МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Кваліфікаційна наукова
праця на правах рукопису

КВАСНЕВСЬКИЙ ВЛАДИСЛАВ МИХАЙЛОВИЧ

УДК 004:728

ДИСЕРТАЦІЯ

МОДЕЛІ І МЕТОДИ ІНФОРМАЦІЙНОГО МОДЕЛЮВАННЯ ОБ'ЄКТІВ У ALLPLAN

05.13.06 – інформаційні технології
(шифр і назва спеціальності)

12 «Інформаційні технології»
(галузь знань)

Подається на здобуття наукового ступеня
кандидата технічних наук (доктора філософії)

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____
(підпис, ініціали та прізвище здобувача)

Науковий керівник –
Бородавка Євгеній Володимирович
д.т.н., доцент

Ідентичність всіх примірників дисертації
ЗАСВІДЧУЮ*:*
Вчений секретар спеціалізованої
вченої ради
/М.І. Цюцюра/

Київ – 2019

АНОТАЦІЯ

Квасневський В.М. Моделі і методи інформаційного моделювання об'єктів у Allplan. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.06 – Інформаційні технології, Київський національний університет будівництва та архітектури, м. Київ, 2019.

На сьогоднішній день існує маса спеціалізованих інформаційних систем автоматизованого проектування. Кожен окремий клас інформаційних систем дозволяє розв'язати ті чи інші проблеми у певній галузі.

Тенденція створення інформаційних систем автоматизованого проектування почалася з кінця XX століття. Саме в цей час і виник підхід створення комп'ютерної моделі будівельного об'єкта. Тенденція продовжується і до тепер, адже в сучасних умовах неможливо обробляти вручну величезні об'єми даних, які продовжують невпинно рости. Кожного року складність проектів зростає, збільшуються потреби інвесторів до представлення проекту на предпроектній стадії з одного боку та потреби конструкторів та архітекторів з іншого. Актуальною стає проблема не лише створення будівельної моделі об'єкта, а і її ведення протягом усіх стадій експлуатації об'єкта та його взаємодія з навколишнім середовищем та іншими об'єктами.

На сьогодні активно розвивається та впроваджується технологія BIM (Building Information Model). Ця технологія являє собою створення спеціальної інформаційної моделі будівлі, яка передбачає збір і комплексну обробку в процесі проектування всієї архітектурно-конструктивної, технологічної, економічної та іншої інформації про будівлю з усіма її взаємозв'язками і залежностями. Будівля і все, що має до неї стосунок розглядається як єдиний об'єкт. Технологія забезпечує управління об'єктом протягом усього його життєвого циклу. Використання цієї технології забезпечує ефективне

управління даними будівельного об’єкта. Все це дозволяє максимально зменшити проектні терміни, детально візуалізувати інтер’єри та екстер’єри у віртуальній реальності. Традиційний підхід до проектування спирається на 2D-моделі, плани та іншу паперову документацію. На відміну від традиційного підходу, BIM технологія додає нові виміри, такі як: плани будівництва, час будівництва, вартість. Всі вони знаходяться в базі даних інформаційної моделі об’єкта. Тривимірна модель будівлі тісно пов’язана з інформаційною базою даних, в якій кожному елементу моделі відповідає певний набір атрибутів, що може бути розширений додатковими атрибутами. При такому підході будівельний об’єкт є одним цілим і зміна кожного його параметру тягне за собою автоматичну зміну інших пов’язаних параметрів: креслень, візуалізацій, специфікацій і календарного графіка. Інформаційне моделювання скорочує витрати протягом всього життєвого циклу об’єкта. Сюди входять витрати на управління фінансами, ресурсами, обладнанням та матеріалами. Накопичені з BIM дані значно спрощують роботу на етапах проектування, будівництва, експлуатації та реконструкції об’єкта.

Архітектурна частина проекту тісно пов’язана з конструктивною частиною проекту. Архітектурна частина відповідає за забезпечення об’ємно-планувальних рішень, відповідність об’єкта своєму призначенню, тобто функціональність об’єкта.

Конструктивна частина відповідає за забезпечення надійності та довговічності об’єкта будівництва. Важливим питанням інформаційних систем автоматизованого проектування в будівництві стає перетворення складної архітектурної моделі насиченої різноманітними архітектурними деталями та елементами, які несуть в собі декоративний характер у строгу конструктивну модель.

На теперішній час в Україні кількість наукових досліджень по розробці та практичній реалізації уніфікованих моделей, методів та технологій інтеграції засобів автоматизованого проектування є доволі незначною. Не

розкрита проблема переходу від архітектурної моделі до конструктивної та проблема усунення колізій, які виникають на певних стадіях переходу між даними моделями. Тому сформульовані та розв’язані в даній дисертаційній роботі науково-технічні задачі уніфікації будівельних елементів та їх властивостей, побудови пластинчато-стержневої моделі (ПСМ) та інтеграції програмних комплексів є актуальними.

Дисертація присвячена вирішенню важливої науково-практичної проблеми автоматизації процесу проектування будівельних об’єктів на різних етапах життєвого циклу. Запропоновані бібліотека уніфікації архітектурних елементів, пластинчато-стержнева модель, методи генерації пластинчато-стержневої моделі, методи усунення колізій на пластинчато стержневій моделі та методи обміну даними між інформаційною системою Allplan та комплексом для управління проектами Microsoft Project.

Бібліотека уніфікації необхідна для формалізації таких понять як: несуча стіна, зовнішня стіна, назви матеріалів і т.н. Адже використання цих понять доволі часто інтерпретується різними системами по різному. Використання бібліотеки уніфікації дозволить не лише уніфікувати певний набір понять, а й надасть можливість гнучко змінювати ці поняття впливаючи на роботу модулів які її використовують.

Запропонована пластинчато-стержнева модель дозволяє автоматизувати перехід від архітектурної моделі до кінцево-елементної, що дозволяє істотно прискорити процес проектування будівельного об'єкта і значно полегшує роботу конструктора. Надалі конструктор може передати пластинчато-стержневу модель в спеціалізовану інформаційну систему проектування, серед них: Ліра САПР, САПФІР.

Використання моделі для інтеграції Allplan і MS Project автоматизує процес управління ресурсами проекту на всіх етапах проектування і дозволяє

наочно побачити зведення будівельного об'єкта в інформаційній системі Allplan.

Ключові слова: будівельний об'єкт, життєвий цикл, модель будівельного об'єкта, інформаційна технологія автоматизації, ПСМ, Allplan, архітектурно-будівельні САПР, САх-системи.

ABSTRACT

Kvasnevskyi V.M. Models and methods of information modeling of objects in Allplan. – Qualification scientific work based on manuscript.

The thesis is for the candidate of technical sciences degree, specialty 05.13.06 – information technology. – Kyiv National University of Construction and Architecture. – Kyiv, 2019.

Today there is a mass of specialized computer-aided design information systems. Each individual class of information systems can solve certain problems in a particular industry.

The tendency to create computer-aided design information systems began at the end of the XX century. It was at this time that the approach of creating a computer model of a construction object emerged. The trend continues to this day, because in today's environment it is impossible to process manually huge amounts of data that continue to grow steadily. Each year, the complexity of projects increases, the needs of investors to present a project at the pre-project stage on the one hand, and the needs of designers and architects on the other, increase. Not only does the construction model of the object, but also its maintenance during all stages of operation of the object and its interaction with the environment and other objects, become urgent.

Today BIM (Building Information Model) technology is being actively developed and implemented. This technology is the creation of a special information model of the building, which provides for the collection and complex processing in

the process of designing all architectural, structural, technological, economic and other information about the building with all its interconnections and dependencies. The building and everything related to it is regarded as a single object. Technology provides facility management throughout its lifecycle. The use of this technology provides effective management of the data of the construction object. All this allows you to minimize project terms, visualize interiors and exteriors in virtual reality. The traditional approach to design is based on 2D models, plans and other paper documentation. Unlike the traditional approach, BIM technology adds new dimensions, such as: construction plans, construction time, cost. All of them are in the object information model database. The three-dimensional model of the building is closely linked to an information database, in which each element of the model corresponds to a specific set of attributes, which can be expanded with additional attributes. In this approach, the construction object is one and the change of each parameter entails the automatic change of other related parameters: drawings, renderings, specifications and calendar. Information modeling reduces costs throughout the lifecycle of an object. This includes the costs of managing finances, resources, equipment and materials. BIM data significantly simplifies the design, construction, operation and reconstruction stages of the facility.

The architectural part of the project is closely related to the structural part of the project. The architectural part is responsible for providing the spatial planning solutions, the suitability of the object to its purpose, that is, the functionality of the object.

The structural part is responsible for ensuring the reliability and durability of the construction site. An important issue of computer-aided design information systems is the transformation of a complex architectural model saturated with various architectural details and elements that carry a decorative character into a strictly structural model.

