Simulation Driven Mass Customization:

by

Edward J. Ferara

B.E., Mechanical Engineering, 1999
The Cooper Union for the Advancement of Science and Art

Submitted to the Department of Mechanical
Engineering in Partial Fulfillment of the
Requirements for the Degree of Master of Science

at the

Massachusetts Institute of Technology.

June 2001

Signature of Author ....
Department of Mechanical Engineering
May 11, 2001

Certified by ....
David Wallace
Esther and Harold E. Edgerton Associate Professor of Mechanical Engineering
Thesis Supervisor

Accepted by ....
Ain A. Sonin
Chairman, Department Committee on Graduate Students

Simulation Driven Mass Customization:

by

Edward J. Ferara

Submitted to the Department of Mechanical Engineering on May 11, 2001 in Partial Fulfillment of the Requirements for the Degree of Master of Science

ABSTRACT

A framework using product design simulations as the basis for product configurators is presented. Configurators are intended to enable the production of highly mass customized products by providing customers with selected design capabilities. It is proposed that reusing design simulations as the logic for configurators will result in configurators that are not only less costly to construct and maintain, but also *provide customers with the power to design.*

A richer form of mass customization, known as *continuous mass customization,* is proposed. Rather than choosing from predetermined lists of options, customers have the ability to make continuous changes to a product's configuration. The product design simulations used in the configurator also provide the customer with real-time performance predictions of the current configuration.

A product configurator for a road bicycle was implemented as an internal pilot project to demonstrate the proposed concepts. The Distributed Object-based Modeling Environment (DOME) was used in the construction of an integrated product model. A website using Java Server Pages was constructed to provide web access to the product model.

An elementary model backsolver was implemented. The model backsolver allows the user to adjust both the outputs (performance characteristics) as well as the inputs (design variables) of a causal model developed for use by product designers. Therefore, customers can interact with the same product model by adjusting the performance characteristics with which they may be more familiar.. The model backsolver searches for the set of inputs yielding outputs that best match the user's desired values. The model backsolver utilized a struggle genetic algorithm optimization (GAO) and preference functions based on acceptability theory.

Thesis Supervisor: David Wallace
Title: Esther and Harold E. Edgerton Associate Professor of Mechanical Engineering

TABLE OF CONTENTS

Herbert: Every day we're losing ground to the Japanese and I want to know why.
Advisor: Oh, unfair trade practices?
Advisor: Mushy-headed one-worlders in Washington?
Advisor: Some sort of gypsy curse?
- The Simpsons

INTRODUCTION

In recent years customers have come to expect custom-tailored, or mass customized products. Mass customization is characterized by dynamic product change and stable process change (Boynton *et al.*, 1993). The traditional product development process of using marketing research to make educated design decisions may be too slow to accommodate mass customized products. While other researchers have sought to make the product design process more responsive to customers by bringing customers and their needs to the designers (Loosschilder, 1988), the philosophy behind this research has been to attempt to bring the power of design to the customer.

In this research, tools have been developed to give customers access to integrated product design simulations through custom web-based interfaces. It is through these interfaces that customers are able to make parametric changes to the product's configuration. The design simulations then provide the customers with real-time performance predictions. It is proposed that this will lead to a new form of mass customization, which we refer to as *continuous mass customization*. Rather than customizing products through discrete changes from predetermined options, customers could make continuous parametric changes to the product's configuration.

Current configuration systems are costly to construct and maintain. The fundamental problem is that the product and its components must be remodeled in the language of the configurator after they have been designed. This is often accomplished with rules that must be explicitly encoded. In addition, the rules must be continuously updated as the components are changed. It is thought that the reuse of design models as the basis for product configurators will reduce this product development bottleneck. The goal of this research was to not only develop technologies that accomplish the previous concepts, but to also identify theoretical issues that would not be obvious unless an implementation was attempted.

"I want a horn here, here, and here. You can never find a horn when you're mad. And they should all play 'La Cucaracha!"

"What about those red balls they have on car aerials so you can spot your car in a park. I think all cars should have them!"

- Homer Simpson

MASS CUSTOMIZATION

Mass customization is perhaps best conceived of as the ability to serve a wide range of customers and meet changing product demands by offering one-of-a-kind custom products and services based upon customer specifications. By tailoring products to the needs of individual customers, companies have used mass customization as a way to differentiate their products and services from competitors.

Mass customization has been extensively applied to services, such as banking, brokering, and Internet portals (*The Economist*, 2000). More recently, physical products that are highly modularized, such as personal computers and telecommunications systems, have become increasingly mass customized. The most prominent firm in the field of mass customization has been Dell Computer Corporation. The Dell website (http:*//www.dell.com)* offers customers the ability to specify the components of and options within their personal computer. As users change the configuration of their computer, they are provided with feedback in the form of a price quote generated by a simple cost model.

Mass customization has led to a business model that exists as an alternative to large-volume mass production. "In Henry Ford's day, Ford made the car and the customer paid for it. In Michael Dell's day, the customer pays for the computer and then Dell makes it." (*The Economist*, 2000)

> Instead of building a single-product, large-volume focused production process, the mass customizer builds a dynamic network of potentially infinite numbers of interchangeable and intercompatible individual unit production processes. Thus, the challenge of alignment in the dynamic network environment of the mass-customization design is to make the unpredictable combinations of processing units function both seamlessly and efficiently. (Boynton *et al.*, 1993)

A company that engages in mass customization is able to gain a competitive advantage by offering value in the form of individually customized products, and this results in the establishment of a close relationship with their customers. The competitive advantage mentioned above is very valuable to a company whose products are facing commodification, as this provides a mechanism to differentiate their products. However, as an organization allows customers to customize their products, they are unintentionally allowing customers to influence the strategic course of the company (Boynton *et al.*, 1993).

If something is too hard, give it up. The moral my boy is to never try anything.

- Homer Simpson

CONFIGURATION DESIGN

The most popular form of design for mass customization today is configuration design. Configuration design is the simplest form of design and is based upon creating a system from a fixed, *pre-defined* set of components (Mittal and Frayman, 1989). David Brown describes it as follows:

Configuring = Selecting + Associating + Evaluating

where:

Selecting	**= Choosing components**
Associating	**= Establishing relationships between components**
Evaluating	**= Compatibility testing + Goal satisfaction testing**

(Brown, 1998)

The first step, *selecting*, requires users to understand what the components are before they begin the configuration process. The process of selecting components to obtain a desired performance requires someone that is knowledgeable in the domain. Companies, such as Lucent, have addressed this problem by designing their configurators to be used by knowledgeable salespeople (Ambler, 1998).

According to Mittal and Frayman's (1989) definition of configuration design, components cannot be designed during the configuration process. This leads to a combinatorial configuration process that resembles the assembling of a puzzle as show in Figure 1.

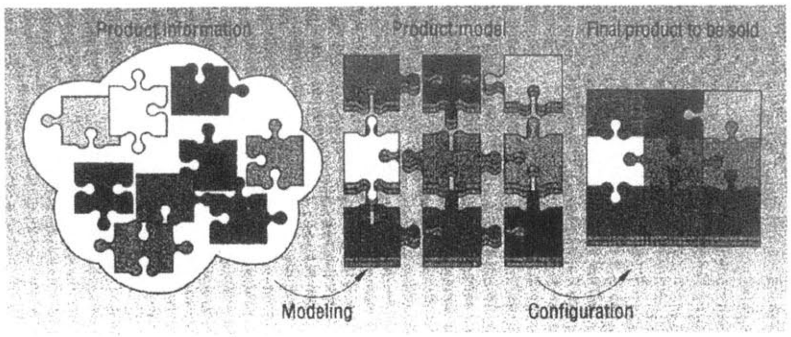


Figure 1: Configuration Design Process (Yu, 1998)

There has been a great deal of previous research in the field of configuration design and artificial intelligence has yielded a number of specific approaches to solving configuration design problems. These approaches included *rule-based* and *model-based* reasoning, which are discussed below.

Rule-based Reasoning

Ruled-based systems, otherwise known as expert systems, use production rules as a mechanism for representing both domain knowledge and control strategy (Sabin *et. al.*, 1998). These rules are constructed by ***if*** *condition* ***then*** *consequence* rules. For example, *if* Component A is selected, *then* Component B must also be selected. Actions are checked versus these rules, and if the conditions are met, the consequences act upon global state information. Rule-based reasoning systems create enormous maintenance problems as components are modified and introduced to the component library. For example, in 1989 Digital's XCON configuration system had more than 31,000 components and approximately 17,500 rules, which changed at an annual rate of approximately 40% (Sabin *et. al.*, 1998).

The explicit encoding of a component's configuration information introduces the product development bottleneck described in Figure 2. Lucent Technologies has used configurators to manage the sale and manufacture of products for over 20 years. Consequently, Lucent experiences a product development bottleneck because engineering specifications must be translated into a format understood by the configurator (Ambler, 1998).

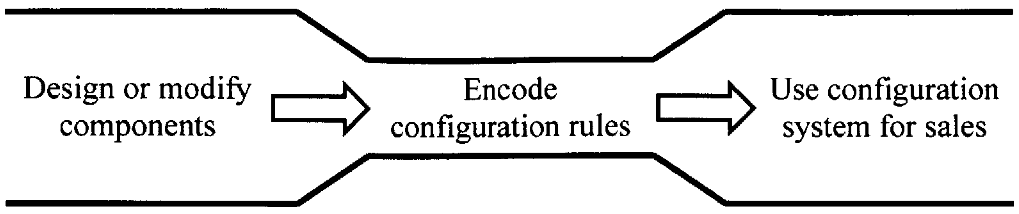

Design or modify components
Encode configuration rules
Use configuration system for sales

Figure 2: Product development bottleneck caused by configuration systems.

