# Toward a multi-agent information management infrastructure for product family planning and mass customisation

Steven B. Shooter*

Department of Mechanical Engineering
Bucknell University, Lewisburg, PA 17837, USA
E-mail: shooter@bucknell.edu
*Corresponding author

Timothy W. Simpson and
Soundar R.T. Kumara

Department of Industrial and Manufacturing Engineering
The Pennsylvania State University, University Park, PA 16802, USA
E-mail: tws8@psu.edu E-mail: skumara@psu.edu

Robert B. Stone

Basic Engineering Department
University of Missouri – Rolla, Rolla, MO 65409–0210, USA
E-mail: rstone@umr.edu

Janis P. Terpenny

Department of Engineering Education
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061, USA
E-mail: terpenny@vt.edu

**Abstract:** Development of complex new products requires numerous decisions by many individuals and groups, which are often geographically and temporally distributed. There is a need to share and coordinate distributed resources and synchronise decisions. Recent advances in Information Technology (IT) pose an untapped potential in assisting the capture, storage, retrieval, and facilitated use of product development information. By sharing assets such as components, processes, and knowledge across a family of products, companies can efficiently develop differentiated products and increase the flexibility and responsiveness of their product realisation process. This paper describes a recent effort in realising an information management infrastructure for product family planning and platform customisation. Particular focus is on current research thrusts:

- an evolutionary approach to product platforming
- a bottom-up approach to product platforming
- industry-based platform case studies

- exploration of ontologies for representing product family information
- filtering techniques to facilitate reuse of manufacturing information in product families.

**Keywords:** information technology; product family; mass customisation; software agents.

**Reference** to this paper should be made as follows: Shooter, S.B., Simpson, T.W., Kumara, S.R.T., Stone, R.B. and Terpenny, J.P. (2005) 'Toward a multi-agent information management infrastructure for product family planning and mass customisation', *Int. J. Mass Customisation*, Vol. 1, No. 1, pp.134–155.

**Biographical notes:** Steven B. Shooter is Associate Professor of Mechanical Engineering at Bucknell University where he teaches design and mechatronics. His research involves information management in design and mechatronic systems' design. As a registered Professional Engineer, he is also actively engaged in applied projects with industry that involve product development or the development of product realisation infrastructure. He received his BSME (1988), MSME (1990), and PhD (1995) from Virginia Tech.

Dr. Timothy W. Simpson is Associate Professor of Mechanical Engineering and Industrial Engineering at Penn State University. Dr. Simpson received a BS degree in Mechanical Engineering from Cornell University in 1994 and MS and PhD degrees in Mechanical Engineering from Georgia Tech in 1995 and 1998, respectively. His teaching and research interests include product family and product platform design, mass customisation, and product dissection. He is the Director of the Product Realisation Minor at Penn State and is an active member of ASME, AIAA, ASEE, and SAE.

Soundar R.T. Kumara is distinguished Professor of Industrial and Manufacturing Engineering. He holds joint appointments with the Department of Computer Science and Engineering and School of Information Sciences and Technology at Pennsylvania State University. He finished his BTech and MTech degrees in India, and PhD in Purdue University. He is an elected active member of the International Institute of Production Research.

Robert B. Stone is currently Associate Professor in the Interdisciplinary Engineering Department of the University of Missouri-Rolla. Dr. Stone's research interests are design theory and methodology, specifically product architectures, functional representations, and design languages. He is Director of the School of Engineering's Student Design Center where he oversees the design competition activities of eight teams and guides the Center's new engineering design and experiential learning initiative.

Janis Terpenny is Associate Professor in the Department of Engineering Education with affiliated positions in Mechanical Engineering and Industrial and Systems Engineering at Virginia Tech. She is co-Director of the NSF multi-university Center for e-Design. Her research interests focus on conceptual design of engineered products and systems. She is currently a member of ASEE (chair of the Engineering Economy Division), ASME, IIE, SWE, and Alpha Pi Mu. She is the Design Economics Area Editor for *The Engineering Economist* journal.

## 1 Introduction

Development of complex new products requires numerous decisions by many individuals and groups that are often separated by distance and time. These decisions are supported by individual and collective background knowledge and new knowledge synthesised from gathered information during the design process. The challenge, then, is to transform gathered information from a vast array of diverse sources into useful knowledge for making effective decisions. From a computational viewpoint, this is a problem of distributed resource sharing and temporal synchronisation. Recent advances in information technology pose an untapped potential for assisting in the capture, storage, retrieval, and facilitated use of product development information.

The Product Family Planning approach to product development is an important factor for success in many markets. By sharing assets such as components, processes, and knowledge across a family of products, companies can efficiently develop differentiated products and increase the flexibility and responsiveness of their product realisation process. Product Family Planning is also a way to achieve mass customisation by allowing highly differentiated products to be produced without consuming excessive resources.

In general terms, a product family is a group of related products that is derived from a product platform to satisfy a variety of market niches. The key to a successful product family is the product platform, where the product family is derived (Meyer and Lehnerd, 1997). As Robertson and Ulrich (1998) point out:

> "By sharing components and production processes across a platform of products, companies can develop differentiated products efficiently, increase the flexibility and responsiveness of their manufacturing processes, and take market share away from competitors that develop only one product at a time."

A product platform can also facilitate customisation by enabling a variety of quick and easily developed products to satisfy the needs and requirements of distinct market niches (Pine, II, 1993). Companies like Sony (Sanderson and Uzumeri, 1997), Volkswagen (Bremmer, 1999), and Black & Decker (Meyer and Lehnerd, 1997) have successfully employed product platform strategies to increase product variety while reducing development costs, manufacturing costs, and time-to-market.

Designing a product platform and corresponding family of products are difficult tasks. They embody all of the challenges of product design while adding the complexity of coordinating the design of multiple products in an effort to increase commonality across the set of products without compromising their individual performance (distinctiveness). Regardless of whether the platform is modular or scalable, the basic development strategy within any product family is to leverage the product platform across multiple market segments or niches.

Successful Product Family Planning places an even greater requirement on effective information management to exploit the potential of shared assets. As such, significant potential rewards can be gained from the exploration and development of information technology for this domain. We view information technology as both a key enabler for future manufacturing enterprises and a transformer of organisations and markets. By reducing barriers to collaboration, compressing lead-time, eliminating physical movement, and enriching decision making, information technology helps manufacturers to achieve their goals of meeting customer needs better, quicker, and cheaper. By

providing global reach and easy connectivity, information technology has fostered cooperation while increasing market competition, and heightened customer expectations. Advances in computer and communication technologies combined with rapid changes in organisations create new opportunities for exploiting information technologies in the entire product realisation process (Balakrishnan *et al*., 1999). Specifically, in the current context of product family design and planning, we address four classes of intelligent information processes:

1 intelligent search
2 collaboration
3 coordination and negotiation
4 understanding and learning.

Our research focus synthesises streams of thought from many related disciplines in engineering, computer science, and management to develop a framework – a computational platform – for examining how information technologies can facilitate and influence product family planning and design.

