INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED 03 STOCKHOLM, AUGUST 19-21, 2003

CHARACTERISTICS OF A MULTI-VIEWPOINT FEATURE-BASED DESIGN AUTOMATION ENVIRONMENT

KwangHoon Lee and Chris McMahon

Abstract

This paper describes an approach to viewpoint-dependent feature-based modelling in computer-aided design developed for the purposes of supporting design automation. The approach uses a combination of a multi-level modelling approach and two stages of mapping between models. The multi-level model approach is implemented in a three-level architecture involving (1) a feature-based description for each viewpoint, comprising a combination of form features and other features such as loads and constraints for analysis, (2) an executable representation of the feature model, for example, in the form of an executable macro file or object model and (3) an "evaluation" of the feature model obtained by executing the representation defined in (2) in a CADCAM system. The mappings involved in the system comprise firstly, mapping between the level (1) feature representations associated with different viewpoints, for example for the geometric simplification and addition of boundary conditions associated with moving from a design model to an analysis model, and secondly mapping between level 1 and level 2 representations in which the feature model is transformed into the executable representation. Because an executable representation is used as the intermediate layer, then the low level evaluation can be active – for example an analysis model which is evaluated and for which results are output.

Keywords: design representations, computer-aided design, feature-based design, design automation, distributed design

1. Introduction

A number of computer-aided design research activities have focused particularly on the problems involved when integrating various previously separate systems and representations into a framework that covers all major aspects of the automation of the product development phases. During the design process, different engineers take information about the design, and manipulate it in order to generate the new information required for the development of the new product. Various engineering representations of the designed artefact are created and used for different engineering tasks. These represent, for example, a manufacturing engineering viewpoint, a structural analysis viewpoint, a process engineering viewpoint, and so on. These representations for the various engineering requirements are obtained by translation between engineering models, augmenting them as necessary with information specific to the specialist viewpoint [1].

The participants in the engineering design process produce information that represents modelled properties of the design. These modelled properties may be divided into two main classes: design parameters that describe the shape, dimension, surface finish and other attributes of the product for subsequent manufacture, typically represented in drawings, diagrams and CAD models; and performance parameters that describe the characteristics and behaviour of the products subject to an external environment that applies loads and other

boundary conditions to the products [2]. The design parameter model is what the design engineer produces, generally today in the form of a CAD model. Specialist engineers, generally using further models –such as Finite Element models – that are used to assist in their estimation, estimate the performance parameters during the design process. The additional models may be computer models, mathematical models, graphical models and even physical prototypes and test pieces.

A significant issue in design is the amount of time and effort that is spent first of all by design engineers in creating CAD models of design parameters and then by specialist engineers in creating their models, based on the design parameter models, so that they may make their specialist judgements. There are various reasons why the work is time-consuming: creating a new design model from geometric primitives can be quite laborious; the modification that is often needed to the design parameter model by a specialist – for example to simplify or approximate the model – can also involve a lot of effort. The manipulation of the model also has to be done at the level of low-level geometric entities such as faces and edge loops, and this contributes to the time taken. However, if model representations can be used that facilitate the automatic or semi-automatic translation between representations, then a good deal of the required effort may be eliminated. Furthermore, if a high-level representation is used, then it may be possible to create design models very rapidly, thus allowing thorough exploration of the design space. An approach to such a high-level representation is multi-viewpoint feature-based design – computer-aided design using features that are compatible with multiple engineering viewpoints.

In this paper a multi-viewpoint feature-based design approach for an automated design environment will be described. The work is a continuation of that presented in [3]. A brief overview of each of the elements of the system will be provided. The system approach is based on building a feature-based model of a design and mapping to executable representations of secondary viewpoint models. The approach to construction of models using the design-by-features approach involves controlling execution of a commercial CADCAM system through commands from the system's macro language with dynamic modification of the macro language by an external control program to allow different artefact parameters to be varied. Variable parameters used to describe a design feature model are defined and a probabilistic assessment of a design evaluated simply by repeatedly varying the values of the describing parameters.