At present in Ukraine, the amount of scientific research on the development and practical implementation of unified models, methods and technologies for the

integration of computer-aided design is quite small. The problem of the transition from the architectural model to the structural model and the problem of collision arising at certain stages of transition between these models are not disclosed. Therefore, the scientific-technical problems of unification of building elements and their properties, construction of plate-beam model (PBM) and integration of software complexes are formulated and solved in this dissertation.

The dissertation is devoted to solving the important scientific and practical problem of automation of the process of designing construction objects at different stages of the life cycle. The library of unification of architectural elements, plate-beam model, methods of generation of plate-beam model, methods of collision elimination on plate-beam model and methods of data exchange between the information system Allplan and the complex for project management Microsoft Project are offered.

A library of unification is necessary to formalize such concepts as: load-bearing wall, outer wall, names of materials, etc. Because the use of these concepts is often interpreted by different systems in different ways. The use of a unification library will not only unify a certain set of concepts, but will also allow the flexible modification of these concepts by affecting the operation of the modules that use it.

The proposed plate-rod model allows automating the transition from the architectural model to the finite element, which allows to significantly accelerating the design process of the construction object and greatly facilitates the work of the designer. In the future, the designer can transfer the plate-rod model to a specialized design information system, among them: Lira CAD, SAPFIR.

Keywords: construction object, lifecycle, construction object model, information technology of automation, PBM, Allplan, architectural and building CAD, CAx-systems.

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Публікації в іноземних виданнях та наукових фахових виданнях:

1. Квасневський В.М. Інформаційна бібліотека уніфікації будівельних конструкцій / В.М. Квасневський, Є.В. Бородавка // The Scientific Heritage. VOL 1. – 2018. №24. – С. 56 - 62.

2. Tsiutsiura S. Implementation of Data Structure for Digital Representation of Building Model/ S. Tsiutsiura, Y. Borodavka, V. Kvasnevskyi//International Journal of Computer Science and Telecommunications. – 2017. – VOL 8 – 2017. № 6 – С. 1-3.

3. Квасневський В.М. Базові принципи побудови УРМ (Універсальної розрахункової моделі). // The Scientific Heritage. VOL 2. – 2016. №6. – С. 82 - 88.

4. Квасневський В.М. Методи сортування геометричних об’єктів та їх реалізація на прикладі плагіна автонумерації для САПР Allplan / В.М. Квасневський, Є.В. Бородавка // Управління розвитком складних систем. – 2014. № 18. – С. 128 - 132.

5. Квасневський В.М. Геометричні методи побудови отворів та гільз для інженерних мереж в САПР Allplan / В.М. Квасневський, Є.В. Бородавка // Управління розвитком складних систем. – 2015. № 22. – С. 128 - 133.

6. Бородавка Є.В. Методи побудови об'єктів в комп'ютерній графіці / Є.В. Бородавка, В.М. Квасневський // Управління розвитком складних систем. – 2015. № 24. – С. 106 - 110.

Матеріали наукових конференцій:

7. Квасневський В.М. Інформаційна модель інтеграції Allplan та Microsoft Ptoject / В.М. Квасневський, Є.В. Бородавка // Тези доповіді III науково-практичної конференції «Modern Methodology of Science and Education». – Дубай, 2017. – С. 18 - 23.

8. Квасневський В.М. Information model of the Allplan and the Microsoft Project integration / Є.В. Бородавка, В.М. Квасневський // Тези доповіді науково-практичної конференції молодих вчених «Build master class 2017» – Київ, КНУБА, 2017. – С. 351.

9. Квасневський В.М. Інформаційна бібліотека уніфікації будівельних конструкцій / В.М. Квасневський, Є.В. Бородавка // Тези доповіді V науково-практичної конференції «Management of the development of technologies» Київ, КНУБА, 2018. – С. 74 - 75.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕННЬ

Позначення	Умовна назва
БО	Будівельний об’єкт
ЖЦ	Життєвий цикл
ЖЦБО	Життєвий цикл будівельного об’єкта
ПСМ	Пластинчато-стержнева модель
BIM	Building Information Model (Інформаційна модель об’єкта)
ЦМО	Цифрова модель об’єкта
БД	База даних
СУБД	Система управління базою даних
UML	Unified Modeling Language (Уніфікована мова моделювання)
IFC	Industry Foundation Classes (Базові промислові класи)
IGES	Initial Graphics Exchange Specification (Початкова специфікація обміну графікою)
DXF	Drawing eXchange Format (Формат обміну графічною інформацією)
STEP	STandard for the Exchange of Product model data (Мова обміну даними про модель продукту)
XML	eXtensible Markup Language (Розширювана мова розмітки)
САПР	Система автоматизованого проектування

ВСТУП

Актуальність теми. На сьогоднішній день існує маса спеціалізованих інформаційних систем автоматизованого проектування. Кожен окремий клас інформаційних систем дозволяє розв'язати ті чи інші проблеми у певній галузі.

Тенденція створення інформаційних систем автоматизованого проектування почалася з кінця XX століття. Саме в цей час і виник підхід створення комп'ютерної моделі будівельного об'єкта. Тенденція продовжується і до тепер, адже в сучасних умовах неможливо обробляти вручну величезні об'єми даних, які продовжують невпинно рости. Кожного року складність проектів зростає, збільшуються потреби інвесторів до представлення проекту на предпроектній стадії з одного боку та потреби конструкторів та архітекторів з іншого. Актуальною стає проблема не лише створення будівельної моделі об'єкта, а і її ведення протягом усіх стадій експлуатації об'єкта та його взаємодія з навколишнім середовищем та іншими об'єктами.

На сьогодні активно розвивається та впроваджується технологія BIM (Building Information Model). Ця технологія являє собою створення спеціальної інформаційної моделі будівлі, яка передбачає збір і комплексну обробку в процесі проектування всієї архітектурно-конструкторської, технологічної, економічної та іншої інформації про будівлю з усіма її взаємозв'язками і залежностями. Будівля і все що має до неї стосунок розглядається як єдиний об'єкт. Технологія забезпечує управління об'єктом протягом усього його життєвого циклу. Використання цієї технології забезпечує ефективне управління даними будівельного об'єкта. Все це дозволяє максимально зменшити проекті терміни, детально візуалізувати інтер'єри та екстер'єри у віртуальній реальності. Традиційний підхід до проектування спирається на 2D-моделі, плани та іншу паперову документацію. На відміну від традиційного підходу, BIM технологія додає нові виміри, такі як: плани будівництва, час будівництва, вартість. Всі вони знаходяться в базі даних

інформаційної моделі об’єкта. Тривимірна модель будівлі тісно пов’язана з інформаційною базою даних, в якій кожному елементу моделі відповідає певний набір атрибутів, що може бути розширений додатковими атрибутами. При такому підході будівельний об’єкт є одним цілим і зміна кожного його параметру тягне за собою автоматичну зміну інших пов’язаних параметрів: креслень, візуалізацій, специфікацій і календарного графіка. Інформаційне моделювання скорочує витрати протягом всього життєвого циклу об’єкта. Сюди входять витрати на управління фінансами, ресурсами, обладнанням та матеріалами. Накопичені з BIM дані значно спрощують роботу на етапах проектування, будівництва, експлуатації та реконструкції об’єкта.

Архітектурна частина проекту тісно пов’язана з конструктивною частиною проекту. Архітектурна частина відповідає за забезпечення об’ємно-планувальних рішень, відповідність об’єкта своєму призначенню, тобто функціональність об’єкта. Конструктивна частина відповідає за забезпечення надійності та довговічності об’єкта будівництва. Важливим питанням інформаційних систем автоматизованого проектування в будівництві стає перетворення складної архітектурної моделі, насиченої різноманітними архітектурними деталями та елементами, які несуть в собі декоративний характер, у строгу конструктивну модель [12].

На теперішній час в Україні кількість наукових досліджень по розробці та практичній реалізації уніфікованих моделей, методів та технологій інтеграції засобів автоматизованого проектування є доволі незначною. Не розкрита проблема переходу від архітектурної моделі до конструктивної та проблема усунення колізій, які виникають на певних стадіях переходу між даними моделями. Тому сформульовані та розв’язані в даній дисертаційній роботі науково-технічні задачі уніфікації будівельних елементів та їх властивостей, побудови пластинчато-стержневої моделі (ПСМ) та інтеграції програмних комплексів є **актуальними**.

Мета і задачі дослідження. Метою дисертаційної роботи є дослідження, розробка та практична реалізація моделей та методів, які дозволяють автоматизувати конкретні процеси проектування і управління в будівництві та надають можливість автоматизованого переходу між окремими процесами.