Model-based Reasoning

Model-based reasoning addresses the limitations of rule-based reasoning, especially the maintenance issues. This approach relies on a system model that is decomposed into entities and the interactions between them. Walter Hamscher expresses the advantages of model-based reasoning as the following:

- A better separation between what is known and how the knowledge is used.
- Enhanced robustness (increased ability to solve a broader range of problems).
- Enhanced compositionally (increased ability to combine knowledge from difference domains within a single model).
- Enhanced reusability (increased ability to use existing knowledge to solve related classes of problems.

(Sabin *et. al.*, 1998)

Some of the more prominent approaches to model-based reasoning are outlined in the following sections.

Resource-based

Resource-based configuration is best suited for configuration tasks in which a specific functionality is desired. The requirements of a solution are expressed in terms of the amount of resources demanded from, or supplied to, the system. The goal is to find a set of components that bring the overall set of resources to a balanced state, in which all demands are fulfilled. The process of configuration is as follows:

1. Start with a set of resources demanded by the requirement specifications.
2. Select one resource type that is not balanced.
3. Create a list of components that can supply the resource.
4. Select one component from the previous list and add it to the current configuration.

5. Repeat this process until all required resource amounts are supplied by the components.
6. If a dead-end is found, backtrack to the last decision point.

Constraint-based

The *constraint-based* approach views the configuration design task as essentially a constraint-satisfaction problem. The elements of the configuration task are modeled as components. Each component has one or more ports which specify how the components can be interconnected. The components and the ports are viewed as the variables of the constraint-satisfaction problem, while the constraints restrict the way components can be integrated into a solution. Although this method is thought to reduce some of the maintenance issues, it has the disadvantage of requiring the user to understand and specify requirements before the configuration process begins.

A major challenge in traditional knowledge-based configuration systems is the acquisition of the rules or constraints that form the logic of the configurator (Sabin *et. al.,* 1998). One of the fundamental difficulties in obtaining the logic for configuration systems it is often *implicitly* encoded during the design process. To be used in a product configurator, the logic of a design must be extracted and *explicitly* re-encoded. It is proposed that the reuse of existing design models as the logic of configuration systems will eliminate this problem.

It measures the pitch, the frequency, and the urgency of a baby's cry, and then tells whoever's around, in plain English, exactly what the baby's trying to say! Everything from "Change me" to "Turn off that damn Raffi record!

- Herbert Simpson

DOME

An integrated modeling environment, is essential for a richer form of product configuration because it can provide the customer with accurate real-time feedback on the performance characteristics of a product. The DOME (Distributed Object-based Modeling Environment) project is intended to address the widespread need to predict the characteristics of large complex products in a dynamic, rapidly changing environment. DOME allows heterogeneous models, such as geometric CAD models, engineering simulations, and cost models that are geographically distributed amongst a company's internal divisions and suppliers to be integrated, via the Internet, to form an integrated system model.

As the margin for improvement within the individual columns of product development has continued to narrow, system engineering has become an increasingly critical facet of the design and production processes. Over the past several years, however, numerous system integration efforts within firms have been attempted with limited success. These endeavors often fail to meet expectations due to the difficulty of creating an explicit model for a very large system involving many suppliers, a rapidly changing product, and an evolving organizational environment (Cooper *et. al.*, 1998). Through DOME, product development participants and organizations are able to publish their core competencies in the form of models, such as geometric, CAE, manufacturing, or marketing models that are available as live services over the Internet. They are able to do this in a effortless manner, without intermediary assistance from computer programmers or departing from their preferred set of product development tools. These service interfaces are made accessible on the Internet through a network of DOME servers, forming a service marketplace. Product developers, small or large, can subscribe to and flexibly inter-relate these services to build 'integrated' system models that allow the prediction and analysis of a larger sub-system's performance. In turn, other product developers might be able to define relationships with the services of subsystems. The result is a distributed network of service exchange relationships from which the overall system model emerges.

Example: Ford movable glass system (MGS)

The following example, is based upon a DOME project for an automotive door moveable glass system, deployed at Ford Motor Company in the summer of 1999. This exampled derived from the work of Wallace and Abrahamson, and will further illustrate the product development service marketplace concept (Wallace *et al*, 1999) .

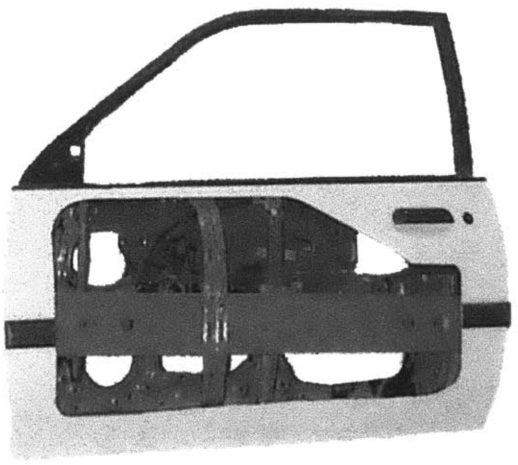


Figure 3: The components of a Ford Escort moveable glass system (MGS).

Figure 4 provides screen images of two different pre-existing models used by engineers and CAD designers at Ford. The Excel spreadsheet model requires a large number of door parameters to predict the glass velocity and stall force while the SDRC I-DEAS geometric model provides the overall physical configuration.

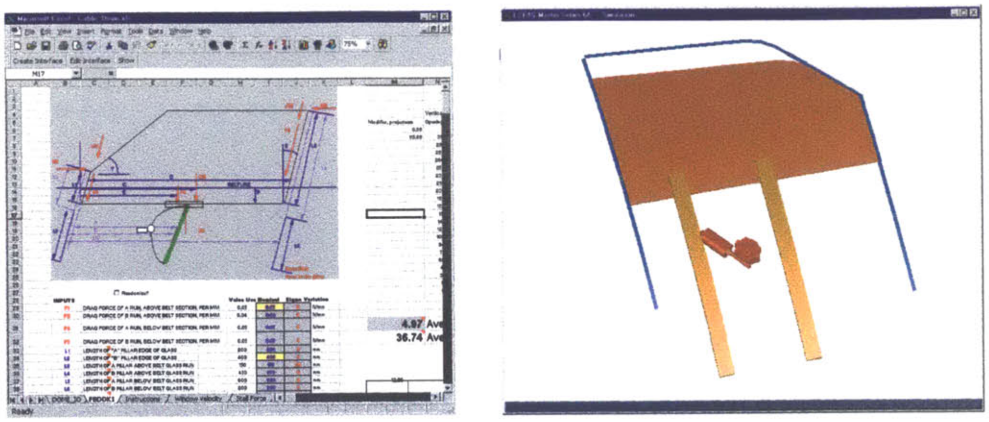
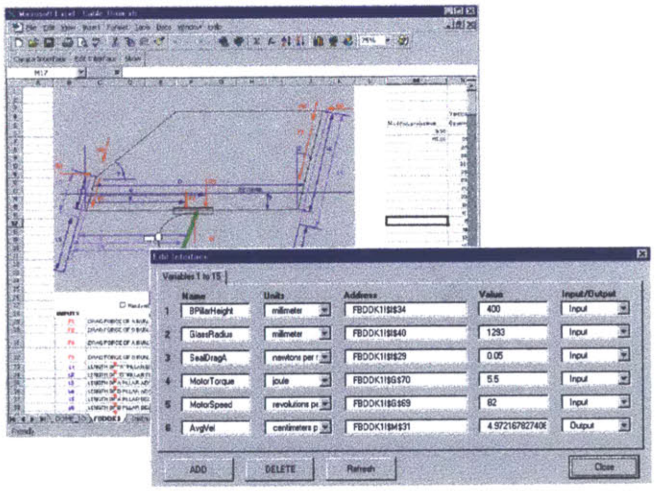


Figure 4: Existing models created and used by engineers and CAD designers for the MGS.

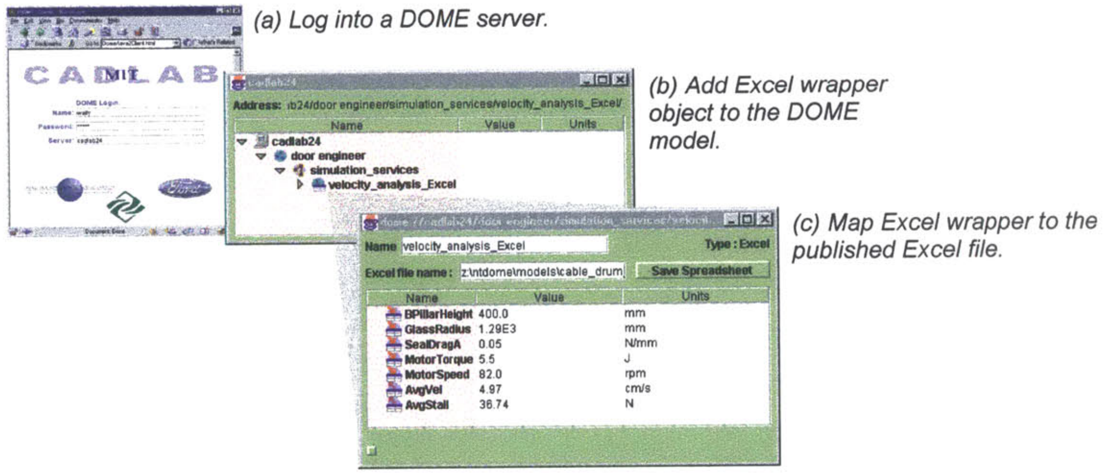
The first goal of DOME is to provide individuals with the ability to conveniently and independently *publish* interactive services so that they are widely accessible. Individuals with particular product development expertise need to be given the capability to create models or model components with services that can be accessed, understood, and manipulated by those who do not have additional skills outside of their traditional domain specific tools. Model owners use simple DOME publishing programs to define interfaces that will mediate how other users will interact with their models. A publisher is a standalone program or macro specific to a third party application. For example, in Excel, a wizard-like publishing macro is used to select cells and define model inputs and simulation outputs as shown in Figure 5.



