Model Compression for Design Synchronization within Distributed Environments

S. H. Bok[1], A. Senthil Kumar[2], Y. S. Wong[3], A. Y. C. Nee[4]

[1] National University of Singapore, engboksh@nus.edu.sg
[2] National University of Singapore, mpeask@nus.edu.sg
[3] National University of Singapore, mpewys@nus.edu.sg
[4] National University of Singapore, mpeneeyc@nus.edu.sg

ABSTRACT

Today's product development practices occur as ad-hoc fragmented value chains across distributed environments. They increasingly require distributed collaborative design capabilities for companies to cooperatively design products to be competitive. There are thus specific middleware mechanisms and capabilities needed to appropriately distribute functionality and data across the network in a complex heterogeneous computing landscape. These are part of the fundamental problems of concurrency and synchronization at various system and support levels. This paper addresses one aspect of the design synchronization problem in digital product modeling and associated design changes due to shape editing to provide timely support in distributed collaborative design. This involves exploiting the geometric/model compression technique for effective visualization and interaction.

Keywords: Distributed Collaborative Design, Design Synchronization, Model Compression

1. INTRODUCTION

Collaborative engineering is a key concern in the global engineering economy. It requires a concerted and continuous combination of product design, management and planning, and realization activities amongst dispersed participants who need to engage early and frequently to work on what-if scenarios amidst different engineering domains, disciplines and perspectives with the associated processes and resources.

This means accessing and using complex, large remote design models and associated datasets and interacting with customers, stylists, suppliers, and domain experts. Each of these groups of people uses different heterogeneous tools to manipulate and evaluate changing product models. Suitable computing infrastructures and associated technologies involving tools and applications are required given the decisive development and potential of the Internet as a collaboration medium in a heterogeneous hardware and software landscape. These are termed as middleware [12] [18]. Involved with this are challenging problems relating to inter-operability, appropriate product and information models and their distribution, and efficient quality communication, all attributable to complete design synchronization support [13][23].

More specifically, the design cycle of complex products has come to rely on upon very complex and repetitive flows of information between the various design groups across varying distributed environments. This often reflects resolving conflicts through such cycles as teams progressively refine constraints. These activities are full of references to the product features (e.g., shape and position of features, manufacturing and maintenance processes, etc.) and are thus difficult to carry out via electronic mail, voice mail, or telephone conversations with consistency.

Early and rapid design change in ad-hoc collaboration is thus always occurring and requiring timely communication and updates to dispersed participants in the product model context in as seamless and as generic as possible way. We note that such collaborations are essentially networks.

According to Wang [23], when a product is designed through the joint and collective efforts of many designers, the design process may be called collaborative design (it may also be called co-operative design, concurrent design and inter-disciplinary or even integrated design, though each term may introduce differentiation in technical requirements and objectives). This may include functions as disparate as design, manufacturing, assembly, test, quality and even purchasing from suppliers and customers.

More technically, collaborative systems can be generally defined as distributed multiple user systems that are both concurrent and synchronized [2]. Concurrency involves management of different processes trying to simultaneously access and manipulate the same data. Synchronization involves propagating evolving data among users of a distributed application, in order to keep their data consistent.

These concepts are generally rather demanding, their difficulty becomes particularly apparent within a collaborative design modeling framework, where the amount of model data that has to be synchronized is typically very large, and the concurrent modeling actions taking place may be very complex.

Specifically, to leverage the Internet, various domain-specific middleware capabilities are required in association with the appropriate formulation of distributed functionality and data architecturally speaking. As well, we note that there are several fallacies associated with networks such as network reliability, zero latency, limitless bandwidth, network security, and fixed network topologies. Such fallacies affect the problem of distributed collaborative design and provide valid consideration in providing the middleware capabilities and support. In particular, with design synchronization, large 3D models and datasets for visualization and interaction must be dealt with in terms of latency and bandwidth.