Основними задачами дослідження є:

– дослідження методів інтеграції комп'ютерних застосунків;

– виокремлення і аналіз основних етапів проектування будівель і споруд та їх послідовності;

– виокремлення і аналіз основних CAD-систем, що використовуються на основних етапах процесу проектування будівель і споруд та порівняння їх можливостей;

– проектування та розробка бібліотеки уніфікації архітектурних елементів;

– розробка методів генерації пластинчато-стержневої моделі (ПСМ);

– розробка та аналіз методів усунення колізій на створеній ПСМ;

– розробка та аналіз методів генерації календарно-планувальних даних та їх передачі в програмні комплекси управління проектами (зокрема у Microsoft Project);

Об'єкт дослідження: процес автоматизації життєвого циклу будівель та споруд різного призначення.

Предмет дослідження: сучасні засоби та методи автоматизації життєвого циклу будівельних об'єктів.

Методи дослідження. Під час проведення досліджень в дисертаційній роботі використані методи системного аналізу для вибору оптимальної технології інтеграції, об'єктно-орієнтованого моделювання з використанням уніфікованої мови моделювання (UML) для створення базових об'єктно-орієнтованих розширюваних моделей комплексної ЦМО, еволюційний метод

пошуку бази розширюваних систем для визначення множини елементів базових моделей комплексної цифрової моделі об’єкта (ЦМО), онтологічні методи дослідження предметної області для визначення ієрархії елементів та сутностей будинків і споруд, теорія множин для визначення ступеня спорідненості окремих моделей будівельного об’єкта, теорія графів для реалізації ієрархічних зв’язків елементів будівель.

Наукова новизна одержаних результатів полягає у тому, що в дисертації вперше:

- запропонована модель уніфікації будівельних елементів, що надає чіткі правила ідентифікації типів елементів за їх цифровими моделями;

- розроблений метод ідентифікації будівельних елементів за їх цифровими моделями, що дозволило автоматизувати процес декомпозиції будівельних об’єктів на складові елементи;

- запропоновані методи побудови пластинчато-стержневої моделі, що дозволило автоматизувати перехід від архітектурної моделі будівлі до конструктивної;

- створені методи виправлення колізій, що виникають під час побудови пластинчато-стержневої моделі;

- розроблені методи автоматичної генерації об’ємно-планувальної інформації на базі даних інформаційної системи Allplan та її подальша передача у Microsoft Project.

Удосконалено:

- модель життєвого циклу будівельного об’єкта, що дозволило формалізувати складності передачі інформації між етапом архітектурного проектування та етапом конструктивних розрахунків на міцність;

- пластинчато-стержнева модель, як проміжна ланка між етапами архітектурного проектування та конструктивних розрахунків на міцність, що дозволило запропонувати ефективні методи її побудови.

Отримали подальший розвиток:

- класифікація будівельних елементів та їх цифрові моделі;

- класифікація колізій перетворення архітектурної моделі в пластинчато-стержневу модель.

Теоретична цінність полягає в тому, що розроблена пластинчато-стержнева модель та методи її генерації, методи усунення колізій, що отримані внаслідок генерації, модель бібліотеки уніфікації, методи генерації календарно-планувальних даних та їх передача у Microsoft Project є внеском в розвиток багаторічних досліджень в цьому напрямку. Одержані результати дозволяють формалізувати перехід від архітектурної моделі до конструктивної, дозволяють ефективно усувати колізії в конструктивній моделі та надають змогу реалізувати автоматизовану видачу календарно-планувальної інформації у системи управління проектами. Також отримані результати надають теоретичне підґрунтя для створення нових моделей та методів, які дозволять зробити процес проектування ще більш автоматизованим.

Практичне значення одержаних результатів полягає в розробці модуля бібліотеки уніфікації, функції якого використовуються групою модулів в інформаційній системі Allplan, розробці модуля генерації пластинчато-стержневої моделі та модуля для генерації та передачі календарно-планувальної інформації у комплекси для управління проектами.

Особистий внесок здобувача. У ході досліджень автором особисто удосконалена розширювана модель будівельного об’єкта, яка відображає стан будівлі на всіх етапах життєвого циклу та формалізує його. Автору особисто належить розробка моделі інформаційної бібліотеки уніфікації. Особисто автором розроблена модель пластинчато-стержневої моделі та розроблена низка методів для її генерації та усунення колізій, які можуть виникнути. В рамках дослідження календарно-планувального модуля Allplan автором особисто розроблена інформаційна модель взаємодії Allplan та комплексів для управління проектами на прикладі Microsoft Project.

Апробація результатів дисертації. Основні результати досліджень обговорювались на III науково-практичній конференції «Modern Methodology of Science and Education» (травень 2017, м. Дубай), науково-практичній конференції молодих вчених «Build master class 2017» (КНУБА, листопад 2017, м. Київ), науково-практичній конференції «Management of the development of technologies» (КНУБА, березень 2018, м. Київ)

Публікації. За результатами дослідження опубліковано 9 друкованих праць, у тому числі 3 статті в наукових фахових виданнях відповідно переліку ВАК, 3 статті у міжнародних фахових виданнях та 3 публікації в матеріалах науково-технічних конференцій.

Обсяг і структура дисертації. Дисертація складається з основного тексту на 130 сторінках, що включає вступ, чотири розділи та основні висновки, список літератури зі 119 найменувань, що розміщений на 10 сторінках. У роботі містяться 69 рисунків та 8 таблиць.

В першому розділі проводиться аналіз процесу проектування будівель та споруд. На основі аналізу виділяться основні етапи процесу проектування та їх основні характеристики. Досліджуються та аналізуються основні типи CAx-систем для моделювання будівельних об’єктів та наводиться їх класифікація. На базі попереднього аналізу виділяються наступні інформаційні системи моделювання, а саме: Allplan, ArchiCAD та Revit.

Проводиться порівняльний аналіз обраних систем інформаційного моделювання будівельних об’єктів за їх основними функціональними та конструктивними особливостями. Розглядається поняття життєвого циклу будівельного об’єкта та проводиться аналіз всіх існуючих труднощів на кожному окремому етапі життєвого циклу. Досліджуються сучасні методи та засоби інтеграції CA-х систем для будівельних об’єктів, детально розглядаються різноманітні нейтральні формати, проводиться їх аналіз та порівняння. Пропонується вдосконалення розширеної моделі будівельного об’єкта та виділяється проблема переходу від архітектурної моделі до конструктивної.

В другому розділі проводиться аналіз проблеми уніфікації будівельних понять в комплексі процесу моделювання будівельного об’єкта на прикладі інформаційної системи Allplan. Досліджується модель об’єкта в Allplan. Пропонується інформаційна модель бібліотеки уніфікації, на базі якої в подальшому можна буде розробити інші моделі, що дозволять автоматизувати процес проектування. Детально розглядаються всі методи необхідні для створення та використання бібліотеки уніфікації.

В третьому розділі детально розглядається та аналізується проблема переходу від архітектурної моделі до конструктивної. Досліджуються сучасні проблеми побудови конструктивної моделі. Пропонується введення проміжної моделі між архітектурною та конструктивною, що значно полегшить процес переходу між цими моделями. Ця модель отримала назву пластинчато-стержнева модель (ПСМ). Наводяться та аналізуються 3 різні підходи до генерації ПСМ, детально розглядаються та описуються методи, необхідні для створення ПСМ. На базі аналізу, визначається, що методів генерації ПСМ недостатньо і після її створення виникає маса різноманітних колізій. Проводиться аналіз всіх можливих колізій ПСМ, та пропонуються методи усунення колізій, що виникли. Детально аналізуються та розглядаються всі методи усунення колізій ПСМ.

В четвертому розділі розглядається загальна архітектура інформаційної системи Allplan. На етапі аналізу календарно-планувального модуля Allplan виділяється проблема генерації та передачі інформації у комплекси для управління проектами. Пропонується розробка інформаційної моделі об’єкта для передачі у комплекси для управління проектами, на прикладі Microsoft Project. Проводиться аналіз та дослідження всіх методів розроблених у попередніх розділах на прикладі системи Allplan, проводиться аналіз їх ефективності.

Автор вважає необхідним висловити вдячність науковому керівнику дисертаційної роботи – доктору наук, професору Бородавці Євгенію Володимировичу.

РОЗДІЛ 1

АНАЛІЗ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ МОДЕЛЮВАННЯ БУДІВЕЛЬНИХ ОБ'ЄКТІВ

1.1. Будівельний об'єкт та його життєвий цикл

Будівельним об'єктом є кожна відокремлена будівля (виробничий корпус або цех, склад, житловий будинок тощо) або споруда (міст, тунель, платформа тощо) з усіма улаштуваннями (галереями, естакадами тощо), устаткуванням, меблями, інвентарем, підсобними та допоміжними пристроями, що належать до неї, а також, за необхідності, з інженерними мережами, які прилягають до неї [1]. Визначаючою особливістю будівельного об'єкта є те, що його будівництво здійснюється за окремим проектом та кошторисом.

У будівельній практиці розрізняють поняття «будівля» і «споруда». *Споруди* – це будівельні системи, пов'язані з землею, які створені з будівельних матеріалів, напівфабрикатів, устаткування та обладнання в результаті виконання різних будівельно-монтажних робіт.

