



Using Rule Based Design in Engineer to Order Industry: An SME Case Study

Siva R. Chavali, Chiradeep Sen, Gregory M. Mocko and Joshua D. Summers

Clemson University, {schaval,csen,gmocko,jsumme}@clemson.edu

ABSTRACT

In this paper, the development and usage of rule-based design (RBD) in an industrial engineer-to-order (ETO) application is presented. First, three different design and geometric modeling processes are discussed for specifying customized bottle packaging systems, assemblies, and components. These processes include: (1) a manual method in which custom design specifications are uniquely created using two-dimension CAD software, (2) a custom in-house Visual Basic automated system built on a commercially available three-dimension solid modeling package, and (3) a commercially available rule-based system integrated with a commercially available three-dimensional solid modeling software tool. The advantages and limitations of the different modeling approaches are presented and evaluated qualitatively. Second, a detailed case study of an ETO sub-assembly is presented in which the third approach is used to create rule-based design applications. The rules for this sub-assembly are formulated and preliminary observations are reported. Guidelines for building and preparing solid models used in RBD are developed.

Keywords: rule-based design, CAD automation, engineer-to-order.

DOI: 10.3722/cadaps.2008.178-193

1. INTRODUCTION AND MOTIVATION

Hartness International (HI) is a successful, world-class small/medium sized enterprise (SME) located in the southeastern United States that designs and manufactures packaging and bottling systems. Each of the bottling systems is customized to meet the unique needs of the customer depending on the specific packaging requirements including the size of the bottle, the bottle configurations, the shape of the bottle, and the wrapping strategies.

HI has developed a product family-based approach using several base-line reconfigurable systems that are modified to address the unique set of customer specifications. This strategy is termed engineered-to-order (ETO). At the heart of ETO is the modification of individual components and assemblies to meet the functional needs of each customer. HI has experimented with three different strategies to enable ETO including: manual creation and modification of 2D and 3D design specification, in-house customized automation program development, and commercial rule-based design software to develop customized “programs”. This paper provides a case study evaluating each of these three strategies, discussing detailed advantages and limitations for each.

Additionally, a detailed example of using the third approach to build automated programs for a sub-assembly within the packaging grid assembly is presented. The RBD program is developed by first analyzing the existing design process of HI products and extracting the reusable knowledge. The reusable knowledge is represented as rules. The reusable rules are then captured in DriveWorks, version 6 (DW6) as equations or conditional (if-then) statements. DW6 is a commercially available rule-based design system for driving parametric SolidWorks 2007 geometric models. DW6 is chosen over RuleStream, other commercially available RBD software, as the implementation environment for several reasons; including success in other ETO applications, low maintenance cost and seamless integration with the 3D solid modeling software (SolidWorks-2007) at HI. The design and implementation of RBD programs represented in DW6 indicate a reduction in design cycle time by 50% (approximately from 12 hours to 6 hours) according to preliminary estimates by the HI RBD implementation team. Additional benefits include enhanced reliability of design information,

uniformity of modeling approach between designers, enforcement of design standards, capturing best practices, and retention of company knowledge in a long run.

2. RULE BASED DESIGN

Knowledge can be captured in machines for automating the design process [5, 6]. Presently, semantic networks, frames and rule based design are a few means of accomplishing this [5, 6, 9, 14]. Knowledge representations in engineering design formalize and organize the knowledge of a designer in form of software programs for designing products [6]. In case of rule based design (RBD), this is accomplished by capturing the knowledge necessary to solve problems in a particular design domain in form of software elements (code, databases, spreadsheets, etc.) and programmatically reusing this knowledge for taking design decisions within the problem domain [8, 10]. With this, much of the designers' domain-specific design tasks can be accomplished by the rule based design systems, and the designers can allocate their time in value-added activities for the enterprise.

Rule based systems are developed using system shells (SECLIPS [4], CLXPERT [7]), providing a skeletal structure for writing rules in simple syntax, and thus provide strategies for solving problems (shell's inference mechanism [5]). Shells can be considered as a platform on which rule based system can be built without any knowledge in the working principles of the shells themselves. When rules are explicitly embedded in shells, the later are ready to take up the task for that problem domain. Some advantages of RBD are presented in Tab. 1.

-
- *Rule based programs are easy for designers to understand and implement because of their relatively simple parametric or conditional (if-then) syntax.*
 - *Many variables in engineering design can be expressed in form of parametric relations. This provides ample opportunity for implementing RBD.*
 - *Many decisions in engineering design are taken by evaluating measurable conditions; hence can be easily represented as conditional rule statements.*
 - *RBD systems can automatically detect conflicts between different rule declarations, thus arresting programming inconsistency inherently.*
 - *For problems that must meet two or more conflicting constraints simultaneously (over-constrained problems), RBD systems can be used to first produce the design from a set of rules and then validate it against the constraints.*
 - *Revision of rules like modification, addition or deletion can be accommodated by the system at different stages without affecting the overall functionality of the system.*
 - *Commercial RBD systems usually offer a means to create custom user interface, making it very convenient for non-experts to use RBD system for solving problems.*
-

Tab. 1: Advantages of Rule Based Design.

However, there are some limitations to rule-based systems [5] (see Tab. 2).

-
- *Rule based design systems are typically built in teamwork between the domain expert (e.g. an expert designer) and the RBD system developer(s). Presently, the process of transferring the domain knowledge to the developer is by verbal interactions. Consequently, it is often difficult to ensure that the transfer of knowledge from the domain expert to the developer(s) is fast and effective.*
 - *The number of rules in complex systems can be high (in the order of hundreds), making debugging and causal analysis difficult.*
 - *RBD systems need a complete prediction of all design variations and possibilities so that decisions for each of those possibilities can be captured in the programs before running them. Hence it is difficult to build RBD systems to solve open ended problems, where the possibilities and design variations are difficult to predict.*
 - *RBD systems greatly depend on their private knowledge representations, resulting into repetition of knowledge across different RBD applications.*
-

Tab. 2: Limitations of Rule Based Design.