	Name	Units	Address	Value	Input/Output
1	BPillarHeight	millimeter	FBDDK1I!$I$34	400	Input
2	GlassRadius	millimeter	FBDDK1I!$I$40	1293	Input
3	SealDragA	newtons per r	FBDDK1I!$I$29	0.05	Input
4	MotorTorque	joule	FBDDK1I!$G$70	5.5	Input
5	MotorSpeed	revolutions pe	FBDDK1I!$G$69	82	Input
6	AvgVel	centimeters p	FBDDK1I!$M$31	4.97216782740€	Output

Figure 5: A macro in Excel is used to define a service interface for spreadsheet models.

After inputs and outputs are defined, model owners use a web browser to log into a DOME server and use special *wrapper* objects to make their published services available over the Internet. A wrapper is an object written as a software plug-in to DOME for third party applications. It interprets the meta-data generated by a corresponding publisher to create a DOME object model that provides a front-end web-based service interface. The wrapper also manages the back-end communication between these DOME objects and data in the third party application. This is illustrated in Figure 6 which depicts an engineer logging into a DOME server and adding an Excel wrapper object. The interface for the Excel wrapper is open and it is being mapped to the published Excel model.


(a) Log into a DOME server.
CADLAB
MIT
DOME Login
Name:
Password:
Server: cadlab24
(b) Add Excel wrapper object to the DOME model.
(c) Map Excel wrapper to the published Excel file.
Save Spreadsheet

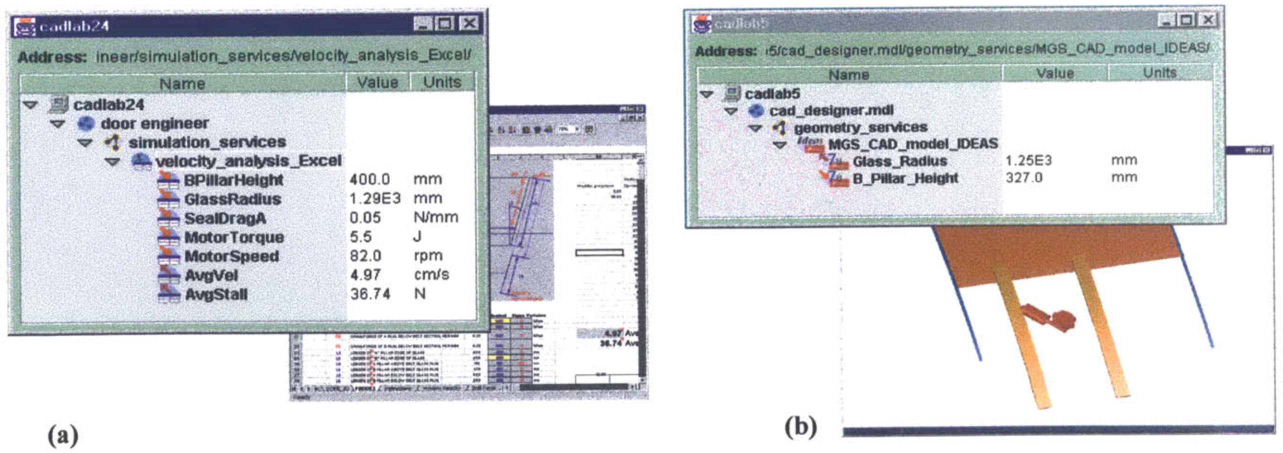
Address: ıb24/door engineer/simulation_services/velocity_analysis_Excel/

Name	Value	Units
cadlab24		
door engineer		
simulation_services		
velocity_analysis_Excel		

Name: velocity_analysis_Excel   Type : Excel
Excel file name : z:\ntdome\models\cable_drum

Name	Value	Units
BPillarHeight	400.0	mm
GlassRadius	1.29E3	mm
SealDragA	0.05	N/mm
MotorTorque	5.5	J
MotorSpeed	82.0	rpm
AvgVel	4.97	cm/s
AvgStall	36.74	N

Figure 6: Adding an Excel wrapper object within a DOME model.



Address: ineer/simulation_services/velocity_analysis_Excel/

Name	Value	Units
cadlab24		
door engineer		
simulation_services		
velocity_analysis_Excel		
BPillarHeight	400.0	mm
GlassRadius	1.29E3	mm
SealDragA	0.05	N/mm
MotorTorque	5.5	J
MotorSpeed	82.0	rpm
AvgVel	4.97	cm/s
AvgStall	36.74	N

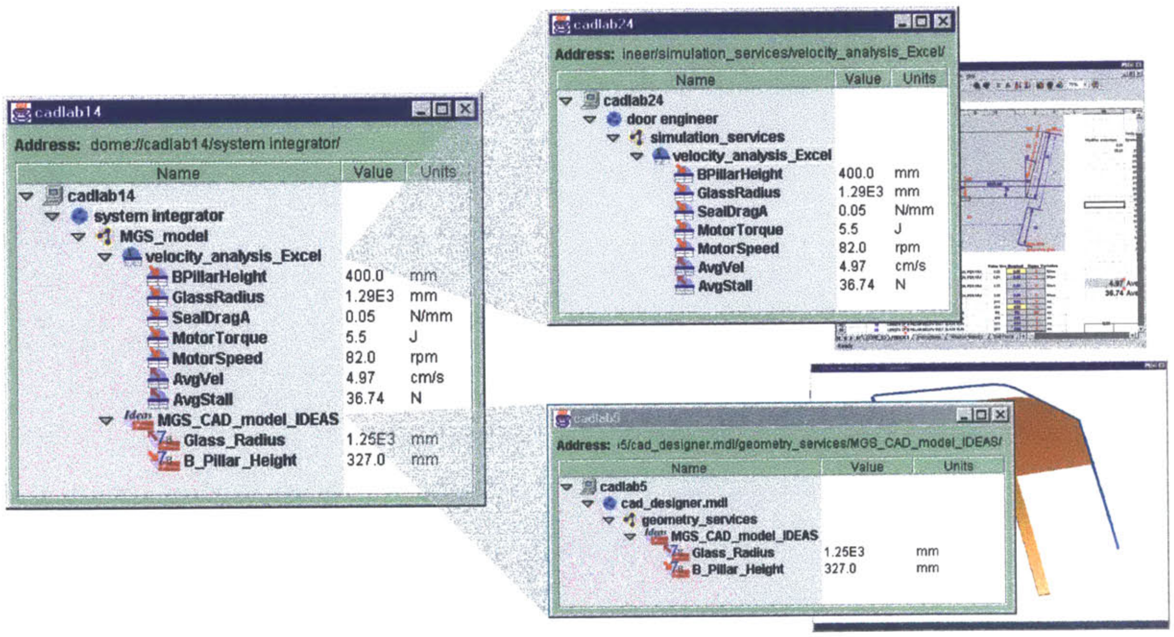
Address: ı5/cad_designer.mdl/geometry_services/MGS_CAD_model_IDEAS/

Name	Value	Units
cadlab5		
cad_designer.mdl		
geometry_services		
MGS_CAD_model_IDEAS		
Glass_Radius	1.25E3	mm
B_Pillar_Height	327.0	mm

Figure 7: Wrapper objects for Excel (a) mapped to the engineer's published spreadsheet and (b) mapped to the published CAD designer's MGS model.

Figure 7 shows two different DOME servers, one providing the door engineer's velocity analysis services (a), and the other providing the CAD designer's geometric modeling services (b). The two modeling services are on different DOME servers. Any design participant with permission to access this DOME server can now use this interface to change inputs to the engineer's model, which will drive the underlying Excel model on the engineer's computer and return corresponding predictive services for velocity and stall force.

The second goal of DOME is to create a mechanism for subscribing to published services and integrating them through relationships that build system models. In Figure 8, a system integrator has logged into yet another a DOME server and is beginning to define an overall MGS simulation model. The system integrator has searched the DOME service marketplace and subscribed to the services provided by the engineer and CAD designer by placing shortcut-like service references (subscriptions) within their own model.



