DOI 10.31891/2307-5732-2018-267-6(2)-141-145
УДК 004.94

Д.І. БІЛИЙ, Т.К. СКРИПНИК
Хмельницький національний університет

СИСТЕМА АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ CAD/CAE ДОДАТКІВ

У роботі було проведено дослідження розробки програмних продуктів для CAD/CAE систем. На основі існуючого архітектурного шаблону Model-View-Operator (MVO) було розроблено новий архітектурний шаблон Model-Creator-ViewOperator (MCVO), що дозволяє, не змінюючи архітектуру системи, масштабувати її, вносити зміни та доповнення до елементів системи.

Ключові слова: система автоматизованого проектування, архітектурний шаблон, MVO, MCVO.

D.I. BILYI, T.K. SKRYPNYK
Khmelnytskyi National University

AUTOMATED DESIGN SYSTEM FOR CAD/CAE APPLICATIONS

Computer Aided Design systems have become an integral part of the machine-building enterprise. Therefore, the creation and upgrading of existing CAD/CAE systems is an actual task that requires constant attention of software developers. The process of improving computer aided design systems constantly continues, so during the construction phase, it is particularly important to select the CAD/CAE architecture of the system, which would allow it to scale, to make changes and additions to the elements of the system without changing the architecture. The architectural patterns are used as one of the most popular approach in creating software. The most well-known architectural pattern is MVC (model-view-controller), which is used when writing server applications. When it comes to create CAD/CAE system the most common pattern is MVO (model-view-operator). The 3D graphical system HOOPS (HOOPS/3dGS) was built based on the MVO architectural pattern based on that it becomes possible to analyze the main components of the pattern and its features. HOOPS/MVO has a model-view-operator architecture that encapsulates various data structures and concepts of HOOPS/3dGS. HOOPS/MVO makes the key capabilities of HOOPS/3dGS more accessible, offering a range of common application-level logic, facilitating efficient development and rapid prototyping of full-featured, high-performance CAD/CAM/CAE applications. The Model-View-Operator (MVO) architecture decouples graphical information from its presentation and manipulation. The model represents the data while the view corresponds to the presentation of this information. An operator can be a set of actions involving querying, creation, editing and manipulation of data. The architecture is implementing via the following three main classes: HBaseModel (model), HBaseView (view) and HBaseOperator (operator). In MVO, the HBaseModel class encapsulates the models found in the Include Library. When application logic needs to build a scene, it would first create a new instance under a Driver segment, which corresponds to an instance of the driver. The MVO class HBaseView stores an information for a given driver instance and managing data such as lights, overlay drawing, selection sets, and camera positioning. The HBaseOperator class and its derived classes designed for rotate the camera, select objects or delete various items after the 3D scene is displayed. When building an application using the MVO pattern, a single instance of the HDB class must be created and initialized during application's initialization phase. MVO supports multiple models as well as multiple views. In the HDB class, 3D graphics information is stored. To load information into the database, you can use HBaseModel. This class includes support for loading a variety of file formats. Once the information is in the database, you can continue to use HBaseModel to manage your 3D objects. When you are ready to save your object information, HBaseModel also includes support for exporting to a variety of formats. The HBaseView class manages the presentation of your models. Although an HBaseModel object can have multiple HBaseView instances, every HBaseView is associated with exactly one HBaseModel instance. In the database, HBaseView encapsulates an instance of a driver segment. The driver instance defines a connection with help of OpenGL or DX9. When HBaseView is initialized, it creates a default segment structure under the driver instance segment, which is used to manage interaction with HBaseModel. The HBaseOperator is the abstract base class that defines the interface for handling user input and operating on the model or view. The methods defined in this class provide the basis for mapping user input to interaction logic. In this context, an object derives from HBaseOperator and then implements methods to handle events such as mouse movements or key presses.

Keywords: computer-aided design, architectural pattern, MVO, MCVO.

