Title:

Pseudo-Singleton Pattern and Agnostic Business Layer for Multi-Engineer, Synchronous, Heterogeneous CAD

Authors:
K. Eric Bowman, ericbowman@gmail.com, Brigham Young University
C. Greg Jensen, cjensen@byu.edu, Brigham Young University
Devin Shumway, devin.shumway@gmail.com, Brigham Young University

Keywords:
Heterogeneous CAD, Neutral Format, MESH CAD

DOI: 10.14733/cadconfP.2016.49-54

Introduction:

Engineering companies are sociotechnical systems in which engineers, designers, analysts, etc. use a wide array of software tools as they follow prescribed product development processes. The purpose of these amalgamated systems is to develop new products as quickly as possible while maintaining quality as well as meeting customer and market demands. Task speed up in a parallelized system can be modeled by Amdahl's law and so is governed by how much of a process can be parallelized [1]. Researchers at Brigham Young University have shortened engineering design cycle times through the development of synchronous collaborative CAD tools [8], [9], [13], [16]. Other research teams have shortened design cycle times by extending seamless interoperability across heterogeneous design tools and domains [2]–[7], [10]–[12], [14], [15], [17]–[19]. Multi-engineer synchronous (MES) collaboration across heterogeneous CAD environments is the focus of this paper. A logical architecture that supports both MES collaboration and interoperability is defined and tested for robustness and proposed as the start of a new standard for interoperability. In particular, a pseudo-singleton pattern is proposed to ensure data stability despite unordered data and a multi-engineer synchronous heterogeneous (MESH) object class pattern is proposed to allow heterogeneous clients to interoperate even if the server has no knowledge of the client. This architecture has demonstrated design and modeling interoperability between Siemens' NX, PTC's Creo and Dassault Systemes' CATIA CAD applications and interoperability between Siemens' NX and Dassault Systemes' CATIA are specifically demonstrated in this paper. The 2D point, 2D line, 2D arc, 2D circle, 2D spline, 3D point, extrude, and revolve features have been demonstrated. Complex models have successfully been modeled and exchanged in real time across heterogeneous CAD clients and have validated the architectural approach proposed for MESH CAD data storage.

Main Idea:

Jensen, Red et al. [20] developed NXConnect which provides real-time MES modeling between NX clients. This technology is currently under commercialization [10]–[12]. Cai created a multi-user SolidWorks experience using similar ideas to NXConnect [2]. Maher did similar research using AutoCAD [16]. While these systems support concurrent collaboration within a homogeneous CAD environment, they do not support a synchronous heterogeneous CAD design environment.

The ideal CAD environment would support concurrent collaboration with the union of CAD features across heterogeneous CAD systems. The purpose of this research is to define a neutral format which merges the principles used to create a MES CAD system with those used to create a heterogeneous CAD system to create a new MESH architecture.

Bowman et al. have developed a neutral parametric database (NPDB) that serves as a neutral storage format for CAD data. [7] It fulfills key requirements that previous neutral formats have not. First, it is based off of the mathematical definition of parametric features rather than the CAD system's proprietary definition. Second, the neutral parametric database is normalized to prevent update anomalies. Third it has been shown to be compatible with an object oriented class structure mapping and finally, it was implemented using industry best practice tools and methods.

The singleton pattern is a pattern of design in software engineering where the instantiation of a class is limited to the creation of one object. This is done primarily to eliminate the use of global objects and variables but still allow the programmer to access static data. It also allows the user to implement interfaces which allows them to pass the singleton as an object into functions. This ability is the main difference that separates the singleton pattern from a static class.

Both MUS homogeneous CAD and single-user CAD translation software exist in the literature, however there has not been MESH CAD software demonstrated. In order to extend multi-user homogeneous CAD software there are a number of problems that must be solved. First was the development of a new data storage standard capable of storing heterogeneous CAD data easily while avoiding update anomalies. This has been developed by Bowman et al. and is known as the NPDB. Another two key problems are first that a MESH client must be able to map a flat list of feature commands to an associative feature tree-structure and second that the system be client-agnostic.

