A Simple Proposal for the Development of Intelligent Design Catalogs

Ian Yellowley, Paul Winkelman

Department of Mechanical Engineering, University of British Columbia
yellowley@mech.ubc.ca

Abstract

Engineers frequently refer to catalogues when designing products and by carefully selecting standard components, they are able to create their own unique systems. Unfortunately, these catalogues tend to serve a limited audience as they favour experienced designers. This research is aimed at developing a software framework that renders catalogue data more accessible to novice designers. The system envisaged is composed of a highly object oriented virtual design environment that allows engineers to develop their products at the conceptual level and then draw on catalogue data as they enter the embodiment and specification phase of the design process. In addition to catalogue data, this design environment must integrate other design aids, such as graphics, simulation and analysis programs. Ultimately, the virtual environment should allow novice designers to develop systems that rival those of experienced engineers using traditional catalogues.

1.0 Introduction

The engineering design process is typically modeled as passing through several phases or stages. Pahl and Beitz (1), for example, describe the process with respect to four phases. During the *clarification* of the task, information is gathered, leading to detailed problem specifications. The *conceptual* phase results in solution variants, establishing functions and principles. During the *embodiment* phase, form is assigned to the concepts and functions. In the *detailed design* phase, the required properties of the individual components are analysed and perhaps optimized.

Within this design model, it is during the third and fourth phases that engineers typically refer to design catalogues. These catalogues may be either computer-based or paper documents and list the components and products (assembled components) of a particular supplier or manufacturer. Product specifications are provided which allow designing engineers to select products or components that, when combined in a new system, produce the required functionality. To further assist the selection process, catalogues often present sample design problems and solutions which show how their products are typically employed in larger systems. Sample calculations indicate how the specifications of the individual components can be used to determine the overall performance of the sample system. Being located

in only the third and fourth phases of the design process, it is evident that a significant amount of the design process (largely conceptual) is already carried out before any reference is made to product catalogues. Hence, product catalogues assume that their users possess considerable design expertise. Users generally need to be knowledgeable in terms of what components are available, what their basic features and functionalities are and how they are typically incorporated into larger engineering systems. Without this knowledge, there is no logical approach than can be employed to locate the proper components.

The novice designer using design catalogues can be likened to a beginner French student trying to understand new words using a dictionary written solely in French. Upon looking up the unknown word in the dictionary, the student encounters additional unknown words in the definition which, too, must be found in the dictionary. This process can continue indefinitely, causing the user to give up in frustration. New words which can potentially serve the needs of the language student remain out of reach. Larger dictionaries attempt to address the problem by including sample sentences to further clarify proper usage, but these sentences can seldom be used unaltered. Thus, to make effective use of the dictionary, the student must possess a certain mastery of the language.

Novice designers face similar difficulties when attempting to use product catalogues to design their products. To increase accessibility, the threshold of knowledge needed to navigate through these catalogues must be lowered. In other words, knowledge that would normally have been presumed to reside in the head of experienced designers must be transferred to the virtual system. Once represented in the virtual space, this knowledge must be made both accessible and understandable to the novice designer.

2.0 Background to the Basic Ideas

The proposed system rests on a long line of previous research conducted in parallel across several disciplines. The earliest approaches to simplifying both design and manufacture so that novice engineers could access the experience of others was through the use of Group Technology. Group Technology began in the USSR in the fifties (Mitrofanov (2)) and was aggressively developed through the 60's and 70's in Europe. The GT

process impacted part design through the creation of part families, (Opitz,(3)), production techniques through part family work plans and plant organization through the use of PFA to create work cells, (Burbidge, (4)). Many of the issues that confront the developer of modern design aids were already considered in this context. The grouping of parts was a major consideration and the emphasis here moved over the years from functional engineering descriptions (shaft, disk, belt etc) to visual form with (later) detailed coding systems and finally to geometric features, (eg taper, thread, bearing surface etc). The geometric features are then added together in a similar but more rigorous and flexible manner than the original composite components of early workers. The use of features in the design area allowed in turn the creation of programming languages to manufacture these parts more easily. In essence the CAM program now became a macro language where the dimensional details were passed to the program that then borrowed the tool sequences and motions from best practice allowing novice programmers to produce credible results.