Вступ

Актуальність. Системи автоматизованого проектування стали невід'ємною складовою частиною машинобудівного підприємства. Тому створення нових та вдосконалення існуючих CAD/CAE систем є актуальною задачею, яка потребує постійної уваги розробників програмного забезпечення. Процес вдосконалення систем автоматизованого проектування продовжується постійно, тому на етапі побудови особливо важливим є етап вибору архітектури CAD/CAE системи, який би дозволяв не змінюючи архітектуру системи масштабувати її, вносити зміни та доповнення до елементів системи.

Одним із підходів, який використовується під час створення програмного забезпечення є використання архітектурних шаблонів. Найбільш відомим архітектурним шаблоном є MVC (model-view-controller), який використовується в ході написання серверних додатків [1]. Для створення CAD/CAE систем існує архітектурний шаблон MVO (model-view-operator) [2].

На основі архітектурного шаблону MVO побудовано 3D-графічну систему HOOPS (HOOPS/3dGS) [3], на основі якої можна проаналізувати основні складові шаблону та його особливості. HOOPS/3dGS має структуру model-view-operator, яка інкапсулює різні структури та концепції даних HOOPS/3dGS. HOOPS/MVO робить ключові можливості HOOPS/3dGS більш доступними, пропонує широкий діапазон загальних логічних рівнів застосування сприяючи ефективній розробці та швидкому створенню прототипів повнофункціональних, високопродуктивних додатків CAD/CAM/CAE систем [4].

Архітектура Model-View-Operator (MVO) відокремлює графічну інформацію від її презентації та маніпулювання. Модель являє собою дані, тоді як представлення даних відповідає представленню цієї інформації. Оператор – це сукупність дій, що включають запит, створення, редагування та обробку даних. Архітектура реалізована за допомогою трьох основних класів: HBaseModel (модель), HBaseView (вид) та

HBaseOperator (оператор) (рис. 1) [2].

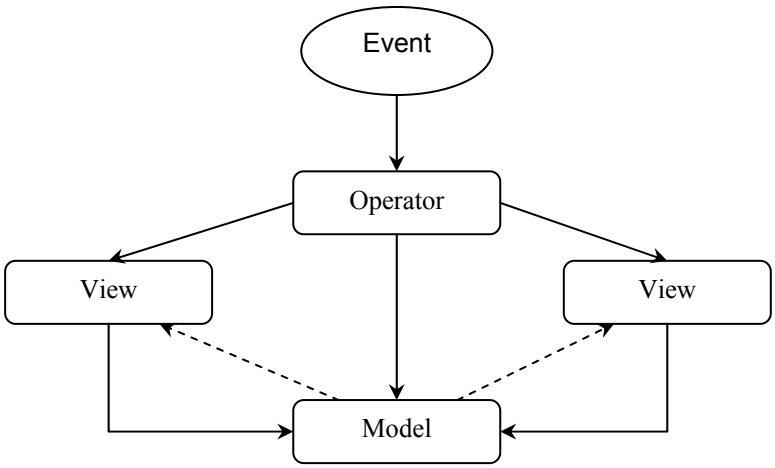

Рис. 1. Взаємозв'язок між окремими об'єктами в архітектурі Model-View-Operator

У MVO клас HBaseModel інкапсулює моделі, що знаходяться в Include Library. Під час побудови сцени спочатку створюється новий екземпляр у сегменті драйверів, який відповідає екземпляру драйвера. Клас MVO HBaseView зберігає інформацію для певного примірника драйверів та дає можливість керувати такими даними, як світло, накладення малюнка, набори виділення та розташування камери. Клас HBaseOperator та його похідні класи призначені для зміни положення камери, виділення об'єктів, або видалення різних елементів після відображення 3D-сцени.