2.1 Practical Benefits of Rule Based Design

The product design process is considered as a search problem and it involves storing the how, why and what of a design [11]. Traditionally, CAD models are used to capture the geometric information about the product, without explicitly capturing these rationales. In RBD, this is achieved by preparing geometric models that are driven by rules that represent the design intent. A geometric model is a computer representation of product design process that stores product and process information for further realizing the product [13]. This information changes with the given product specification and necessitates doing repetitive design tasks that contain information of the previously designed models. This repetitive work can be reduced by building automated systems that work on previously captured design intent in the form of rules. Rule based design systems acts as productivity tool in reusing design knowledge and reducing design lead times by 90% [1, 12].

RBD systems provide a reliable platform for enforcing enterprise-level best practices as well as standards. The use of lists and databases allow RBD systems to choose standard values instead of calculating them as design variables. The concept of preferred numbers can be used to validate the design against standards such as ISO, BSI or ASTM .

Designers make errors even when provided with the best of training or with best design equipment [3]. These errors can be avoided if detected at the time of their occurrence. RBD systems can be built to predict the intention of designer and flag the errors when they detect deviation from this intention.

In addition, corresponding suggestions for correcting those errors can also be provided to the designer. In situations where minimization or detection of errors is not possible, care should be taken to at least minimize the effect of these errors. It is possible to develop rules in designing process by grouping these errors into classes and then analyzing them for minimizing their occurrence and their negative effects [3]. This can be done by providing higher and lower bounds to design variables and or making systems insensitive to values beyond them.

Retention of human expertise and knowledge is a major problem that enterprises are facing in today's fast-moving job market. Knowledge of an experienced human expert is an asset to an organization and the organization incurs a substantial loss when it loses human experts. Rule based systems provide a means to capture the knowledge of human experts, thus saving the organization from losing knowledge and expertise in case of employee turnover [3].

2.2 Knowledge Description

In this research, knowledge is represented in two ways. The first, are traditional if-then rules. Antecedent (condition), the "IF" part and consequent (conclusion), the "THEN" part are required for building a rule. A simple rule is represented as:

$$\begin{aligned} &IF \text{ <antecedent>} \\ &THEN \text{ <consequent>} \end{aligned} \quad (2.1)$$

An antecedent is made up of an object and its value is linked together by an operator. Object is a feature parameter that drives the rule depending upon the value that it is assigned. Operator is used for assigning value to an object. Conditional operators such as not equal to (!=), greater than (>), less than (<) etc., are used to compare the value of a numeric variable against a reference value, while assignment operators such as equal to (=) are used to assign a numerical value to a design variable. Multiple antecedents are joined by logical operators such as AND (conjunction), OR (disjunction) or combination of both. Following is an example that shows the rule for deciding when to use ribs in the design of a flat plate:

$$\begin{aligned} &IF \text{ 'thickness_plate' < 3mm} \\ &AND \text{ 'length_plate' > 30cm} \\ &OR \text{ 'width_plate' > 30cm} \\ &THEN \text{ use_ribs = TRUE} \end{aligned} \quad (2.2)$$

The second knowledge representation used in ETO applications is parametric relations [12]. This is accomplished by expressing a design variable in terms of mathematical relations between other design variables. The relation can be expressed in form of an equation. The variable being declared (on the left hand side of the equality) is the dependent variable, which in turn can drive other variables in other equations. This can create a large tree of variables arranged in order of dependencies, in which a large number of variables can be defined as functions of a few independent variables. In any particular design problem only these independent variables are manipulated by the designer and the RBD system determines the remaining dependent variables.

For example, in designing a table for a conference hall, the length of the table is dependent upon the number of intended people attending the conference. Here the length of the table is a function of number of people who attend the conference. Assume that the table design problem is defined as follows:

“Determine the length and width of a long table for use in conference rooms. Each person needs to maintain at least 75% of the width of the chair from the next person or from the corners of the table to sit comfortably. The conference room configuration may allow one person to sit at one or both ends of the table. The conference room may allow sitting on one or both longitudinal edges of the table.”

The parametric expressions for this design can be expressed as below:

$$\begin{aligned}
 \text{chair_width} &= 30 \\
 \text{num_person} &= 20 \\
 \text{num_sides} &= 2 \\
 \text{num_ends} &= 1 \\
 \text{space_comf} &= \text{chair_width} \times 0.75 \\
 \text{seat_side} &= \text{ceiling}((\text{num_person} - \text{num_ends}) / \text{num_sides}) \\
 \text{table_length} &= \text{seat_side} \times (\text{chair_width} + \text{space_comf}) + \text{space_comf} \\
 \text{table_width} &= \text{chair_width} + 2 \times \text{space_comf}
 \end{aligned} \tag{2.3}$$

Here the variables have been given self-explanatory names. We can see that the table design problem needs only four independent variables to determine the length and width of the table satisfying the design problem (see Eqns. 2.1 - 2.3). Note that even if the number of persons is 20 and one person is sitting at one end of the table (leaving the rest 19 for the long sides) the rules determine the number of seats per side (“seat_side”) as 10, allowing one vacant space. This is a reflection of the design intent captured in the rules.

In this example, we have used the “ceiling” function from the ANSI-C standard functions library; however, it is understood that the availability of this function to the RBD environment of the shell may vary between commercial systems. In case the function was not available, the expression “seat_side” would involve conditionals.

Apart from design decisions, design constraints can also be encoded in RBD systems by writing verification (check) rules. Typically, problems that are over-constrained (i.e. must meet two or more conflicting constraints simultaneously) need these rules. A primary set of rules are first used to produce the design, which is then checked against the constraints encoded in the verification rules. Rules can also represent relations, recommendations, directives, strategies and heuristics for coming up with satisfying design solution based on the given design constraints and specifications. However, it is noted that these rules can all be represented in one of the two descriptions (if-then or parametric) discussed above.

There are multiple ways for representing rules outside the scope of this research. The two types of representations (parametric and conditional) described here are based on an iterative development of the ETO case study.