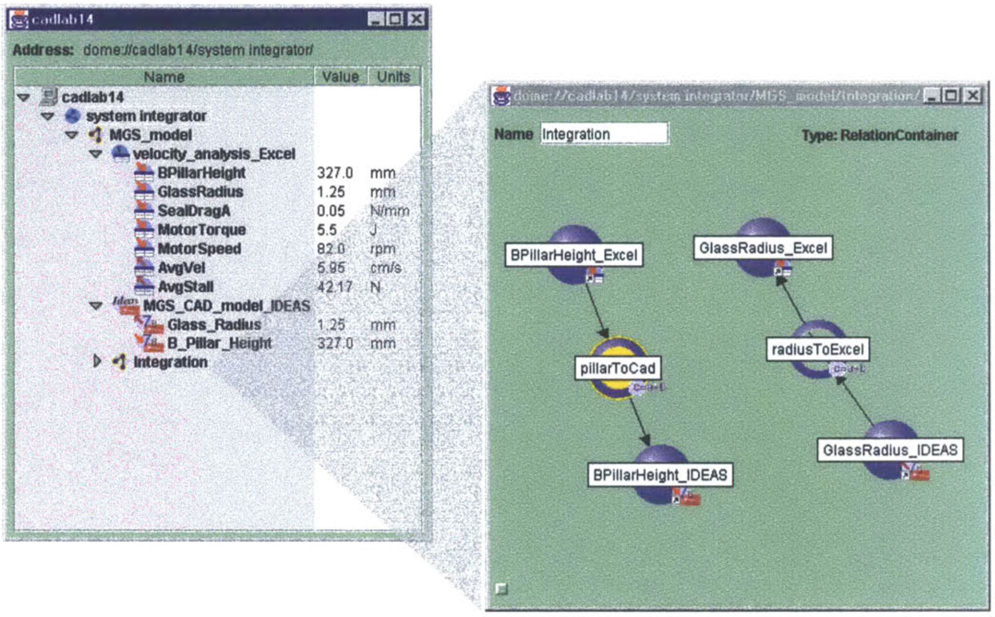
```
cadlab14
Address: dome://cadlab14/system integrator/
Name                                      Value    Units
▽ cadlab14
  ▽ system integrator
    ▽ MGS_model
      ▽ velocity_analysis_Excel
           BPillarHeight                  400.0    mm
           GlassRadius                    1.29E3   mm
           SealDragA                      0.05     N/mm
           MotorTorque                    5.5      J
           MotorSpeed                     82.0     rpm
           AvgVel                         4.97     cm/s
           AvgStall                       36.74    N
      ▽ MGS_CAD_model_IDEAS
           Glass_Radius                   1.25E3   mm
           B_Pillar_Height                327.0    mm
```

```
cadlab24
Address: ineer/simulation_services/velocity_analysis_Excel/
Name                                      Value    Units
▽ cadlab24
  ▽ door engineer
    ▽ simulation_services
      ▽ velocity_analysis_Excel
           BPillarHeight                  400.0    mm
           GlassRadius                    1.29E3   mm
           SealDragA                      0.05     N/mm
           MotorTorque                    5.5      J
           MotorSpeed                     82.0     rpm
           AvgVel                         4.97     cm/s
           AvgStall                       36.74    N
```

```
cadlab5
Address: 5/cad_designer.mdl/geometry_services/MGS_CAD_model_IDEAS/
Name                                      Value    Units
▽ cadlab5
  ▽ cad_designer.mdl
    ▽ geometry_services
      ▽ MGS_CAD_model_IDEAS
           Glass_Radius                   1.25E3   mm
           B_Pillar_Height                327.0    mm
```

Figure 8: A system integrator subscribes to the velocity analysis and MGS CAD model services by creating shortcut-like references within their own DOME model.

At this point, the system integrator wishes to relate the modeling services of the engineer and CAD designer. A relationship manager object is added to the MGS model, providing an object for relating existing services and creating new services as illustrated in Figure 9. It is important to note that the models remain on the computers of engineers and CAD designers', thus not requiring the integrator to own the software on which the models run. In addition, the models and all of the intellectual property they contain are never transferred between servers. Although DOME allows clients to swap and evaluate suppliers as if their products and services were commodities, it also allows the suppliers to retain all of their propriety models.


cadlab14

Address: dome://cadlab14/system integrator/

Name	Value	Units
cadlab14		
system integrator		
MGS_model		
velocity_analysis_Excel		
BPillarHeight	327.0	mm
GlassRadius	1.25	mm
SealDragA	0.05	N/mm
MotorTorque	5.5	J
MotorSpeed	82.0	rpm
AvgVel	5.95	cm/s
AvgStall	42.17	N
MGS_CAD_model_IDEAS		
Glass_Radius	1.25	mm
B_Pillar_Height	327.0	mm
Integration		

dome://cadlab14/system integrator/MGS_model/integration/

Name: Integration

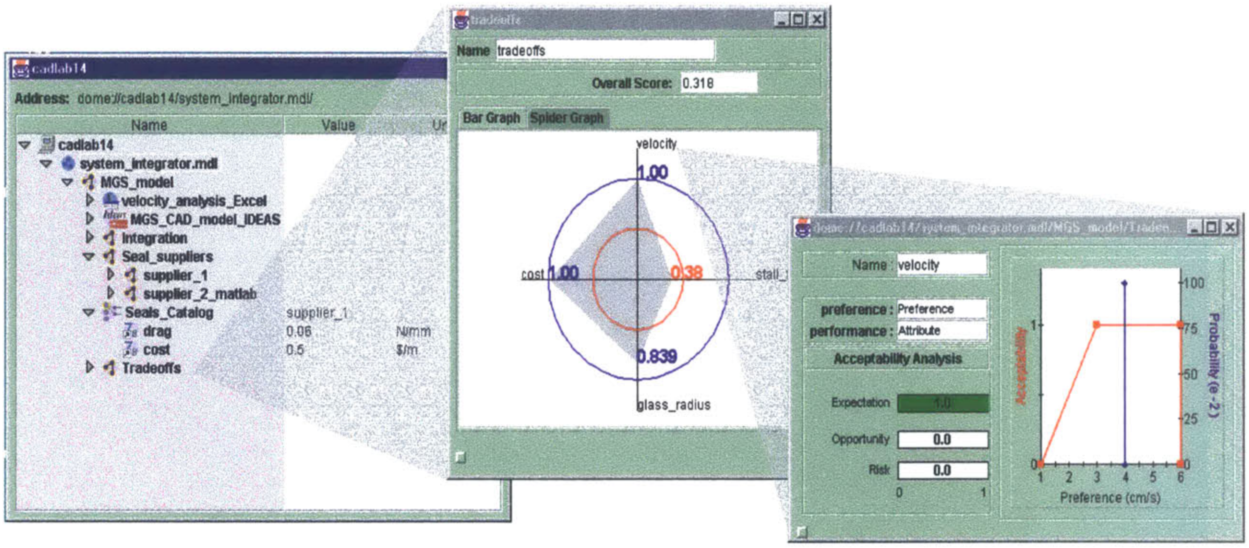
Type: RelationContainer

Figure 9: Adding an integration module and defining relationships between the models of the engineer and CAD designer to create an integrated system.

Simple equality relationships have been defined using a visual editor so that parameters in the Excel spreadsheet drive parameters in the MGS CAD model. This particular relationship manager resolves relations according to directed graph rules. Relations between services can be arbitrarily complex and different manager objects will offer different types of relation coordination behaviors.

The third goal of the concept is to introduce product development tools to support the integration and management of system models. For example, how does one understand the structure or behavior of the emergent product models? How may one select solutions once a large number of options become available?

Once a system model has been created, several special DOME objects can be added for purposes of analysis. In Figure 10, a decision support object has been added (Kim, 1999; Kim *et. al.,* 1997). This provides the integrator with a real-time view of system-wide tradeoffs as different participants make local design changes. A spider diagram shows performance assessments on four axes, while the expanded detail window shows the glass velocity prediction relative to its design specification. Additionally, an object has been developed using the Design Structure Matrix (DSM) to visualize service interactions as they evolve (Smith and Eppinger, 1997; Steward, 1981; Abrahamson *et al.*, 1999). A genetic optimization object has also been implemented to automatically search for models states that best meet design goals (Senin *et al.*, 1999a). A full set of papers arising from the DOME project may be found at http://cadlab.mit.edu.


cadlab14

Address: dome://cadlab14/system_integrator.mdl/

Name	Value	Ur
cadlab14		
system_integrator.mdl		
MGS_model		
velocity_analysis_Excel		
MGS_CAD_model_IDEAS		
Integration		
Seal_suppliers		
supplier_1		
supplier_2_matlab		
Seals_Catalog	supplier_1	
drag	0.06	N/mm
cost	0.5	$/m
Tradeoffs		

tradeoffs

Name: tradeoffs

Overall Score: 0.318

Name : velocity

preference : Preference

performance : Attribute

Acceptability Analysis

Expectation: 1.0

Opportunity: 0.0

Risk: 0.0

Figure 10: The services of a decision support object are added to visualize system-wide design tradeoffs.

Vision: A Product Development Service Marketplace

Development of the DOME concept has led to a new vision for product development. In abstract, when one subscribes to services in the DOME marketplace one is choosing resources that form the organization responsible for simulating and delivering the product. Thus, the product development organization becomes a loose affiliation of individuals or organizations sharing services driven by the needs of a particular product. Individuals have the capacity to develop complex products by building upon the services of other organizations. Many product development service capabilities will become commodities, much like many physical components are today. It will be possible to rapidly interchange equivalent design service providers so that the detailed development of the product and the definition of the product development organization will become part of the same process.

DOME's mechanisms for publishing, subscribing, and synthesizing relationships provide the underpinnings of a *service marketplace* for the producers and consumers of product development models and data. The marketplace facilitates the matching of producers and consumers, as well as the linking and augmentation of services to create new systems. Each participant in the marketplace brings expertise and formal representations in the form of data and models, ranging from an individual offering finite element analysis to an application engineer offering a catalog of electric motor simulations. System integrators are able to flexibly define and alter relationships between different services in the marketplace on-the-fly, without hard coding software connections between the applications of service providers.

The DOME is useful to product configuration because it allows product models to be flexibly and quickly integrated. The ability to construct integrated product models at a reasonable cost and in a distributed manner is essential to the feasibility of simulation-driven mass customization.

All my life, I have searched for a car that feels a certain way. Powerful like a gorilla, yet soft and yielding like a Nerf® ball. Now, at last, I have found it.
- Homer Simpson

CONTINUOUS MASS CUSTOMIZATION

Current configuration systems are developed after the product's components have been designed. If customers were given access to DOME-like system models, they could engage in their own tradeoff analysis while receiving accurate real-time performance predictions. This may lead to a new form of mass customization, known as *continuous mass customization,* that is much richer than today's combinatorial mass customization.