In Section 2, several collaborative design systems in research are reviewed. In Section 3, the proposed system architecture is briefly discussed as a framework. We refer to our past and present efforts in this area of providing for distributed collaborative design [6-9]. In particular, we have also indicated some related research involving downstream design synchronization in an Integrated Product and Process Design (IPPD) context, exemplified with fixture planning with some overlapping information [9]. This leads to Section 4 as a problem overview in this paper context and several considerations supporting the leverage of the model compression technique to provide ‘end-to-end’ capability supporting design synchronization in our research approach. Some results are presented in Section 5 followed by Section 6 on conclusions.

2. COLLABORATIVE DESIGN SYSTEMS SURVEY

Several approaches were developed in creating an integrated environment or frameworks for product and process design based on traditional standalone systems. One approach is the use of standard file formats such as STEP and IGES for CAD models located at central databases. Roy [20] proposed a Web-based collaborative design framework for the use of a translator to convert CAD models into VRML based models which can then be viewed over the WWW onto such standalone systems. The VRML models are stored in an existing product data repository. The translator resides on a main central server and can be accessed remotely by a designer. Most of such frameworks including [15, 10] are regarded as under proof-of-concept development stage [23].

One drawback in the use of standard file formats is that the approach provides a static (rigid) interface to applications [8]. It is also important to note that although VRML is treated as a neutral representation, it in itself is not a geometric model and can only be used to display geometric models with no editing capability [23]. We also wish to note that like HTML for 2D web page layout and display for publishing, VRML is the equivalent of 3D display or multimedia publishing. It is also not an appropriate method for product model and data representation. Technically it also mirrors a static scene graph approach which is not very efficient and cumbersome in the realm of design changes. In summary, these are drawbacks to synchronization.

For more relevant means to design synchronization, we note the efforts of Hoffman [9] with their net shape association to monitor design changes. Our work [15] is similar with more detail reported therein. Notable is the role of the server-based cellular model coupled with semantic feature modeling and feature conversion for design synchronization, as report by de Kraker [5] and led by Bronsvoort. Related is the extension work on the Internet, WebSPIFF, carried out by Bidarra [2]. We note that design synchronization on an ‘end to end’ principle that is from server to client has not been discussed elsewhere in the distributed and collaborative design to the best of our knowledge. In particular, the role of interactive 3D facet models integral to design changes effected from a geometric modeling server is not exploited. We use the next 2 sections to introduce architectural considerations to formulate distributed functionality and associated data for supporting synchronization.

3. DISTRIBUTED DESIGN ARCHITECTURE

The requirements in a distributed and collaborative design context lead almost inevitably to the adoption of a *client-server or more generally distributed computing* architecture, in which the server provides the participants in a collaborative design session with the indispensable communication, coordination and data consistency tools, in addition to the necessary basic modeling facilities.

A recurrent problem in client-server systems lies in the conflict between limiting the complexity of the client application and minimizing the network load. In a collaborative design context, client complexity is mainly determined by the type of modeling and interactive facilities implemented at the client, whereas network load is mainly a function of the kind and size of the model data being transferred to/from the clients [5]. Thus, unless special or specific measures are introduced, the abovementioned conflict cannot be optimally balanced or compromised. We note that either extreme of so-called thin clients and fat clients, respectively presents the problem of heavy network traffic for unintelligent image rendering, and that of massive data inconsistencies. The former is not a novel approach and the latter is too close to the condition of standalone systems. Both pose extreme synchronization issues without attempting to provide appropriate middleware capabilities.

In conclusion, the principle for a good compromise to such difficulties is a client-server approach, where the server coordinates the collaborative session, maintains a shared model and repository, and provides all

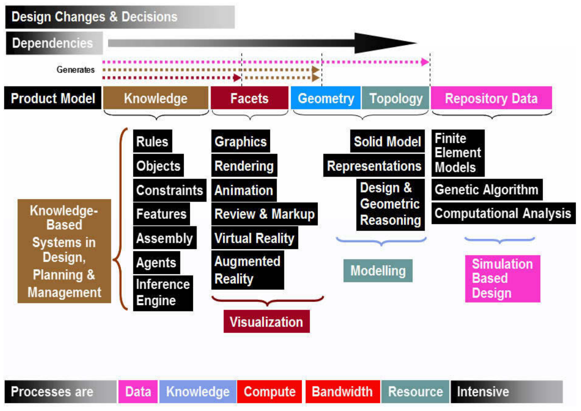

Figure 1: Product Model-Centric Characteristics

functionality that cannot, or should not, be implemented on the client. The clients then perform operations locally as much as possible, and only high level semantic messages, and limited amounts of information necessary for updating the client, will be sent over the network. This keeps the network load relatively low, while guaranteeing good client interactivity at acceptable response times. An important advantage of this architecture is there is only one product model in the system. Clients send their modeling operations to the server, and receive feedback after any such operation has been performed on its central feature model, avoiding inconsistency between multiple versions of the same model.