2. Research background

The objectives of the work presented here were, firstly, to explore whether, by using features to construct design models and as a basis for specialist models, the models could be constructed much more simply so as to avoid lengthy initial construction, and to avoid time-consuming conversion between models. Secondly, it was wished to examine whether the construction and manipulation of features could be carried out programmatically so that complete design and analysis procedures might be incorporated into optimisation or probabilistic analysis processes. Each of these aspects will now be reviewed.

2.1 Approaches to multiple viewpoint feature-based design

Feature-based design

In feature-based design, models of the designed artefact are represented as collections of features - model elements that have engineering significance [4]. Feature based approaches encompass both design using features (design-by-features) and automatic identification of

features on a conventional geometric model. In design by features, features are entities that can be modified by changing feature parameters, typically related to the geometric form of the feature. Feature-based design is regarded as a promising approach to design representation for various product development phases. Nevertheless, improvements are sought through increased capability for design (e.g. especially geometry specification and modification of complex shapes such as castings or pressed panels) and a better ability to act as the integration for manufacturing applications such as process planning, assembly planning and analysis.

Design by features comprises all operations that are part of the process of creating a feature-based model and transformation of the model to different applications. The approach provides two principal advantages in the design process in that (1) the designer can store in the feature model non-geometric information which is available at the design stage and can later be applied to various engineering domains, and (2) features can be used to access information associated with particular feature types during the design process. This makes it possible to implement advanced applications such as concurrent design, real time geometry modification, and integration with other applications. The general purpose of the modelling methodology is the support of methods for automatically generating models for carrying out specialist activities, especially engineering analysis tasks. In general, however, the geometric and other attributes of a component may be combined into features in a variety of ways that reflect the needs of different design and manufacturing applications at different product development phases. This is currently a problem in feature representation: different specialists may choose to use different feature sets for the same part because they assign different engineering meaning to the elements of the part. This is known as viewpoint dependency. In this regard there is a need for feature definitions, or for a method of mapping or converting features, that allows wide coverage of different viewpoints in the design process.

Representations for viewpoint-dependent models

There have been a number of approaches to the representation of multiple viewpoints in design by features. An algorithm has been developed by Jha [5] that propagates feature modifications automatically across different domains. In this "feature-tree" algorithm, multiple feature models of the part are maintained, and changes made in one feature model are propagated to other feature models of the same part. The feature-tree makes available to the propagation algorithm a history of the extraction process that results in the multiple feature models. The modifications are however limited to the geometry (not the topology) of volumetric features. A mechanism for maintaining consistent product views in a distributed product information database was proposed by Hoffmann [6]. In his approach, a single repository called a "master model", in which all-relevant product data resides was proposed for the integration of different product information domains while the other views must be updated to maintain consistency. The architecture builds on different applications distributed over different CAD or CAM systems, and a master model is also used to associate the model. Bronsvoort [7] proposes a multiple viewpoint feature modelling approach to overcome the shortcomings that current multiple viewpoint modelling is done only for form features. The approach supports conceptual design, assembly design, part detail design and manufacturing planning by providing a viewpoint-dependent interpretation of the product for each of these applications. De Kraker [8] describes a multi-level model approach in which viewpoint dependent models are defined at three levels: at the highest level is a feature-based description; at the intermediate level features are described by their canonical shape as a CSG-tree; at the lowest level is the evaluated geometry for the model.

2.2 Design automation.

Achieving design automation involves the integration of the information processing required by the various disciplines involved at the different stages of the design process [3]. For well-established designs, e.g. in the automotive industry, it may be possible to model the flow of information between different disciplines and stages, and to use these models to assist in the automation of aspects of the design process. For example, the optimisation or probabilistic design assessment of an engineering part may involve generation of a geometric model of the part, use of the model to generate an analysis model, and use of the analysis model to produce information to guide the modification of the geometric model. However, using current technology, considerable human input is needed to set up geometric models or parametric models for use in such activities, and then to use these models in the preparation of analysis models for the activities.

A major issue in reducing the human effort required in information processing in design is the use of model representations that are sufficiently semantically rich. Feature based modelling is regarded as a key technology in this respect. Computer-aided design with feature representation as the mechanism may be used to define and maintain product design information for analysis and simulation of products from the early stage of the product development cycle. By using a feature-based description of engineering parts, the automation of such processes as optimisation and probabilistic design may be assisted – the human input required in the modelling of the geometry and then in the preparation of the analysis model may be very much reduced.