3. CASE STUDY: HARTNESS INTERNATIONAL (HI)

Hartness International Inc., headquartered at Greenville, South Carolina, is a family owned enterprise that designs, manufactures and services packaging equipment and solutions for bottled packaged goods. Started in 1940, the company now has about 450 employees, with manufacturing facilities in North America, Europe and China and sales and regional offices all around the world. Hartness International’s solution portfolio includes a wide variety of products, including case packing solutions for bottled goods and several high-efficiency conveyor solutions, shrink wrapping systems, bottle filling solutions, bottling line monitoring solutions with video capture, and packaging system integration solutions for its customers. Common across many of the products designed and manufactured at Hartness International are the followings:

- Commonality across product lines: the products designed and developed at HI fall into product families.
- Functional and geometric commonality between components, assemblies, and sub-assemblies.
- Customization of product lines based on specific customer requirements.

These product characteristics lend themselves to the usage of rule-based design systems for engineered-to-order (ETO) applications across the Hartness product line. The grid assembly analysis is a pilot project in which the effort required to capture rules, develop RBD programs, and use the RBD programs are analyzed and evaluated. The experiences and initial findings from the pilot study are summarized in the following sections.

3.1 Study Scope: Grid Assembly

The HI Rule Based Design project was scoped to automate the design of the grid subassembly in the case packaging line at HI. A typical case packing machine is shown in Fig. 1. The machine is fed with filled bottles and empty cases and delivers cases filled with bottles.

A typical case packing machine has two rows of conveyors running in the same direction along the length of the machine at two levels. In the packaging system the flow of bottles is from right to left, with the operator positioned in front of the machine. The bottles run in clusters on the top conveyor, while the cardboard cases run along the bottom one. The grid assembly, located toward the left end of the machine (shown with the ellipse in Fig 1), transfers the bottles from the top conveyor to the cases on the bottom conveyor.

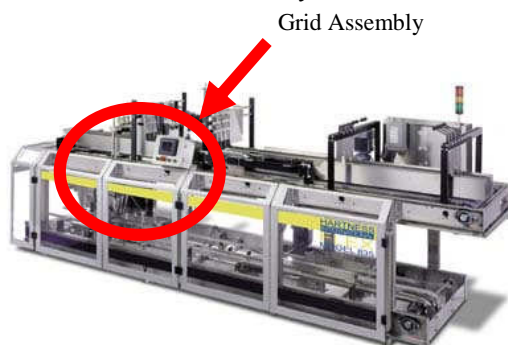


Fig. 1: Hartness Model-825 Case Packer.

The grid assembly shown in Fig. 2 is a door-type grid assembly in which the bottles are released to the case by opening a pair of doors. The bottles are fed by the conveyor on top of the door assembly, which is a part of the grid assembly. Simultaneously, a case is fed by the bottom conveyor and aligned right below the door. An elevator then lifts the case so that the fingers hanging from the door subassembly are partially inserted into the case. The doors are then turned open by a pneumatic cylinder-actuated mechanism about the door hinges. The bottles fall into the fingers, which guide them into the right locations in the case.

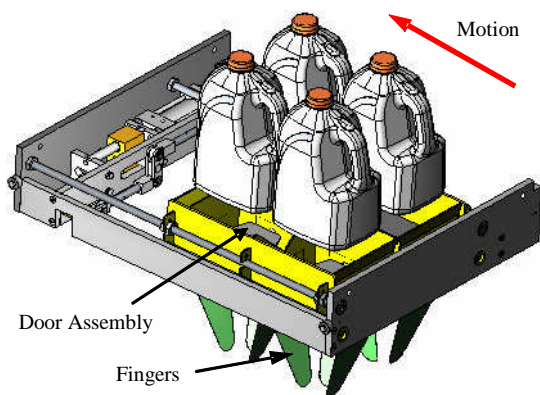


Fig. 2: Grid Assembly for Model-825.

3.2 Analysis of the Grid Assembly: Choice of RBD

Before undertaking the project efforts, a study was conducted by the Clemson team to analyze the design process in Hartness International. The following observations were made:

1. The case packing machines are typical examples of variant design problems. HI maintains a series of product lines of case packers depending on technology and application. However, each case packer is suited to the requirements of the customer's production requirement.
2. A case packer can be used to pack various types of bottle products in multiple arrangements in the cases. For example, the same machine can be used to pack large bottles (e.g. detergents) in a 2x3 arrangement and

also small bottles or cans (e.g. soft drink cans) in a 4x6 arrangement in different cases. However, this variation can be accommodated by changing grid assembly and lane divider spacers.

3. The grid assembly needs to be redesigned for each packing arrangement and thus called grid change over parts. This means that all grid assemblies must be designed to suit the same case packer, for a specific bottle-case configuration. This creates a lot of opportunity for information reuse between different grid designs. for a specific bottle-case configuration
4. Grid assemblies are functionally and component-wise similar, but no two grids are identical. This leads to a situation where the specific designs can be thought as variants of a basic design.

The above mentioned repetitive nature of the grid design problem provides a suitable ground for implementing RBD solutions. The functional and geometric similarity enable the assemblies to be modeled as a single seed model in CAD, and the specific instances to be generated using rule based variations on the seed model. Since the variations are finite and predictable, the RBD system can be built to capture the design intent for all possible variant designs.

3.3 ETO Process Approaches at Hartness

In the following three sections, three design approaches of information reuse in HI are described and their significant characteristics are pointed out.

3.3.1 Manual Information Reuse Approach

The process of designing grid assemblies in this approach in Hartness International is depicted in Fig 3. The process relied heavily on the capabilities of AutoCAD, which was used as the primary CAD platform. During a new design the designer was required to design the components and the assemblies by referring to the customer specifications manually.