Будівлі – це споруди, що складаються з несучих та огороджувальних або сполучених (несучо-огороджувальних) конструкцій, які утворюють наземні або підземні приміщення, призначені для проживання або перебування людей, розміщення устаткування, тварин, рослин, а також предметів.

До будівель відносяться: житлові будинки, гуртожитки, готелі, ресторани, торговельні будівлі, промислові будівлі, вокзали, будівлі для публічних виступів, для медичних закладів та закладів освіти тощо. На рисунку 1.1 наведено класифікацію будівель відповідно до Державного класифікатора «Будівлі та споруди» [2].

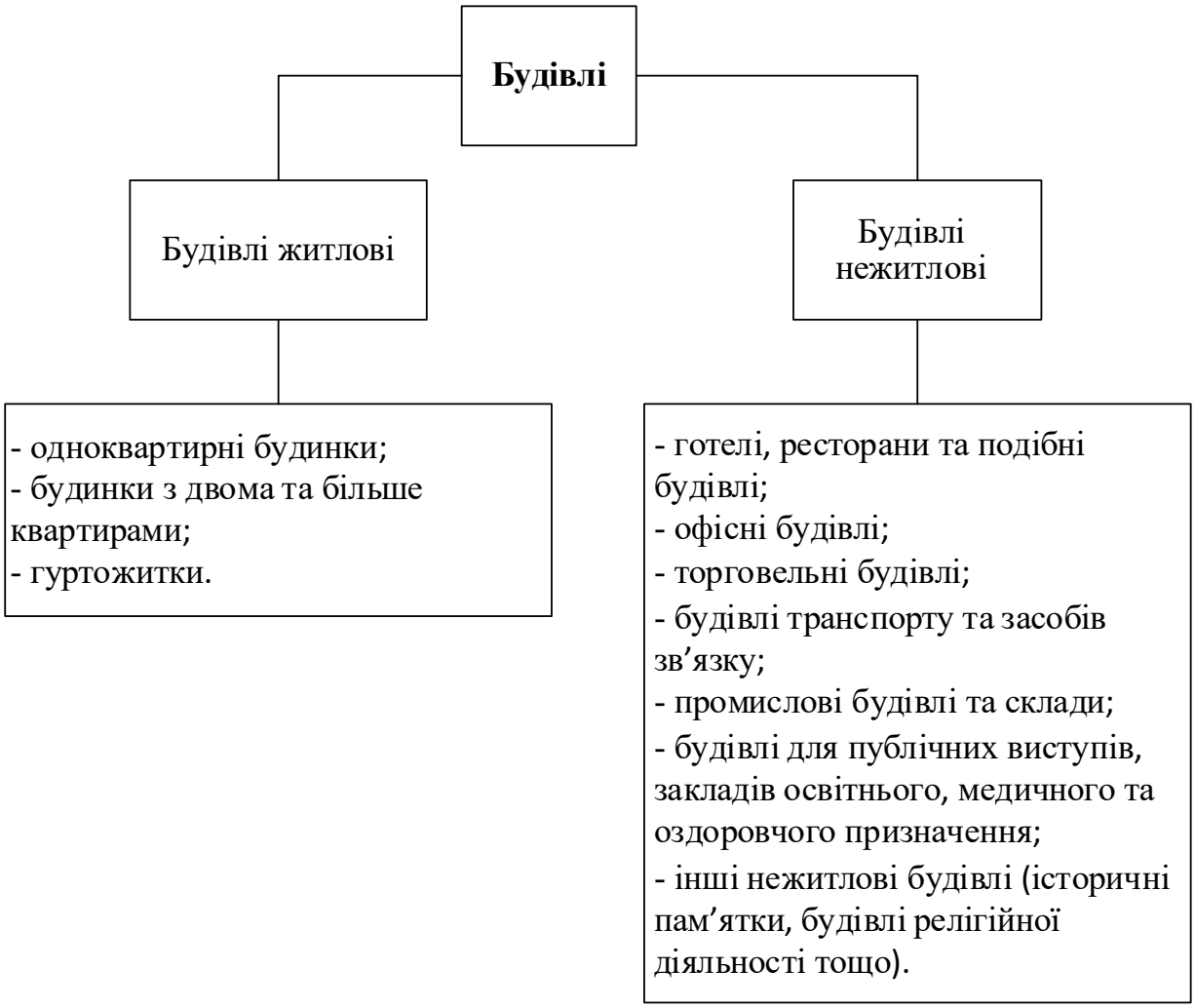


Рис. 1.1. Класифікація будівель відповідно до Державного класифікатора «Будівлі та споруди»

Інженерні споруди – це об'ємні, площинні або лінійні наземні, надземні або підземні будівельні системи, що складаються з несучих та в окремих випадках огороджувальних конструкцій і призначені для виконання виробничих процесів різних видів, розміщення устаткування, матеріалів та виробів, для тимчасового перебування і пересування людей, транспортних засобів, вантажів, переміщення рідких та газоподібних продуктів тощо. В них приміщення або відсутні, або вбудовані в них приміщення не визначають їхнього основного призначення [2].

Інженерні споруди класифікуються в основному за інженерним задумом, що визначається цільовим призначенням об'єкта. До інженерних споруд відносяться: транспортні споруди (залізниці, шосейні дороги, злітно-

посадкові смуги, мости, естакади тощо), трубопроводи та комунікації, дамби, комплексні промислові споруди, спортивні та розважальні споруди тощо. На рисунку 1.2 наведено класифікацію інженерних споруд відповідно до Державного класифікатора «Будівлі та споруди».

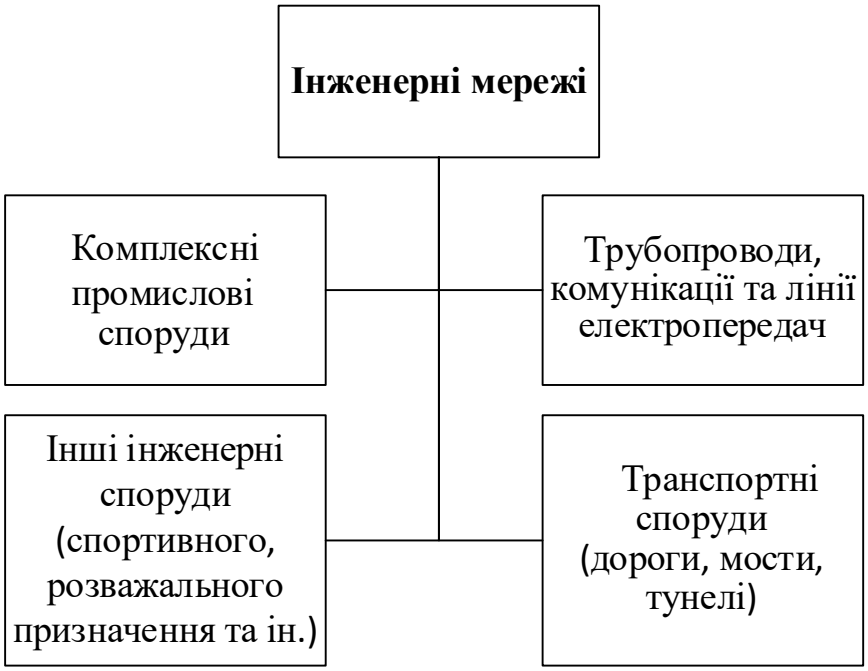


Рис. 1.2. Класифікація інженерних споруд

Кожен будівельний об’єкт (БО) можна розділити на структурні частини, які є сукупностями окремих взаємозалежних об’ємно-планувальних і конструктивних елементів, що виконують задані функції.

Різним типам будівель і споруд відповідає свій набір будівельних елементів, що визначається призначенням, сферою застосування та особливостям будови об’єкта. В деяких випадках ці набори можуть суттєво відрізнятись. До них можна віднести інженерні споруди. Це пов’язано з тим, що інженерні споруди мають більш вузьку функціональну направленість. Конструкція кожної окремої інженерної споруди визначається рядом специфічних функцій, які вона виконує. Ці відмінності унеможливлюють розробку єдиної універсальної моделі будівельного об’єкта для різних типів споруд.

Що стосується будівель, то тут ситуація зовсім інша. Для всіх видів будівель можна чітко виділити набір спільних конструктивних елементів та навіть цілих структурних частин, а вже потім, виходячи з особливостей конкретного типу будівельного об’єкта, доповняти його.

Основні конструктивні елементи або частини цивільних, промислових і сільськогосподарських будівель – це фундаменти, стіни і окремі опори, перекриття, перегородки, дахи, сходи, вікна, двері, ворота та ін. [1].

На рисунку 1.3 зображено основні конструктивні елементи, які є спільними як для житлових, так і для нежитлових будівель.