Figures 1, 2 and 3 illustrate the general consideration of a product data centric architecture, a system architecture and framework to support distributed collaborative design and the ‘network stack’ of middleware layers. We note that design changes should now be seen to generate and drive the essential functionality and data in distributed collaborative design with the corresponding need to handle large complex sets of models and data. Such design changes occur in distributed environments involve different users and heterogeneous tools and formats.

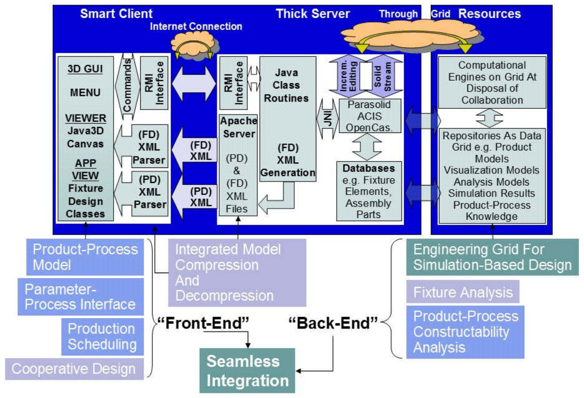

Figure 2: Overall System Architecture

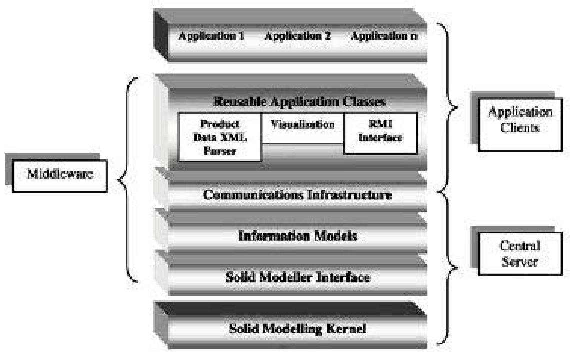

Figure 3: Middleware Perspective and Layers

Accordingly by middleware design, we have developed suitable and reusable Java classes and interfaces. More information can be found in [14-15], [17], [22]. By functionality and data considerations, this approach accounts for the problems of compatibility and inter-operability so that we may develop an extensible computing environment supporting seamless integration. This refers to the Java-based geometric kernel modeling interface, the use of Extensible Markup Language (XML) to model and represent an augmented product data representation, and on the client side, the interfaces for extension into a modeler independent association relationship management capability to help propagate design changes and maintain domain-specific application view consistency, similar to [9] and discussed in [15] to achieve adaptive process responses and solutions.

In the context of this paper, we would like to deliberate on the 'end-to-end' requirement for design synchronization given the augmented product data XML representation conceived and used in exchange between server and client. We note now that interactive graphics or facet models for visualization are crucial in a distributed design situation for this synchronization.

4. MODEL COMPRESSION FOR DESIGN SYNCHRONIZATION

In this section, we review computer graphics simplification techniques and evaluate relevant issues related to distributed collaborative design. We then arrive at the choice and leverage of the model compression algorithm for integration to support 'end-to-end' design synchronization.

Interactive 3D computer graphics play an important role in human-computer interaction in design and manufacturing, amongst many other areas. In many of these applications and a general context, human productivity or satisfaction would be significantly enhanced by the possibility of immediate or rapid access to remotely located 3D data sets for visual inspection or manipulation.

