HiLL PUBLISHING

Engineering Advances, 2023, 3(6), 454-459
https://www.hillpublisher.com/journals/ea/
ISSN Online: 2768-7961

Enhancing Engineering Product Design Using a Knowledge-based Game Engine Platform

Guolong Zhong[1,*], Venkatesh Chennam Vijay[2], Noel Perera[2]

[1]Guangdong University of Technology, Guangzhou, Guangdong, China.
[2]Birmingham City University, Millennium Point, Birmingham, UK.

How to cite this paper: Guolong Zhong, Venkatesh Chennam Vijay, Noel Perera. (2023) Enhancing Engineering Product Design Using a Knowledge-based Game Engine Platform. *Engineering Advances*, 3(6), 454-459.
DOI: 10.26855/ea.2023.12.003

Received: November 30, 2023
Accepted: December 27, 2023
Published: January 19, 2024

***Corresponding author:** Guolong Zhong, Guangdong University of Technology, Guangzhou, Guangdong, China.

Abstract

Traditional CAD tools and systems cannot explain real-world concepts by themselves and require the users to have knowledge and design experience of the product in order to understand design rules and judge the correctness of the changes. Knowledge-based engineering (KBE) has been introduced to address these issues; however, existing KBE methodologies offer limited instantiation steps and enabling tools for implementation. In this paper, a knowledge-based product modelling prototype system is proposed to overcome the above problems. This system is developed based on the Virtual Product Modelling framework using a game engine platform to aid in capturing, reusing, and exchanging the existing product information and provide knowledge reasoning in the product modelling process. The findings of this research have shown the potential of using the developed prototype system to help save time and prevent engineers from making mistakes in the product design process.

Keywords

Product design, Knowledge-based engineering, Game engine, Design rules

1. Introduction

1.1 Background

Computer Aided Design (CAD) has been introduced as a Design Engineering Automation (DEA) method for completing product design. Nonetheless, CAD tools and systems cannot understand and explain real-world design concepts by themselves. To judge the correctness of the design, CAD tools and systems require users to have sufficient knowledge and design experience of the product. The trend of product design has evolved from CAD to Computer Aided Product Modelling and then to Knowledge-Based Product Modelling. This has required the product design software and environment to provide more interaction between end-users and the product modelling process through the reuse of existing knowledge to support the product modelling. In the last 20 years, Knowledge Based Engineering (KBE) has shown its advantages in product development in different engineering areas such as automation, mechanical engineering, civil engineering and aerospace engineering in terms of modelling and cost reduction. It helps automate the repetitive design tasks by capturing, integrating, utilising and reusing existing knowledge required in various aspects of the product design [1, 2]. The use of KBE methods and techniques has played an important role in design engineering automation for the development of a product in the industry [3, 4]. A critical issue of knowledge-based product modelling is capturing, classifying, structuring, and managing the captured knowledge. Since there is no clear formalised link between a generic product model and an interoperable format in the KBE environment [5-7], it is essential to provide well-defined knowledge classes and a formalised knowledge capture method for individuals, enterprises and industries to capture and share knowledge instead of using informal oral communication or notes and spreadsheets in different formats. In this paper, geometric data contained in a CAD file is regarded as “geometry information”, and non-geometric information, such as experience, expertise and design rules, is called “knowledge”.

1.2 Interactive application development using a game engine

The development of an interactive application consists of two main components: the application and the content [8]. The application aims to provide information in real-time to end-users and the ways to interact with it. The content contains the information through which the application navigates and provides a view to the users. According to the interactive qualifying project report [9], developers have recently realised that game engines can be successfully used for non-game applications development, such as architecture prototyping, interactive applications and research data visualisation. Gaming engines are generally used as an integrated development environment to enable the rapid development of game applications and build interactive applications.

Unity is a cross-platform game engine that was announced in 2005. It has become more popular and adopted by a growing number of users in recent decades due to its easy accessibility, user development support and strength in making 2D and 3D simulations [10]. Apart from the gaming industry, Unity is also used by industries such as automotive, architecture, engineering, and construction [11, 12]. A survey [13] showed that Unity has been playing an active role in the game development field, and the usage and industrial devotion to Unity is increasing and amplifying.