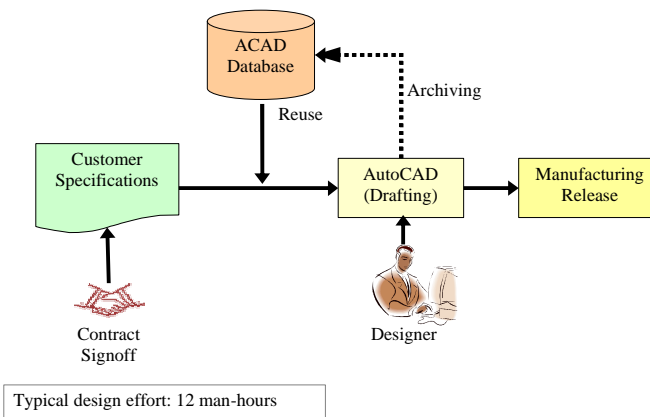


Fig. 3: The Manual Information Reuse Approach with AutoCAD.

Typically, the results of each design were stored for reference in the default file structure of the computers (shown by the dashed arrow), which gradually grew up to be an informal “AutoCAD database”. The only means of re-using design information in this process was by manually referring to AutoCAD designs of previous contracts from this database (shown by the thick arrow in Fig 3). The dashed arrow indicates the process of archiving AutoCAD data in the informal database. This approach of reuse was characterized by the following:

- The reuse was informal, often inspired by the individual designer’s interest towards speed.
- It depended heavily on individual designer’s familiarity with the database and searching proficiency.
- The database was informal and unsecured.
- The design process did not have any built-in mechanism to ensure consistency between different design outputs.

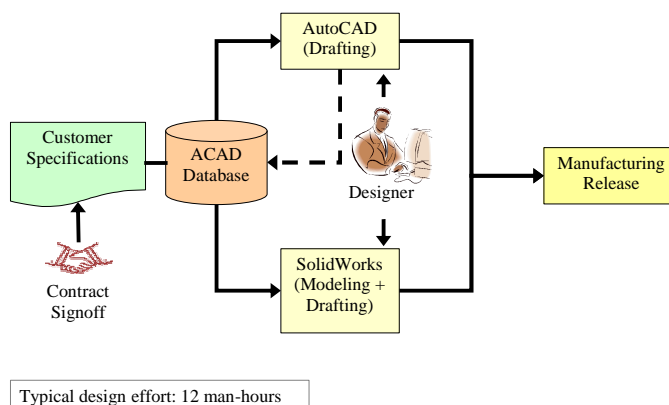


Fig. 4: The Manual Information Reuse Approach with AutoCAD and SolidWorks.

At a later stage while using this approach, the company upgraded its design software platform from AutoCAD to SolidWorks, the mid-range 3-D CAD software marketed by Dassault Systems. However, this upgrade did not have much influence on the design process flow, other than the fact that the designers now referred to the legacy designs from the AutoCAD database to produce models and drawings in SolidWorks (by manually translating the AutoCAD data into SolidWorks). AutoCAD is continued as an auxiliary CAD platform till date since it often provides means for fast design release by modifying the legacy data for new contracts. The modified process flow with both AutoCAD and SolidWorks is shown in Fig. 4.

The thick arrows indicate direction of information flow during informal reuse of AutoCAD data. The dashed arrow indicates the process of archiving AutoCAD data in the informal database.

3.3.2 The In-house Rule Based Design Approach

Soon after the introduction of SolidWorks, the business benefits, opportunities and feasibilities of information reuse in design were becoming apparent and the first attempts were made for capturing design information in software tools. This started as more of an exploration project from a single designer that matured into a company-wide design automation project. However, this model did not rely on dedicated knowledge representation tools. Design rules and knowledge were captured in Visual Basic (VB) software applications. The VB applications are integrated with SolidWorks through the application programming interface (API) and macros. However, this exercise required that the design process of the case packing machines be studied in search of reusable components and formalize them as rules. Consequently, the company identified the grid assembly as a good candidate for rule based design and carried out preliminary analysis of the design parameters, their dependencies and the decision points that control them. The company had a total 88 product lines in grid assembly, and VB-based macros were written to drive nearly 40 of them. This approach of information reuse is depicted in Fig. 5.

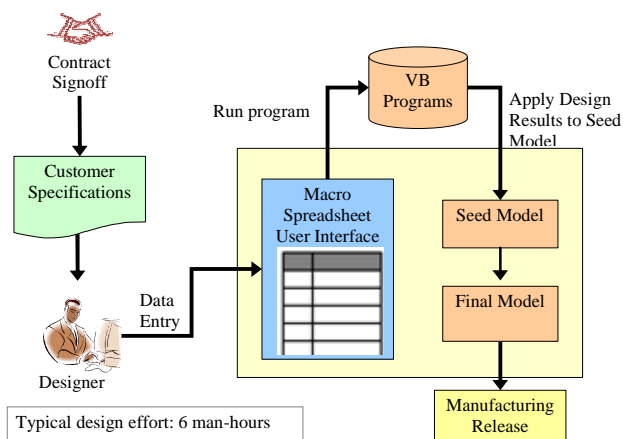


Fig. 5: The In-house Rule Based Design Approach.

This model of information reuse can be characterized by the following:

1. It was based on a formal exploration of reusable activities and rules within the design process.
2. Effort was spent in resolving design knowledge in form of rules that could be expressed as equations connecting variables. This needed identification of all possible design variations and the design parameters (dependent and independent variables, constants, etc).
3. Consequently, the components and assemblies for each product line were modeled in SolidWorks as a general representation, called a seed model, which was repeatedly updated using the VB-based SolidWorks macros to produce specific design solutions.
4. This scheme required a “programmer” for developing the programs in VB. Designers were unable to control the system due to lack of programming knowledge.
5. Since the system was an in-house development, its maintenance depended greatly on the developer, hence increased the company’s dependency on an individual.

3.3.3 Commercial Rule Based Design Approach

In the Hartness International Rule Based Design project, the team from Clemson University improved upon the VB-RBD model by introducing industry-standard Rule Based Design software, DriveWorks-6 (DW6), to drive the SolidWorks-2007 models. The project scope included those configurations of the grid assembly, which were not covered by the VB programs.