3D computer graphics are dominated by polygonal or facet models due to their mathematical simplicity. This results in simple and effective rendering algorithms which embed well in computer hardware leading to widely available graphics accelerators. The number and complexity, measured by the number of facets and vertices, of these 3D models and data sets is growing rapidly, due to improved design and model acquisition tools. This growth seems to be faster than the ability of graphics accelerators to render them interactively. As well, anticipated increases in phone and network bandwidth will not, by themselves, suffice to offset the explosion of complex 3D models. This observation is applicable to design and manufacturing as products have grown in complexity and given the need to carry out distributed collaborative design. Thus there is always a need for product models as parts and assemblies to be viewed and edited interactively. This has created the need for polygonal simplification techniques [4].

Briefly, these methods can simplify the polygonal geometry of small, distant, or otherwise unimportant redundant portions of the model, seeking to reduce the rendering cost without a significant loss in visual content such as in flight simulation. Alternatively, they can reduce model complexity without introducing geometric error such as in volumetric information stemming from medical imaging useful for surgical simulation. In the case of complex engineering analysis and simulation problems that require a model to go through subdivision or partitioning, simplification is employed to remove unnecessary geometry. If the problem is to improve runtime performance in visualization by simplifying the polygonal scene being rendered, the most common use of polygonal simplification is to generate levels of detail (LODs) of the objects in a scene [12]. By representing distant objects with a lower LOD and nearby objects with a higher LOD, applications from video games to CAD visualization packages can accelerate rendering and increase interactivity. In the latter case, this would be evident in a factory simulation situation involving spatial factory design and planning where losses in geometric accuracy and detail can be tolerated.

However, in the context of distributed collaborative design, we have noted that frequent design changes and updates need to occur across distributed environments. From the perspective of synchronization, simplification techniques involving time consuming preprocessing efforts to generate multiple LODs would not be suitable as multiple time-consuming updates are incurred in distributed collaborative design, even though this is has been termed as progressive transmission or streaming [7]. LODs also severely compromise the geometric and visual fidelity required in design and would not be advisable especially when co-design involving distributed teams members take place.

In addition, simplification techniques that drastically allow for topology modification, compromise or loss [6, 21] are also inappropriate in distributed collaborative design as this would also create misunderstanding of the original topology in the B-rep model. The CAD model for a product design would become grossly misinterpreted when design features are 'lost' during communication. We also note that unlike say flight or factory simulation, collaborative design requires more static or passive as opposed to dynamic scene model viewpoints.

Nevertheless we should indicate that LODs have been a key aspect in the design specification of the Virtual Reality Markup Language (VRML) standard and other programmatic scene graph techniques for visualization. Their original context being more related to multimedia uses would be inappropriate for distributed collaborative design involving complete descriptions and augmentation issues such as design intent and product-process modeling.

With the consideration and requirement for integration, distributed collaborative design capabilities in distributed environments require an approach to leveraging a simplification algorithm that is transmission or bandwidth friendly, can avoid losses or compromises as mentioned above and is able to accommodate product model representations and design changes. We thus require a simplification algorithm known as model or geometry compression that in general can take original highly detailed and complex models and reduce its size to a bandwidth-acceptable level of complexity without compromising visual and topology fidelity.

In general, combining different simplification techniques can still be relevant in the context of product assembly modeling and manipulation. This is because we can consider parts in a product assembly as either 'passive' or 'active'; the former not being contextually subject to design changes or shape editing – thus allowing compromises in fidelity, and conversely the latter, as is our present problem definition. If therefore there are large product assembly models to be handled, for say assembly simulation, the role of LODs for example might be useful.

Much of the work done in model or geometry compression is based on clever encoding of the topological relationship between nodes in the meshes. These encodings minimize the repeated references to nodes, thereby achieving a compact description of topology. An interesting observation made in meshes representing manifolds is that, on an average, the number of triangles is twice the number of vertices and each vertex is referenced in 5 to 7 triangles. Hence, a lot of research has concentrated on aggressive attack on the problem of encoding of topological relationship between vertices.

Early examples of compact encoding of a mesh were seen in triangle rendering engines such as OpenGL, in the form of triangle-strips and triangle-fans. A lot of research has been carried out in generating maximal triangle-strip decomposition of given meshes, minimizing the repetitions in the references to vertices. For our purpose, we note that Rossignac's work on the Edgebreaker algorithm achieves even greater compression by compact representation of topological relationship between vertices of a mesh [19].