У відповідності до шаблону MVO під час створення додатка один етап класу HDB повинен бути створений та ініціалізований під час фази ініціалізації програми [2]. MVO підтримує кілька моделей, а також декілька виглядів. У класі HDB зберігається інформація про 3D-графіку. Щоб завантажити інформацію в базу даних, можна використати HBaseModel. Цей клас включає підтримку для завантаження різноманітних форматів файлів. Після того, як інформація знаходиться в базі даних, можна продовжувати користуватися HBaseModel, для керування 3D-об'єктами. При необхідності зберегти інформацію про об'єкт, HBaseModel включає підтримку експорту в різні формати.

Клас HBaseView керує презентацією моделей. Хоча об'єкт HBaseModel може мати кілька примірників HBaseView, кожен HBaseView асоціюється з рівномірним екземпляром HBaseModel. У базі даних HBaseView інкапсулює екземпляр сегмента драйвера. Екземпляр драйвера визначає з'єднання з OpenGL або DX9. При ініціалізації HBaseView створюється структура сегменту за умовчанням в сегменті екземпляра драйвера, який використовується для керування взаємодії з HBaseModel.

HBaseOperator є абстрактним базовим класом, який визначає інтерфейс для обробки вводу користувача та роботи на моделі або вигляді (view). Методи, визначені в цьому класі, служать основою для відображення вхідних даних користувача до логіки взаємодії. У цьому контексті об'єкт походить від HBaseOperator, а потім реалізує методи для обробки подій, таких наприклад, як рух миші або натискання клавіш.

Мета і постановка завдання

Мета роботи полягає у вдосконаленні архітектурного шаблону проектування для розробки систем автоматизованого проектування. Вдосконалення архітектурного шаблону проектування для розробки додатків до систем автоматизованого проектування необхідно для прискорення розробки спеціалізованих програмних продуктів та знаходження рішення, яке дозволяло б не змінюючи архітектуру системи вносити зміни та доповнення до елементів системи. Для досягнення поставленої мети визначені наступні завдання:

1) дослідити сучасні архітектурні шаблони;
2) розробити новий архітектурний шаблон;
3) розробити структуру програмного продукту на основі розробленого архітектурного шаблону;
4) виконати алгоритмічну та програмну реалізацію додатку для системи автоматизованого проектування для підтвердження його життєздатності.

Основна частина

Зважаючи на такі недоліки шаблону MVO, як: відсутність частини для обробки даних та побудови 3D моделі, можливість взаємодії оператора напряму з моделлю, було запропоновано змінити його структуру, а саме: об'єднати представлення та оператор в одне ціле та додати ще один шар для обробки введених даних та створення нового об'єкту (рис. 2).

В результаті було розроблено навий шаблон, який складається з наступних компонентів: представлення-оператор (ViewOperator), створювач (Creator) та модель (Model).

Представлення-Оператор (ViewOperator) вміщує в собі функції як представлення, так і оператора із шаблону MVO. Даний елемент відповідає за створення та відображення як вікон або полів для введення даних про модель, так і за відображення самої моделі, а також реалізує функцію взаємодії користувача з

моделлю, тобто обробляє введені користувачем дані та передає їх у модель для подальших дій з ними.