Inspired by the market success of 'customer-oriented' companies over the past decade, many manufacturing organisations have successfully inculcated customer focus as a philosophy at all management levels and in all functions. Starting with an emphasis on meeting specifications and commitments at downstream stages of the supply chain, customer orientation slowly propagated to upstream functions such as product design to better reflect customer preferences in design. The next step, mass customisation (Pine, II, 1993), aims to service increasingly finer-grained market segments by, *e.g.*, designing modular products that can be assembled in myriad combinations for individual niches. Companies have gone one step forward by offering made-to-order products (*e.g.*, jeans, shoes, and bicycles) whose physical dimensions 'perfectly' fit individual customers. The product architecture is fixed, and customers must still choose from an available (predetermined, but extensive) menu of choices at the time of purchase. Moreover, the customers who provide inputs to product design are not necessarily the product purchasers, *i.e.*, design is not truly customised. It is clear that future organisations will seek to achieve far greater levels of customer involvement, culminating in continuous customer engagement at all stages of manufacturing. Therefore, there is a need to support these activities with an open information management infrastructure.

The strategy for the formulation of this infrastructure is based on three foundations:

1 The development of a generalised information management infrastructure, with particular emphasis on capturing information regarding component-sharing and reuse within a family of products.
2 Creation of a corresponding graphical modelling environment.
3 Formulation of a software agent-based synthesis framework for product family planning and customisation.

This paper begins by describing the vision for the information management infrastructure. Then, efforts toward a graphical modelling environment are described, followed by the formulation of an agent-based framework for product family planning and customisation. The paper highlights some industry cases that are being reviewed, as well as examples from actual products. Closing remarks and future work are discussed in the final section.

## 2 Foundations for advancing the state of the art

### *2.1 Information infrastructure development*

The primary research objective is to develop a generalised information management infrastructure with particular emphasis on capturing information regarding component-sharing and reuse within a family of products. The key issue here is determining what constitutes critical design knowledge and how to archive it. There are many ways to represent a product, but to support knowledge reuse for future product design efforts and component-based representations of a product are needed. Based on our prior research (Hirtz *et al.*, 2002), we have identified functionality, physical parameters (*e.g.*, dimensions, geometry type, material) manufacturing process, input/output flows, performance parameters (and possibly a mathematical description of the performance), and interfaces as needed component knowledge for archival and reuse. Taken together, the abovementioned heterogeneous design knowledge constitutes a design repository.

There is currently no technology on the market that is truly a design repository. However, there are several packages that contain elements of a design repository. Such computerised design packages can be grouped into three basic categories:

1 Mechanical CAD (MCAD) packages that augment traditional CAD models with more abstract design knowledge.

2 Systems engineering tool-sets that contain higher level of design information so that may be used to generate CAD models.

3 Systems modelling and simulation packages which have no interaction with traditional CAD packages.

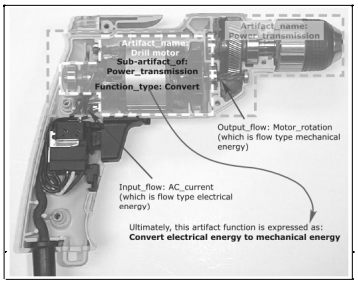
Though substantial results are often promised with such systems, they require significant resources to implement and have been applied to very specific applications; making their general acceptance and expansion difficult.

The review of current commercial offerings indicates that the concept of a design repository is extremely useful to automated design storage and retrieval packages and that industry is moving towards the vision of design repositories. A standard repository information management structure, supported by fundamental functional and architecture modelling research, is clearly needed to guide work in this area. The information management structure must support conceptual design, or the fuzzy front-end of the design process, since as much as 70% of product costs are determined during design (Barton and Love, 2001). One traditional hurdle to defining such structures is the evolving strategies and methodologies that exist for this phase of design.

Functionality is a critical piece of the knowledge representation puzzle. It allows products/components to be searched based on their function. This can be used to identify families of products (*i.e.*, products that perform a significant portion of the same functions) and overlapping functions can be used as a basis for defining a product's core platform. Analogy techniques that exist to find families of products are essentially simple mathematical operations, *i.e.*, project one product's functional vector onto a repository of product vectors. The higher the value of the projection, the more similar the two compared products are. This could also be easily applied on a component level.

Consider the drill shown on the left of Figure 1. A common component used in many product families is the drill motor (Meyer and Lehnerd, 1997). The middle entity in Figure 1 is an *Artifact* representing the drill motor that is a member of a family of other artifacts and *subartifacts*. The artifact information is also shown as an overlay on the actual drill. For instance, the motor includes a reference to a function object, providing a pointer from the artifact domain into the function domain. The last entity represents the motor *Function*, which is to 'convert' with an *input_flow* and *output_flow*. Each of these contains pointers to further levels of information. For example, *input_flow* points to an object named *AC_current* that has its own set of objects. This example is only a portion of the representation (as evidenced by the presence of pointers to other data structures); a broader description can be found in Refs. (Shooter *et al*., 2000; Szykman *et al*., 2001). Ultimately, these objects such as functions and flows can be categorised in the structured language of the functional basis (Hirtz *et al*., 2002). Then, the next step is to extend the current representation scheme to include higher-level objects that signify component-sharing and platform inheritance within a family of products derived from a common platform.

**Figure 1** Drill example with artifact and Transfer_Function for drill motor




Artifact
{

| | |
|---|---|
| name | Drill_motor |
| information | Drill_motor_info |
| type | Motor |
| config_info | Drill_motor_config |
| function | Drill_motor_fcn |
| form | Drill_motor_form |
| behaviour | Drill_motor_behaviour |
| subartifact_of | Power_transmission |
| is_source_of | {Motor_rotation} |
| is_destination_of | {AC_current} |

}

Transfer_function
{

| | |
|---|---|
| name | Drill_motor_fcn |
| information | Drill_motor_fcn_info |
| type | Convert |
| fcn_of_artifact | Drill_motor |
| input_flow | {AC_current} |
| output_flow | {Motor_rotation} |

}

## *2.2 Creation of a graphical modelling environment*

A recent conference by the Integrated Manufacturing Technology Initiative on 'Applying Knowledge to Manufacturing and Design' (Integrated Manufacturing Technology Initiative, 2003) brought together parties from industry, government agencies, and academia to outline necessary initiatives to advance technology in this realm. Two top areas identified were:

1 the formulation of standard knowledge representations

2 the development of intuitive graphical user-interface technology.

Representation and visualisation go hand-in-hand. Design, in principle, is a synthesis activity. Hence, a graphical modelling environment that allows for artifacts to be connected together through visual representation with due consideration to functionality will be very useful. The preceding section addresses the first area. The graphical modelling environment discussed in this section is a critical aspect of the development of a coordinated information management infrastructure for product family planning.