The choice is made of model or geometric compression, that is basically loss-less and capable of high compression ratios, is relevant to the bandwidth and transmission constraints of the Internet as a shared resource and expedient to the need for synchronization for timely updates. To do that, it is proposed that the model compression algorithm should be integrated in a middleware framework for distributed collaborative design that may also support incremental design change. More specifically, integrated model compression across distributed environments requires incremental compression on the server side, followed by transmission and decompression into our augmented product data XML representations for use on the client end user side. To the best of our knowledge, this aspect has not been reported elsewhere in the distributed collaborative design context.

5. RESULTS & DISCUSSIONS

The Edgebreaker algorithm was mainly developed in the context of visualization of complete 3D models independent of the origin of those models [11]. Thus the compressed data format does not contain modeling information such as face tags and B-rep topological information that are useful as augmentation for product data representation and integral portrayal with the modeling kernel to effect distributed and collaborative design. The initial augmented product data approach using XML that was conceived has been necessary and sufficient enough to demonstrate the feasibility of Internet-enabled Fixture Design [22]. Design change and synchronization were not accounted for.

This approach, without the leverage of model compression and augmentation for distributed design, is similar to Hoffman although he had used the term 'characteristic point' and the concept of 'geometry certificates' in the Master model repository context as an abstraction from the geometric server.

We have integrated the Edgebreaker compression algorithm into our architecture. We have also modified the initial augmented Product Data representation's XML schema (Figure 4). The sequence of computing events for arriving at the Product Data XML Schema for complete models is shown in Figure 5. In general, such augmented information is important for engineering collaboration as clients interact with the modeling kernel and can also easily visualize, interrogate and interact on the client side. In a more advanced and relevant context, we note the value of semantic feature modeling [1] and its complement to our middleware oriented approach noting also that facets are standard descriptions with no compatibility issues. As facet models are characteristic of engineering and product models and need to be kept as a repository, it would be advisable to employ such compression methods to these repositories.

In this sequence of events, when a modeling operation is carried out, a tessellated mesh of the model is created by invoking a function call on Parasolid. The mesh data from Parasolid is then formatted as required for the Edgebreaker algorithm to work. The data required for Edgebreaker are the number ~~of ve~~rtices, the coordinates of the vertices, the number of triangles and the indices of the vertices that belong to each triangle.

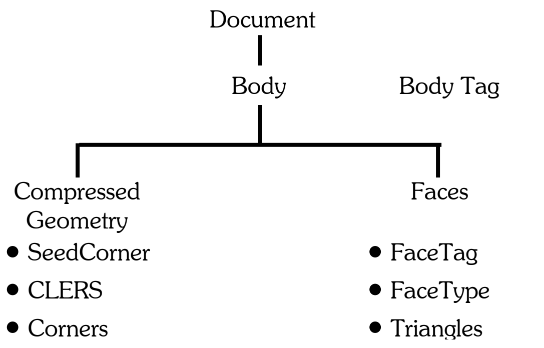

Figure 4: Revised Product data schema incorporating compressed geometry

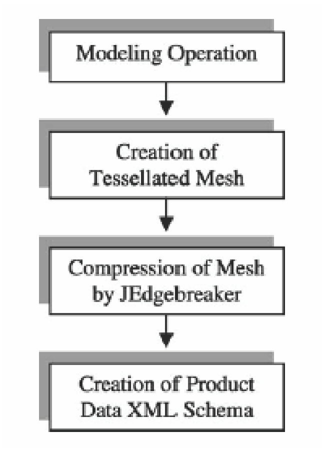
Modeling Operation
Creation of Tessellated Mesh
Compression of Mesh by JEdgebreaker
Creation of Product Data XML Schema

Figure 5: Basic integration and sequence of creating the augmented product data schema.

The coupling of 'model compressed' data with the augmented product data representation is logical and obvious given that design changes cause changes to the boundary representation and as such boundary and topological information extracted in such changes would have been transmitted across to the client side. The essential difference is the re-organization and compression format introduced into the augmented product data XML representation whilst respecting their linkage with the boundary information. With design changes caused by shape editing, this new schema will also allow for concurrent changes in both B-rep topological information and model compressed data.