This research aims to use a game engine platform to develop a knowledge-based product modelling system that could capture, reuse and integrate the associated knowledge applied by designers to provide interactions with knowledge reasoning and to enhance the product design process. The rest of the paper will present in detail how the design engineer's knowledge is captured, reused, and mapped into the system in the applied use case to enhance the product modelling process for design engineering automation.

2. Methods

A Virtual Product Modelling (VPM) framework for developing a knowledge-based product modelling environment that enables existing knowledge to be captured and reused in product modelling has been developed and discussed in a previously published paper by the authors [14]. This framework (see Figure 1) consists of the following five stages:

1) Product model development
2) Knowledge capture of non-geometric information
3) Knowledge capture of geometry information
4) Knowledge mapping
5) Product visualisation and validation

This framework provides a set of activities for design engineers to conduct for implementing the knowledge-based engineering technique in the modelling process. It also provides the ability for parametric geometry representation as components are associated with parametric values after the implementation.

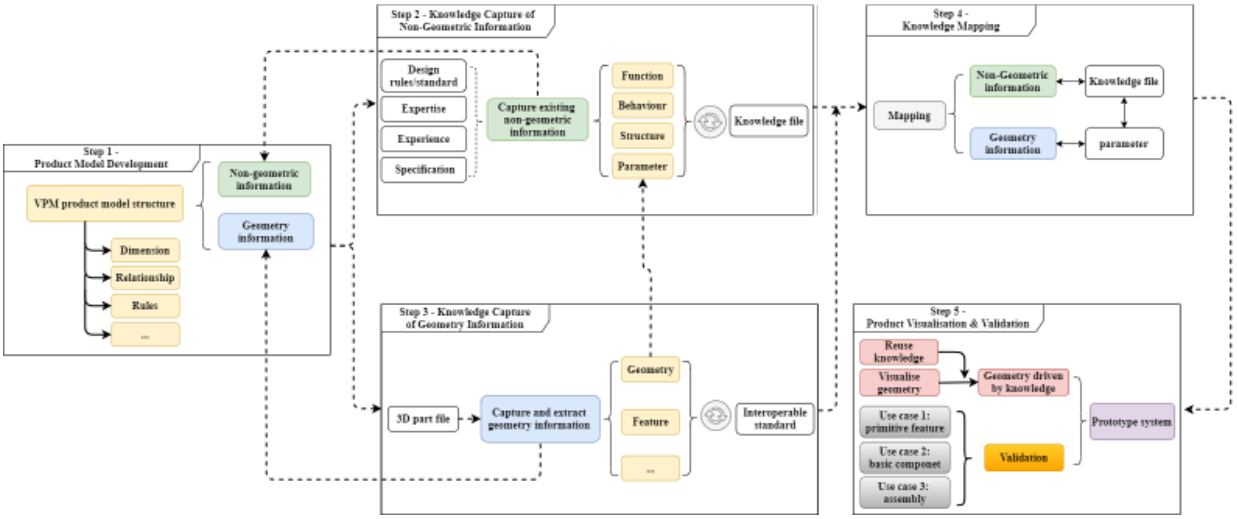


Figure 1. Virtual product modelling framework [14].

3. Implementation and Validation

The proposed system was tested through a use case that is adapted from the primitive design feature examples for the basic engineering feature modelling [15]. This use case is selected because primitive design features are the fundamental geometric features applied in the actual product modelling process in general CAD environments.

Primitive features are basic geometric features from which many other design features can be created. The basic primitive design features are block, cylinder, sphere and cone. Four parts with primitive design features are selected based on these features in this use case (as shown in Figure 2).