In a similar manner, feature-based design research of the late 1980s (Ansaldi *et al*., 1985; Forrest, 1986) and early 1990s (Anderson and Chang, 1990; Bronsvoort and Jansen, 1993; Chen and Hoffman, 1995) led to the current generation of commercial CAD systems. It is necessary to develop information management and higher-level modelling artifacts for function-based systems that can reason about families of products. Effective graphical modelling environments must support the planning and development of families of engineered products and systems for mass customisation through three key services including:

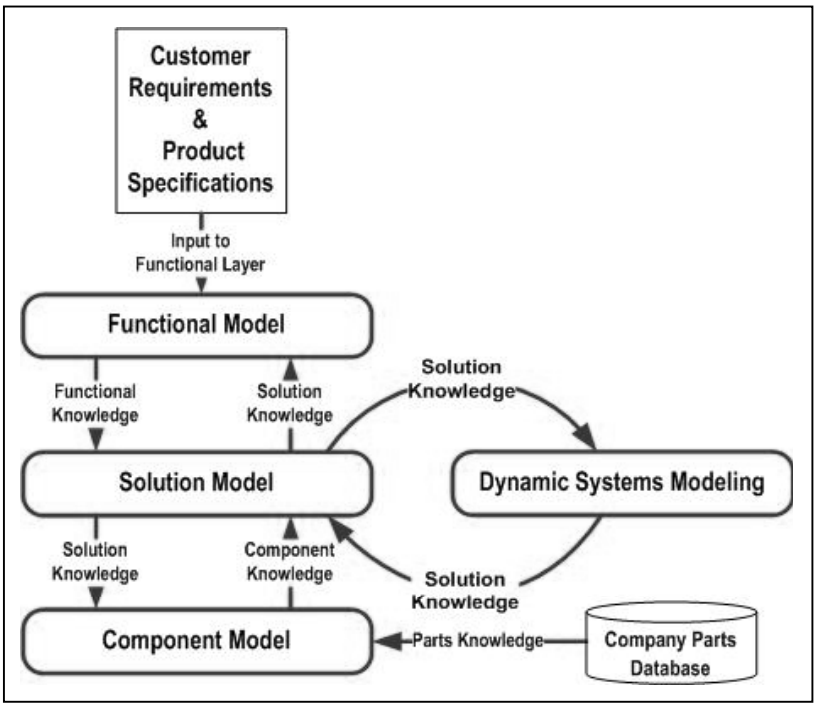
1 a graphical means to select, place, and semantically connect modelling objects representing design functions of the product together in a cohesive model

2 the means to capture requirement specifications and customer preferences that can be used to constrain this model

3 the means to visualise results of solution synthesis and regenerate new solution alternatives as needed.

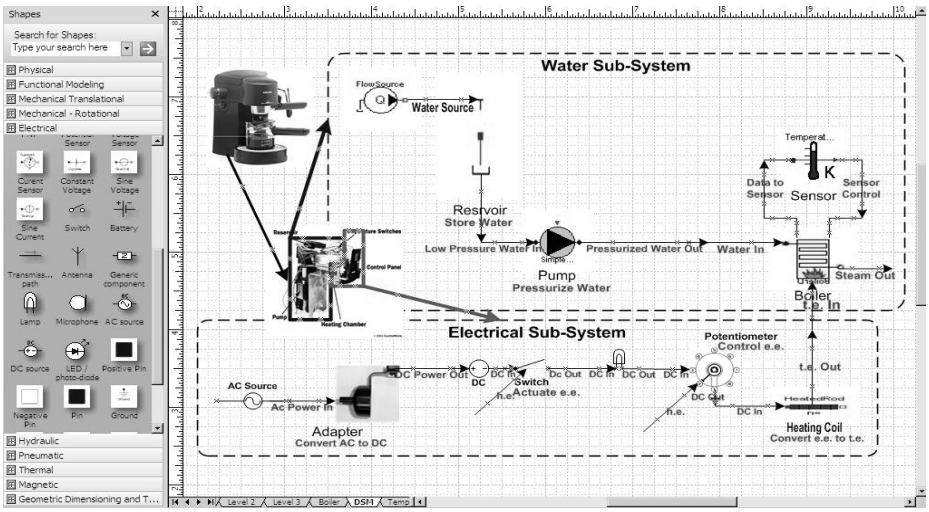
A Windows form-based system enables a top-down function-based modelling methodology that is supported by a web-based interface for knowledge management and modelling (Sikand and Terpenny, 2001; 2003). This approach provides organisations with a means to create their own classification system and repository of reusable solutions to function in design. Addressing the needed transformation – among the intuitive environment that contains user-defined terminology and artifacts and the information management infrastructure of a generalised repository – is a challenging area for exploration. Figure 2 provides an overview of the essential elements needed in the proposed graphical modelling environment.

**Figure 2** Elements of proposed graphical modelling environment



Functions, Solutions, and Components are modelled as objects in this environment. The environment supports modelling across multiple levels of abstraction. At the highest level of abstraction is the Functional Model where the designer gives inputs to the system about the customer requirements and product specifications. At the next lower level of abstraction is the Solution Model where solutions to functions are identified. Solutions are not explicitly tied to functions; rather, they emerge with the satisfaction of requirements. Solutions are represented in terms of a Dynamic Systems Model to facilitate solution synthesis. Figure 3 shows an example of a graphical modelling environment used to describe the Dynamic Systems Model of the Electrical and Water Sub-Systems in a coffee maker. As shown, the coffee maker can be described with several layers, providing the greatest freedom to designers in representing design concepts in terms of abstraction or detail. As shown on the left-hand side of the figure, six domains define the types of artifacts that can be used in modelling, including mechanical, electrical, hydraulic, pneumatic, magnetic, and thermal. Decoupling of knowledge from data is achieved with the separation of the Component Model, invoked during configuration, and the company parts' database. This provides for greater generalisation as well as reduced system maintenance.

**Figure 3** Coffee maker described in the graphical modelling environment




## *2.3 Agent-based synthesis framework for product family planning*

Software agents serve as the ideal mechanism to implement the core intelligent information processes in design and manufacturing. A software agent is an entity that functions continuously and autonomously in an environment that is often inhabited by other agents and processes (Shoham, 1993; 1997). Software agents may be viewed as virtual representatives of humans, processes, or products. Constructed as independent software entities, agents act as participants, sentinels, and negotiators. Agents embody three main functional components – communicating, understanding and processing (reasoning), and learning. The capabilities of an agent depend on its endowments along three dimensions:

1 *autonomy* or the extent of independent decision making

2 *intelligence*, the amount and type of knowledge, reasoning, and learning capabilities

3 *mobility*, the ability to move across systems.

We envision an agent-based synthesis framework for the configuration of platform concepts and customised variants that will support the following steps in the design process:

1 Given a customer's functional requirements, appropriate design configurations will be retrieved from the design repositories and presented to the customer.

2 From the product family set, components will be retrieved through appropriate functionality matching (closest functionality match).

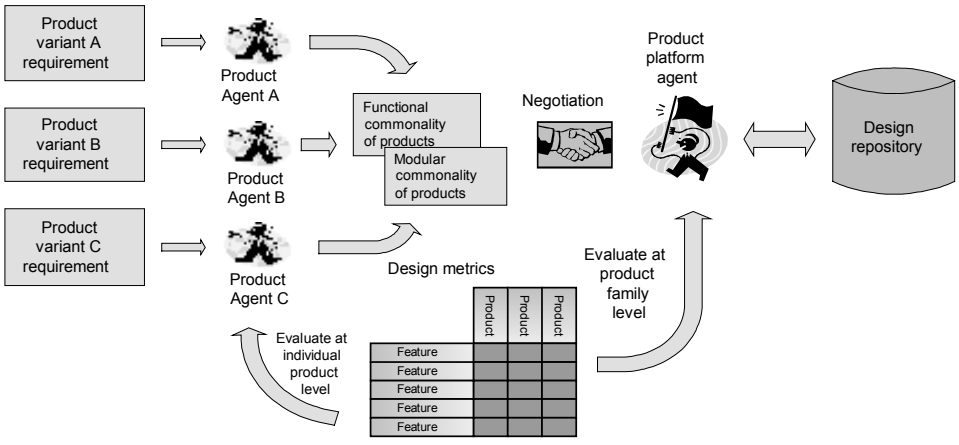
3 The selection of appropriate components (products from product platform) will be based upon the customers' requirements, and the product will be iteratively configured.