We have verified the effectiveness of the Edgebreaker algorithm in reducing the data required for visualizing a 3D model by comparing the sizes of the augmented product data XML files with compressed geometry format and uncompressed mesh data. The results of our experiments are presented in Tables 1 and 2 for some basic primitive models and complex realistic product models (Figures 6 and 7). The experimental results show a significant compression of the data required, validating the effectiveness of using the Edgebreaker algorithm for model compression in reducing data sizes (Table 1). Initial timing tests to compare the visualization taken for without compression as in the prior system, and with compression with Edgebreaker incorporated are shown in Table 2. We note that this is for complete model compression, rather than for design changes. As well, these have not taken place in a networked context though caution should be applied to the consideration that the Internet has always been a shared traffic 'highway' of resources, one essential characteristic of collaboration. Adequate predictable performance would really relate to the concept and practice of 'Quality of Service' (QOS), an important research concern in network research. Still, it is clear that when no special measures are undertaken in this regard, the viability of distributed and collaborative design would be less, just as in the same regard, research on graphics simplification and compression.

Model Type	File Size		Compression Ratio
	Before Compression	After Compression	
Cube	5KB	1KB	5
Prism	7KB	2KB	3.5
Sphere	178KB	31KB	5.74
Torus	379KB	66KB	5.74
Chuck	492KB	79KB	6.23
Flange	137KB	39KB	3.5

Table 1: Size reduction tests with model compression

Model Type	Visualization Time		Percentage Difference (%)
	IFD (secs)	JEdgebreaker (secs)	
Cube	1.74	1.74	0.0
Prism	1.49	1.49	0.0
Sphere	6.73	3.53	47.5
Torus	13.07	4.00	70.0

Table 2: Timing tests for visualization

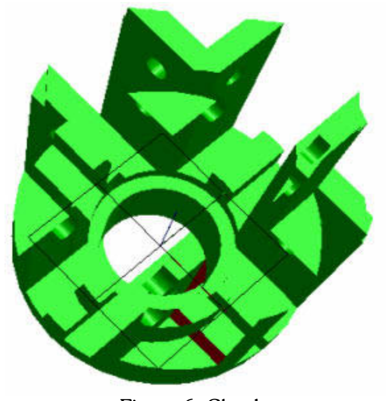
Figure 6: Chuck.

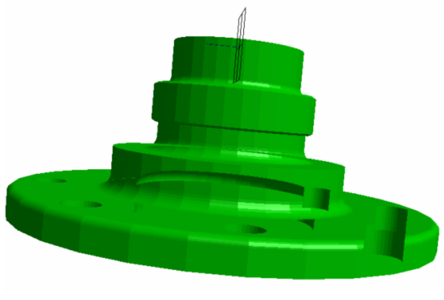
Figure 7: Flange.

6. CONCLUSIONS

In today's context of investigating the Internet as a medium for distributed collaborative design, the primary concern for seamless integration require a middleware oriented environment and architecture to address a number of issues. These cover compatibility, inter-operability, reusability and extensibility, and techniques on design synchronization and concurrency in general.

In this paper, we explain how in the formulation of distributed functionality and data, there is a need to handle 3D data sets inherent in the visualization and interaction of product models to account for timely updates and exchanges in the use of network bandwidths on the Internet. This resulted in the choice and testing of the geometric or model compression algorithm. We note the need to develop integrated incremental compression to support design changes. In an overall arrangement for effecting synchronization, we should note that timely updates may require a distributed coordination mechanism to ensure that relevant dispersed servers and clients are operating in a lock-step fashion.

7. REFERENCES

[1] Bidarra, R., and Bronsvoort, W. F., Semantic Feature Modeling, Computer-Aided Design 32 (2000) 201–225 202

[2] Bidarra, R., van den Berg, E. and Bronsvoort, W.F, 2001, Collaborative Modeling with Features, In: CD-ROM Proceedings of the 2001 ASME Computers and Information in Engineering Conference, 9-12 September, Pittsburgh, PA, ASME, NY.

[3] Blatecky, A; West, A; Spada, M., 2002, Middleware – The New Frontier. EDUCAUSE Review, Jul-Aug, 25-35.