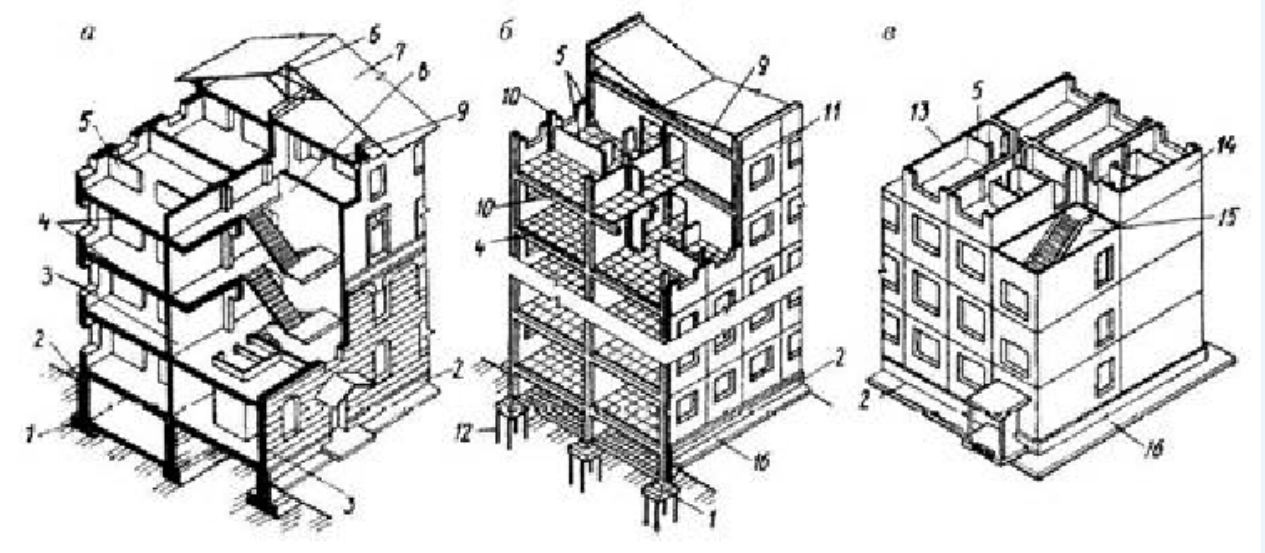


Рис. 1.3. Основні конструктивні елементи цивільних будівель

а – стара будівля; б – каркасно-панельна будівля; в – будівля із об’ємних блоків; 1 – фундамент; 2 – цоколь; 3 – несучі поздовжні стіни; 4 – міжповерхові перекриття; 5 – перегородки; 6 – крокви даху; 7 – покрівля; 8 – сходова клітка; 9 – горищне перекриття; 10 – ригелі і колони каркасу; 11 – навісні стінові панелі; 12 – палі; 13 – 15 – об’ємні блоки (13 – кімнати, 14 – санвузли та кухні; 15 – сходові клітки); 16 – відмостка.

В сучасній міжнародній та українській практиках, у зв’язку із впровадженням стандартів якості ISO 9000, введено поняття життєвого циклу

виробу (ЖЦ), яке дозволяє оцінити ефективність функціонування об’єкта на будь-якому етапі його існування. Термін ЖЦ застосовний також і для об’єктів будівництва. Життєвий цикл будівель – це час від моменту обґрунтування необхідності їх побудови до настання економічної недоцільності подальшої експлуатації. Життєвий цикл (ЖЦ) будь-якої будівлі або споруди (будівельного об’єкта - БО), починаючи від технічно-економічної задумки і закінчуючи ліквідацією, реалізується через організаційно-технологічні цикли – взаємопов’язані, неперервні і повторювані процеси (проектування, монтаж и т.д.). Життєвий цикл об’єкта складається із послідовності етапів проектування, будівництва, експлуатації, реконструкції та утилізації. Всі ці етапи можуть перекриватися, на кожному етапі задіяно багато людей та організацій. На цих етапах формуються всі основні розділи об’єкта будівництва, всі види його забезпечення (технічне, програмне, інформаційне, математичне, організаційне та ін.) [11]

На основі всіх цих даних, виділяють наступні етапи життєвого циклу об’єкта:

I – етап техніко-економічного обґрунтування зведення будівлі

II – проектування і конструювання

III – зведення з розробкою технології, організації і технологічних регламентів виробництва робіт

IV – перед-експлуатаційне освоєння

V – експлуатація будівлі, яка дозволяє забезпечити окупність засобів, вкладених в його створення та освоєння

VI – підтримка конструктивних елементів та інженерних систем будівлі в нормальному технічному стані шляхом проведення планово-попереджувальних і капітальних ремонтів.

VII – період фізичного і морального зносу, що потребує проведення модернізації, реконструкції або ліквідації будівлі.

VIII – період реконструкції, відновлювальні фізико-механічні та експлуатаційні характеристики будівель, включаючи: I, II – техніко-економічне обґрунтування та розробку технічної документації. [11]

На рисунку 1.4. зображені етапи життєвого циклу будівельних об’єктів.

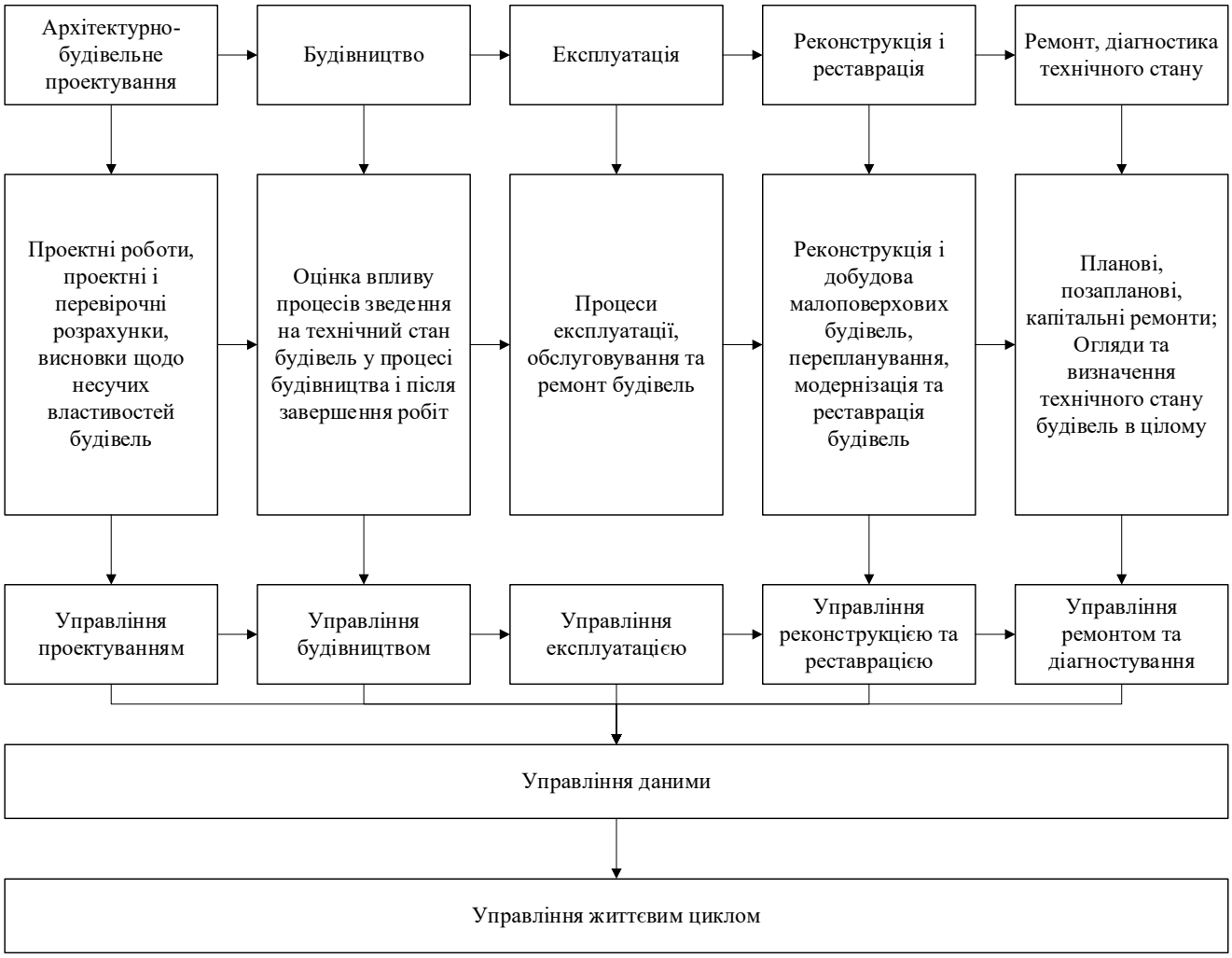


Рис. 1.4. Етапи життєвого циклу будівельних об’єктів

1.2. Дослідження сучасних CAx-систем для моделювання будівельних об’єктів

Спочатку проведемо класифікацію існуючих CAx-систем. Існує декілька класифікацій систем автоматизованого проектування:

1. *За рівнем формалізації розв’язуваних задач*: системи побудовані на повністю формалізованих методах розв’язання проектних задач; системи, що проводять

проектні роботи, які не піддаються повній формалізації; системи, що організовують пошук розв’язання неформалізованих задач.