In order for a system to support parametric CAD models properly, it must represent them as a directed acyclic graph (DAG) where the nodes are features and the edges are parent-child relationships between features as shown in Fig. 1.

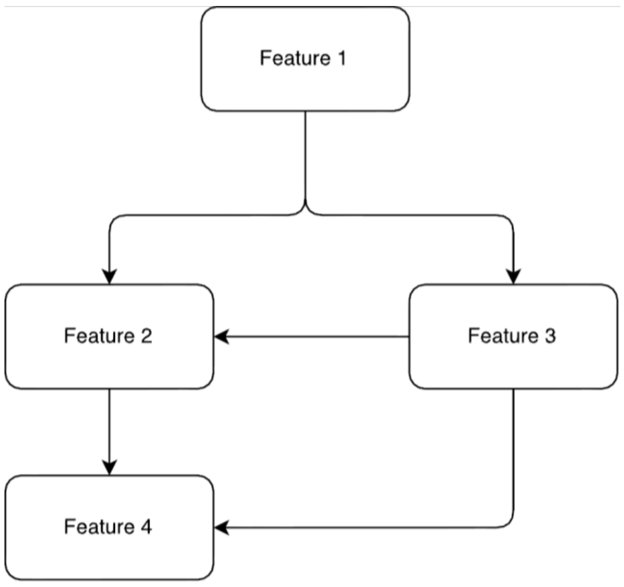

Fig. 1: CAD Data should be represented as a Directed, Acyclic Graph.

This is difficult in a client-server architecture because computer queries typically return a list of features with no regard for their dependencies. The listed features not only don't clearly display dependencies but they can also be returned in a random order. For example, if you queried all of the features from **Fig.** the result might look like the following numbered list:

1. Feature 2
2. Feature 3
3. Feature 4
4. Feature 1

Unfortunately, the part model cannot be built in this order because feature 4 depends on features 2 and 3, feature 3 depends on feature 1, and feature 2 depends on feature 1. So, the order this model must be built is as follows:

1. Feature 1
2. Feature 3
3. Feature 2
4. Feature 4

Any other build order would result in a fatal application exception. Finding the correct build order for a part is a critically important and difficult task.

The intuitive way to solve a problem like this is to explicitly find and maintain the correct feature order using consistency managers, but the logic for this type of architecture is difficult to develop and must be changed whenever feature definitions are modified. It also requires significant computation time to determine the dependency order and significant memory to store all dependencies for all features of all parts within the assembly. This approach is unlikely to scale to a large assembly with hundreds of thousands of parts and millions of features.

Another way to solve this issue is through the pseudo-singleton pattern which is a development from the singleton pattern discussed in the background. The pseudo-singleton pattern makes its class constructor private and only allows uniquely identified instances of itself to be queried. If the uniquely identified instance exists it is returned from a static dictionary. If the instance has not been created the class queries the server for the instance, adds it to the static dictionary and returns it. In this way feature instances can be used by a client developer without taking concern for their associations. In other words, any feature built from the pseudo-singleton pattern maintains its own dependencies with no further logic. Sample code that illustrates this pattern is shown in Fig. 1.

```
public class PseudoSingleton
{
    private static Dictionary<Guid, PseudoSingleton> Instances;

    private PseudoSingleton(Guid Id) { }

    public static PseudoSingleton GetInstance(Guid Id)
    {
        if (!Instances.ContainsKey(Id))
            Instances.Add(Id, new PseudoSingleton(Id));

        return Instance[Id];
    }
}
```

Fig. 1: The Pseudo-Singleton Pattern.

The utility of this pattern can be demonstrated by assigning features to the DAG shown in Fig. 2. Below in Fig. 2 is a DAG representation of a part containing a coordinate system (CSYS), a 3D Point built from that CSYS, another point built from the CSYS and relative to the first point, and a line connecting the two points.