4 The new product (customised platform variant) will be presented to the customer after the iterated negotiation process.

5 In case the platform does not have the components and/or products belonging to the family that will satisfy the requirements, based upon the repository knowledge, a new product will be synthesised and presented to the customer.

Figure 4 illustrates the proposed agent-based approach to product platform planning. Agents in the environment will represent product variants and negotiate with one another to determine the components and processes that can be made in common and those requiring unique design. The agents will be autonomous to automate the negotiation process for the common platform, building upon the design repository. The platform optimisation will be realised in an evolutionary approach by iteratively updating the common base. This will depend on the evaluation feedback from each self-interested agent, based on its own product variant derivation from the platform. In this manner, global optimality at the product family level is anticipated.

**Figure 4** Proposed agent-based framework for product family planning



The following computational tasks are needed to perform these design tasks:

- Retrieve design components and related knowledge from the design repositories – this involves *intelligent search* based on very detailed low-level functions and features.
- Retrieve products from the product family that will satisfy the customer requirements – this also involves *intelligent search* based on high-level functional and cost features.
- In the event of synthesis of a new product from the design repository knowledge, there is a need to develop auction-based mechanisms since there can be several components that satisfy the same functional requirements. Each of the components will 'bid' to be included in the product, and this can be modelled as a market economy where configurations are generated through negotiation with the customer.
- Once a new product is synthesised, it needs to be added to the product family.

The design process can be computationally modelled using software agents. Table 1 summarises the agents that must be designed, developed, and implemented. Table 1 also summarises the research issues for building a computational infrastructure (Cols. 3 and 4). The following research lists the issues that must be addressed for the development such an agent-based infrastructure:

- *Developing design domain (product platform) ontology for defining agent communication language*: Agent communication requires a vocabulary, spawning the need for the development of an ontology for platform design which we are currently developing (Nanda *et al*., 2004).
- *Standardisation of agent communication, and reasoning*: Interoperable agents in the current research can only be established with the standardisation of agent systems. Standardisation should consider agent communication, multi-modal message understanding, and reasoning. This language will consider negotiation protocols for product family formulation.
- *Multiagent behaviour evolution and learning*: This must consider simple statistical regression techniques and multiattribute utility theory to build appropriate tools for modelling agent behaviour evolution and learning.
- *Integrating platform functions through the web*: Distributed design facilitation is an important area for exploration, wherein customers, design repositories, and product platforms are distributed; making autonomous business process integration very complex. This requires the building of an agent communication language to ensure easier and seamless integration.

In the past decade, agent research has advanced considerably. Several software agent building platforms are available both as open source and commercial packages. In terms of Technology Readiness Level,[1] currently agent software is at Level 7 and ready to be rated at Level 8 in about three years. Cyebele, developed by Intelligent-Automation Incorporated,[2] is currently being used for the development of agent based air traffic control under NASA's leadership, with about 10,000 agents. Cognitive Agent Software (Cougaar) developed under the direction of DARPA is a complex system which

addresses robustness, security, and performance (survivability) and is currently being used for deploying agent based military logistics and planning systems.[3] These large-scale systems prompt the authors to be confident that the research proposed in this paper is realisable and not just a conceptualisation.

**Table 1** Software agents for product family planning and platform customisation

| *Agent* | *Platform design domain task* | *Computational task* | *Computational techniques addressed* |
|---|---|---|---|
| Retrieval agent | • Retrieve the components from the repository | • Database search<br>• Function-based matching | • Similarity distance (between the required and specified functions in the design repository)<br>• Partial information-based matching (approximate reasoning-Bayesian) |
| Customisation agent | • Customer interface<br>• Facilitate negotiation between the design platform and customer<br>• Learn customer design preferences | • Extract functional requirements from customer input<br>• Present the configuration provide interaction<br>• Profile learning | • Content-based function matching<br>• Event based message passing<br>• Parametric and non-parametric techniques for customer classification |
| Platforming agents | • Match components with product platform<br>• Design synthesis using different components<br>• Perform negotiation with the customer for acceptance | • Matching<br>• Inference<br>• Auction-based negotiation | • Rule-based inference<br>• Market economy based mechanism design for auctions |
| Configuration agents | • Collect different products satisfying the customer and form the platform | • Similarity matching | • Clustering<br>• Grammar-based methods (syntactic pattern recognition) |
| Archival agents | • Add new components to design repositories<br>• Add new products to product families | • Database updating<br>• Data dictionary updating<br>• Forming meta-databases | • Meta database design<br>• Concept graphs for new concept addition |

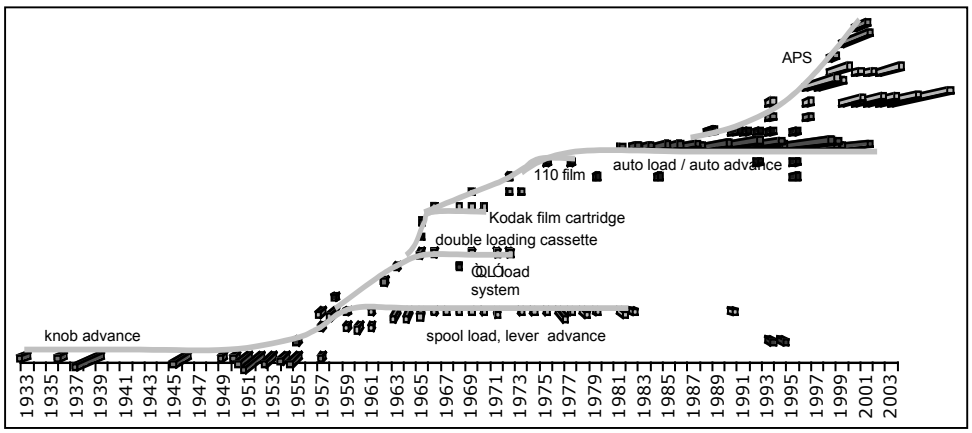
## 3 Current research thrusts

It is clear that there is a multitude of possible approaches for developing an information management infrastructure for product family planning and platform customisation. It is also essential that such development includes best practices from industry as well as recent research advancements. Our current research is pursuing several avenues for identifying product platform leveraging strategies to support future product family planning. In particular, we are examining an evolutionary approach to product platforming (see Section 3.1), a bottom-up approach to product platforming (see Section 3.2), and industry-based platform case studies (see Section 3.3). Research thrusts aimed at ontology development for product families to support reuse of product design information (see Section 3.4) and filtering techniques to facilitate reuse of manufacturing information in product families (see Section 3.5) are also discussed.

### *3.1 Thrust 1: evolutionary approaches to product platforming*

Our first research thrust is to examine the extent of product evolution and variety which influences the development of the product platform. The intent is to determine what is common among platforms and how platforms and differentiating elements are distinguished as a product evolves. Our hypothesis for this thrust is that product platforms form (or, are formed) during periods of slow advancement in an evolution of technological performance. Based on a study of Canon cameras, platforms appear to be associated with S-curve plateaus representing periods of slow evolution. In Figure 5, which represents *S*-curves for 70 years of Canon cameras, consider the lowest *S*-curve that begins at the label 'knob advance' and continues to the label 'spool load, lever advance.' This curve represents a camera's evolution over time in terms of mechanical film advance mechanism performance. (Note: Each increase in performance corresponds to a distinct increment in loading and advancing capability, Kurtadikar *et al*., 2004). Note that in the plateaus of any given *S*-curve, there are typically more variants of the camera (indicated by the axis out of the page). During the steeply sloped portions of the curve, fewer variants are found. Thus, if a company's position on the *S*-curve can be determined in real time, then designers can anticipate when it is appropriate to platform and when it is appropriate to prepare for a technological jump.