The limiting factor in mass customization is often thought to be the ability to manufacture a mass customized product once it has been designed. In the past, this problem has been addressed by intentionally modularizing power supplies and components, such as hard drives, and motherboards in personal computers. The individual components are mass-produced, while the final product is mass customized from the components. This leads to products that are not only constructed, but also designed, in manner that is not much different that playing with Lego® blocks (*The Economist*, 2000).

Perhaps the greatest challenge in mass customization is design products in amid a highly dynamic environment in which the components themselves are designed during the configuration process. This requires customers to interact with product models that are much more complex than the simple rules and databases of current configuration systems. By using actual product models to provide the logic of a configuration system, the product development bottleneck of encoding engineering specifications into the configurator's language is eliminated.

When companies employ configuration design, in which users select from a fixed set of components or options, they are aware of the possible combinatorial configurations that customers can produce. As the relationships of a configuration system are explicitly stated, the configuration system is limited to the vision of those who encode the configuration rules. In continuous mass customization, customers are empowered to develop products in a manner that the company may never have conceived of. Other research efforts, such as the elusive KIKon* framework, have referred to this type of configuration as *explorative design* (the * in KIKon* symbolizes exploration) (Rhamer *et al.*, 1998).

In the early 1980's, a parallel to continuous mass customization occurred in the integrated circuit (IC) industry. IC products were growing increasingly complex and the costs of correcting errors in custom designs were very high. LSI Logic was a relatively small venture competing against the much larger, Fujitsu. LSI Logic surprised Fujitsu when they released their proprietary software design tools to their customers. Although the tools were difficult to use in the beginning, they provided customers with the value of knowing that the IC's they designed were what they actually wanted (von Hippel, 1999).

It is not realistic to expect companies to give customers their proprietary design models. In addition, companies may want to control the set of parameters with which customers can interact. There are often valid reasons for this, such as retaining their brand image. For example, Bose would not want customers to design a speaker that did not meet their conception of the minimum audio quality requirements for a Bose product. In order to preserve their corporate integrity, organizations must be able to administrate the parameters that a customer can change. The architecture of DOME addresses both of these issues by allowing an organization to flexibly create and publish an interface to the model of their product. Customers can interact with the product's model through this interface without requiring the organization to reveal the underlying propriety model structure to the customer.

Advisor: What about a separate soundproof bubble-dome for the kids with optional restraints and muzzles?
Homer: Bull's-eye!

- Homer Simpson

DOME MODEL CONFIGURATOR

The use of product simulations in a configurator presents several design challenges. The term customer is very vague and could range from a automotive OEM engineer that is the customer of a part supplier, to the person that actually purchases a car. In addition, divisions within a large OEM may be customers of other divisions within the same company. To accommodate different types of customers, one of the first tools that's needs to be implemented is the ability to flexibly create a graphical user interfaces to DOME models.

Implementation

Several technologies were used to construct graphical interfaces to DOME models. While the following technologies are based on the Java programming language, other technologies could be used to provide similar functionality.

JavaBeans

The DOME Configurator Module allows a user to construct a graphical interface by choosing from a library of interface components and then associating these components with DOME services. The model configurator uses interface components that implement the JavaBeans™ standard. By implementing the JavaBeans™ standard, the Configurator Module can interact with an unknown interface component to discover its properties and methods. Therefore, any component that implements the JavaBeans™ standard can be acquired through the web and used in the Configurator Module. Once the components are combined into an interface, a customer can download and use it remotely it as a Java Applet.

Advantages

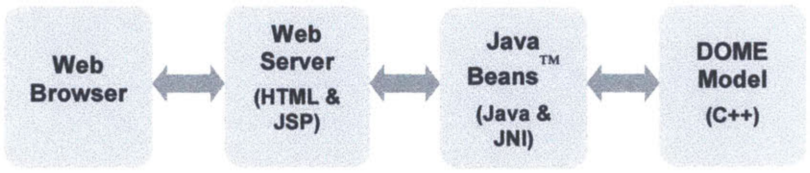
- Ability to use off-the-shelf interface components that can be highly customized.
- Interface components are relatively powerful.

Disadvantages

- Requires user to have special software installed. (Java Runtime Environment)
- The initial downloading of the applet can be slow.

Java Server Pages (JSP)

Java Server Pages technology allows the content of web pages to be dynamically generated using the Java programming language. JavaBeans™ were developed to allow Java Server Pages to interface with DOME models. The Java Native Interface was used to interface JavaBeans™ with the C++ kernel of DOME.



While the interfaces created with Java Server Pages are not as powerful as those created with Java Beans, they require very little programming experience and no knowledge of how DOME actually works. In addition, it is a widely used technology that requires no special software on the part of users.

Advantages

- Requires little to no programming experience.
- Only requires user to have a web browser installed.

Disadvantages

- Only simple interfaces can be created.
- Requires a web server that supports the JSP standard.

Herbert Simpson: All a man needs is an idea.
Bum: Then how come you're still a bum?
- The Simpsons

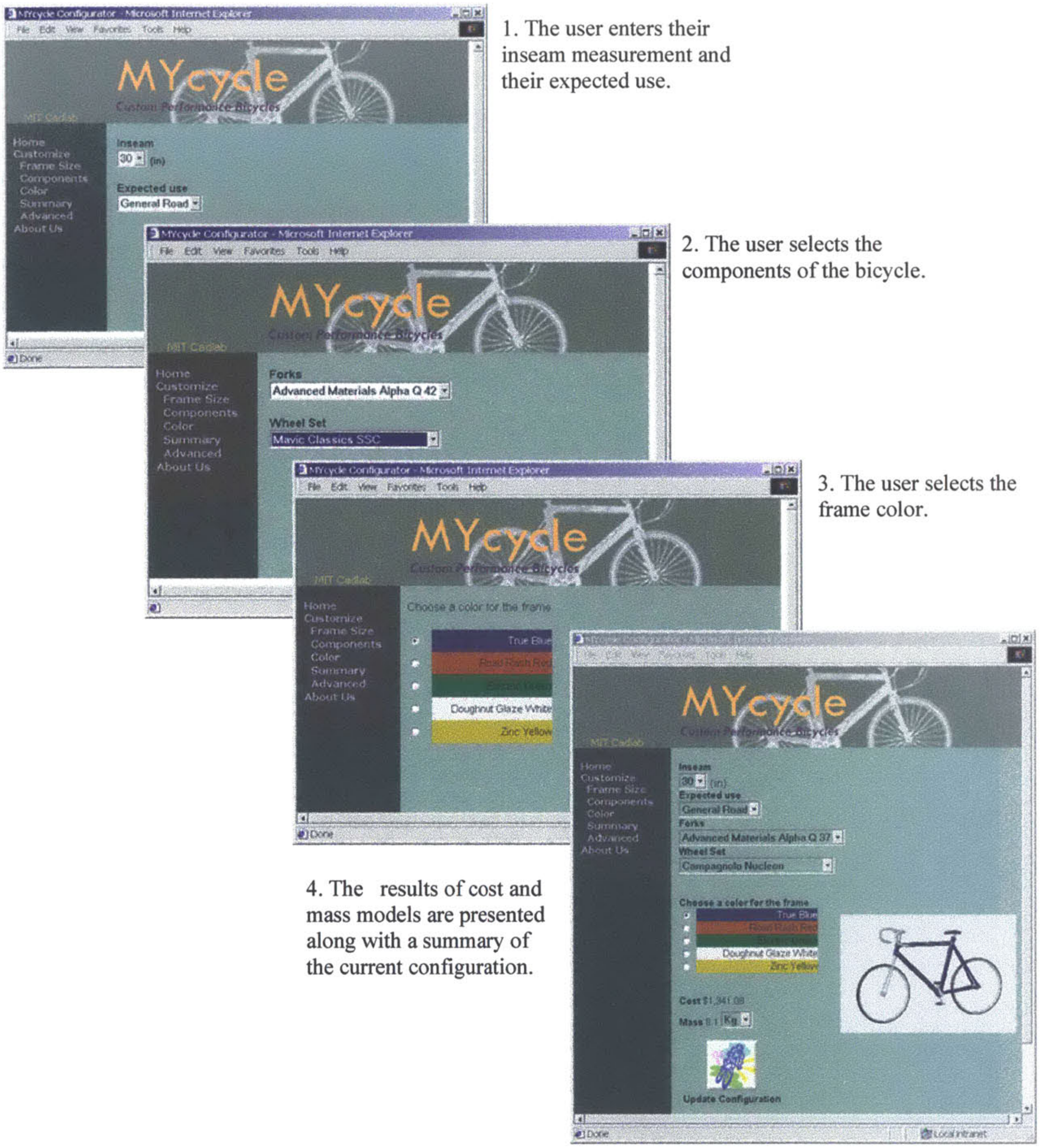
MYCYCLE EXAMPLE

To test the feasibility of a simulation-driven product configurator, an internal pilot project was prepared. A configurator for road bicycles under the fictitious company MYcycle was implemented. The customer is assumed to be a cycling enthusiast that is interested in purchasing a custom bicycle. The product choice and the anticipated sophistication of the user found in this example are irrelevant to the basic concepts of this research. The following is a list of some of the simulations used.