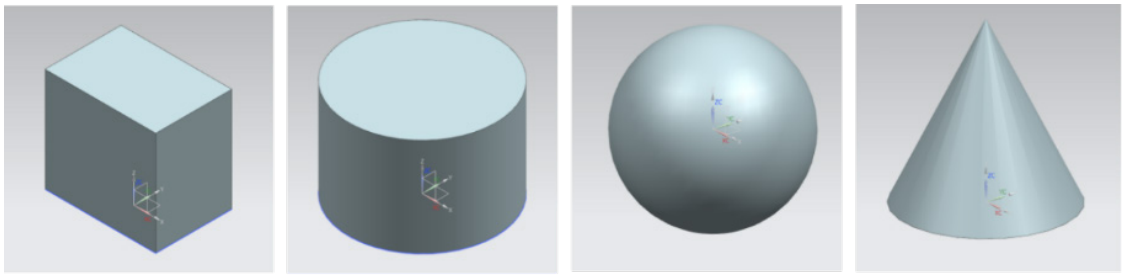


Figure 2. Four simple parts with primitive design features (modelled in Siemens NX 10).

Existing information on this use case (example shown in Table 1) is collected from the part library of one of the current product modelling systems – Siemens NX 10. However, the Siemens NX part library does not provide all the information that fits the classified VPM classes. Hence, for some VPM classes, such as material, behaviour, fit, and relationship, the entities are given as "None" or "Not defined".

Table 1. Existing information of use case 1- block part example

VPM knowledge class	Existing product information
Product	Block
Feature	Primitive design feature - block
Description	The Block is a cube.
Function	None
Behaviour	None
Form	Primitive design feature - block
Material	Not defined
Design intent	Primitive design feature to create other design features.
Geometry	From STEP file
Dimension	Length =100mm, width =100mm, height=100mm
Rules	Block Length L = Block Width W = Block Height H
Fit	None
Constraint	None
Relationship	None
Reference	None

The implementation follows the same process as listed in Methods. The function of making changes to product geometry is limited to basic parameters based on the use case. In this use case, different simple parts are used to test the effectiveness of the methodology and the workability of each function. In the block part example, one testing scenario is identified - "single dimension changed (block length) with the single rule applied". The design rule that has been captured and applied to the block part is "Length(L) = Width(W) = Height (H)". In the object-oriented programming (*IF-THEN-ELSE* statement), this rule can be expressed as:

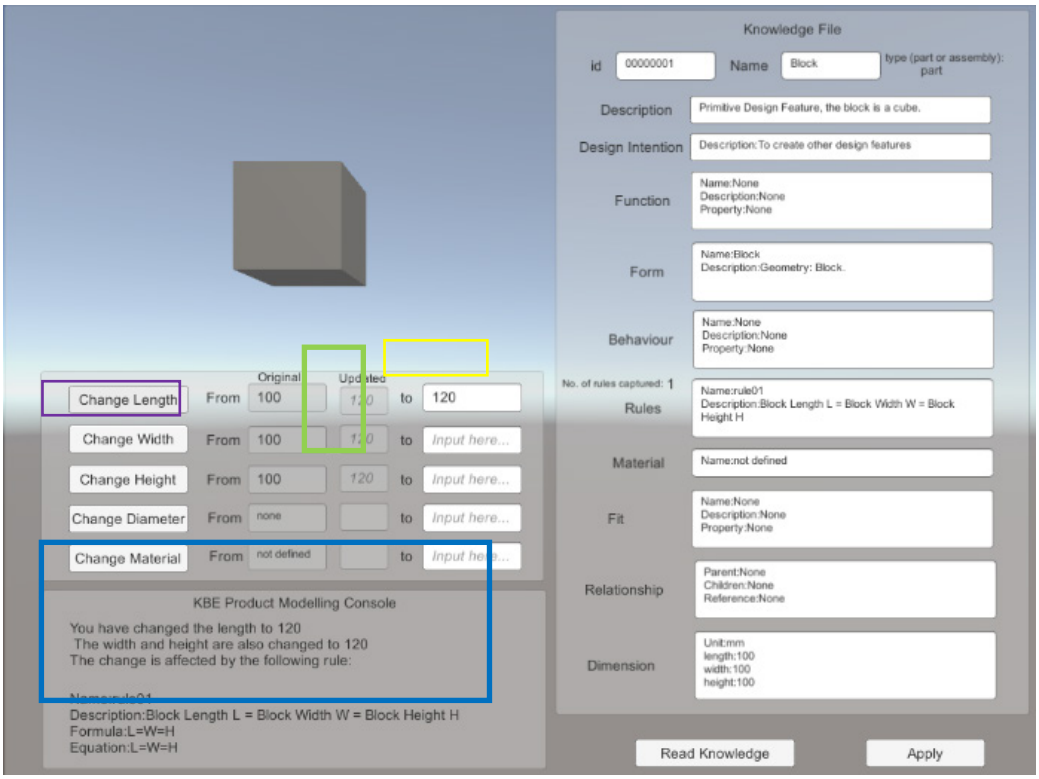
"*IF* block length is changed, *THEN* block width and block heights need to be changed;