In this approach, the design rules for each product line are captured in the rules environment of DW6. SolidWorks models are built compatible to the design output produced by DriveWorks. During a new design, the designer has to interact with a custom user interface in DW6 that asks specific questions about the grid assembly design parameters. These parameters are then used by DW6 to process the rules that evaluate the remaining design parameters, including configurations, mating conditions, dimensions and tolerances for all the components in the assembly. The results of this rule based evaluation are automatically stored by DW6 in the file system. The design outputs (models and drawings) are produced in SolidWorks-2007 automatically by first connecting the SolidWorks-2007 session to the stored design parameters through an interface and then updating the parametric seed models to those parameters as shown in Fig. 6. This approach of rules reuse is characterized by the following:

1. Rules are captured in a dedicated RBD software tool, as opposed to a general purpose programming language, making declaration, access and retrieval of rules easy.
2. Rules are declared and edited using the DriveWorks graphical interface, as opposed to the code in VB. This greatly enhanced rules readability and interpretation by designers and eliminated dependency on programmers.
3. The declaration syntax is simple, typically stated in IF-THEN format, with only one variable being controlled in a statement. This makes the rules structure modular and makes control easy.
4. Effort was spent in resolving design knowledge in form of rules that could be expressed as equations connecting variables. This needed identification of all possible design variations and the design parameters (dependent and independent variables, constants, etc).
5. All possible design options were predicted before building the RBD system, allowing declaration of decisions for each situation in IF-THEN format.
6. The seed models in SolidWorks are built in synch with the DriveWorks output format. This needed special attention toward modeling.
7. The rules are stored and managed in an environment outside the CAD system, thus enforcing design standards and reducing the risk of errors by inexperienced designers. At the same time, the rules are simply declared, allowing an experienced designer (with proper rights) to easily change them.

3.4 Comparison between the Three Approaches of Information Reuse

A comparison between different models of information reuse in design reveals the following observations:

1. The commercial RBD systems are costlier to implement than homegrown RBD systems or manual information reuse approaches, owing to the high initial investment and long implementation times.
2. However, the commercial RBD systems produce a much faster return on investment (owing to reduction in running cost and design cycle time).

3. Additional benefits come from uniformity of design output from different designers, enforcement of design standards and company best practices, enhanced reliability and retention of company's knowledge in the long run.
4. The commercial RBD systems provide a more sustainable development environment because of continuous support from their vendors.

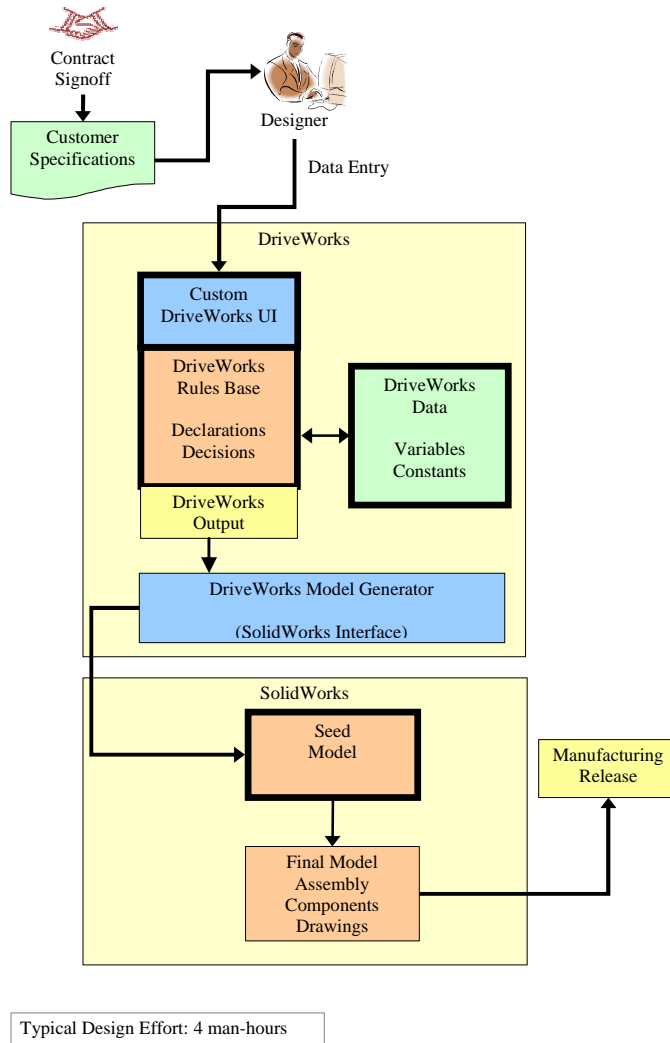


Fig. 6: The Commercial Rule Based Design Approach.

3.5 Project Execution

The project was executed following a modified version of the typical Software Development Life Cycle (SDLC) model. The process for developing an RBD system for a grid product line consisted of four phases:

1. Rule elicitation & documentation
2. Geometric modeling in SolidWorks-2007
3. Rule building in DriveWorks-6
4. User Interface creation in Driveworks-6
5. Testing

The activities and deliverables of each phase are explained in the following sections.

3.5.1 Rule Elicitation & Preparation

This is the first phase in the project life cycle. It requires detail understanding and analysis of the product, their components and assemblies, their functions, manufacturing processes, and specifically, the current design process. During the study of the current design process, all the design variables in the component and assembly part files are identified. The rules (parametric and conditional) governing the variables are identified along with the conditions for applying them. Once rules are formed they are documented in a predetermined format.

In the following example, the process of converting the functional requirements into design rules is explained using a component (turn buckle) of the grid assembly. The intent of this example is to determine the length of the turn buckle. Fig. 7 shows the arrangement of various links that operate the doors in the grid assembly in the typical case packing machine shown in Fig. 2. The center block moves along the cylinder axis when the cylinder is actuated. This in turn moves the center link, which is rigidly fixed to the center block. The doors are connected by the cranks and clevises to the center link by means of the turn buckles.