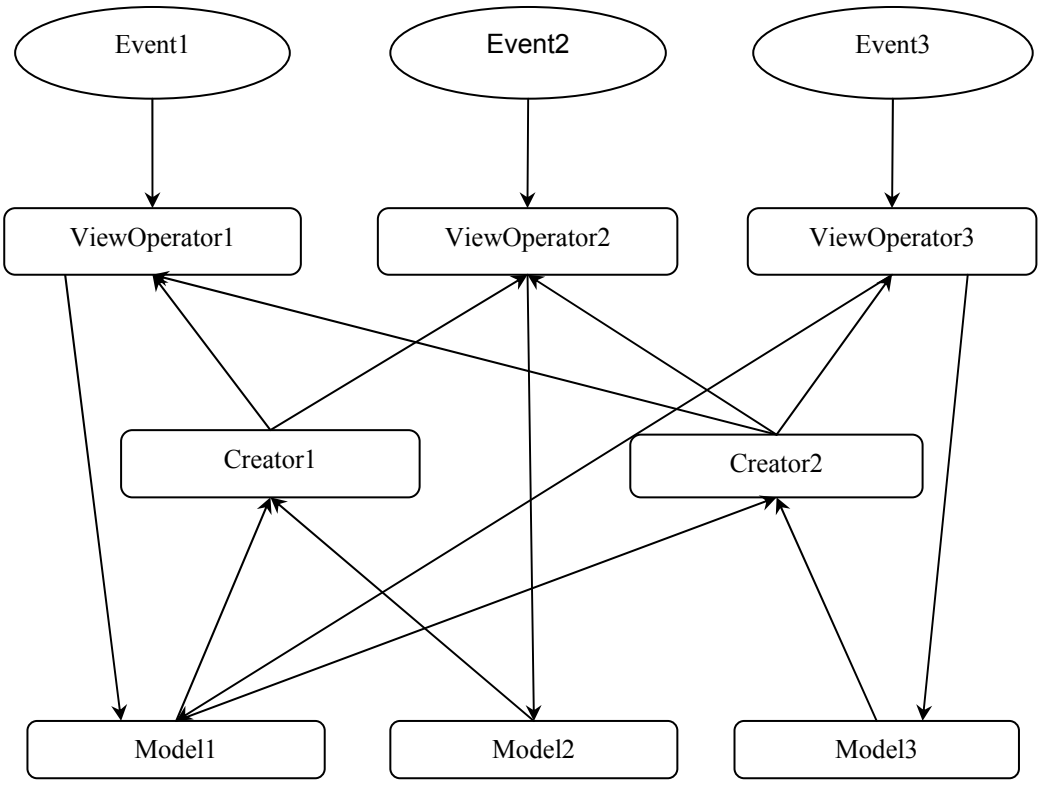

Рис. 2. Графічне зображення архітектурного шаблону MCVO

Створювач (Creator) реалізує функції зчитування даних з моделі та побудови 3D об'єкта на основі отриманих даних.

Функція моделі залишилась незмінною, тобто збереження у собі інформації про об'єкт, надання можливості збереження інформації у базі даних з можливістю їх експорту в подальшому.

Принцип роботи архітектурного шаблону MCVO можна описати наступним чином. При появі певної події (Event) ініціалізується створення представлення-оператора (ViewOperator). Подією може виступати як натиск на певну кнопку, так і введення певних даних чи зміна параметрів в текстовому полі. Далі в роботу вступає представлення-оператор, який відображає певний вигляд чи вікно програми, в якому є можливість введення параметрів для побудови 3D моделі. Потім введені користувачем дані опрацьовуються, приводяться до потрібного вигляду, щоб модель змогла їх отримати без помилок. Модель може зберігати отримані дані як в пам'яті, так і в базі даних.

На наступному етапі створювач (Creator) отримує дані від моделі та виконує операції по побудові 3D моделі. Після побудови створювач передає представлення на представлення-оператор для відображення створюваної моделі. Створювач має можливість взаємодіяти з декількома представленнями-операторами, що дає змогу відображати зміни на кількох представленнях. Але так, як в операційній системі або CAD/CAM системі існує лише один графічний потік для відображення вікон та моделей, неможливо одночасно відобразити зміни з декількох представлень-операторів.

Перевагами створеного шаблону є:

- збільшення швидкості роботи завдяки тому, що представлення та оператор становлять одне ціле та відсутній проміжок передачі даних між цими шарами;
- можливість обробки даних та створення 3D моделі завдяки новому шару створювача (Creator);
- можливість взаємодії створювача з декількома представленнями-операторами, що дозволяє відображати зміни на кількох представленнях;
- можливість моделі взаємодіяти з декількома створювачами, завдяки чому дані, що зберігаються у моделі можуть опрацьовуватися в декількох місцях.