ELSE IF block width is changed, *THEN* block length and height need to be changed;

ELSE IF block height is changed, *THEN* block length and width need to be changed."

The resulting visualisation in the developed user interface is shown in Figure 3. After importing the STEP file of the block part into the user interface, a 3D model of the block can be visualised. By clicking the "Read Knowledge" button, the developed tool will automatically parse the knowledge file and display all the captured information in the interface. After the user selects to change the block length by clicking the button "Change Length" in the tool interface, the resulting

changes will be shown in the "KBE Product Modelling Console" panel. For instance, when the user inputs 120 in the interface to change the length from the original 100 to 120 (unit: mm), the "KBE Product Modelling Console" will analyse if this change can be made by checking the rules. In this testing scenario, the rule that has been applied to the block part is "Length(L) = Width(W) = Height (H)". Therefore, the width and height of the block are also changed to 120 automatically. The affecting rule is shown correctly in the "KBE Product Modelling Console", which fulfils the knowledge reasoning for this product modelling process.



Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Purple box –button pressed to apply the change; Blue box - knowledge reasoning.

Figure 3. Results of validation – use case 1: Block part.

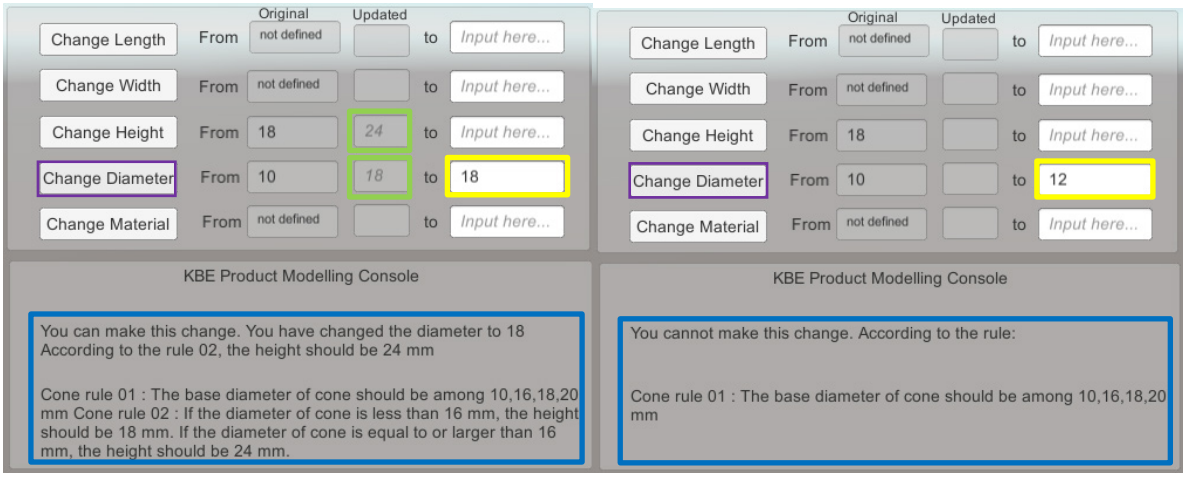
Similarly, different testing scenarios have been defined to test the function of "Change Height", "Change Diameter", and "Change Material" by using the cylinder part, cone part and sphere part. The rules applied in these testing scenarios are listed in Table 2.