Distributed design

Competitive markets require rapid product development, and product development over computing networks is a new development made in modern engineering design. Design today is undertaken in a distributed fashion by teams that may be on multiple sites. The computing environments used are also distributed, with computers connected in networks that allow them, in principle, to communicate with other machines all over the world. Software approaches have been developed to exploit these distributed networks. In particular, processes may be executed on one computer under control of another computer, allowing different modelling, analysis and simulation components to be distributed over a network with heterogeneity of computer platforms and languages, and to be executed as desired to achieve design automation by automatically invoking processes in an appropriate order. Hence, computationally expensive computing activities such as optimisation and probabilistic design may be distributed in the most appropriate way. For this reason, large-scale design work related to the product development life cycle is moving to distributed computing. The distributed system can be thought of as a design environment that provides support for collaboration in the product development stages. However, in order to coordinate such activities, a mechanism for coordination of processes in a distributed working environment is required.

Tool Management Systems (TMS)

A TMS is a system for handling the flow of data and the control of multi-process execution for distributed design activities. It achieves this by invoking user-specified tools as necessary and by maintaining consistent versions of data in some problem-solving application. Generally, TMS employ an object-oriented approach and use such mechanisms as CORBA [9] or other distributed object system for data exchange and coordination. Examples of TMS are NetBuilder [10], BossQuattro [11] and ADAPRES_NET [12].

In a TMS, a standard interface to the controlled applications should be provided to allow them to be invoked and for data transfer to take place. The terms "tool integration" and "tool encapsulation" are used to describe alternative tool control procedures. Tool integration describes the process by which tools are directly executed under control of a TMS, for example via the tool's API, source code or some other mechanism. In this way, the tool can directly obtain data to operate. In tool encapsulation, software wrappers between the TMS and the controlled tools provide an interface layer through which tool control and data exchange takes place.

Coordination of the controlled tools is typically based on a network model of the required sequence of tool executions. For example, ADAPRES_NET effects the control using a Petri net model of the interactions. TMS typically provide inbuilt control mechanisms allowing optimization and sensitivity analyses to be carried out under the control of the TMS.

3. Research methodology.

3.1 Multi-level feature mapping approach to design automation.

In order to use features to construct design models and as a basis of specialist models, it is necessary to be able to specify feature models and their manipulation into different viewpoints in such a way that they may be constructed and manipulated easily. Ideally, some sort of language would be provided for the specification of features, and of the way in which they are assembled together into a model and then manipulated for different applications. This language would provide mechanisms for the definition of features, for the combination of features into a model, for the manipulation of features (simplification, approximation etc.) and for the association of such aspects as loads and boundary constraints with a feature model. These mechanisms eventually have to be translated into operations for a CAD system. So, for example, a feature model of a link, which comprises an "eye", a "shank" and another "eye" needs to be translated into the system commands needed to construct these features. When the model is then manipulated into a FE model, then there may need to be simplification or approximation of the feature models and then loads and boundary conditions need to be applied to the features of the design model. In practice this means firstly that a different feature model may need to be created with alternative geometric representation of the features (to account for the simplification/approximation) and secondly that loads and boundary conditions need to be applied to the features, or more precisely to the faces of the features. The viewpoint-dependent manipulation should involve execution of CAD system commands in some way such that the appropriate geometric entities and FE entities are created.

Viewpoint dependency means that the features needed and the way in which they are defined may vary with the engineering viewpoint. Feature mapping is therefore seen as an important requirement for the flexibility of feature-based design systems including multiple viewpoint modelling [2]. The approach that has been adopted in this work has been to develop an approach for viewpoint-dependent feature modelling, called Dynamic Feature Evaluation [DFE] and implemented using a commercial CADCAM system (SDRC I-DEAS) in a distributed automation environment [13]. The approach uses a combination of mapping between representations and a multi-level modelling approach, similar to that employed by de Kraker [8], but here comprising:

1. A feature-based description for each viewpoint, comprising a combination of form features and other features such as loads and constraints for analysis.