[4] Cignoni, P.; Montani, C.; and Scopigno, R., "A Comparison of Mesh Simplification Algorithms," Computers & Graphics, vol. 22, no. 1, 1998, pp. 37-54.

[5] De Kraker, K. J.; Dohmen, M. and Bronsvoort, W. F., Maintaining multiple views in feature modeling. In 4th Symp. on Solid Modeling and Applications. ACM Press, 1997, pp. 123–130.

[6] El-Sana, J.; and Varshney, A., "Controlled Simplification of Genus for Polygonal Models," Proc. IEEE Visualization 97, IEEE CS Press, Los Alamitos, Calif., 1997, pp. 403-412.

[7] H. Hoppe, "Progressive Meshes," Computer Graphics (Proc. Siggraph 96), vol. 30, ACM Press, New York, 1996, pp. 99-108.

[8] Han, J.H. and Requicha, A.A.G., 1998, Modeler-independent feature recognition in a distributed environment, Computer Aided Design, 30(6), pp 453–63.

[9] Hoffman, C.M. and Joan-Arinyo, R., 1998, CAD and the product master model, Computer Aided Design, Vol. 30, pp. 905-919.

[10] Huang, G.Q. and Mak, K.L., 1999, Web-based collaborative conceptual design, Journal of Engineering Design, 10(2), pp 183-194.

[11] J-Edgebreaker, Triangle Mesh Compression Software in Java. http://www.igd.fhg.de/~coors/JEdgebreaker/

[12] Lindstrom, P. et al., Real-Time, Continuous Level of Detail Rendering of Height Fields, Computer Graphics (Proc. Siggraph 96), vol. 30, ACM Press, New York, 1996, pp. 109-118.

[13] Maropoulos, P. G., 2003, Digital enterprise technology: defining perspectives and research priorities. Int. J. Computer Integrated Manufacturing, 16(7-8), 467–478.

[14] Mervyn, F., Senthil, K.A.; Bok; S.H. and Nee; A.Y.C., 2003, Developing distributed applications for integrated product and process design, Computer Aided Design (in press).

[15] Mervyn, F., Senthil, K.A.; Bok; S.H. and Nee; A.Y.C., 2003, Design change synchronization in a distributed environment for integrated product and process design, Submitted to CAD'04, International CAD Conference and Exhibition, May 24-28, 2004.

[16] Pahng, G.D.F.; Bae, S. and Wallace, D., 1998, A web-based collaborative design modeling environment, Proceedings of the IEEE Workshops on Enabling Technologies Infrastructure for Collaborative Enterprises (WET ICE'98), pp 161-167.

[17] Ratnapu, K.K., 2001, Web Based CAD System, NUS Mechanical and Production Engineering. MEng Thesis.

[18] Richard E. Schantz and Douglas C. Schmidt, 2001, Middleware for Distributed Systems: Evolving the Common Structure for Network-centric Applications, Encyclopedia of Software Engineering, edited by John Marciniak and George Telecki, Wiley and Sons.

[19] Rossignac, J., "Edgebreaker: Connectivity Compression for Triangle Meshes," IEEE Trans.Visualization and Computer Graphics, vol. 5, no. 1, pp. 47-61, 1999.

[20] Roy, U., and Kodkani, S.S., Product modeling within the framework of the World Wide Web, IIE Transactions 1999; 31(7), pp 667–677.

[21] Schroeder, W., "A Topology-Modifying Progressive Decimation Algorithm," Proc. IEEE Visualization 97, IEEE CS Press, Los Alamitos, Calif., 1997, pp. 205-212.

[22] Senthil, K.A.; Bok, S.H.; Tan, B.C.; Kumar, R.K.; Nee, A.Y.C., 2000, The development of an internet enabled interactive automated fixture design system. Proceedings of the 7th International Conference on Mechatronics, edited by Charles Ume, pp. 1-7, 6-8 Sep 2000, Atlanta, U.S.A.

[23] Wang, L.H.; Shen, W.M.; Xie, H.; Neelamkavil, J.; and Pardasani, A., Collaborative conceptual design – state of the art and future trends, 2002, Computer-Aided Design, Vol. 34, pp 981-966.