The design variables involved in this calculation are identified in Fig. 8, Fig.9 and Fig. 10. The independent variables include the bottle width (W), the number of bottle lanes (N) that goes on the top of these doors and the bottle lane clearance (C). Here clearance C is described as the total clearance allowed on both sides of the bottle. Thus the distance between door rods (D) is defined by the rule:

$$D = (W + C) \cdot N \quad (3.1)$$

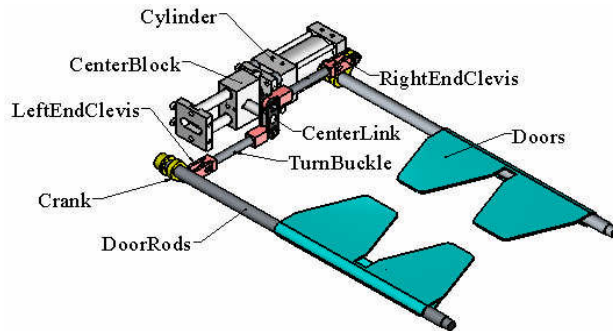


Fig. 7: Typical Door Assembly.

When the doors are horizontal, the design requires that the cranks are positioned 45 degrees with the horizontal as shown in Fig. 9. The crank on the left is 45 degrees below horizontal and the one on the right is 45 degrees above horizontal, both facing right. This requirement leads to the value of the horizontal distances, LED and RED , in terms of crank length (CrL) and the clevis length (CvL).

$$LED = CvL + CrL \cdot \cos(45) \quad (3.2)$$

$$RED = CvL - CrL \cdot \cos(45) \quad (3.3)$$

The length of the center link (CLD) is an independent variable in this calculation. Thus the distance available for the two turn buckles (AD) is calculated as:

$$AD = D - LED - CLD - RED \quad (3.4)$$

The length of the turn buckle (L) can be calculated by adding half of available distance (AD) and thread depth (TD) on each side as shown in Fig. 10.

$$L = \frac{AD}{2} + 2TD \quad (3.5)$$

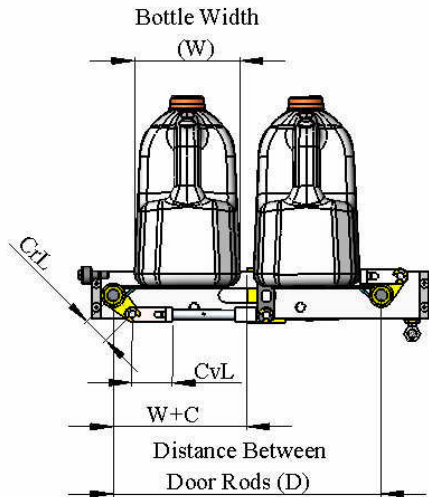


Fig. 8: Door Links and Bottles.

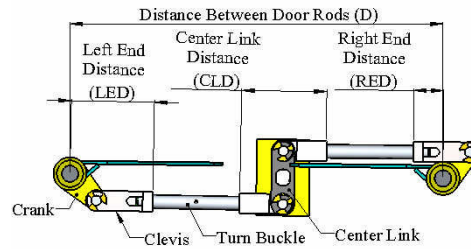


Fig. 9: Door Links and Turn Buckles.

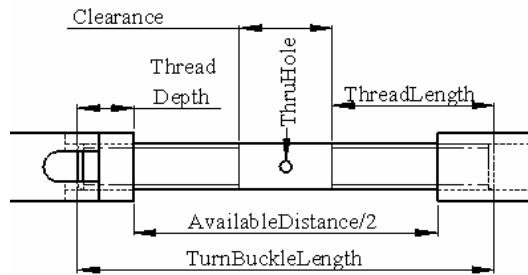


Fig. 10: Turn buckle.

The turn buckle has threads on both ends and a through hole at the center. A design criterion requires the threads to be at least equal to $\frac{3}{4}$ inch (fastening requirement) and the distance between the two threads to be at least 1 inch (hole clearance requirement). The minimum length constraint of the turn buckle then becomes $1 + \frac{3}{4} + \frac{3}{4} = 2.5$ inch. If this condition is not met by the length produced by the rules in Eqn. 3.4, an offset type center link is to be used allowing the additional room for satisfying the minimum length condition, as shown in Fig. 11. However, the offset distance (OD) is fixed at 0.88 inches. Hence, if the minimum length criterion for the turn buckle does not meet even after using an offset link, then the design is rendered infeasible.

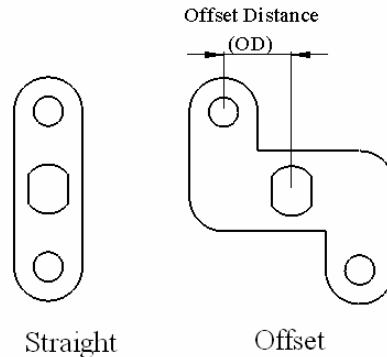


Fig. 11: Center Link Configurations.

This offset link thus gives an additional distance (0.88 inch) to accommodate the buckle in odd situations. This is also a driving parameter and recommendation for validating this design can be done. This is captured by the following rules:

$$\begin{aligned}
 &L_{\min} = 2.5 \text{ (constant declaration)} \\
 &IF(L \geq L_{\min}) \\
 &\quad THEN(\textit{straight_center_link} = TRUE) \\
 &ELSE IF (L < L_{\min} \text{ AND } L \geq (L_{\min} - OD)) \\
 &\quad THEN (\textit{offset_link} = TRUE \text{ AND } L = L_{\min}) \\
 &ELSE("design_not_possible")
 \end{aligned} \tag{3.6}$$

3.5.2 Geometric Modeling in SolidWorks-2007

During this phase, fully parametric geometric models and drawings of the components and assemblies of the grid assembly were built using SolidWorks-2007. Feature parameters and other design variables (number of holes in a pattern, pitch, etc.) that needed control by DW6 were identified and named using a prescribed naming convention. In order to make the models compatible for DW6, a certain modeling approach was undertaken. The following points characterize the approach.