The essence of GT, (from a design perspective) is the reuse of design and manufacturing expertise. The experience was presented in an easy to understand, high level, abstracted manner. A GT approach provides less experienced engineers the opportunity to create “new” products and saves experienced designers time while providing a high degree of confidence in the result.

The area of software engineering has many of the same basic intellectual issues as design and manufacturing in other disciplines. Not surprisingly then a movement parallel to GT started in that area and these days it is considerably better and more widely known. The idea of Object Oriented programming began with Simula and Simula 67 in the 60’s and progressed through several well known languages (the list is long but Small Talk (in the 70’s), C++ and Eiffel (in the 80’s) and Java (in the 90’s) provide a good cross section of the efforts that have been made over the last 40 years.

The two basic tenants of Object Oriented programming are termed encapsulation and inheritance; they have exact equivalents in the ideas of group technology. Encapsulation suggests that objects hide their variable values and present a simple high-level interface to the user who does not need to understand the internal mechanisms. (Note for the curious, of course one can add methods to the class that allow the passing of such detailed internals if one wishes too). The idea of inheritance is to allow sub classes to inherit the methods of higher level classes, (as well as adding their own new methods where needed). So what results is a system which allows one to benefit from code written by other experts and at the same time be presented with a high level interface to the process that avoids the consideration of the detailed methods required to achieve the end result.

It is curious perhaps that relatively little work has progressed along the idea of using these new OO languages to reinforce the ideas of GT in design and manufacture. Notable attempts to combine an Object Oriented approach with the design process have been suggested by Zhang et al ((5),(6)) and Ma et al (7). At the same time several researchers have attempted to use Object Oriented techniques in the development of design environments for specialized areas of application (8), (9). The authors in this paper are in the process of trying to ascertain the effectiveness of the use of such systems in the development of intelligent design catalogs to be used in the later stages of design, perhaps in the training of designers and (even more hopefully!) as an aid within the conceptual design phases.

The proposed design environment will be comprehensive and new versions of the environment will build on a common base framework; the base framework in turn is based upon an extensible programming language that facilitates an OO approach. The overall framework proposed is based on previous work, (Ardekani et al (10)), in which a high-level object oriented virtual environment was developed to allow the automated configuration, dynamic reconfiguration and system design of complex automation systems. The high level of abstraction within the framework allows programmers to operate at the level of the design requirements and avoids the need to have detailed technical mastery of the underlying electronic hardware and software. The system architecture then provides a link from the high level functional needs of the designer or programmer and the detailed software and hardware behavior of individual components without requiring the designer to be an expert. At the same time the user is relieved of a considerable amount of the effort to build systems (automated code reuse) and this in turn improves system reliability since one has standardized the methodology in line with that specified by the supplier of the low level component.

The environment for each high level system will typically reflect industrial practice and by extension the language developed to drive the system will employ the vocabulary of the user, as opposed to that of the component supplier. The possibilities for customization and extension are then available both at an industry sector level and at the company level where designers will wish to add specific company enhancements and features to the overall industry sector standard, Figure (1) indicates how this has been done within a general design framework for embedded systems. In Figure (1), the overall standard framework for automation is eCAF which supplies a range of generic classes and methods for motion, data acquisition, communication and plc tasks. ECAF is written in FORTH, which is the interpreter shown in Figure (1), this then allows the language to inherit all the vocabulary of that language. Since eCAF itself is extensible then it is expected that companies build their

own extensions as shown and since it has a well defined API then companies would normally build simple customized interface programs which simply send FORTH strings to eCAF from another application written in the language of their choice. To more clearly demonstrate the idea Appendix (1) shows the motion vocabulary provided within eCAF and describes the mode of use with a very simple example. A novice programmer will understand the intent of each statement

The creation of interactivity in a general manner is best achieved if components from a range of manufacturers are able to integrate automatically into the system described above. The technique used to achieve this is termed OPEN_CONFIG (11), which derives much from the pioneering ideas of Open Firmware (12). Each low level component carries it's own code that provides the detailed executable instructions to allow the

accomplishment of methods commanded from the high level (virtual) environment. To allow reconfiguration and the possibility of choosing optimal performance the architecture developed in the previous work also provides a binding table which allows the system to consider multiple choices in the provision of resources to achieve each of the high level methods. (The binding table simply allocates hardware components to functions in the virtual environment). Changes in allocation are made by reassignment in the binding table. The selection of a particular option in the binding table then results in the vectoring of high level instructions to execute code corresponding to different devices. The application of the system to a machine tool environment is shown in Figure (2).