2. *За функціональним призначенням*: системи розрахунково-оптимізаційні; графічні системи; системи автоматизованого проектування конструкцій; графоаналітичні системи; системи підготовки технічної документації; системи обробки результатів експериментальних досліджень; інформаційні системи; системи технологічної підготовки програм для верстатів з числовим програмним управлінням (ЧПУ).

3. *За спеціалізацією*: спеціалізовані системи; інваріантні системи.

4. *За технічною організацією*: системи з центральним процесорним керуванням; системи, що комплектуються автоматизованими робочими місцями конструктора (АРМ); системи з власними обчислювальними ресурсами. Така класифікація придатна до будь-яких інформаційних систем проектування, але ми будемо розглядати її лише в розрізі систем автоматизованого проектування в будівництві. Якщо поглянути на дану класифікацію з точки зору існуючих сучасних інформаційних систем проектування, то легко зробити висновок, що вона не зовсім адекватна сучасним реаліям. Звичайно ніхто не каже, що вона неправильна, але якщо ми будемо намагатися застосувати дану класифікацію до сучасних інформаційних систем, то в нас нічого не вийде з однієї простої причини – кожна з існуючих систем підпадає під декілька категорій. А все тому, що інформаційні системи проектування постійно удосконалюються і об’єднують в собі все більше різноманітних функцій. І це добре, тому що ми рухаємося в напрямку універсалізації всіх систем автоматизованого проектування. Хоча очевидно, що на даному етапі розвитку абсолютна універсалізація неможлива з різних причин.

У вітчизняній науці та літературі поняття автоматизованої систему проектування досить широке і охоплює великий спектр задач в різних галузях.

В західній літературі застосовуються більш вузькоспеціалізовані поняття, які в останні роки все частіше проникають і в нашу літературу. Зараз поширений англомовний загальний термін, що позначає різноманітні технології автоматизації за допомогою комп'ютера CAx – Computer Aided. В свою чергу технології CAx поділяються за функціональними напрямами. Серед них виокремлюють три основні складові:

1. Computer Aided Design (CAD) – автоматизоване проектування. Термін використовується для позначення широкого спектру комп'ютерних інструментів, які допомагають інженерам, архітекторам та іншим професіоналам у здійсненні проектування. Будучи ключовим інструментом у рамках концепції управління життєвим циклом виробу, системи автоматизованого проектування включають в себе безліч програмних і апаратних засобів – від систем двовимірного креслення до тривимірного параметричного моделювання поверхонь і об'ємних тіл. Найчастіше саме цей термін і ставлять у відповідність нашому терміну САПР. По областях застосування CAD традиційно поділяються на: архітектурно-будівельні (ArchitectureEngineeringand Construction CAD – AECCAD); механічні (Mechanical CAD – MCAD); електронні (Electronic CAD – ECAD або ElectronicDesignAutomation – EDA); технологічні (Computer-Aided ProcessPlanning – CAPP).

2. Computer Aided Engineering (CAE) – автоматизоване конструювання. Використання спеціального програмного забезпечення для проведення інженерного аналізу міцності та інших технічних характеристик компонент і складань, виконаних в системах автоматизованого проектування (CAD). Програми автоматизованого конструювання дозволяють здійснювати динамічне моделювання, перевірку та оптимізацію виробів та засобів їх виробництва. Традиційні галузі аналізу включають в себе: аналіз напруженості деталей і складань методом скінченних елементів; аналіз теплових та рідинних потоків методами обчислювальної гідродинаміки; аналіз

кінематики; моделювання динамічних механічних взаємодій; моделювання виробничих операцій (лиття, пресування та ін.). У процесі проведення будь-якого виду аналізу в системах CAE традиційно виокремлюються три етапи його проведення: попередня обробка даних (побудова за геометричною моделлю виробу (CAD-даним) потрібної моделі досліджуваного процесу – наприклад, сітки скінченних елементів, точок докладання зусиль та їх векторів); аналіз моделі за допомогою спеціалізованого обчислювача; заключна обробка результатів (візуалізація результатів розрахунків математичної моделі).

3. Computer Aided Manufacturing (CAM) – автоматизоване виробництво. Термін використовується для позначення програмного забезпечення, основною метою якого є створення програм для керування верстатами з ЧПУ (Computerized Numerical Control – CNC). Вхідними даними CAM – системи є геометрична модель виробу, розроблена в системі автоматизованого проектування (CAD). В процесі інтерактивної роботи з тривимірною моделлю в CAM-системі інженер визначає траєкторії руху різального інструменту по заготовці виробу (так звані CL-дані, від cutterlocation– положення різця), які потім автоматично верифікуються, візуалізуються (для візуальної перевірки коректності) і обробляються постпроцесором для отримання програми управління конкретним верстатом. Ще один спосіб класифікації САПР – за характером базової підсистеми. В цій класифікації розрізняють такі САПР [5].
1. САПР на базі підсистеми машинної графіки та геометричного моделювання. Ці САПР орієнтовані на пристрої, де основною процедурою проектування є конструювання, тобто визначення просторових форм та взаємного розташування об’єктів. Тому до цієї групи належить більшість графічних ядер САПР в області машинобудування.

2. САПР на базі СУБД. Вони орієнтовані на програми, в яких при відносно нескладних математичних розрахунках обробляється великий об’єм даних. Такі САПР переважно зустрічаються в техніко-економічних програмах,

наприклад при проектуванні бізнес-процесів, але застосовуються також при проектуванні об’єктів, що подібні до щитів управління в системах автоматики.

3. САПР на базі конкретного прикладного пакету. Фактично це автономно використовувані програмно-методичні комплекси, наприклад, імітаційного моделювання виробничих процесів, розрахунку міцності за методом скінченних елементів, синтезу і аналізу систем автоматичного управління і т.д. Часто такі системи належать до систем CAE. Прикладами можуть слугувати програми логічного проектування на базі мови VHDL, математичні пакети типу MathCAD.

4. Комплексні (інтегровані) САПР, що складаються із сукупності підсистем попередніх видів. Характерними прикладами комплексних САПР є CAE/CAD/CAM системи в машинобудуванні або САПР великих інтегральних схем (ВІС). Так, САПР ВІС включають в себе СУБД і підсистеми проектування компонентів, принципових, логічних і функціональних схем, топології кристалів, тестів для перевірки придатності виробів. Для керування такими складними системами застосовують спеціалізовані системні середовища. Іншим, більш простим, варіантом класифікації САПР є класифікація за вартістю: – САПР нижнього рівня: $500–$2000 за робоче місце; – САПР середнього рівня: $2000–$20000 за робоче місце; – САПР високого рівня: понад $20000 за робоче місце. Звичайно таким методом можна класифікувати всі наявні САПР, але така класифікація малоінформативна по відношенню до їх спеціалізації і функціонального призначення. Така класифікація може використовуватися як додаткова до вже існуючих, розширюючи інформативну частину ціновими показниками. Хоча, зважаючи на нестабільність курсів валют і цінової політики взагалі, покладатися на абсолютну точність цих показників не варто. З усіх наведених класифікацій, найбільш точною і зрозумілою, є класифікація, що використовується в західній літературі. Вона розділяє всі САПР за напрямами, що робить її досить зручною та зрозумілою. Для більшої зручності розуміння ієрархії систем CAx

пропонується їх класифікацію подати у вигляді схеми (рис. 1.5). Це далеко не повний перелік всіх напрямків CAx-систем, декомпозицію можна продовжувати і далі, але нас цікавить лише один напрямок – архітектура і будівництво. САПР цього напрямку в подальшому будемо називати просто AECCAD або архітектурно-будівельні САПР, щоб уникати неоднозначності поняття САПР, яке в цілому набагато ширше**.**