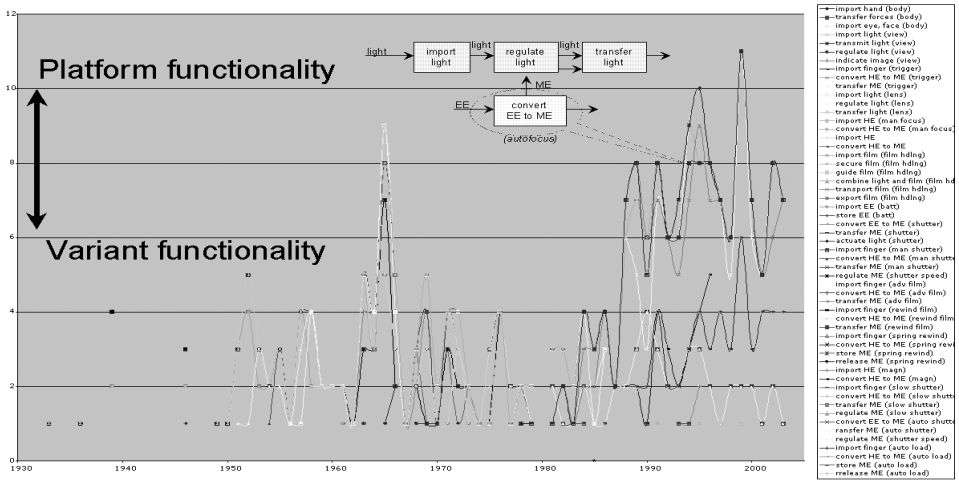
**Figure 5** Evolution of film load and advance function in Canon cameras



The aforementioned strategy is based on the S-curve effect in which a given technology generally exhibits a plateau during both its initial inception and during its optimisation. While the decision itself to platform (and therefore stabilise by choice) a given technology during any period along this S-curve may actually cause a plateau to appear that otherwise would not, the inherent plateaus during startup and final optimisation of a technology can, we claim, be used and leveraged as a basis for a platform. The rationale is that the slow technological progress associated with plateaus offers a prime opportunity to platform. Difficulty arises since this strategy relies on real-time knowledge of technological performance, which is error-prone. Additionally, usage of such an evolutionary approach is based on the assumption that particular opportunities afforded by inherent technological plateaus are relatively significant compared to opportunities caused by many other factors affecting the decision to platform. For example, while NiCad battery technology is relatively stable and therefore indicative of a platform based on NiCad batteries (as evidenced in the VersaPack tool family), there are clearly other factors that create an impact on the platform decision. When taken with its underlying assumptions and despite other factors involved that must be taken into account during platform selection, the use of an evolutionary approach appears promising in certain cases.

Looking at the camera data from a slightly different perspective, Figure 6 summarises the impact of variant functionality on platform design. In Figure 6, the occurrence of all product subfunctions found in the 70 years of data is plotted. This analysis shows that over time, frequently occurring functions (near the top of the graph) tend to become part of the platform and, thus, are solved by the platform instantiation. Less frequently occurring functions (near the bottom of the graph) tend to be part of the differentiating elements of the products. In several cases, certain functions tend to become more frequent, indicating that they started as variant functionality and have been accepted as basic or required functionality in future products. As with the *S*-curve analysis, if designers can spot trends in needed functionality across several product variants, then they can predict what functionality to capture in the product platform.

**Figure 6** Platform and variant functionality within Canon cameras



### *3.2 Thrust 2: bottom-up approach to product platforming*

Our second research thrust is to analyse existing consumer products that are readily available in the market in an attempt to reverse engineering product platform strategies employed by different companies. We believe that it is possible to identify platform strategies through dissection of existing product lines by analysing common, variant, and unique components (and connections) within a set of products. Figure 7 shows five modules (rows) from seven different single-use cameras (columns) that were dissected and analysed. Each component and module in each product are photographed, weighed, and measured, and material and manufacturing processes are noted as best as possible (*e.g.*, plastic injection moulded piece, stamped brass, machined aluminium). The components and modules are also categorised as being common, variant, and unique components; and numbers and colours are used in each row to identify which products share components. Using this information, we can compute a variety of commonality indices for the product family as discussed in Ref. (Thevenot and Simpson, 2004) to assess the amount of common assets within the family of products.

**Figure 7** Example of single-use camera dissection

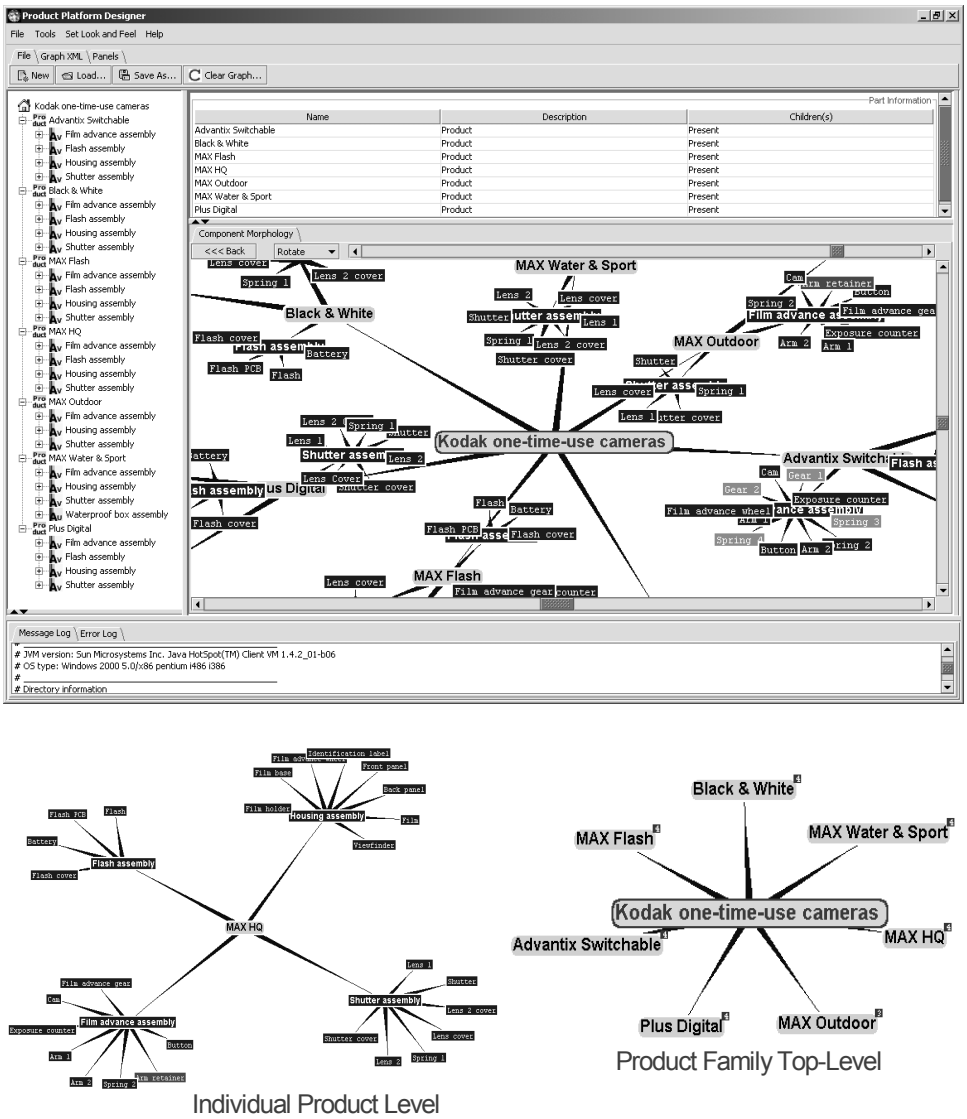
| Module | Module type | MAX Outdoor | MAX Flash | Max HQ | Plus Digital | Black & White | Max Water & Sport | Advantix switchable |
|---|---|---|---|---|---|---|---|---|
| Flash | Variant | | 1 | 1 | 1 | 2 | | 3 |
| Flash cover | Variant | | 1 | 1 | 1 | 2 | | 3 |
| Flash PCB | Variant | | 1 | 2 | 2 | 3 | | 4 |
| Front cover | Unique | | | | | 1 | | |
| Front panel | Variant | 1 | 2 | 3 | 3 | 4 | 5 | 6 |