Model	Tool Used
Frame Geometry	SolidWorks
Cost of Frame Material	Excel
Power Required from Rider	Excel
Steering Stability	Excel
Wheelset Components	DOME Catalog
Fork Components	DOME Catalog

The simulations were integrated using the conventional DOME model-building tool. A website was then built and integrated to the DOME model using Java Server Pages. Two interfaces were constructed on the website. The first is referred to as the beginner configurator and represents the state of the art. The second is referred to as the advanced configurator and incorporates the concepts of continuous mass customization.

The beginner configurator (Figure 11) walks the user through a series of three questions. It collects information such as the expected use and anatomical dimensions of the user. This information is then mapped to tables that contain default frame sizes and other settings. This type of application is often mistakenly referred to as mass customization; however, the user has virtually no control over how their inputs are interpreted. The product is not *customized* by the user, but is rather *fitted* to the user.



1. The user enters their inseam measurement and their expected use.

Inseam: 30 (in)
Expected use: General Road

2. The user selects the components of the bicycle.

Forks: Advanced Materials Alpha Q 42
Wheel Set: Mavic Classics SSC

3. The user selects the frame color.

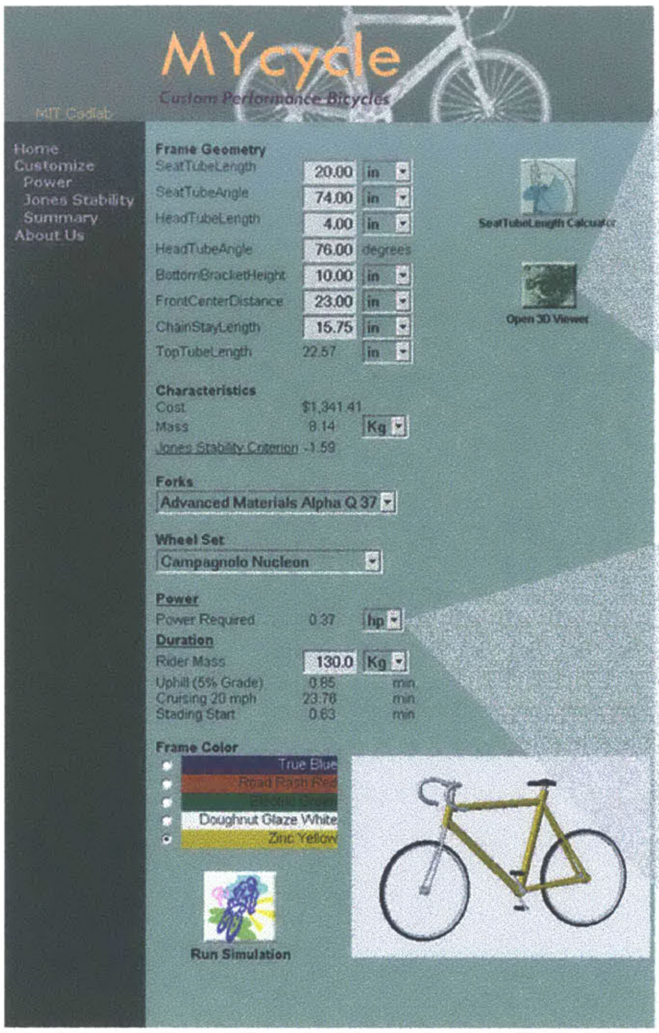
Choose a color for the frame
- True Blue
- Road Rash Red
- [illegible]
- Doughnut Glaze White
- Zinc Yellow

4. The results of cost and mass models are presented along with a summary of the current configuration.

Inseam: 30 (in)
Expected use: General Road
Forks: Advanced Materials Alpha Q 37
Wheel Set: Campagnolo Nucleon
Choose a color for the frame: [✓ True Blue]
Cost $1,341.08
Mass 8.1 Kg

Figure 11: The beginner configurator.

The advanced configurator (Figure 12) gives the user direct control over the bicycle's geometry and components. Unlike the beginner configurator, which only provides the user with the mass and cost of the current design, the advanced configurator provides the user with the results of steering stability and power simulations.



Frame Geometry

SeatTubeLength	20.00	in
SeatTubeAngle	74.00	in
HeadTubeLength	4.00	in
HeadTubeAngle	76.00	degrees
BottomBracketHeight	10.00	in
FrontCenterDistance	23.00	in
ChainStayLength	15.75	in
TopTubeLength	22.57	in

Characteristics

Cost	$1,341.41	
Mass	8.14	Kg
Jones Stability Criterion	-1.59	

Forks
Advanced Materials Alpha Q 37

Wheel Set
Campagnolo Nucleon

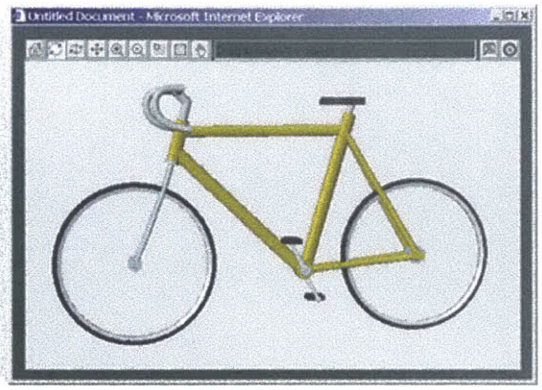
Power

Power Required	0.37	hp

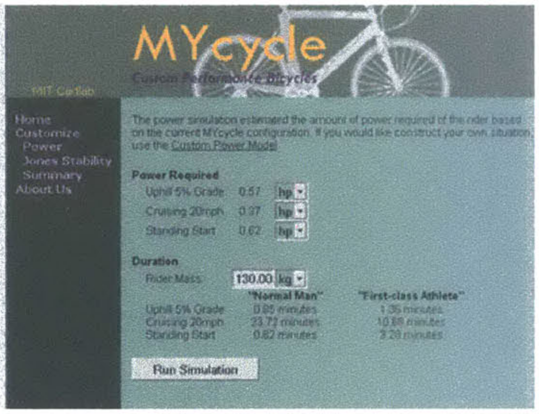
Duration

Rider Mass	130.0	Kg
Uphill (5% Grade)	0.85	min
Cruising 20 mph	23.78	min
Stading Start	0.63	min

Frame Color
True Blue
Road Rash Red
Electric Green
Doughnut Glaze White
● Zinc Yellow



The RealityWave viewer allows the user to examine a three-dimensional model of the current configuration.



The power simulation estimated the amount of power required of the rider based on the current MYcycle configuration. If you would like construct your own situation, use the Custom Power Model

Power Required

Uphill 5% Grade	0.57	hp
Cruising 20mph	0.37	hp
Standing Start	0.62	hp

Duration

Rider Mass 130.00 kg

	"Normal Man"	"First-class Athlete"
Uphill 5% Grade	0.85 minutes	1.35 minutes
Cruising 20mph	23.72 minutes	10.88 minutes
Standing Start	0.82 minutes	3.28 minutes

The power simulation estimates the amount of power required from the user in different situations.

Figure 12: The advanced configurator.

The maximum latency for the system is approximately 10 seconds. Most of the latency, however, is in the rebuilding of the CAD model. The current implementation of DOME has many flaws that also contribute to the latency. The kernel of the current implementation of DOME is single-threaded. Therefore the simulations can not be run in parallel. In addition, the current implementation is unable to solve the causal structure of the model to efficiently determine when to execute the simulations. As a result, many simulations are executed more times than needed. The implementation of a DOME system that automatically solves the causal structure of the model and efficiently executes simulations will address this limitation.

Herbert: Hey Homer, how's your car coming?
Homer: Oh, fine. They were putting in an onboard something-or-other and rack-and-peanut steering.

- The Simpsons

MODEL BACKSOLVING

As discussed earlier, it is not realistic to give customers access to design models and to expect them the to understand the inputs of the models and how they effect the outputs. In practical terms, a customer may not understand how the amount of memory in their computer affects the cost and the performance of a computer, even if they do understand that they want a cheaper computer. The model backsolver encapsulates a DOME model and allow users to flexibly change independent and dependant parameters. The marketing demands of translating engineering tradeoff decisions into language that the customers can understand are reduced by allowing customers to interact with performance variables.

Current DOME models typically require a model administrator to publish a rigid input/output interface. A rigid model requires the model administrator to strictly define the inputs and outputs of the model when a model is published, and therefore only allows a client to change the inputs of the model. This requires the model administrator to anticipate which inputs a user will be able to provide and will want to access. By using the model backsolver, a model administrator could publish a variational model without changing the underlying simulation. A variational model allows a user to change both independent and dependent parameters of a model. In addition, product models could be reused without having to rewrite the model for a new interface.

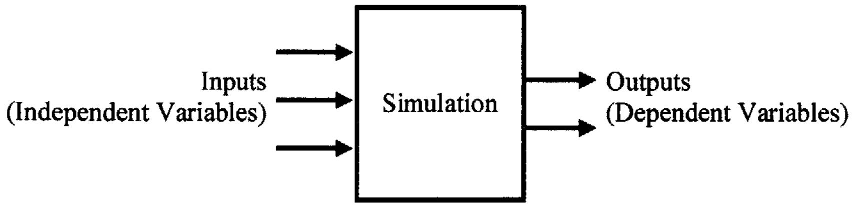


Figure 13: Rigid Model: User can only change inputs.

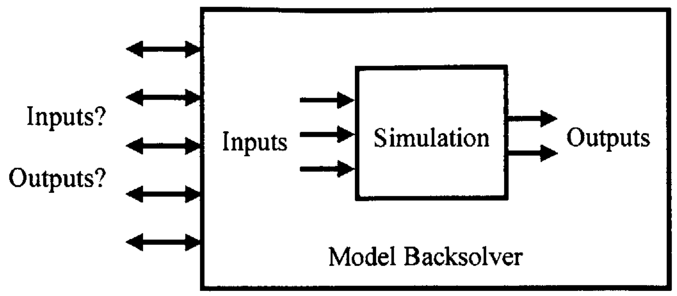


Figure 14: Variational Model: User can change both inputs and outputs.