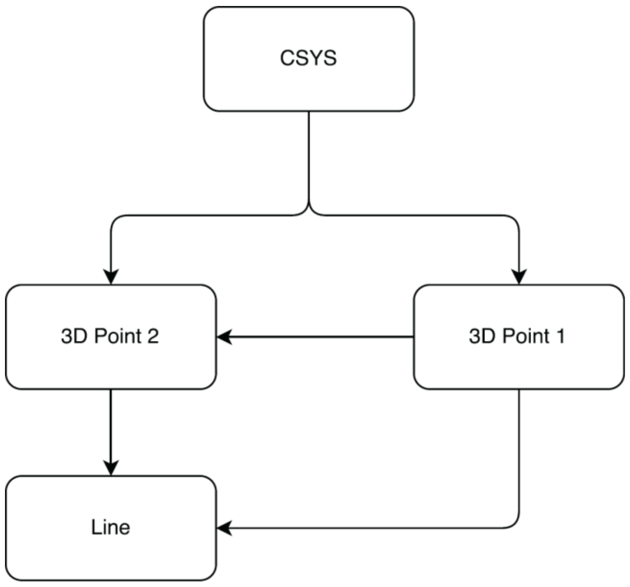

Fig. 2: Sample CAD Part DAG.

Assigning those features to the queried feature list mentioned above we would get the following list of feature messages:

1. 3D Point 2
2. 3D Point 1
3. Line
4. CSYS

The first feature, *3D Point 2* has a dependency on *CSYS* and a dependency on *3D Point 1*. When this message is processed, *3D Point 2* calls the "GetInstance" method, passing in *CSYS's* GUID. Since *CSYS* is not yet in the feature dictionary its constructor is called, it is added to the feature dictionary and it is returned to *3D Point 2*. *3D Point 2* then calls the "GetInstance" method passing in *3D Point 1's* GUID Since *3D Point 1* is not yet in the feature dictionary its constructor is called. *3D Point 1* is built off of *CSYS* so *3D Point 1's* constructor calls the "GetInstance" method passing in *CSYS's* GUID. Since *CSYS* is already in the feature dictionary it is simply returned. *3D Point 1* is then added to the feature dictionary and returned. The dependency graph for *3D Point 2* has automatically been assembled. The point is created and added to the feature dictionary.

The second feature, *3D Point 1* has already been placed in the feature dictionary so when the loading algorithm reaches its message it simply verifies that it is already in the feature dictionary and moves on.

The third feature, *Line* has a dependency on *3D Point 1* and *3D Point 2*. When this message is processed, *Line* calls the "GetInstance" method, passing in *3D Point 1's* GUID. Since *3D Point 1* is already in the feature dictionary it is simply returned. The same happens for *3D Point 2* since it is also in the feature dictionary. The dependency graph for *Line* has automatically been assembled. The line is created and added to the feature dictionary.

The final feature, *CSYS,* has already been placed in the feature dictionary so when the loading algorithm reaches its message it simply verifies that it is already in the feature dictionary and moves on.

Conclusions:

This research defines a logic architecture that enables a CAD client-server architecture to work with unordered feature data and irrespective of which commercial CAD client or clients are being used. The architecture of the program developed at BYU to validate this approach consists of database, communication, logic, and client tiers in a standard N-tiered architecture. Data is stored in a NPDB using a TPT mapping method. When changes are made to the database, the communication tier handles messages to and from the server, client, and database. The logic tier contains classes with methods for creating and updating features bi-directionally between client and server. The client tier tracks a user's actions and passes features to the logic tier for neutralization and storage. This architecture supports concurrent collaboration between the NX, CATIA, and Creo CAD systems. This paper focused on interoperability between NX and CATIA only, however Creo is now in the testing phase. The 2D point, 2D line, 2D arc, 2D circle, 2D spline, 3D point, extrude, and revolve features were developed. All users work at the same time creating features on their own CAD client. As a user exits their sketch the changes made by other users are pulled to their client in real time so that the model on all screens stays in sync. Any conflicts can be prevented using the methods proposed by Hepworth et al. [11]. For example, Fig. 3 shows a collaborative session between one NX user, one CATIA user and one Creo user. All are working together in the same session. They can see the operations their collaborators have completed as they complete them and all can edit simultaneously.