На основі вдосконаленого архітектурного шаблону було розроблено програмний продукт, роботу якого можна подати у вигляді спрощеної блок-схеми, зображеної на рис. 3.

Після запуску користувач має обрати тип частини валу: циліндричний, конічний чи багатокутник. Якщо користувач обере тип частини валу циліндричний чи багатокутник, то йому потрібно буде вказати діаметр та довжину частини валу, а якщо обере конічний тип, то потрібно буде вказати діаметр на початку і в кінці частини валу та її довжину. Якщо якісь дані будуть некоректними, то поле із цим значенням підсвітиться червоним.

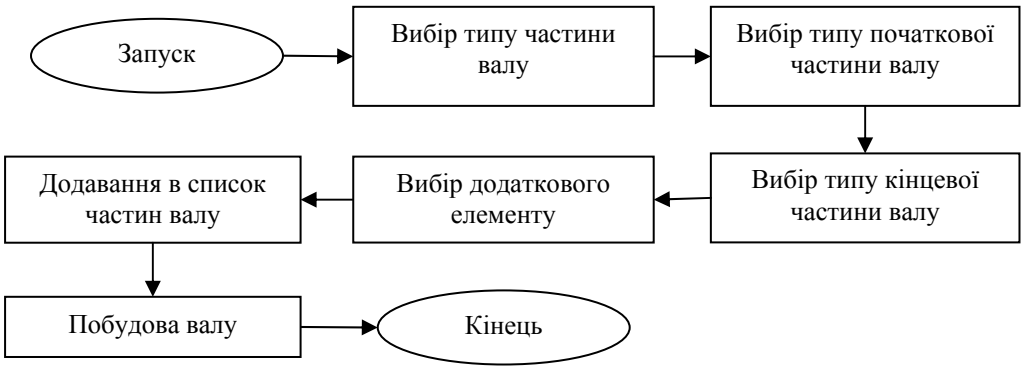

Рис. 3. Спрощена блок-схема програмного продукту

Для того, щоб запустити систему автоматизованого проектування валів необхідно запустити Autodesk Inventor, перейти на вкладку «Інструменти» та натиснути кнопку «Shaft».

Після того, як користувач вибрав тип частини валу, йому надається можливість вибрати тип початкової та кінцевої частини валу. Якщо користувач обере округлення, то йому потрібно буде ввести радіус округлення, якщо ж користувач обере фаску, то перед ним знову постане вибір, який тип фаски обрати: з однією відстанню, з відстанню та кутом, з двома відстанями.

Далі користувач має можливість додати на частину валу додатковий елемент: шпонковий паз (рис. 4), канавку, наскрізний отвір чи різьбу.

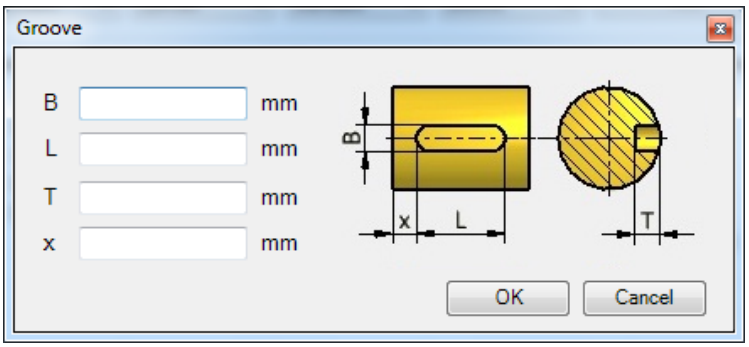

Рис. 4. Вікно для введення параметрів для шпонкового пазу

Потім користувач може додати в список створену частину валу (рис. 5).