Figure (1) Opportunities for Extension and Customisation (10)

3.0 Application to the Design Environment

The conceptual model outlined above, serves as the framework for the proposed new design environment. A common Object Oriented, extensible language will be used to allow the creation of multiple custom design languages, (i.e. each will use the same base). The environment will always possess a common structure and will drive through a binding table to physical components. The physical components will in many cases be common, however the class structure and the vocabulary of methods used to design will vary with the industry sector that is being considered. The hardware-independent master control system is replaced with the form-independent conceptual design environment. The virtual automation environment allows for dynamic reconfiguration; the virtual design catalogue environment should allow for the change of particular components while leaving the rest of the design

unaffected. As the automation domain accommodates process constraints, the catalogue domain incorporates design constraints, such as cost, weight or safety factors. Vendor neutrality is similar in the two domains: both virtual systems can accept physical devices from a wide range of suppliers.

In the design domain, the abstraction capability is an equally essential feature of the virtual environment. First, abstraction opens the door for the analysis of design. Every design created has the potential to be unique. Analysis, however, demands that the design fit a pre-determined template as defined by the analysis tools available. Standard features must therefore be extracted from the unique design that, once identified, can be aligned with the appropriate analysis tools (e.g., mathematical equations). Within a unique conveyor system, for example, there exists an abstract element "shaft" which can be analyzed for bending and torque tolerances.

Figure (2) A High level OO Framework that Receives Code from Components to Facilitate Configuration, (11)

The abstraction capability can also provide aid in the very early stages of design. By enforcing the use of consistent language in class, method and variable names then the class structure also provides a data base to examine extremely abstract concepts such as searching by function (method); it will also allow browsing (graphically) by shape, size and material. For those who obtain inspiration through observation and search for similar (better or cheaper) elements the system should prove to be a very viable tool.

4.0 A Brief Overview of the Proposed System

The system described is in the first stages of development. The idea is that the overall system has links to existing engineering software systems that allow visualization, drawing, manufacturing, simulation and analysis. This infers that there must be mechanisms (software) in place to parse and translate data between appropriate formats. One wishes ideally to acquire an advantage through providing detailed domain specific knowledge about objects. The author's propose that this be done by having a design catalog which contains code

as well as data. This will operate in the same fashion as the automation system example. When a designer selects a particular component the server will send instructions and data to the high level environment on the user machine. The instructions are in the form of "FORTH source" and are interpreted and executed in a transparent manner. The user can now draw any view of the component, apply loads to it, print specifications or perhaps simulate performance of a system that will contain the component.

To be specific one might imagine the problem of designing a simple positioning system. If the designer has made initial decisions to use a table on rolling element ways and to drive a ballscrew through a servomotor then the low level process will involve the comparison of components from a variety of manufacturers in terms of the overall system performance and cost. A high level language here would likely have classes, variables and methods to examine resolution, velocity, acceleration etc., each o these are dependent on several parameters as follows:

Figure (3) Simplified Flow of Code and Data from Catalog to Design Environment

Resolution:: encoder and ballscrew pitch

Maximum Acceleration:: Motor torque, ballscrew pitch, ballscrew buckling load, bearing axial loads. (Perhaps too, guideway stiffness and offset between ballscrew thrust and center of mass)

Maximum Velocity:: Motor speed, maximum ballnut speed, ballscrew pitch, ballscrew diameter (whirling), maximum guideway speed

There are other issues depending on how far one wishes to look. Accuracy for instance will bring one to consideration of additional parameters:

- Quality of screw
- Characteristics of Closed Loop controller
- System compensation
- Thermal errors
- Alignment
- Static and dynamic deflections