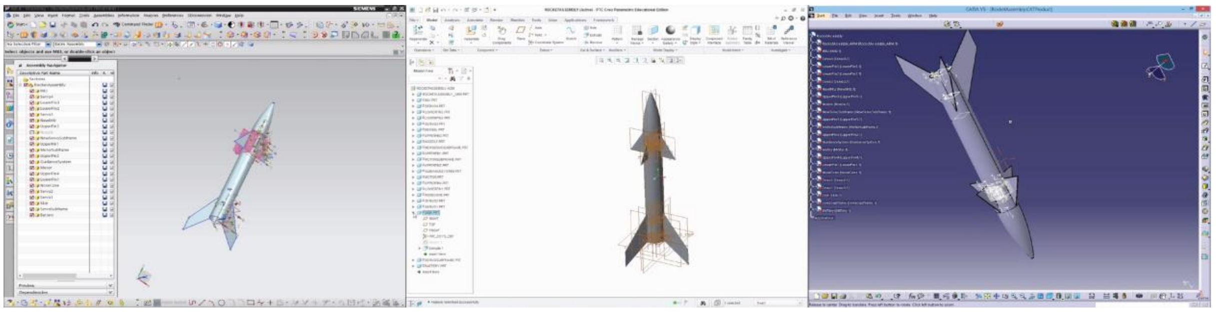

Fig. 3: A Catia user, NX user and Creo user collaborating on a rocket assembly.

As mentioned before, the industry members we work with all strive to develop high-quality products as quickly as possible. The aim of the research done in the Brigham Young University site of the National Science Foundation Center for e-Design is to help them shorten their product development lifecycle by making their engineering methods more efficient, enabling them to reduce non-recurring costs without quality loss. BYU researchers have done this by allowing engineers to work in a more synchronous way than before. A number of other researchers have worked toward the same goal by studying seamless interoperability between heterogeneous engineering CAD clients. The goal of this paper was to demonstrate the steps BYU has taken to merge these two areas of research. A functional prototype MESH CAD client-server architecture with logic rules to support unordered feature data and a client-agnostic server was developed. A core set of features were chosen for implementation that allow for non-trivial model generation. Moderately complex modeling tasks were completed by small teams using the prototype in order to demonstrate that the client server approach taken and the NPDB data storage method are functionally sound.

Acknowledgements:

Research of this type does not happen in a vacuum. We would like to thank those who have worked on this problem before us because their ideas were inspirational. We would like to thank all of the members of the BYU CADLab and Dr. Tew for his contributions to our server. We could not have done this without Raytheon Missile Systems or our members in the National Science Foundation center for e-Design: Belcan, Boeing, PCC Airfoils, Pratt & Whitney, CD-adapco, UTC Aerospace Systems, and ARDEC.

References:

[1] Amdahl, G. M.: Validity of the single processor approach to achieving large scale computing capabilities, in AFIPS spring joint computer conference, 1967, 187–196.

[2] Chen, X. C. X.; Li, M. L. M.; Gao, S. G. S.: A Web services based platform for exchange of procedural CAD models, Proc. Ninth Int. Conf. Comput. Support. Coop. Work Des. 2005., 1(310027), 2005, 605–610. http://dx.doi.org/10.1109/CSCWD.2005.194241.

[3] Choi, G. H.; Mun, D.; Han, S.: Exchange of CAD part models based on the macro-parametric approach, Int. J. CAD/CAM, 2(1), 13–21, 2002.

