Functional Design, NIST (2003).

[64] Gero, JS and Kannengiesser, U (2002), The situated function-behaviour-structure framework, in JS Gero (ed.), AI in Design'02, Kluwer, Dordrecht, pp. 89-104.

[65] Rosenman, M. A. and Gero, J. S., CAD modeling in mul-tidisciplinary design domains, in I. Smith (ed.), Artificial Intelligence in Structural Engineering, Springer, Berlin, pp.335-347, 1998.

[66] Ashby, M. F., Materials Selection in Mechanical Design, 3rd ed. ISBN 0-7506-6168-2.

[67] Christoph Hoffman and Robert Joan-Arinyo, CAD and the Product Master Model, Computer-Aided Design, Vol. 30, No. 11, pp 905-918 (1998).

[68] Yoshioka, M., and Tomiyama, T. (1997) "Pluggable Metamodel Mechanism: A Framework of an Integrated Design Object Modelling Environment", Computer Aided Conceptual Design, Proceedings of the 1997 Lancaster International Workshop on Engineering.

[69] Teresa De Martino, Bianca Falcidieno and Stefan Haßinger, Design and engineering process integration through a multiple view intermediate modeller in a distributed object-oriented system environment, Computer-Aided Design, Volume 30, Issue 6, pp 437-452.

[70] T. Gruber. Towards principles for the design of ontologies used for knowledge sharing, International Journal of Human-Computer Studies, 43(5/6):907{928, 1995.

[71] Alexander, Ishikawa, Silverstein, Jacobson, King, Angel, A Pattern Language, 1977.

[72] Bingolin and Brezillon, An Experience using Context in Translation from System's Requirements to Conceptual Model, Journal of Human computer studies 48(3)

[73] Szykman, S., J.W. Racz, and R.D. Sriram, The Representation of Function in Computer-Based Design, Proceedings of the 1999 ASME Design Engineering Technical Conferences, 1999.

[74] Julie Hirtz, Robert Stone, Daniel McAdams, Simon Szykman, Kristin Wood, A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts, Journal of Research in Engineering Design, 2002, vol. 13, number 2, pp 65-82.

[75] David G. Ullman, A Taxonomy for Mechanical Design, Research in Engineering Design, 1992, vol. 3, number 3, pp 179-189

[76] Dixon, J. R., Duffey, M. R., Irani, R. K., Meunier, K. L., Orelup, M. F., A proposed taxonomy of Mechanical Design problems, Computers in Engineering 1988 - Proceedings, 1988, p 41-46.

[77] Iyer, G., Mills, J.J., Barber, S., Devarajan, V., Maitra, S., Using a Context-Based Inference Approach to Capture Design Intent from Legacy CAD,

Computer-Aided Design and Applications, Vol. 3, Nos. 1-4, 2006, to be published, http://www.cadanda.com.

[78] Stauffer, L.A., Ullman, D.G., Dietterich, T.G., Protocol Analysis of Mech Engg Design, ICED 87, pp 74-85.

[79] URL: http://www.aspire3d.com, Imagecom Inc, Arlington, TX.

[80] http://www.cs.odu.edu/~jzhu/courses/content/logic/

[81] Wikipedia contributors, "Philosophical logic," Wikipedia, The Free Encyclopedia

[82] http://www.ruleml.org/

[83] Tool and Manufacturing Engineers Handbook, Editors: William H. Cubberly, Ramon Bakerjian, CMfgT, SME, , Vol 5. ed. 4, ISBN: 0-87263-351-9, 1989.

[84] Mechanical Engineering Design, Shigley, Joseph Edward, Charles Mischke, Richard Budynas, 7th Ed. ISBN 0-07-252036-1.

[85] Machine Design: An Integrated Approach, Robert L. Norton, 3rd Ed. ISBN: 978-0131481909.

[86] Machine Elements in Mechanical Design, Robert L. Mott, 3rd Ed. ISBN: 0-13-841446-7.

[87] Mechanics of Materials, R. C. Hibbeler, 5th Ed. ISBN: 0-13-00-8181-7.

[88] Materials Selection in Mechanical Design, Michael Ashby, 3rd Ed. ISBN: 0-7506-6168-2

[89] URL: http://www.efunda.com/

[90] URL: http://www.matweb.com/

[91] URL: http://www.google.com/coop/docs/cse/faq.html#1

[92] URL:http://wordnet.princeton.edu, Cognitive Science Laboratory, Princeton, NJ.

[93] URL: http://www.matweb.com, Material Property Data.

[94] Tool and Manufacturing Engineers Handbook – Desk Edition, vol 5, 4th ed, 1989, ISBN: 0-87263-351-9.

BIOGRAPHICAL INFORMATION

Ganeshram Ramji Iyer has attained a Master of Science degree and a Doctor of Philosophy degree in the field of Mechanical Engineering from the University of Texas at Arlington. His current research interests lie in the domain of Computer-aided Design (CAD), Product Data Management (PDM) and Product Lifecycle Management (PLM). He is currently working at Imagecom Inc on CAD products for interoperability and next generation Product Data Management.