2. An executable representation of the feature model, for example, in the form of an executable Macro file or object model.
3. An "evaluation" of the feature model obtained by executing the representation defined in (2).

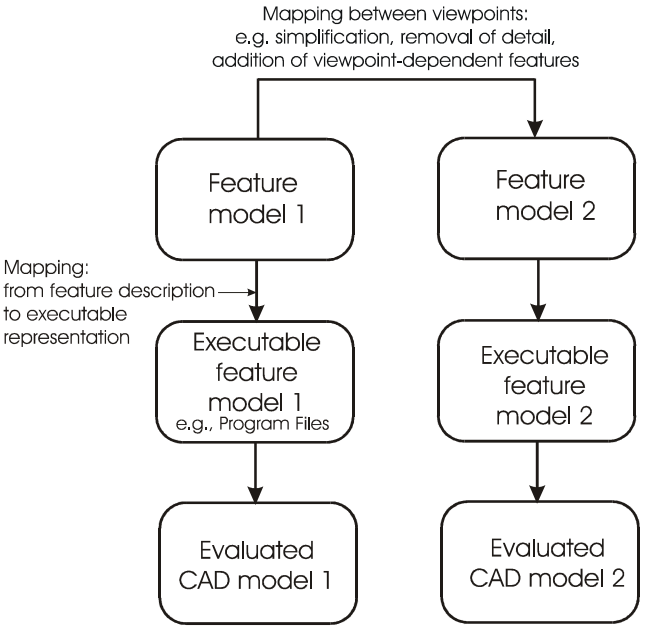

Figure 1. *Architecture and mappings of the multi-level feature-based modelling system*

This three level architecture is shown in figure1, and there are two sets of mappings associated with this architecture: firstly, mapping between the level 1 feature representations, for example for the geometric simplification and addition of boundary conditions associated with moving from a design model to an analysis model, and secondly mapping between level 1 and level 2 representations in which the feature model is transformed into the executable representation. Note that because an executable representation is used as the intermediate layer, then the low level evaluation can be active – for example an analysis model which is evaluated and for which results are output.

The work reported here involves both mappings, although it has concentrated on the second of the two mappings. It is assumed that techniques such as that described by Kugathasan [14] would be used for complex mapping of form features between viewpoint representations. Simple mapping, limited to addition of analysis features, will be described. The question that has been primarily addressed concerns how feature models should be mapped to executable intermediate representations, and then how these representations may be used in design automation applications. This has been done through experiments that explore the automated construction of feature models from different viewpoints.

3.2 Design automation with a feature-based model.

Automated construction of feature models requires the feature description to be translated into an executable set of commands for the CAD system to evaluate. It was considered that this could be done in a number of ways: by instructing a CAD system application to execute commands one-by-one, by translating the feature description into code for the system's API to execute, or by translating it into the system's macro language (that in the case of I-DEAS is called Program Files and uses command mnemonics for automated system operation). All

three approaches were explored, but the first method is cumbersome, the system used did not support a full set of finite element operations through its API, and therefore the mnemonic approach was concentrated on. It is that approach that will be described here.

The use of a macro language for the evaluated feature model is achieved by first creating mnemonic sequences for features and for feature operations either by writing them directly or by capture of interactive commands and then editing the captured files as required. Mapping from feature model to macro language is then achieved by creating a file combining code segments corresponding to features and other operations. Figure 2 shows the code segments corresponding to the development of a finite element model of a bicycle crank.