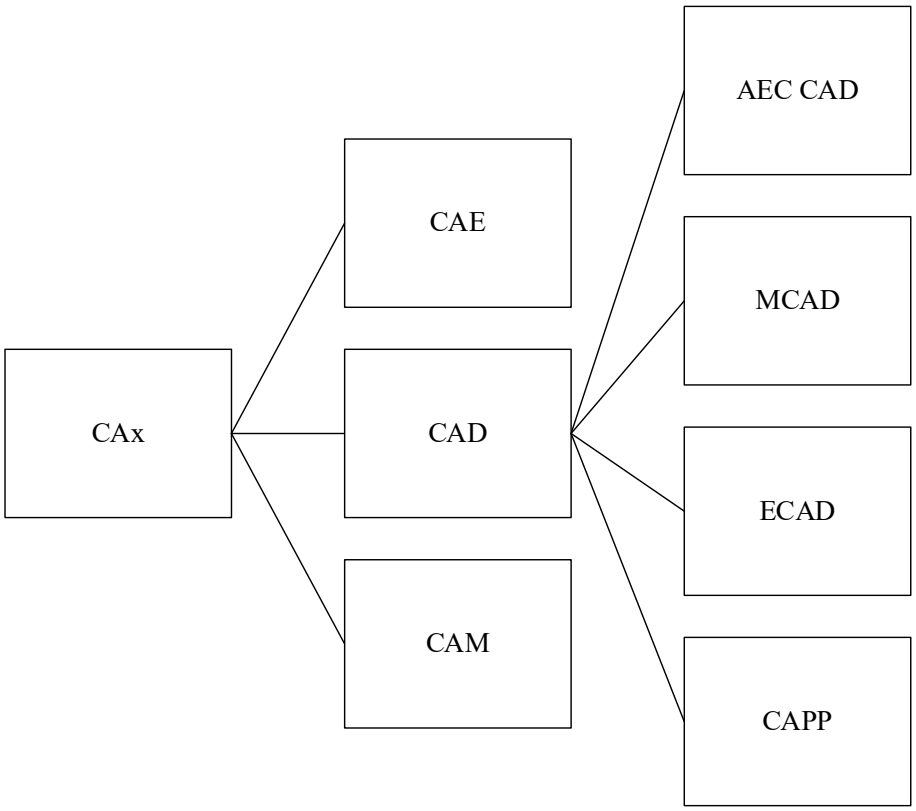


Рис. 1.5. Класифікація CAx-систем

Слід звернути увагу на такий момент – хоча в наведеній класифікації CAx-систем можна знайти найбільш близький відповідник визначенню САПР (це – CAD), все ж не варто відкидати всі інші. У визначеннях CAE і CAM зазначається, що вони в якості вхідної інформації використовують результати проектування в CAD-системах, а значить їх спільне використання забезпечує автоматизацію всього життєвого циклу продукту (об’єкта). Тому всі CAx-системи є складовою управління життєвим циклом продукту – Product Lifecycle Management (PLM).

В Україні та країнах східної Європи найбільш розповсюдженим є програмний пакет AutoCAD від компанії Autodesk. Розроблений Autodesk більш ніж 35 років тому назад, AutoCAD довгий час відповідав більшості вимог проектувальників. Але дивлячись на сьогоднішній день, навіть маючи величезний інструментарій та багато різноманітних можливостей адаптації до потреб користувачів, AutoCAD не задовольняє потреб більшості проектувальників. Цей пакет може застосовуватись лише при розробці дуже малих і достатньо простих проектів, автоматизуючи лише рутинну роботу кульмана і не більше. AutoCAD – хороший інструмент для креслення з великою кількістю можливостей, але сучасному проектувальнику потрібно набагато більше, ніж просто швидке та красиве створення креслень. До найбільших недоліків можна віднести відсутність можливості створення параметричної моделі та слабкі 3D можливості, а також моральну застарілість даного продукту.

Серед CAD-систем, які спеціалізуються на автоматизації процесу створення архітектурної моделі будівельного об’єкта виділимо наступні:

- ArchiCAD (Graphisoft);
- Revit Structure Suite (Autodesk);
- Allplan (Nemetschek);

Кожна із систем використовує концепцію віртуального будівельного об’єкта.

«Віртуальним будівельним об’єктом» називається інженерна комп’ютерна система будівлі, споруди або будівельного комплексу, призначена для вирішення завдань створення, управління, моніторингу та аналізу поведінки в кожен момент життєвого циклу реального об’єкта від задуму до утилізації. Згідно з цим визначенням «Віртуальний будівельний об’єкт» призначений для збору і використання інформації про об’єкт будівництва. Він радикально спрощує доступ до повної, достовірної та актуальної інформації в кожен

момент життєвого циклу об’єкта будівництва. «Віртуальний будівельний об’єкт» дозволяє:

• реалізувати нову технологію інтерактивного комп’ютерного проектування і управління всіма процесами життєвого циклу об’єкта будівництва в будь-який момент часу від задуму до утилізації;

• підвищити якість проектування об’єкта будівництва і пов’язаного з ним інвестиційного процесу;

• швидко і якісно приймати обґрунтовані планові та надзвичайні рішення з використанням повної, актуальної та достовірної інформації, накопиченої по об’єкту будівництва за час його існування;

• реалізувати стійкий інтерактивний зв’язок з інженерними системами об’єкта будівництва і ефективно управляти ними на основі теорії функціональних систем;

• реалізувати дистанційне керування окремими приладами, механізмами, інструментами, датчиками та іншим обладнанням, що знаходяться в будівлі або споруді;

• влаштовувати комп’ютерні презентації об’єкта будівництва в цілому і будь-яких його систем з показом змін в часі, а також в умовах віртуальної реальності;

• проводити математичні дослідження моделей об’єкта будівництва, розробляти і застосовувати нові моделі, методи і алгоритми досліджень.

Виходячи з визначення і призначення «Віртуального будівельного об’єкта» можна стверджувати, що він складається з трьох основних частин: програмно-інформаційного забезпечення, технічного забезпечення та інтерфейсу з користувачами.[11]

Перераховані вище системи реалізують BIM-технологію, тобто реалізують підхід при якому в єдину базу вноситься інформація про всі аспекти

будівельного об’єкта. Особливістю підходу BIM є те, що в процесі проектування він передбачає комплексний збір і обробку всієї архітектурно-конструкторської, технічної, економічної інформації про будівлю з усіма її взаємозв’язками та залежностями, будівля і все що має до неї відношення розглядається як один об’єкт. Тривимірна модель будівельного об’єкта пов’язана з інформаційною базою даних, у якій кожному елементу моделі є можливість присвоїти додаткові атрибути. За рахунок того, що об’єкт розглядається як єдине ціле, зміна одного параметру призводить до автоматичної зміни інших параметрів та об’єктів, пов’язаних з цим об’єктом, аж до креслень, візуалізацій, специфікацій та календарного графіка. До основних переваг застосування BIM-технологій можна віднести: точність проектів, їх зрозумілість для замовника за рахунок 3D-візуалізації, економія часу проектування і будівництва, зменшення вартості будівництва і експлуатації.

Розглянемо детальніше кожну із систем.

ArchiCAD - програмний пакет, що забезпечує розробку будь-яких архітектурно-дизайнерських рішень. В ArchiCAD можна одночасно працювати над створенням проекту і складати супутню будівельну документацію, оскільки програма зберігає інформацію про будинок, що проектується: плани, розрізи, перспективи, перелік необхідних будматеріалів, а також зауваження архітектора, зроблені в процесі роботи. На будь-якому етапі роботи можна побачити спроектовану будівлю у 3D вигляді, в розрізі, в перспективі, зробити анімаційний ролик. У 1982 році Graphisoft стала першою компанією, яка запропонувала 3D-моделювання на персональному комп’ютері. Основним поняттям ArchiCAD є «Віртуальна Будівля» (Virtual Building). «Віртуальне Будівля» – це тривимірна модель проекту, яка містить всю інформацію необхідну для роботи з ним, та існує разом з проектом протягом всього часу проектування. Таким чином, під час роботи в ArchiCAD не просто створюються окремі креслення. Отримується повний набір

проектної документації в одному файлі: поверхові плани, розрізи і фасади, дані про приміщення, специфікацію матеріалів і виробів, будівельно-технічна документація, малюнки та зображення фотореалістичної якості, демонстраційні відеоролики і сцени віртуальної реальності. Серед основних концепцій ArchiCAD – здатність обробляти 3D-модель і 2D-креслення в одному робочому документі. Це досягнуто за допомогою поняття «Інтелектуальні Компоненти Будівлі» (Intelligent Building Components), які містять повну інформацію про об'єкт, і є частиною глобальної «Віртуальної Будівлі». Завдяки тому, що використовується «Вбудована база даних Віртуальної Будівлі» (Virtual Building's Integrated Database), будівельно-технічна документація та інші файли можуть бути отримані автоматично безпосередньо з 3D-моделі будівлі без використання додаткового програмного забезпечення і зайвих дій. Використання бази даних «Віртуальної Будівлі» має ще одну позитивну рису, – якщо в проекті щось виправити, ця зміна вноситься тільки один раз в ArchiCAD, при цьому всі документи, а саме плани, розрізи / фасади, 2D та 3D види , автоматично змінюються. Такий підхід дозволяє скоротити час розробки проекту і уникнути можливих помилок. На відміну від інших САПР, написаних для інженерів, а пізніше адаптованих для архітектури, ArchiCAD спочатку розроблявся виключно для роботи в галузі будівельної індустрії. Тому інтерфейс та інструментарій програми є засобами, звичними для архітекторів. До переваг цього програмного комплексу можна віднести наступні фактори: найпопулярніша і нейтральна архітектурна платформа, що володіє зрозумілим, інтерактивним і легко засвоюваним призначеним для користувача інтерфейсом.[11]