Table 2. Rules applied in the cylinder, cone, and sphere parts

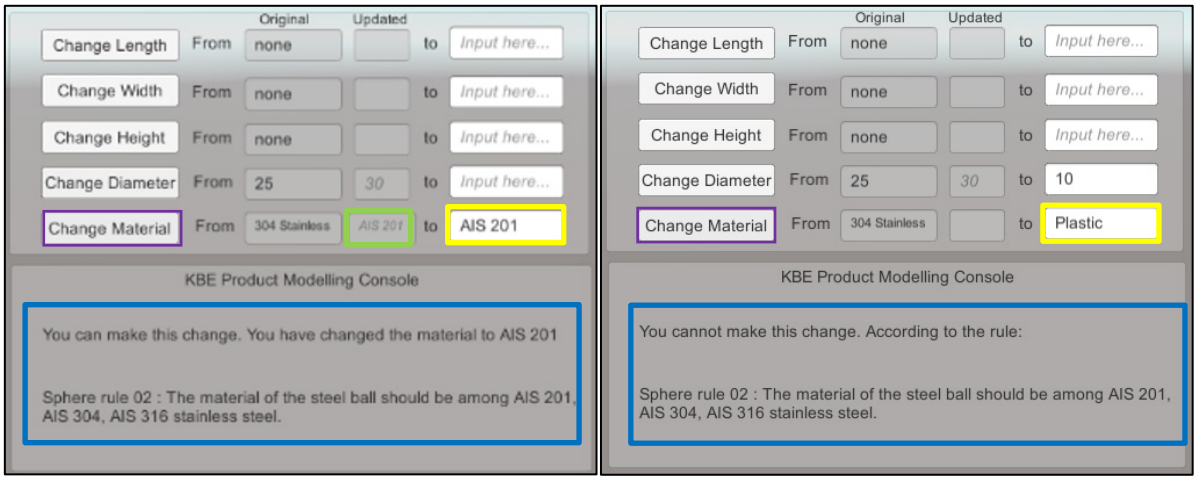
Rules		Description
Cylinder	Cylinder rule 01	The height of cylinder should not be larger than 200.
	Cylinder rule 02	The diameter of cylinder should not be larger than 80.
Cone	Cone rule 01	The base diameter of cone should be 10, 16, 18, 20 mm
	Cone rule 02	If the diameter of cone is less than 16mm, then the height should be 18mm. If the diameter of cone is equal to or larger than 16 mm, then the height should be 24 mm.
Sphere	Sphere rule 01	The diameter of sphere should be among 19, 20, 21, 22, 25, 30, 35, 40 mm
	Sphere rule 02	The material of the steel ball should be among AISI 201, AISI 304, AISI 316 stainless steel.

The implementation processes remain the same as described before. Some of the validation results for different scenarios are shown in Figures 4 (a) (b) (c), respectively.

(a) Scenario: change of height with rule 01 – cylinder part



(b) Scenario: change of base diameter- cone part



(c) Scenario: change of material under rule 02 - sphere part

Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Red box - propagated parameter (changes not allowed by rules); Purple box –button pressed to apply the change; Blue box - knowledge reasoning.

Figure 4. Results of validation for different scenarios in cylinder, cone, and sphere part.

4. Discussion

The developed knowledge-based product modelling system has shown extended capabilities of knowledge reasoning,

reuse, and exchange to enhance the product modelling process. The successful implementation of the selected use cases has proved the effectiveness of the prototype system in reusing the existing product information as knowledge to provide knowledge reasoning in the product modelling process. The visualisation of product geometry and its associated knowledge has validated the proposed data exchange method for exchanging the geometric data and knowledge (non-geometric data) of a product.

The implementation and validation results also help recognise the limitation of the developed knowledge-based product modelling prototype system in visualisation. The possible changes of dimensions that the users in this interface can make are limited to the use case. The results of changes to the product, such as changes to the length, width, height, diameter, and material, are presented through the text description in the interface. The text description is not as effective as a graphical 3D display in visualising geometry. This deficiency is caused by the lack of available enabling technologies that support editing the geometry data in the STEP file and displaying the graphical changes directly in the modelling environment. However, the limitation in visualisation is acceptable as the system has shown its effectiveness in propagating the changes of simple parts geometry by reusing the captured knowledge.