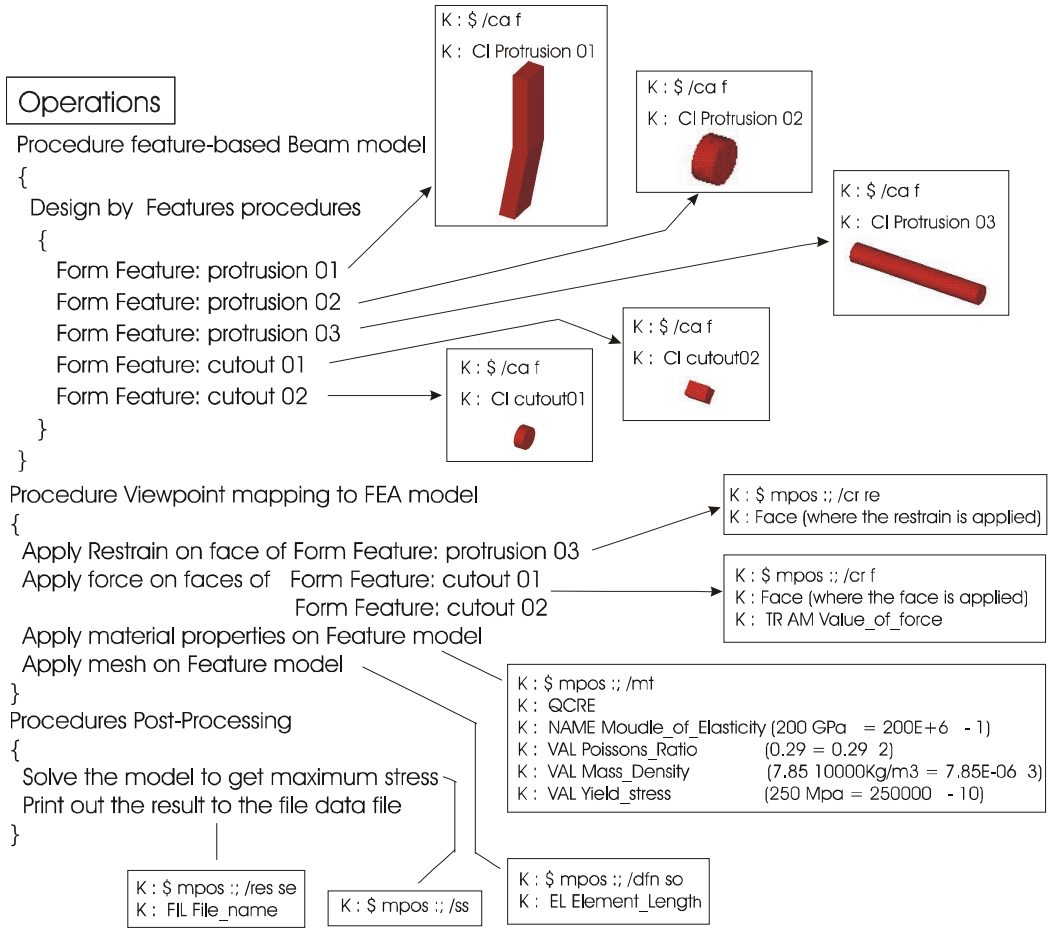

Figure 2. Macro operations for the FE model of a bicycle crank

In this example, a finite element analysis feature model is represented as a collection of executable commands describing feature modifications, feature parameters and locations, and then each step of the mapping is described in terms of operations on the whole part (e.g., meshing, materials), or on named elements of features (e.g. loads, constraints applied to specified faces), all described using executable mnemonic commands. In this way, repetitive design automation tasks (e.g. repeatedly constructing models for the purposes of optimisation) may be carried out by writing, modifying and executing mnemonic files.

A further advantage of using a macro file approach is that model variables may be modified easily by editing the file, and that this may be done programmatically. Distribution of execution may also be achieved easily, simply by instructing a remote workstation to execute a program file. This approach to remote execution also reduces the necessary network traffic.

There are negative aspects of using macro files. The most serious difficulty (shared with the other approaches) was that, for the system used, it was not possible to use user-supplied persistent names for geometric entities (e.g. faces of features) to be used in later manipulation of the model. System-supplied names could be used, but these were not persistent and keeping track of them was difficult. Also, the mnemonic files are difficult to read, and macro execution is potentially much slower than programmatic execution using an API. Nevertheless, unless absolute performance is an issue (and it may be in some optimisation applications), macros offer a convenient way of constructing feature models.

Macro and command-by-command approaches also require different ways of arranging for the distributed execution of tasks. In each approach presented here the architecture is that a central computer application works at feature model level, and generates and accumulates the data used for example for optimisation or probabilistic analysis. Construction of the CAD models, execution of finite element analyses and so on is carried out by CAD system software, either on the same or separate workstations. The nature of the interaction is different between the two applications however. In the command-by-command approach there is continuous communication between the central controller and the CAD system command server, and the controller passes a stream of commands to the remote processes. In the macro approach the central controller again deals with the optimisation or probabilistic analysis data, and uses this data to create or edit macro files, which it then passes for execution to the remote process. In this way the process is more asynchronous, and the central controller could manage many remote processes simultaneously.