[4] Dou, W.; Song, X.: Operation Command Transformation of Synchronized Collaborative Design Upon Heterogeneous CAD Systems, J. Algorithm. Comput. Technol., 7(4), 423–448, 2013. http://dx.doi.org/10.1260/1748-3018.7.4.423.

[5] Dou, W.; Song, X.; Zhang, X.: A language of neutral modeling command for synchronized collaborative design among heterogeneous CAD systems, 2009 1st Int. Conf. Inf. Sci. Eng. ICISE 2009, 12–15. http://dx.doi.org/10.1109/ICISE.2009.52.

[6] Ganapathi, S.: A Software Model for Interoperability, The University of Texas at Arlington, 2002.

[7] Gu, H.; Chase, T. R.; Cheney, D. C.; Bailey, T.; Johnson, D.:Identifying, Correcting, and Avoiding Errors in Computer-Aided Design Models Which Affect Interoperability, J. Comput. Inf. Sci. Eng.,1, June, 2001, 156,2001. http://dx.doi.org/10.1115/1.1384887.

[8] Hepworth, A. I.; Nysetvold, T.; Bennett, J.; Phelps, G.; Jensen, C. G.: Scalable Integration of Commercial File Types in Multi-User CAD, Computer Aided Design & Applications, 11(4), 2014, 459–467. http://dx.doi.org/10.1080/16864360.2014.881190.

[9] Hepworth, A.; Tew, K.; Trent, M.; Ricks, D.; Jensen, C. G.; Red, W. E.: Model Consistency and Conflict Resolution With Data Preservation in Multi-User Computer Aided Design, J. Comput. Inf. Sci. Eng., 14, 2014, 021008. 2014,http://dx.doi.org/10.1115/1.4026553.

[10] Iyer, G. R.: Development of API-Based Interfaces to Enable Interoperability Between CAD Systems During Design Collaboration, The University of Texas at Arlington, 2001.

[11] Leach, L. M.: A Language Interface for Data Exchange Between Heterogeneous CAD/CAM Databases, Rensselaer Polytechnic Institute, 1983.

[12] Li, M.; Yang, Y.; Li, J.; Gao, S.: A preliminary study on synchronized collaborative design based on heterogeneous CAD systems, Coop. Work Des. 2004,no. 310027,2004.

[13] Moncur, R.; Jensen, G. C.; Teng, C. C.; Red, E.: Data consistency and conflict avoidance in a multi-user CAx environment, Computer Aideded Design & Applications, 10(5), 2013, 727–744. http://dx.doi.org/10.3722/cadaps.2013.727-744.

[14] Mun, D.; Han, S.; Kim, J.; Oh, Y.: A set of standard modeling commands for the history-based parametric approach, Computer-Aided Design, 35, 2003, 1171–1179. http://dx.doi.org/10.1016/S0010-4485(03)00022-8.

[15] Rappoport, A.: An architecture for universal CAD data exchange, Proc. eighth ACM Symp. Solid Model. Appl. - SM '03, 2003, 266. http://dx.doi.org/10.1145/781644.781648.

[16] Red, E.; Jensen, G.; French, D.; Weerakoon, P.: Multi-user architectures for computer-aided engineering collaboration, 2011 17th Int. Conf. Concurr. Enterprising, no. Ice, 2011, 1–10.

[17] Sun, L. J.,; Ding, B.: Heterogeneous CAD data exchange based on cellular ontology model, 2009 WRI World Congr. Softw. Eng. WCSE, 1, 2009, 46–50. http://dx.doi.org/10.1109/WCSE.2009.106.

[18] Tessier, S.:Ontology-Based Approach To Enable Feature Interoperability Between Cad Systems, Georgia Institute of Technology, 2011.

[19] Zhang, X.; Dou, W.:An approach of constructing neutral modeling command set of synchronized collaborative design upon heterogeneous CAD systems,Proc. - Int. Conf. Manag. Serv. Sci. MASS 2009, 0–3. http://dx.doi.org/10.1109/ICMSS.2009.5302072.