5. Conclusions and future work

This paper presented a knowledge-based product modelling prototype system developed in a game engine platform (Unity) to enhance engineering product design. This prototype system is developed and implemented based on the Virtual Product Modelling framework introduced by the authors. One use case has been implemented to validate the workability and effectiveness of the proposed framework and prototype system. and the proposed system proved the capability of knowledge reasoning and reuse of the developed knowledge-based product modelling system. The resulting enhancement to the product design process by using the proposed system lies in modelling products with more complexity regarding existing product information, parameters, internal and external rule constraints, and relations. The proposed system can be further tested using a complicated use case.

References

[1] G. La Rocca. “Knowledge-based engineering: Between AI and CAD. Review of a language-based technology to support engineering design.” *Adv. Eng. Informatics*, vol. 26, no. 2, pp. 159-179, 2012.

[2] C. Chapman, S. Preston, M. Pinfold, & G. Smith. “Utilising enterprise knowledge with knowledge-based engineering.” *Int. J. Comput. Appl. Technol.*, vol. 28, no. 2-3, pp. 169-179, 2007.

[3] E. M. Shehab & H. S. Abdalla. “Manufacturing cost modelling for concurrent product development.” *Robot. Comput. Integr. Manuf.*, vol. 17, no. 4, pp. 341-353, 2001.

[4] I. O. Sanya & E. M. Shehab. “An ontology framework for developing platform- independent knowledge-based engineering systems in the aerospace industry.” *Int. J. Prod. Res.*, vol. 53, pp. 1-27, 2014.

[5] I.-S. Fan & P. Bermell-Garcia. “International Standard Development for Knowledge Based Engineering Services for Product Lifecycle Management.” *Concurr. Eng.*, vol. 16, no. 4, pp. 271-277, 2008.

[6] M. Cederfeldt, F. Elgh, & I. Rask. “A Transparent Design System for Iterative Product Development.” *J. Comput. Inf. Sci. Eng.*, vol. 6, pp. 300-307, 2006.

[7] R. Curran, W. J. C. Verhagen, & M. J. L. Van Tooren. “The KNOMAD methodology for integration of multi-disciplinary engineering knowledge within aerospace production.” *48th AIAA Aerosp. Sci. Meet. Incl. New Horizons Forum Aerosp. Expo.*, pp. 1-16, 2010.

[8] M. Barnes. “Introduction to Collada.” 2007. [Online]. Available: https://www.gamedeveloper.com/art/introduction-to-collada. [Accessed: 06-Dec-2021].

[9] J. Haas. “A History of the Unity Game Engine for An Interactive Qualifying Project.” p. 44, 2014.

[10] S. K. Arora. “Unity vs Unreal Engine: Which Game Engine Should You Choose?” 2021. [Online]. Available: https://hackr.io/blog/unity-vs-unreal-engine. [Accessed: 06-Dec-2021].

[11] Unity Technologies. “Unity solutions for architecture, engineering and construction.” 2021. [Online]. Available: https://unity.com/solutions/architecture-engineering-construction.

[12] Juliani, et al. “Unity: A General Platform for Intelligent Agents.” no. February, 2018.

[13] Hussain, H. Shakeel, F. Hussain, N. Uddin, & T. L. Ghouri. “Unity Game Development Engine: A Technical Survey.” *Univ. Sindh J. Inf. Commun. Technol.*, vol. 4, no. 2, pp. 73-81, 2020.

[14] G. Zhong, V. C. Vijay, & I. Oraifige. “A Game-Based Product Modelling Environment for Non-Engineer.” *ICSGGBL 2021 23rd Int. Conf. Serious Games Game-Based Learn.*, vol. 15, no. 4, pp. 308-315, 2021.

[15] M. C. Leu. “NX10 FOR ENGINEERING DESIGN.” *Design*. p. 207, 2016.