So what seemed to be a well defined problem will become a minefield for an inexperienced design engineer. If one is also required to move out from the base positioning table to provide the amplifier, power supply, operator interface etc or to look at alternate drive system configurations, then one starts to realize that even at this late stage there are very real challenges. A company (designing transfer lines for instance), would build a high level system that allowed the analysis of alternatives and directed the designer through the most likely directions, checking constraints

at each stage. The Object Oriented environment would have high level abstract methods such as Resolve Position: or Accelerate: that would have the generic code to calculate requirements. The specific code and data within the design catalog then inserts itself, through the binding table and as many individual components as required can be tested. One should note here that in a design environment, things are a little more complex than in a virtual machine environment! In the design environment one often needs to be able to vary requirements and examine cost and performance in a critical manner. This then means that a single high level method in the designer's vocabulary will map into the requirements for several methods at the low level. So the code inside the catalog is written to provide the pieces of the program and data needed by the higher-level method to examine the performance of the overall system. As an example the high level method Accelerate would require that the designer had selected a range of screws, motors and bearings. Each selection from the design catalog of a specific hardware component then loads the code and data to allow the required parameters to be generated and stored at the higher level for the design environment, (see Figure (3))

It is seen then that as long as an individual component can supply the code that parallels the high level method there is no need for the designer to understand the "internals" of the device and once a high level design language is available to direct the designer and query results the system should be simple to use. The authors believe that the system can be used to help in meaningful

ways during earlier design stages by providing designers with comparisons between various types of approach and at an even earlier stage giving them the opportunity to look for alternatives by browsing key words in catalogs. Figure (4) shows a typical portion of a class diagram for fasteners with methods and variables associated with each class which could be used for searching. Note that

generic methods for drawing and simulation would be inherited from the highest level classes (above those shown in Figure (4)).

Figure (4) A Simple Example Showing Classes, *Methods*, and both *geoetric* and *technological* variables

5.0 Concluding Remarks

Computer technology has provided engineers with many tools to help them carry out their design activities with greater ease. Unfortunately, the majority of these tools

are aimed at those activities associated with the latter, more analytic, stages of the design process. There are relatively few aids specifically created to support the

earlier, more synthetic stages. The reason for the bias is that, whereas the latter stages of design are concerned with the quantifiable, the concrete, and the specific, early stages must focus on the conceptual, the abstract and the ill-defined. How can one develop computer applications capable of dealing with such uncertainty?

Object-oriented programming, with its tenets of inheritance and encapsulation, provides an opportunity to create just such an application. Inheritance allows for a hierarchical structure where one may travel between different levels of abstraction, from the very abstract (high level) to the very concrete (low level). Encapsulation means that extensive data can be linked to objects without necessarily imposing a thorough understanding on the part of the user. Novices stand to gain greatly by this feature.

The proposed system is aimed at using object-oriented technology to link product catalog data (the concrete) to a conceptual (abstract) design environment. The significance of bringing the two within the same environment means that functional descriptions can be translated to existing, well-defined components. It is hoped to build on these ideas to produce a system that will be of great benefit to experienced and novice designer alike.

6.0 References

- [1] Pahl G. and W. Beitz, Engineering Design: A Systematic Approach, trans. by Pomerans, Arnold and Ken Wallace, Springer-Verlag, London, 1988
- [2] Mitrofanov, S.P., Scientific Principles of Group Technology, English Translation, National Library for Science and Technology, Washington, D.C., 1966.
- [3] Opitz, H., A Classification to Describe Workpieces, Pergamon Press, Oxford, 1970.
- [4] Burbidge, J. L., (1989) Production Flow Analysis for Planning Group Technology. Oxford University Press. 1989
- [5] Zhang, W.Y., S.B. Tor and G.A. Britton, Automated Functional Design of Engineering Systems. Journal of Intelligent Manufacturing, Vol. 13, pp. 119-133, 2002.
- [6] Zhang, W.Y., S.B. Tor, G.A. Britton and Y.-M. Deng. EFDEX: A Knowledge-Based Expert System for Functional Design of Engineering Systems, Engineering with Computers, Vol. 17, pp. 339-353, 2001.
- [7] Ma, Yong-Sheng, Shu Beng Tor and Graeme A. Britton. The Development of a Standard Component Library for Plastic Injection Mould Design Using an Object-Oriented Approach, International of Advanced Manufacturing Technology, Vol. 22, pp. 611-618, 2003.
- [8] Speck, Andreas Reusable Industrial Control Systems. IEEE Transactions on Industrial Electronics, Vol. 50, No. 3, pp. 412-418, 2003.
- [9] Zhang, W., C.W. Chuen, C.M. Wong, and L.H. Lam, Feature Representation and Database Schema of an