1. During creation of sketches, care was taken to avoid dimensions of detail features from absolute reference frames. This helps maintain the parametric relation of size and positioning of the features with respect to other features in the body. When a change in the driving parameters is imposed by DW6, the driven dimensions are properly updated with respect to each other, ensuring successful model regeneration.
2. For multiple occurrences of features (e.g. an array of holes), the array or pattern function of the CAD tool was used. This allowed DW6 to control the pitch and number of occurrences in the patterns, ensuring successful model regeneration.
3. Though SolidWorks-2007 allows combining the profiles of multiple features in a single sketch, this was avoided. Designing separate sketches for separate features decoupled the features from each other and allowed DW6 to control them independent of each other.

3.5.3 Rule Building in DriveWorks

In this phase, the design rules are captured in DW6 and links are built to the corresponding parameters in SolidWorks-2007. Fig. 12 shows a sample screenshot of the DriveWorks Model Wizard, which is a DW6 interface invoked from SolidWorks-2007. It shows the mapping between DW6 variables (left column) to corresponding SolidWorks-2007 parameters (right column).

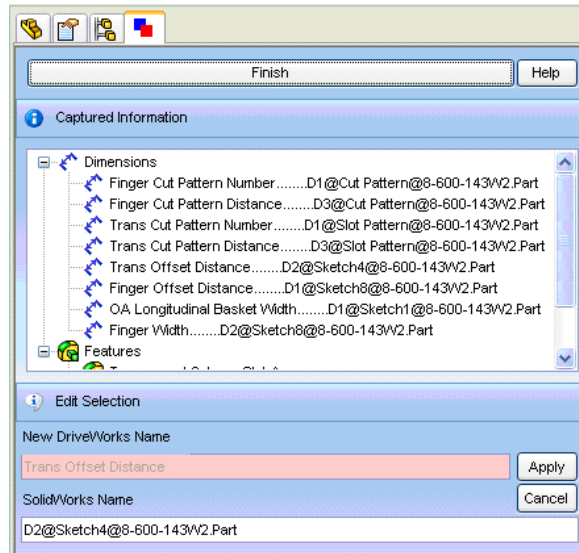


Fig. 12: Design Parameters in DriveWorks Model Wizard.

The sample set of rules for controlling the parametric models of the door assembly, identified in Section 3.5.1, are implemented in DW6 using the model wizard. Each of the rules that control the assembly must be encoded in DW6 to provide the linkage between the solid models and the rule interface. Fig. 13 shows a screenshot of the DW6 rules statements.

Variable Name	Variable Formula
Angle in Radians	=BentAngleOfFingerReturn * 3.14 /180
AT Avail Fing Length	=BottleChainToCaseChainDistanceReturn-5.063-(H
AT Case Plate Dimension	=BottleChainToCaseChainDistanceReturn -2.063
AvailDistance	=(DwVariableOALaneCenter + 2 * DwConstantCy
BasketLengthNoTrsvrsl	=(NoofBottlesperLaneReturn * BottleLengthReturn
BasketLengthWithTrsvrsl	=(NoofBottlesperLaneReturn * DwVariablePatternD
Clearance for finger bend	=(0.5/COS(1.57-DwVariableAngleInRadians)-0.5)**
End Finger Tab Offset	=IF(EndFingerWidthtoUseReturn > 1.47,0.5,IF(Endf
Lane Center	=BottleWidthReturn + BottleLaneClearanceReturn +
LongitudinalLength	=IF(NumberofTransversalReturn >0,DwVariableBas
NoOfTrsvrslReq	=IF(OR(NoofCasesToFillReturn>1,NoOfPartitionsCa
OA Lane Center	=DwVariableLaneCenter * NumberOfLanesReturn
OffsetDistanceWithoutTrsvrsl	=IF(NoofBottlesperLaneReturn >1, ((NoofBottlesperl
OffsetDistanceWithTrsvrsl	=IF(NoofBottlesperLaneReturn >1, ((NoofBottlesperl

Fig. 13: DriveWorks Rules.

3.5.4 User Interface Creation

DriveWorks-6 (DW6) supports the creation of custom user interface forms. This capability was exploited to create the design process wizards. The UI was designed in such a way that only the independent variables of the design problem were asked in the UI wizard dialogs. Fig. 14 shows a sample user interface form.

825/900 Platform Lowering Head LH

Note: Specify All Dimensions in Inches

Packer Information

Machine Work Order #

Packer Type-Height

AT-Low Bag

AT-Mid Bag

Wear Strip

Lane Divider Thickness

No Of Lane Dividers

Bottle Chain To Case Chain Distance

Secondary Shift Frame? If Yes, Check box

Check Available Finger Height Manually!!!

Case Information

Width (Inside) Does Case Have Flaps? If Yes, Check box

Length (Inside) Are Flaps Vertical? If Yes, Check box

Height (Inside) Does Case have Partition? If Yes, Check box

Flap Height

Partition Height

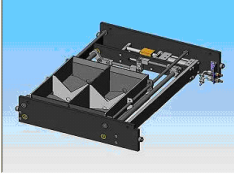


Fig. 14: Design Parameters in DriveWorks Model Wizard.

3.3.5 Testing

A detailed test protocol was developed by the Clemson team in conjunction with HI for testing the RBD system. Emphasis was given to test all possible cases, in order to trap bugs in the rules, the user interfaces and the models. The system is tested in terms of:

- Correctness: checking whether the rules produce the design in accordance with the design requirements in call cases, repeatedly.
- Consistency: testing whether the rules can lead to conflicting design intents in any case, or produce impossible design.
- Robustness: testing whether the system can withstand intentionally inconsistent input values. The system is supposed to either arrest such input (by imposing ranges on values, etc.) or feedback to the user about the inconsistency before applying the rules to the models.