An example of repeatedly evaluated part geometry is shown in Figures 3 and 4, which show the features and key parameters (values and standard deviations) respectively for a connecting rod. In order to explore the likely weight variation of the connecting rod owing to variation in the dimensions, models of connecting rods were repeatedly constructed for different values of the feature parameters in order to define a response surface for the weight of a rod, and then this response surface was used in a Monte Carlo simulation to compute the weight of the rod as a probability distribution function.

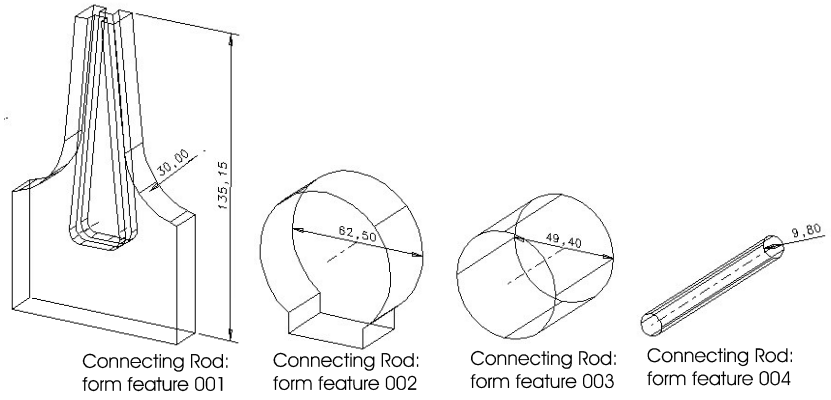

Figure 3. *Form features of Connecting Rod*

4. Conclusions

The experiments that were carried out demonstrated how features models could be created programmatically, using a variety of interfaces to a standard CADCAM system. Initially, purely design features were used, and then it was shown how a feature-based model could be taken and extended to another viewpoint (e.g. a finite element model or geometric analysis

model) using the same approaches. From these experiments, it was understood not just how to take one feature configuration and develop another viewpoint, but also in principle how different parts can be modelled using the same feature set and again further viewpoint models developed for these parts. The outcome of the work was a multi-stage system for the construction and execution of feature-models comprising:

- parametric design to construct generic features;
- complete feature definitions for design, finite element analysis and geometric analysis viewpoints;
- design by features for the automated design method for all feature types by a macro approach, operated in a distributed design environment in which different workstations in a network can create models from different viewpoints;
- a multi-level mapping operation for feature description and executable representation;

The work has shown that a framework for collaboration allowing the generation of viewpoint models for different product development stages is possible, but that in order to achieve this it is necessary to properly structure product information flow, and enhancements to CAD environments are needed, in particular a capability to attach persistent names to geometric primitives of features, if a full range of functions are to be supported. Since one of the main advantages from using feature technologies over conventional geometric modelling is the ability to associate functional and engineering information with aspects of product models these enhancements will be important if the full benefits of computer-aided engineering are to be achieved.