The model backsolver reduces the publishing requirements of model administrators by allowing the users of the models to flexibly interact with the model. As a result, the model administrator does not have to anticipate which parameters a user will be able to provide.

The basic premise of this implementation of the model backsolver is to use a search technique to find a set of inputs to a model that will result in the desired output. When a desired value for an output is specified, there may be a large set of inputs that result in this output. For instance, if a box with a volume of 2.0 m^3 is desired, there is an infinite set of lengths, widths and heights that result in a box of this size. Given this infinite set of possible solutions, a decision must be made on behalf of the user as to which solution, or solutions, the user would be most interested in. In this implementation, this decision is made using acceptability theory. Acceptability theory is used to assign preferences to the parameters of the interface. The search technique then optimizes the aggregation of the preference functions.

Search Techniques

The model backsolver can be incorporated with any search technique. In this implementation, however, a struggle genetic algorithm (GA) was used (Senin, N. *et. al.,* 1999). Each interface parameter has a preference function associated with it. The derivation of the preference function is described in the section below. The search algorithm looks for the set of inputs which maximizes the aggregate of the preference functions. While any type of aggregation could be used, in this implementation a summation aggregator was employed.

Preference Functions

Acceptability theory was used because each preference function is constructed on an absolute scale. Therefore, there is no need for weighting functions in multiattribute decision support. Although this approach has more restrictive assumptions than utility theory, it was chosen because it is easier to use and requires less input from the user (Kim *et. al.*, 1997).

The following sections details different methods of constructing the preference functions. Each method varies in the level of sophistication expected from the users.

Direct Specification of Preference Functions

In this scenario, the user must explicitly specify the preference function for each parameter of the model. This requires the user to not only understand what each parameter is, but also to have considerable knowledge about how a preference function works. While direct specification allows the user the greatest control over how the backsolver works, it also demands the greatest level of sophistication. The specification of preference functions by customers has been proposed by other researchers (Tseng *et al.* 1996). It is doubtful, however, that consumers will have the required level of sophistication.

Parametrically Generated Preference Functions

In this method, the user specifies sensitivity parameters from which a preference function is generated. The sensitivity parameter can refer to their willingness to allow the value to increase or decrease. The preference functions are always generated with an acceptability of 1.0 at the desired value. If the user has not specified a desired value, the previous value is used, the rationale being that if the user does not enter a desired value because the user may find the previous value to be acceptable and therefore has no reason to change it.

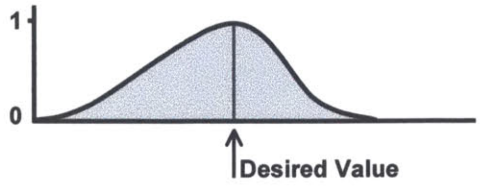


Figure 15: Preference function generated with a greater sensitivity to increase than to decrease.

In the implementation of the backsolver, the preference functions can be generated using either the logistic equation or an exponential decay. The user supplies a sensitivity parameter that is a constant for the given equation. The constant affects the shape of the curve. Since the sensitivies are different for different equations, they should perhaps be normalized between zero and one. This however would require the normalized value to be mapped to a bounded range. If the sensitivity for a value to increase and the sensitivity to decrease were equal, the preference function would be symmetrical and require only one sensitivity value from the user.

Box Design Example

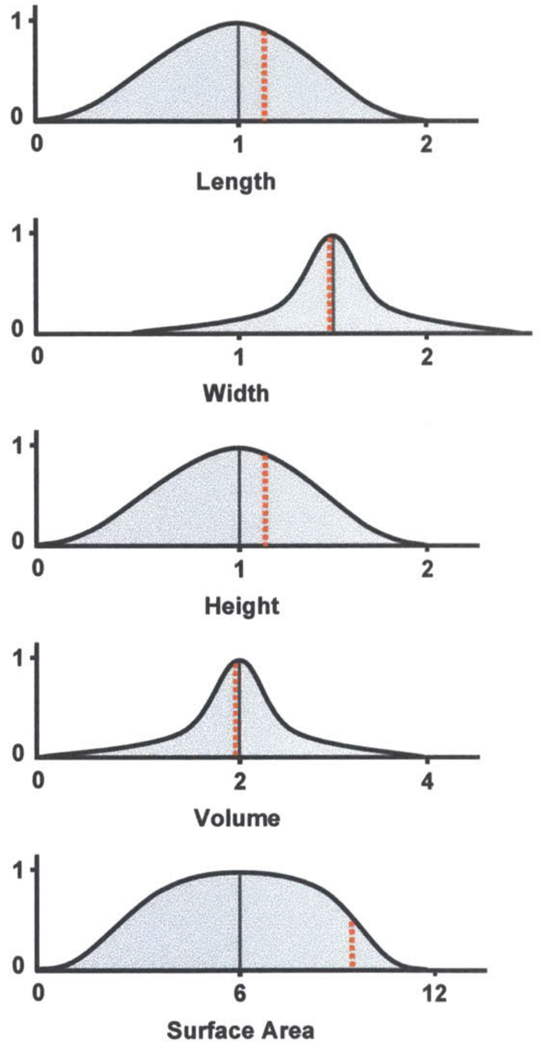
In this example we consider the model of cube that accepts the length, width and height of the box as inputs and provides the volume and surface area as outputs.

Previous Model State	
Length	1.0 m
Width	1.0 m
Height	1.0 m
Volume	1.0 m^3
Surface Area	6.0 m^2

The user adjusts the variables to the desired values.

Desired Model State	
Length	No preference
Width	1.5 m
Height	No preference
Volume	2.0 m^3
Surface Area	No preference

Preference functions are parametrically generated.



The preference functions are generated about the desired value. If no change is made, the previous value is used. The values that have been altered by the user are made more sensitive to changes, while those that have not altered are made less sensitive.

A search is run to find the solution that maximizes the aggregate of the preference functions

Final Model State	
Length	1.14 m
Width	1.49 m
Height	1.15 m
Volume	1.95 m^3
Surface Area	9.45 m^2

In the previous box example, the variables that the user changed were weighted to be more sensitive to changes. Correspondingly, the variables that the user had not changed were weighted to be less sensitive. This is important because the *primary* goal of the search is to find solutions based upon the user's desired values. The *second* goal is to limit the deviation from the previous solution. The ratio between the weighting of the variables that were and were not changed determines the relative weighting between these two goals.

If in the previous example, the user only desired the volume to be 2.0 m^3, the solution would likely be less than 2.0 m^3. This is because the length, width, and height would be "holding back" the solution because their preference functions would prevent them from deviating from their previous values. Since the surface area is determined exclusively by the length, width, and height, the preference function of the surface area would be adding to the amount they are "holding back" the solution. An algorithm to adjust sensitivities based on the number of inputs that a particular dependent variable depends upon should be developed. This is currently not possible to implement in DOME however, because it would require the model to be accessible on both an interface and parameter level.

The previous example required the specification of the range over which the preference function was generated. During the process of implementation, this was done by taking a percentage of the current value. If this value is set too low, the search algorithm may miss a reasonable solution. To find the solution, the user may have to run the backsolver again. While it would eventually converge to the correct solution, the user may find an iterative process to be frustrating.

Caching Techniques

A caching system was used to store the state of the model's interface in a relational database. The database was then used to find the best solutions with which to seed the initial population of the genetic algorithm. If the cache data were large enough, the solutions in the cache could be used exclusively, thus greatly decreasing latency. It is thought that this would reduce the search time to find a solution. The effectiveness of this technique however, was never fully evaluated.

The caching system was implemented using a JDBC compliant relational database. After each execution of the model, the states of all of the inputs and outputs were stored in the database. This was difficult in the current implementation of DOME because it only allows access to individual parameters, not interfaces. Implementing a version on DOME that allows access to simulations on the individual parameter level and the interface level is a needed development.

Indirectly Generated Preference Functions

The history of the users changes might also be utilized to infer the weights. An algorithm that determined the preference functions based entirely on the history of the users changes and desired values would probably be very effective for customers that have little to no understanding of how the backsolver works. The changes to a parameter and the order in which these changes occur could be used to infer its sensitivity. The previous method of parametrically generating preference functions could then be used to construct a preference function with this sensitivity value.

Future Considerations

The concept of constraints and specifications is often used in the selection of components in product design. A typical design problem would be the following:

> *"Select a bolt from a catalog of bolts that can withstand a bearing stress of 15 psi and has a diameter of 0.5 in."*

The problem would then be to find a bolt that satisfies the two constraints. When a model backsolver is used within a DOME-like system, the question might be instead posed as:

> *"Search for a bolt and supplier whose simulations predict that the bolt can withstand the bearing stress specified by the FEA analysis, while having the diameter specified by the CAD model."*

In this implementation, the backsolver can only search by changing the input values that predict the desired performance. However, it should also be capable of searching by changing the services and simulations. Currently, the network of simulations can only be made dynamic through a catalog of predetermined simulations to actively alter the models structure. More advanced methods should be considered to incorporate unknown suppliers. For example, an ontology could be overlaid on DOME models so that design services could be autonomously discovered and integrated into product models.

The current implementation of the model backsolver is a goal-oriented approach. The values of the outputs are always computed using the actual simulations, and are therefore necessarily valid. As a result, the notion of the system being over-constrained or under-constrained is irrelevant. The backsolver will always attempt to find better solutions, and while the solutions that are generated may not exactly match the desired values, they at least provide the user with a set of valid options. As the search algorithm continued to search further and new resources were added to the service marketplace, better solutions could always be found.