The information is also entered into a database for which a graphical visualisation interface (see Figure 8) is being developed to facilitate storing, retrieving, and analysing information from the design repository. The visualisation interface is with the graph-based Java Web Start[4] programme, which is developed using swing components and the TouchGraph library.[5] TouchGraph library is an open source graph visualisation tool that uses spring layout and presents node content on focus. Java Web Start allows the applications to be deployed with a single click over the internet using Java™ Network Launching Protocol (JNLP), a web-centric provisioning (distribution of software components) protocol. Java Web Start technology architecture is both browser and web server independent, ensuring that the most current version of the application is deployed on the client's desktop. Product part information is presented in a hierarchical tree structure using swing's 'JTree' component. The node structure is mapped to the graph library as an alternate display. For persistent storage and retrieval, any node in the tree structure can be serialised, or the whole graph can be exported as XML document. The graph structure and the tree structure can also be captured graphically and exported in to various common graphics formats (*e.g.*, bmp, jpg, gif, png). The panel-based graphical user interface contains tools for manipulating the product tree structure. The following operations can be performed on the product tree:

- Add node
  - a New components can be added as nodes by the supplied graphical user interface.
  - b Serialised nodes can also be attached to any existing node.
- Modify Node
  - a Component name and details can be modified and stored.
- Delete node
  - a Any node and all its child nodes can be deleted.
- Serialise/deserialise node
  - a Any Node with all its child nodes can be serialised, *i.e.*, stored in disk and can be retrieved later
- Plot graph in independent window(s)
  - a Any node along with its child nodes can be plotted in a separate window. Display depth of the graph can be specified. The graph can be zoomed in a particular area. By focusing over a node, the details of the nodes can be displayed. The whole tree can be traversed by clicking on the nodes.

Recent efforts on design repositories are capable of storing function-form descriptions at the individual product level, and we are currently developing enhancements to include a product platform and product family levels along with an associated ontology for reasoning about new product variants (Nanda *et al.*, 2004).

**Figure 8** Visual representations of the single-use camera family

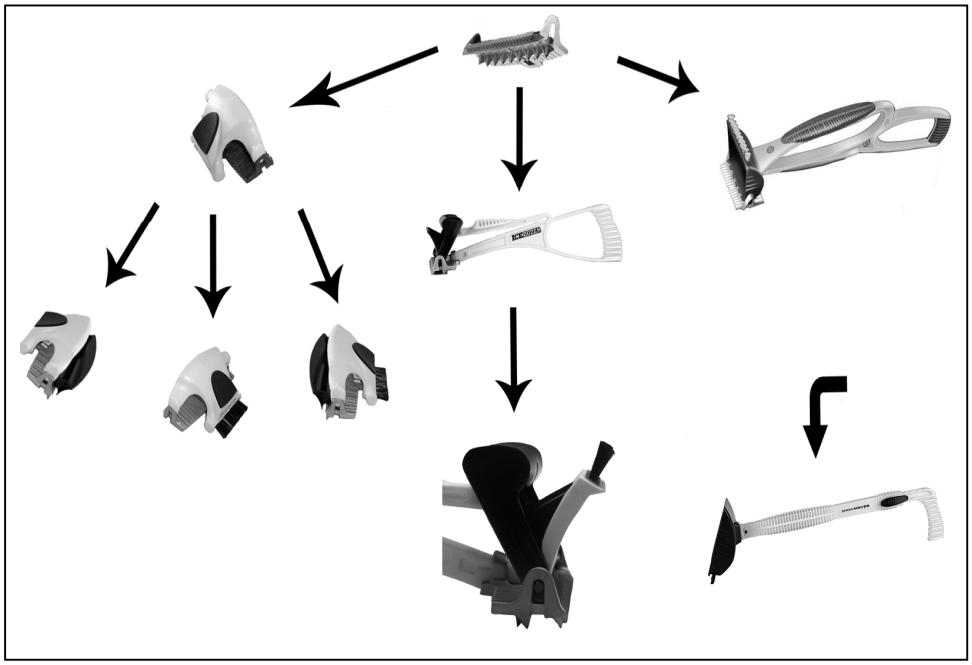



## *3.3 Thrust 3: industry-based platform case studies*

Our third research thrust is in developing interactions with company personnel to explore their approach(es) to product platforms. The goal of this activity is to identify what is common about platform strategies that are being employed within different companies. As part of this process, we are examining the information capture, storage, and retrieval needs for implementing platform approaches by direct interaction with companies' personnels.

One interesting case involves a small company, Innovation Factory®, which used a platform strategy when developing an innovative ice scraper product line (shown in Figure 9). Note the common scraper blade, which was intentionally designed as a platform. The case study (Shooter, 2005) explores the development of this product with a focus on the platform approach. The case examines topics such as market research, ideation and conceptual design, detailed design, testing, manufacturing and production, marketing, and distribution. In preparing the case, the range of the information materials involved in the development is examined, such as design notebooks, drawings, market research, test data, meeting notes, and cost information. The case studied the information stored and its modes of communication. The study of this case has prompted updates to the design repository to better support product family planning.

**Figure 9** Ice scraper platform approach of innovation factory®

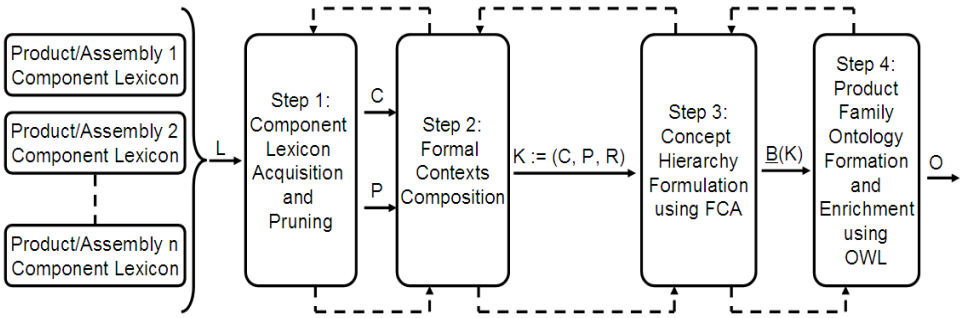


### *3.4 Thrust 4: product family ontology development*

We have proposed a methodology called Product Family Ontology Development Methodology (PFODM) (Nanda *et al*., 2004) for developing formal product ontologies using the semantic web paradigm. Formal Concept Analysis (FCA) is used to develop the conceptual models of product design artifacts upon which the product family ontology is formalised using Web Ontology Language (OWL). FCA is used to identify similarities among a finite set of design artifacts based on their properties and is used to develop and refine the ontology using OWL. FCA borrows its mathematical foundation from order theory, the theory of complete lattices in particular and is used to identify similarities among a finite set of design artifacts, referred, based on their properties, as 'formal concepts'. Formal product representation using OWL can not only store the structure of

the product family but also help in capturing the evolution of different components in the product family. In the proposed PFODM shown in Figure 10, special importance is given to the tabular and graphical representation of product design artifacts and their properties using FCA to readily tie to an existing design repository.