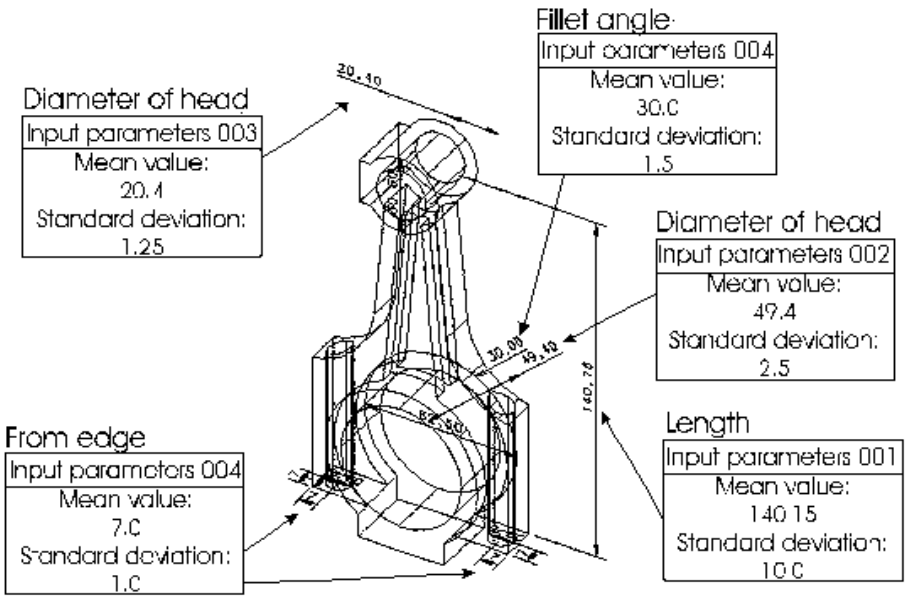

Figure 4. *Probabilistic design viewpoint of geometric analysis model.*

References.

[1] MG-IT Coordonnateur: Lèon, J.C., "Contribution to a multi-views, multi-representations design framework applied to a preliminary design phase", IDMME '98, 2nd International Conference on Integrated Design and Manufacturing in Mechanical Engineering, Compiè- France, May 27-29, 1998

[2] Suh, N. P. "The Principles of Design", Oxford University Press, New York, 1990

[3] Lee, K.H., Kaymaz, I., McMahon, C.A., "The use of distributed viewpoint-dependent feature-based modelling and the response surface method in design assessment", International Conference on Engineering Design ICED, Glasgow, August 21-23, 2001

[4] Shah, J.J., and Mäntylä, M., "Parametric and Feature-based CAD/CAM", Wiley-Interscience, New York, 1995

[5] Jha, K. and Gurumoorthy, B., "Automatic propagation of feature modification across domains", Computer-Aided Design, Vol. 32, pp 691-706, 2000

[6] Hoffmann, C.M., and Joan-Ariyo, R., "Distributed Maintenance of Multiple Product Views", Computer-Aided Design, Vol. 32, pp.421-431, 2000

[7] Bronsvoort, W.F., Noort, A., van den Berg, J., and Hoek, G.F.M., "Product development with multiple-view feature modelling", CD-ROM Proceedings of the IFIP Conference on Feature Modelling and Advanced Design-For-The-Life-Cycle Systems, 12-14 June, Valenciennes, France, 2001

[8] De Kraker, K., D., Maurice, and Bronsvoort, W. F., "Multi-way feature conversion to support concurrent engineering", Solid Modelling '95, third symposium on solid modelling and application, Salt Lake City, Utah, May 17-19, 1995

[9] Henning, M. and Vinoski, S, "Advanced CORBA(R) Programming with C++", Addison-Wesley, Reading, MA, 1999

[10] Dabke, P., Cox, A., and Johnson, D., "NetBuilder: an Environment for Integrating Tools and People", Computer-Aided Design, 30, 1998, pp.465-472

[11] Samtech, "BOSSquattro", http://www.samcef.com/pdf/boss-va.pdf

[12] Kaymaz, I., "An Adaptive Response Surface Method for Engineering Analysis", Department of Mechanical Engineering, University of Bristol, UK, PhD Dissertation, 2001

[13] Lee, K.H., "Engineering design representation by feature-based design in design automation – multiple viewpoint dependent models in product development" Department of Mechanical Engineering, University of Bristol, UK, PhD Dissertation, 2002

[14] Kugathasan, P., "A feature-based approach to design information management – Multiple viewpoint dependent models for the product introduction process", Department of Mechanical Engineering, University of Bristol, PhD Dissertation, 1998

Chris McMahon
University of Bath
Department of Mechanical Engineering
Bath BA2 7AY
UK
Tel: Int +44 1225 384026
Fax: Int +44 1225 826928
Email: c.a.mcmahon@bath.ac.uk
URL: http://staff.bath.ac.uk/enscam/

KwangHoon Lee
Kwangju Institute of Science and Technology
(K-JIST), Department of Mechatronics
1 Oryong-dong, Puk-gu
Kwangju, 500-712
Republic of Korea
Tel: Int +84 11 9147 8555
Email: mekl2002@intizen.com