Although the implementation of the backsolver was rather elementary, it presented two major areas of concern worthy of future study. The first area is the development of algorithms to automatically generate preference functions on behalf of the user. The second area is the exploration of advanced techniques to actively alter the structure of a model, and their incorporation into search algorithms.

"People are afraid of new things. You should have just taken an existing product and put a clock on it or something."

- Homer Simpson

MASS CUSTOMIZED CATALOGS

The use of engineering simulations as the logic for product configurators produces very powerful and scalable configurators. However, the difficulties in modeling an entire product may prove too difficult for some organizations. In addition, the latency costs associated with executing the entire system model may be too great to provide reasonable, real-time performance predictions. Instead, the simulation-driven configurators could be used to generate catalogs of predetermined configurations. The configurator could be a tool used by account managers to generate a catalog of configurations specially tailored for a particular buyer. While this may appear to defeat the purpose of simulation-driven mass customization, it represents a practical next step. Given this situation, the problem of a sales representative promising an undeliverable configuration could easily be avoided.

In developing mass customized catalogs, the first step would be to decide which configurations to include in the catalog. These decisions would likely take into account the product portfolio and other strategic issues. While these decisions are rather subjective and may change often, new catalogs could be generated relatively quickly.

To implement mass customized catalogs, a means of saving the state of an model is required. A simple mechanism to do this was implemented in the caching system of the model backsolver. For each execution, the respective states of the inputs and outputs of the model were saved in a database. In reality however, a more robust system of distributed version control on each model server should be considered. The implementation of mass customized catalogs was not explored because, with the exception of the distributed version control system, such a model does not represent any interesting technical challenges.

Bart:	*Dad?*
Homer:	*What is it, boy?*
Bart:	*I thought your car was really cool.*
Homer:	*Thanks boy! I was waiting for someone to say that.*

- The Simpsons

CONCLUSIONS

The MYcycle pilot project revealed several areas of future research, while demonstrating that it is possible to use design simulations as the basis for a product configurator. A pilot project that is conducted in conjunction with an industry partner should be pursued to determine the exact effectiveness of the reuse of design models in reducing the construction and maintenance costs of configuration systems.

Latency

The model used in the MYcycle project was relatively simple, yet its latency averaged about 10 seconds. The issue of latency can not be assumed to be solved by improved processing power. As more processing power becomes available, the models will simply grow to be more complex. The simulation environment must have the ability to efficiently execute simulations. In additions, simulations should be able to be executed asynchronously. Caching techniques should also be incorporated on a fundamental level; however, this requires the simulations to be accessible at the granularity of both interfaces and variables.

Model Backsolving

The implementation of model backsolver was not developed to a level of maturity required to be included in the MYcycle project. It is thought that a model backsolver would be useful to product configuration because it would allow customers to alter the performance variables that they are familiar with. The backsolver presents two areas for future research. The first involves developing algorithms to determine which solutions the user would be interested in if an exact solution can not be found. A pilot project should be pursued to determine it customers could deal with the backsolver's inability to always find the desired values. The second area of research revolves around the incorporation of dynamic simulation networks into the backsolver's search techniques. Therefore, the model backsolver could search by not only changing the inputs of the model, but also by actively changing the structure of the model.

Finally, a simulation-driven product configurator may change the focus and process of product development. If an organization is able to successfully simulate their products, the goal of the product development process may become the integrated product model, not the product itself. The actual products would merely be states of the model, and customers could use the model to generate unforeseen products with unexpected uses. In addition, difficult tradeoff decisions encountered during the product development process would be essentially "outsourced" to customers.

You have joined the Sacred Order of the Stonecutters who, since ancient times, have split the rocks of ignorance that obscure the light of knowledge and truth. Now let's all get drunk and play ping pong!
- The Simpsons

Lisa: So even if a man takes bread to feed his starving family that would be stealing?
Rev.: No. Well, it is if he puts anything on it. Jelly, for example.
Lisa: I see.

- The Simpsons

REFERENCES

Abrahamson, S., Wallace D., Senin, N., Borland, N., (1999) "Integrated Engineering, Geometric, and Customer Modeling: LCD Projector Design Case Study", *Proceedings of the ASME DT Conferences*, DETC/DFM-9084, September 1999, Las Vegas, Nevada.

Abrahamson, S., Wallace, D., Senin, N., Sferro, P., (2000), "Integrated Design in a Service Marketplace", *Computer-aided Design,* **32**(2), 97-107.

Abrahamson, S., Wallace, D., and Borland, N. (1999) "Design Process Elicitation Through the Evaluation of Integrated Model Structures", *Proceedings of the ASME DT Conferences*, DETC/DFM-8780, Las Vegas, NV.

Abrahamson, S., Wallace, D., and Borland, N. (1999b) "Object-based Design Modeling and Optimization with Genetic Algorithms", GECCO-99*: Proceedings of the Genetic and Evolutionary Computation Conference*, July 13-17, Orlando, FL.

Ambler, B. (1998) "Improving the product realization process" *IEEE Intelligent Systems,* July/August 1998. pp 29.

Borland, N., Kauffman, H., Wallace, D. (1998) "Integrating Environmental Impact Assessment into Product Design: A collaborative modeling approach", *Proceedings of the ASME DT Conferences*, DETC/DFM-5730, Atlanta, GA.

Borland, N., Wallace, D. (1999) Environmentally-Conscious Product Design: a Collaborative Internet-Based Modeling Approach, to appear in *the Journal of Industrial Ecology*.

Boynton, A.C., Victor, B., Pine II, B.J. "New competitive strategies: Challenges to organizations and information technology." *IBM Systems Journal,* 32(1), 40-64.

Brown, D. (1998) "Defining Configuration", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*. 12, pp. 301-205.

Coates, R. "Mass Customization – Manufacturing Postponement for the Next Century" *Perspectives?* CSC Consulting, 1997.

Cooper, R. and S. Kaplan, R. S. (1998) "The promise and peril of integrated cost systems", *Harvard Business Review*, July-August, pp. 109-119.

Huffman, C., Kahn, B. "Variety for Sale: Mass Customization of Mass Confusion" *Marketing Science Institute* Working Paper Report No. 98-111 June 1998.

Kim, J., and Wallace, D. (1997) "A Goal-oriented Design Evaluation Model", *Proceedings of the ASME DT Conferences*, 97-DETC/DTM-3878, Sacramento, CA.

Loosschilder, G. H. "The Interactive Concept Test: Analyzing Consumer Preferences for Product Design." UT Delft, Netherlands, 1988.

Mittal, S., Frayman, F. (1989) "Towards a generic model of configuration tasks" *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, (IJCAI-89), pp. 1395-1401. Morgan Kaufmann, San Mateo, California.

Pahng, F., Senin, N, Wallace, D. (1998) Distributed object-based modeling and evaluation of design problems, *Computer-aided Design*, volume 30, number 6, pp. 411-423.

Rahmer, J. and Voss, A. (1998) "Supporting Explorative Configuration." *Artificial Intelligence in Design '98*, 483-498.

Sabin, D. and Weigel, R. (1998) "Product Configuration Frameworks –A Survey", *IEEE Intelligent Systems*. 43-48.

Senin, N., Wallace, D., Jakiela, M.J., (1996), "Mixed Continuous Variable and Catalog Search Using Genetic Algorithms", *Proceedings of the ASME Design Automation Conference*, 96-DETC/DAC-1489, August 1996, Irvine, California.

Senin, N., Wallace, D., Borland, N. (1999b) Distributed Object-based Modeling of Design Problems, under review by the *ASME Journal of Mechanical Design.*

Senin, N., Wallace, D., Borland, N. "Object-based Design Modeling and Optimization with Genetic Algorithms." GECCO99 – Real World Applications, July 1999, Orlando, Florida.

Smith, R. P. and S. Eppinger (1997). "Identifying Controlling Features of Engineering Design Iteration." *Management Science,* 43(3).

Steward, D. V. (1981). "The design structure system: a method for managing the design of complex systems." *IEEE Transactions on Engineering Management* EM-28(3): 71-74.

The Economist, April 1, 2000 pp. 57-58.

Tseng, M., Du, X. (1996) "Design by Customers for Mass Customization Products", Annals of the CIRP, Vol. 45/1/1996.

Von Hippel, E. (1994) "Sticky Information and the Locus of Problem Solving: Implications for Innovation" *Management Science,* 40(4), 429-439.

Von Hippel, Eric. (1998) "Economics of Product Development by Users: The Impact of "Sticky" Local Information" *Management Science,* 44(5), 629-644.

Von Hippel, Eric. (1999) "Toolkits for User Innovation: The Design Side of Mass Customization" *MIT Sloan School of Management Working Paper #4058,* February, 1999.

Yu, Bei., Skovgaard, Jørgen. "A Configuration Tool to Increase Product Competitiveness" *IEEE Intelligent System* 13(4) 1998. pg 34-39.

"Brother, Can You Spare Two Dimes?" 8F23 The Simpsons. Writer John Swartzwelder. Dir. Rich Moore.

"Homer the Great" 2F09 *The Simpsons*. Writer John Swartzwelder. Dir. Jim Reardon.

"Homer vs. Lisa and the 8th Commandment" 7F13 *The Simpsons*. Writer Steve Pepoon. Dir. Rich Moore.

"Oh Brother, Where Art Thou?" 7F16 *The Simpsons*. Writer Jeff Martin. Dir. W. M. "Bud" Archer.