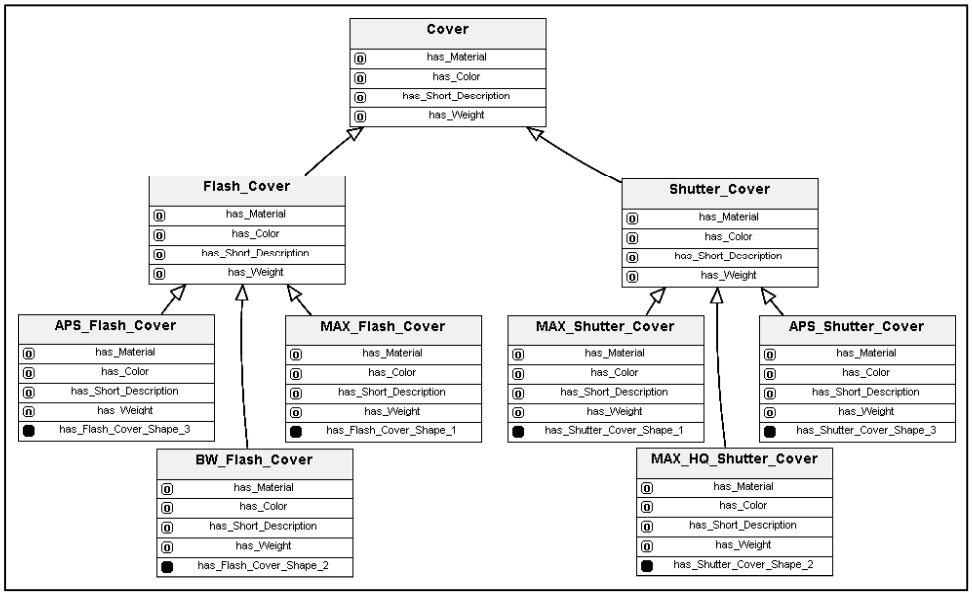
**Figure 10** Product Family Ontology Development Methodology (PFODM)




As an illustration, a group of one-time-use cameras, containing several products from the Kodak one-time-use camera family, is represented in OWL format. PFODM can support an explicit and fully documented way to develop product family design ontologies and is elaborated in this paper. Product family design representation using OWL promotes better learning of product features across products and reduced development time, system complexity, and product design lead-time. The ontology is illustrated in Figure 11 for the cover of the cameras. The class 'Flash_Cover' is defined with properties such as 'has_Weight'. All the child classes (*i.e.*, sub-classes) inherit this property. In the example, the 'APS_Flash_Cover' class and the 'MAX_Flash_Cover', sub-classes of 'Flash_Cover', also have 'has_Weight' as a property. All the sub-classes have an 'is a' relationship with their parent classes. An individual of type 'MAX_Flash_Cover' is a 'Flash_Cover' which belongs to the 'Cover' domain. All the properties of class 'Cover' will show up in 'MAX_Flash_Cover'. When designing new components in the one-time-use camera product family, the designer extends (creates sub-class of) one or more of the existing components. Thus, the class diagram not only captures the relationship among components but also traces the evolution of the components in a product family.

The method of building ontologies using PFODM imposes traceability on the development of formal ontologies for a product family domain. It also provides a systematic method of comparing alternative cross tables and restructuring the class hierarchy where new classes/properties can be integrated, or subtracted from existing representations. From the collection of these design artifact *classes*, a rather large number of combinatoric spaces can be formed from all possible composite descriptions or contexts to help the designers explore, redesign, and improve a product family.

**Figure 11** OWL representation of *Kodak: cover* class

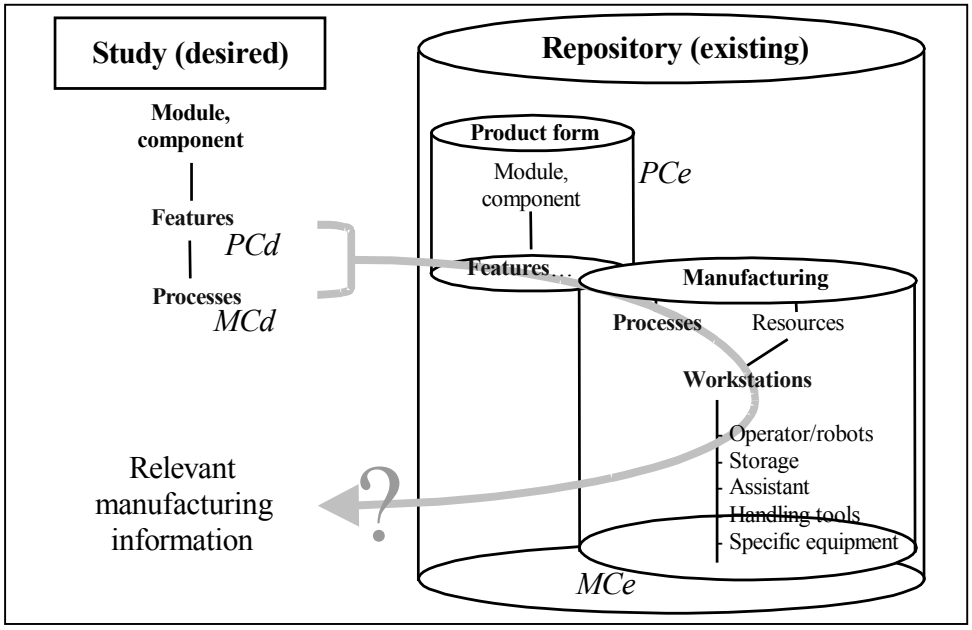



### *3.5 Thrust 5: reuse of existing manufacturing information for product families*

A considerable problem is the reduction of costs and time for the development of the product platform and manufacturing of the associated products. As many companies have already designed and manufactured many individual products, there is an advantage to exploring the reuse of relevant information to design the manufacturing facility required to produce the associated products. The REUSE (Reuse Existing Unit for Shape and Efficiencies) framework is based on the relevancy, efficiencies, and configuration of existing knowledge. This initiative will move the process designers and factories closer to take advantage of the strong rules of designs and the experience from manufactured products. This framework is applicable from the preliminary stage to the end of the design process.