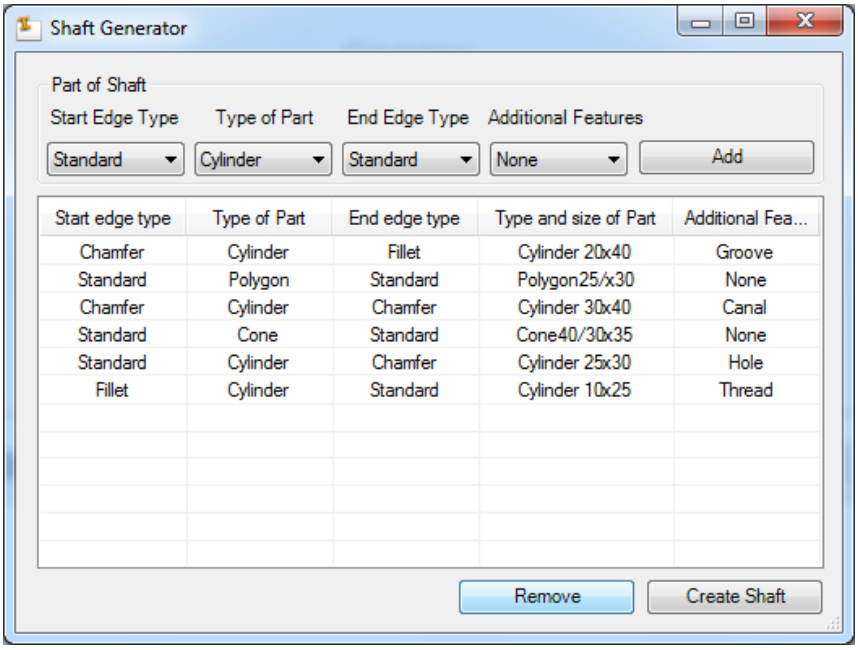

Рис. 5. Головне вікно програми, у якому зображено список частин валу

Після додавання всіх частин валу користувач має можливість видалити певну частину зі списку та побудувати вал (рис. 6).

Рис. 6. Вал, створений за допомогою системи автоматизованого проектування валів

Як видно із рис. 6, розроблена програма дає можливість створювати вал, що містить частини різних типів та додаткові елементи на них.

Висновки

Досліджено архітектурний шаблон MVO для створення програмних додатків для систем автоматизованого проектування. На основі шаблону MVO розроблено новий архітектурний шаблон Model-Creator-ViewOperator. В результаті було створено програмний продукт для системи автоматизованого проектування CAD/CAE додатків. Застосування технології дає можливість масштабувати систему, не змінюючи її архітектуру, вносити зміни та доповнення до елементів системи.

Література

1. MVC для веб: проще некуда [Електронний ресурс]. – Режим доступу : https://habr.com/post/181772/

2. HOOPS/MVO Technical Overview [Електронний ресурс]. – Режим доступу : https://docs.techsoft3d.com/visualize/3df/latest/build/tech_overview/mvo_technical_overview.html

3. Hoops Visualize Technical Documentation [Електронний ресурс]. – Режим доступу : http://techsoft3d.jp/developers/technical-documentation/hoops-visualize/

4. 10 Ways to Unlock the Power of HOOPS Visualize [Електронний ресурс]. – Режим доступу : https://www.prototechsolutions.com/hoops-visualize-application-development/

References

1. MVC dlya web: proshche necuda [Electronic resource]. – Access mode : https://habr.com/post/181772/. [in Russian]

2. HOOPS/MVO Technical Overview [Electronic resource]. – Access mode : https://docs.techsoft3d.com/visualize/3df/latest/build/tech_overview/mvo_technical_overview.html

3. Hoops Visualize Technical Documentation [Electronic resource]. – Access mode : http://techsoft3d.jp/developers/technical-documentation/hoops-visualize/

4. 10 Ways to Unlock the Power of HOOPS Visualize [Electronic resource]. – Access mode : https://www.prototechsolutions.com/hoops-visualize-application-development/

Рецензія/Peer review : 3.11.2018 р. Надрукована/Printed :19.12.2018 р.
Рецензент: д.т.н., проф. Сорокатий Р.В.