3.4 Observations

The case study presented in this paper reveals several key characteristics of rule-based design adoption and development in an engineer-to-order company. The observations are summarized as follows:

1. The knowledge engineer, designers, and drafters must work in conjunction to ensure the solid models are “rules-ready”. The solid models encountered in this case study early in the process did not capture design intent correct, thus make it difficult to effectively control the design. As the project progressed, guidelines were established as best practices for solid modeling.
2. A significant learning curve was present in identifying rules and design intent. Rules-based design is a significant change for developing customized products. Thus in addition to learning a RBD software, engineering designers must learn to explicitly represent design intent.
3. An economic assessment for adopting RBD for a product line is required to ensure value to a company. However, it is difficult to assess the economic impact of RBD implementation. In order to complete a preliminary assessment a pilot project must be completed. In this pilot project, the time and effort required eliciting rules, developing documentation, implementing rules programs, and test and debug must be recorded. This time information must be used in conjunction with frequency of product usage to estimate the economic impact

4. CONCLUSION

RBD systems are a means of capturing design domain knowledge in form of rules in software and reusing them in variant design problems.

In design, repetitive components can frequently be identified in various proportions. For industries involved in variant designs (obtaining a specific design from a general set of solutions), the share of repetitive design is higher than the

other industries. These design situations provide an opportunity for automation through rule based design, since rules are easier to write for finite and predictable number of variations.

It has been shown in this paper with the help of a case study, in which three approaches of information reuse were compared, that commercial rule based design systems produce higher business benefits in terms of higher design speed, fast implementation (due to high compatibility with commercial CAD tools), user-friendliness (due to custom UI) and sustainability (due to continuous support from vendors).

The homegrown RBD software (Visual Basic macros) developed on general purpose programming tools make rules declaration, edition and retrieval difficult for designers who are not proficient in programming. This approach also increases the company's dependency on individual employees. On the other hand, commercial RBD systems do exist to provide dedicated RBD interface to commercial CAD systems. These systems come with corporate support and provide sustainable development environment.

Further, since product design is carried out using commercial CAD software, the RBD system should be chosen based on its compatibility with the CAD system. In the case study, DriveWorks-6 was chosen because of its compatibility to support RBD on SolidWorks-2007 CAD software.

The task of developing an RBD system should be undertaken as a formal software development activity. Standard software development lifecycles should be followed in principle, with moderate modifications to suit the nature of the project.

5. FUTURE WORK

The future work can be summarized in the following points:

- Presently, RBD systems greatly depend on their captive knowledge representation. All knowledge necessary for solving problems in the specific domain need to be captured while building the RBD system. This entails repetition of knowledge capture task and provides no means for reusing knowledge between individual RBD systems. Implementation of the concept of similarity in RBD systems will enable reusability of rules across the systems.
- At present, the typical model of implementing RBD systems involves a domain expert (designer), who supplies the design knowledge, and the software developer, who collects the knowledge and converts them into the RBD system. Research effort is necessary toward the development of a shell that allows developing RBD systems through visual programming languages. This will enable the designers themselves to build RBD systems, thus eliminating the dependency on the software developer.
- A case study should be conducted to learn how designers work with Rule Based Design systems and the systems should be redesigned such that they are friendlier to work with, even for untrained personnel.

6. ACKNOWLEDGEMENTS AND DISCLAIMER

We would like to thank Hartness International for allowing us to conduct the case study on Rule Based Design. We would especially thank Mr. Mickey Dorsey, Mr. Marc Monaghan, Mr. Cliff Bardsley and Mr. Clayton Rowley for their cooperation and support. Last but not least we would like to thank the grid designers at HI for sparing their precious time in explaining the current design process. No approval or endorsement of any commercial product, service or company by Clemson University is intended or implied.

7. REFERENCES

- [1] Callot, M.; Kneebone, S.; Oldham, K.; Murton, A.; Brimble, R.: 1998, MOKA - A Methodology for developing Knowledge Based Engineering Applications, Proceedings of European Product Data Technology conference, Watford, UK, 1998.
- [2] CIM data, The Value of Rules-Driven Product Management, CIM data, <http://www.CIMdata.com>, 2005
- [3] Donald, A.: Design Rules Based on Analyses of Human Error, Communications of ACM, 26(4), 1983.
- [4] Hadlock, F.: SECLIPS: A Structured English Interface for an Expert System Shell, Proceedings of the 17th International Florida Artificial Intelligence Research Society Conference, Miami Beach, Florida USA, 2004.
- [5] Kingston, J.: Rule Based Expert Systems and Beyond: An Overview, AIAI-TR-25, 1997, United Kingdom.
- [6] Kulon, J.; Broomhead, P.; Mynors, D. J.: Applying Knowledge-Based Engineering to Traditional Manufacturing Design, The International Journal of Advanced Manufacturing Technology, 30(9-10), 2005, 945-951.
- [7] Lee, J. S.; Hung, C.: 1995, CLXPRT: A Rule-Based Scheduling System, Expert Systems with Applications, Pergamon Press Ltd, 9, 1995, 153-164.

- [8] Rao, Z.; Hsien, C.: A Matrix Representation and Mapping Approach to Knowledge Acquisition for Product Design, Proceedings of 9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems, Melbourne, Australia, 2005, 311-317.
- [9] Rattanaprateep, C.; Chittayasothom, S.: Expert Database System Architecture and Implementation on Object Relational Databases, WSEAS Transactions on Computers, 5(3), 2006, 560-567.
- [10] Richter, K. J.; Blemel, K.; Fink, J.; Grewe, C.; Hashmi, A.: The Aries Project for Rule-Based - Design Knowledge Acquisition, IDA paper, 1995, 1034.
- [11] TechnoSoft Inc., Knowledge-Based Engineering (KBE), <http://www.technosoft.com/kbe.php>, Accessed on Jan 23, 2007.
- [12] Yeun, Y. S.; Yang, Y. S.: Design Knowledge Representation and Control for the Structural Design of Ships, Knowledge-Based Systems, Elsevier Science, 10(2), 1997, 121-132.
- [13] Zeid, I.: Mastering CAD/CAM, McGraw-Hill, New York, 2005, 69-70.
- [14] Zuozhi, Z.; Jami, S.: Modeling and Representation of Manufacturing Knowledge for DFM Systems, Proceedings of the ASME Design Engineering Technical Conference (CIE), Salt Lake city, Utah, US, 2004, 723-733.