In order to assess relevancy, the characteristics of both the product and the process must be considered, because the product and manufacturing are inextricably bound together and because it is preferable, in terms of optimisation, to consider the whole set of parameters from the beginning of the study. Thus, for each module or component, the Product Characteristics (PC) are used to define the product information, and the Manufacturing Characteristics (MC) represent the information process. Two categories of information must be considered: first the *desired* information, represented by the Product Characteristics desired (PCd) and the Manufacturing Characteristics desired (MCd). Second, the *existing* information identified by the existing Product Characteristics (PCe) and the existing Manufacturing Characteristics (MCe) are available in a repository (see Figure 12). The aim is to match both the product-manufacturing desired characteristics and the existing product-manufacturing characteristics to reuse the relevant existing manufacturing information. This effort has been described in (Alizon *et al*., 2005) and includes examples from the battery and air conditioning modules from vehicles.

**Figure 12** Proposed REUSE framework for manufacturing of families of products




## 4 Closing remarks and future directions

It is clear that the formulation of an information management infrastructure for product platforms and mass customisation requires a concerted effort with multiple thrusts. Described here are five of the current thrusts undertaken by this multi-university team to better understand the fundamentals of product platforming:

1 evolutionary approaches to product platforming

2 bottom-up approaches to product platforming

3 industry-based platform case studies

4 exploration of ontologies for representing product family information

5 filtering techniques to facilitate reuse of manufacturing information in product families.

The long-term goal is develop representations of information that are currently unavailable in traditional CAD/CAM/CAE tools to support information and knowledge exchange in the new product development paradigm and to help avoid a proliferation of proprietary formats in next-generation commercial software tools. While not a standard development effort, it is an attempt to provide a generic representation schema that can serve as a foundation for development of new systems.

Future work will continue to progress on multiple fronts with coordination efforts occurring through bi-monthly conference calls and 3–4 face-to-face meetings throughout the year. Industry involvement continues to play a major role in guiding our development

efforts. For instance, we are currently dissecting and analysing a family of refrigerators donated by a leading appliance manufacturer to test the scalability of our Product Family Ontology Development Method on a large-scale system. We are also in discussions with several companies to identify case studies related to their recent efforts in shifting to platform-based new product development. Finally, educational efforts are underway to improve students' understanding of the importance of product platform planning and mass customisation.

## Acknowledgements

This work was funded by the National Science Foundation through Grant Nos. IIS-0325402, IIS-0325321, IIS-0325279, and IIS-0325415. Any opinions, findings, and conclusions or recommendations presented in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

## References

Alizon, F., Shooter, S.B. and Simpson, T.W. (2005) 'A collaborative framework for knowledge reuse in manufacturing of platform-based products', *ASME Journal of Computing and Information Science in Engineering*, under review.

Anderson, D.C. and Chang, T.C. (1990) 'Geometric reasoning in feature-based design and process planning', *Computers and Graphics*, Vol. 14, No. 2, pp.225–235.

Ansaldi, S., De Floriani, L. and Falcidieno, B. (1985) 'Geometric modeling of solid objects by using a face adjacency graph representation', *Computer Graphics*, Vol. 19, No. 3, pp.131–139.

Balakrishnan, A., Kumara, S.R.T. and Sundaresan, S. (1999) 'Manufacturing in the digital age: exploiting information technologies for product realization', *Frontiers of Information Systems*, Vol. 1, No. 1, pp.25–50.

Barton, J.A. and Love, D.M. (2001) 'Design determines 70% of cost? A review of implications for design evaluation', *Journal of Engineering Design*, Vol. 12, No. 1, pp.47–58.

Bremmer, R. (1999) 'Cutting-edge platforms', *Financial Times Automotive World*, September, pp.30–38.

Bronsvoort, W.F. and Jansen, F.W. (1993) 'Feature modeling and conversion key concepts to concurrent engineering', *Computers in Industry*, Vol. 21, No. 1, pp.61–86.

Chen, X.P. and Hoffmann, C.M. (1995) 'On editability of feature-based design', *Computer-Aided Design*, Vol. 27, No. 12, pp.905–914.

Forrest, A. (1986) 'User interfaces for three-dimensional geometry modeling', in F. Crow and S. Pizer (Eds.) *Proceedings 1986 Workshop on Interactive 3D Graphics*, New York: ACM Press, pp.237–249.

Hirtz, J., Stone, R., McAdams, D., Szykman, S. and Wood, K. (2002) 'A functional basis for engineering design: reconciling and evolving previous efforts', *Research in Engineering Design*, Vol. 13, No. 2, pp.65–82.

Integrated Manufacturing Technology Initiative (2003) *Notes from Applying Knowledge to Design and Manufacturing: The Fall 2003 Workshop*, Nasville, Tennessee, October.

Kurtadikar, R., Stone, R., Van Wie, M. and McAdams, D. (2004) 'A customer needs motivated conceptual design methodology for product portfolios', *ASME Design Engineering Technical Conferences – Design Theory and Methodology Conference*, Salt Lake City, UT: ASME.

Meyer, M.H. and Lehnerd, A.P. (1997) *The Power of Product Platforms: Building Value and Cost Leadership*, New York, NY: Free Press.

Nanda, J., Thevenot, H., Simpson, T.W., Kumara, S.R.T. and Shooter, S.B. (2004) 'Exploring semantic web technologies for product family modeling', *ASME Design Engineering Technical Conferences – Design Automation Conference*, Salt Lake City, UT: ASME, Paper No. DETC2004/CIE-57683.

Pine, II, B.J. (1993) *Mass Customization: The New Frontier in Business Competition*, Boston, MA: Harvard Business School Press.

Robertson, D. and Ulrich, K. (1998) 'Planning product platforms', *Sloan Management Review,* Vol. 39, No. 4, pp.19–31.

Sanderson, S.W. and Uzumeri, M. (1997) *Managing Product Families*, Chicago, IL: Irwin.

Shoham, Y. (1993) 'Agent oriented programming', *Artificial Intelligence*, Vol. 60, No. 1, pp.51–92.

Shoham, Y. (1997) 'An overview of agent oriented programming', in M. Bradshaw (Ed.) *Software Agents*, Menlo Park, CA: AAAI Press/MIT Press.

Shooter, S.B. (2005) 'Product platform development at innovation factory', in T.W. Simpson, Z. Siddique and J. Jiao (Eds.) to appear in *Product Platform and Product Family Design: Methods and Applications*, Kluwer Academic/Plenum Publishers, expected publication June.

Shooter, S.B., Keirouz, W., Szykman, S. and Fenves, S. (2000) 'A model for the flow of design information in product development', *Journal of Engineering with Computers*, Vol. 16, pp.178–194.

Sikand, A. and Terpenny, J.P. (2001) 'Collaborative and distributed design: current status and research opportunities', *Proceedings of Flexible Automation and Intelligent Manufacturing (FAIM) International Conference*, Dublin, Ireland, 16–18 July, pp.389–398.

Sikand, A. and Terpenny, J.P. (2003) 'A web-based framework for product knowledge exchange and distributed design', *IIE Transactions on Design and Manufacturing*, under review.

Szykman, S., Fenves, S., Keirouz, W. and Shooter, S. (2001) 'A foundation for interoperability in next-generation product development systems', *Journal of Computer Aided Design*, Vol. 33, pp.545–559.

Thevenot, H. and Simpson, T.W. (2004) 'A comparison of commonality indices for product family design', *ASME Design Engineering Technical Conferences*, Salt Lake City, UT: ASME, Paper No. DETC2004/DAC-57141.

## Notes:

1 www.asc.nasa.gov/aboutus/trl-introduction.html

2 www.i-a-i.com

3 www.Cougaar.org

4 http://java.sun.com/products/javawebstart/

5 http://touchgraph.sourceforge.net/index.html