Object-Oriented Feature-Based Product Model for Hydraulic Manifold Blocks. The International Journal of Advanced Manufacturing Technology, Vol. 16, pp. 182-188, 2000

[10] Ardekani R., Oldknow K., and Yellowley I. The Design of a Flexible, Modular System Architecture for Monitoring and Control, 1st CDEN/RCCI Conference, Montreal, 2004

[11] Oldknow, K.D. and Yellowley, I., "The Design, Implementation and Validation of a System for the Dynamic Reconfiguration of Open architecture Machine Tool Controls", Int. J. Mach. Tools and Manufacture, Vol 41, pp 795-908, 2001.

[12] Open Firmware organisation (IEEE1275) <http://playground.sun.com/1275/home.html>

[13] C. Duff and N. Iverson, "Forth Meets Smalltalk," Journal of Forth Application and Research, 2, 1, 1984

Appendix (1) Original Embedded System Framework. (11)

INTERPOLATOR Methods, (Coordinated Motion, Linear and Circular modes)

describe: version: resume: suspend: show: set_delay:
set_priority: get_priority: get_delay: get_status:
get_thread: lin: cw: ccw: set_feed: get_master_prd:
set_accel: get_accel: set_incremental set_absolute:
get_move_mode: set_distance_mm: set_distance_inch:

set_distance_rev: get_distance_unit: set_time_sec:
set_time_min: get_time_unit: set_angle_rad:
set_angle_deg: get_angle_unit: home:
put_radius: get_radius: put_direction: get_direction:
put_sangle: get_sangle: put_eangle:
get_eangle: dwell: abort: get_busy?:
get_moving?: feed_limit_on: feed_limit_off:
get_feed_limit:

AXIS Methods (basic single axis, control and kinematic settings)

describe: version: show: get_btable: Register:
set_mm/rev: get_mm/rev: set_BLU/rev: get_BLU/rev:
get_BLU/dist: get_min_distance: set_min_distance:
get_max_distance: set_max_distance:
get_max_velocity:
set_max_velocity: get_velocity: set_velocity:
get_ref_position: set_ref_position:
set_in_position_range:
get_in_position_range: set_radius_dist:
get_radius_BLU:
put: get: clear: check_status: get_ts: get_tm: set_filter:
reset: halt: enable: send_increment: read_position:

read_error: flush_buffer: get_buffer_stat:
get_buffer_count: zero_on_next_z: zero_in_progress?:
in_position?: get_pwm_output: set_pwm_output:
get_msratio: send_header: set_lcmode: get_lcmode:
get_lcfreq: set_lcfreq: get_seq_num:

The **2Axis-interpolation** class supplied with *the system* provides a set of methods to help specify the required motion as well as the algorithm to perform coarse first stage interpolation for 2 axis linear and circular motion profiles. Versions for 3, 4 and 5 axis interpolation as well as special cases such as a gantry axis are also available. Once the axis objects are instantiated they can be bound to the interpolator class. For example the two axes defined earlier are used to instantiate the interpolator object **intxy** as follows:

```
x_axis y_axis 2axis-interpolator intxy
```

This will create a new thread that will perform motion commands such as:

```
set_incremental: intxy  
set_distance_BLU: intxy  
set_time_sec: intxy  
30000e0 set_feed: intxy  
10000e0 20000e0 lin: intxy drop
```

The above commands execute an incremental vector move of 10000 BLU (basic length unit) along the x_axis and 20000 BLU along the y_axis with at a vector feed rate of 30000 BLU/sec.

The important thing to realize is that by using an OO language to build a high level virtual environment one can enhance the language using the base capabilities and classes of the language. The language used by Cameleon Controls in this example and proposed by the authors for the design development is FORTH, (it is often used where extensible, interpretive, multitasking environments are required). Object orientation is added to FORTH, which has always had such ideas through the CREATE/DOES construct. The model used derives from NEON a very early OO Forth implementation (13).