Недоліки ArchiCAD. Система не інтуїтивна і, все-таки, досить чужорідна до Windows. Багато незручностей від того, що програма була колись розрахована на слабкі комп'ютери і тому має принципові обмеження. Будівля розглядається тільки по поверхах, використовуються шари, погана система підказок при побудові. Є параметричні обмеження при моделюванні, а також

обмеження масштабування. Оскільки конструктори, проектувальники інженерних мереж та інші суміжники працюють в AutoCAD, доводиться зберігати інформацію в форматі dwg. А модуль експорту / імпорту directDWG не доопрацьований, так що присутні проблеми з помилками в переданих файлах. Бібліотечні об'єкти не вносяться в проект, тобто при передачі доводиться їх копіювати окремо або синхронізувати всі бібліотеки. Через це часто виникають різночитання одного і того ж проекту на різних комп'ютерах. Модуль експорту в розрахункові програмні комплекси відсутній, створити аналітичну модель в ArchiCAD неможливо. [11]

Revit. Спеціалізоване рішення для архітектурного і будівельного проектування на базі AutoCAD може застосовуватися архітекторами, будівельниками, проектувальниками і дизайнерами. Програма є незалежним «вертикальним» рішенням на базі AutoCAD. В Revit додані власні «інтелектуальні» параметричні об'єкти, на підставі яких ведеться концептуальне і робоче проектування. REVIT – це спеціалізоване рішення архітектурного та будівельного проектування з підтримкою обміну даними з AutoCAD. REVIT реалізує BIM-технологію. Нативний експорт в DWG-формат. Є можливість зробити весь проект на базі однієї тривимірної платформи. [11]

Недоліки Revit. Недостатня параметризація, багато ручної роботи при внесенні змін. Обмежений зв'язок з іншими промисловими платформами. Слабо реалізовані вітчизняні норми, проблематично створити автоматичні специфікації по вітчизняним стандартам. Обмежене масштабування, низька продуктивність при роботі з великими моделями. [11]

Allplan. Програмний комплекс Allplan від Nemetschek поєднує в собі зручність при користуванні обміном даними і ефективним плануванням витрат. Allplan, в якості платформи BIM, забезпечує основу для проектування 3D-моделі. Allplan САПР дозволяє виявити і усунути розбіжності на ранній стадії при створенні 3D-моделі на основі будівлі. Якщо модель правильна, то відповідні креслення (розташування і армування створеної арматури також будуть

правильними. У Allplan реалізована підтримка багатокористувацької платформи і конкурентних операцій при роботі з одним об’єктом, але поки ідея реалізована недосконало, оскільки є маса неконтрольованої інформації. Програма дуже добре пов’язує всі розділи проектування (архітектуру, інженерію, водопостачання, каналізацію і т.д.). Позитивним є наявність величезної кількості бібліотек і постійне їх розширення. Є можливість створення своїх елементів і бібліотек. Хороша 3D візуалізація. Не можна не відзначити високу продуктивність при деталізації залізобетонних конструкцій. [11]

Недоліки Allplan. Система дуже громіздка, досить складний для користувача інтерфейс, змішана технологія 2D-3D плану; недостатньо розвинене проектування металоконструкцій; відсутня хороша зв’язка з розрахунковими програмними комплексами. [11]

Для наочності наведемо порівняння описаних інформаційних систем у вигляді таблиці.

Таблиця 1.1

Порівняння інформаційних систем автоматизованого проектування за їх основними функціональними особливостями

№	Функціональні особливості САПР	Autodesk Revit Structure Suite	Nemetschek Allplan	ArchiCAD
1	Відкритий API	+	-	+
2	Динамічна зміна специфікацій	+	-	+
3	Параметризація об’єктів	+	+	+
4	Підтримка концепції BIM	+	+	+

5	Зв’язок із кошторисними програмами	± (необхідне доопрацювання баз даних)	+ (Додатковий модуль)	-
6	Генерація укрупненої 3D моделі та перевірка колізій	+	+ (Додатковий модуль)	-
7	Автоматична генерація аналітичної моделі	+	-	-
8	Двосторонній зв’язок із розрахунковими програмами	+ (відсутній імпорт результатів розрахунку для залізобетонних монолітних плит перекриття)	+	-
9	Односторонній зв’язок із розрахунковими програмами	+ (Revit Structure, AutoCAD SD Залізобетон)	+ (підтримка різноманітних форматів обміну даними: PDF 8 (2D та 3D імпорт/експорт); • IFC 2x3 (імпорт/експорт); • IFC 4 (імпорт/експорт); • DWG (імпорт/експорт);	-

			• DGN V8 (імпорт/експорт); • SKP; • ODBS; • AVI; • gbXML; BMP, JPG, TGA, TIF)	
10	Автоматичне створення детальної моделі ЗБК	+	+	-
11	Автоматичне створення специфікацій на основі 3D моделі	+ (Revit Structure, AutoCAD SD Сталь)	+ (недостатньо локалізовані вітчизняні норми)	±
12	Відповідність актуальним нормам проектування	+	+	±
13	Автоматичне створення детальної 2D моделі ЗБК	+ (AutoCAD SD Залізобетон)	+	-
14	Автоматичне створення специфікацій на основі моделі	+ (AutoCAD SD Залізобетон) ± (Revit Structure, необхідне	-	±

		створення баз даних)		

Проаналізуємо системи на можливості та переваги з конструкторської точки зору.

Таблиця 1.2

Порівняння інформаційних систем автоматизованого проектування за основними конструктивними можливостями

	Autodesk Revit Structure Suite	**Nemetschek Allplan**
Можливість зміни параметрів конструктивних елементів	+	+
Генерація розрізів конструктивних елементів	-	-
Автоматична генерація абсолютно твердих тіл	-	-
Можливість створення розрахункових моделей капітелей	-	-
Графічне задання навантажень	+	-
Автоматичний збір вітрового навантаження	-	-
Задання умов спирання	+	+

Моделювання процесу зведення	-	-

1.3. Аналіз існуючих труднощів в інтеграції різних стадій життєвого циклу будівельного об’єкта

В теперішній час для проектних організацій і фірм, а також підрядних та субпідрядних підприємств, незалежно від форми власності, стало актуальним завдання підвищення ефективності процесу проектування і зведення будівель і споруд, а також забезпечення нової якості керованості цим процесом за рахунок створення єдиного інформаційного простору підприємства. Досягти цього можна лише володіючи повною достовірною оперативною інформацією про всі етапи проектування та зведення споруди. Реальним інструментом для досягнення поставленої мети є комплексна інтеграція окремих підсистем всього підприємства [91].

Своєчасність інтеграційних процесів на підприємстві обумовлена такими факторами як:

- підвищення ефективності виробництва можлива тільки на основі об’єктивної картини технічних і технологічних параметрів;
- існуючі інформаційні та організаційні бар’єри між управляючими і технологічними рівнями підприємства приводять до блокування інформації, важливої для аналізу діяльності підприємства, а також різко знижають оперативність прийняття управлінських рішень;
- ринок засобів і систем автоматизації пропонує всі необхідні компоненти для здійснення комплексної інтеграції, тобто для побудови «Інтегрованої автоматизованої системи управління» (ІАСУ).

Окрім цього, комплексна інтеграція сприяє створенню в рамкам підприємства єдиного банку даних, який містить в собі інформацію про продукцію, технологічні процеси, дані допоміжних виробництв,

знижує ступінь дублювання інформації і забезпечує стандартизацію усієї діяльності підприємства. [91].

На сьогоднішній день, інтеграція програмних застосунків в ті чи інші виробничі процеси є першочерговим завданням інформаційних технологій. Зараз на ринку існує маса різноманітних програмних застосунків та цілих інформаційних платформ, метою яких є вирішення тих чи інших проблем інтеграції.

У промисловості виділяють два типи інтеграції, а саме: вертикальна та горизонтальна. Ці види інтеграції справедливі і для будівництва.

Вертикальна інтеграція – це інтеграція систем управління підприємством і систем управління технологічними процесами з метою забезпечення максимальної ефективності всіх систем автоматизації. Вона ґрунтується на організації потоків інформації від нижнього рівня (датчики і контролери технологічного устаткування) у внутрішні і зовнішні комп'ютерні мережі підприємства і через них в адміністративні системи управління. Дане завдання вирішується на основі об'єднання промислових і адміністративних мереж [91].

Горизонтальна інтеграція – це об'єднання між собою всіх автономних систем автоматизації технологічних і виробничих процесів, а також адміністративних підрозділів цехового рівня в єдину інформаційну мережу. Інтеграція по горизонталі тісно пов'язана з вирішенням проблем удосконалювання господарського механізму [91].

На сьогодні в числі основних принципів інтеграції інформаційних ресурсів виділяють наступні: інтеграція корпоративних застосунків (EAI), інтеграція між організаціями (B2BI), інтеграція бізнес-процесів (BPI), інтеграція інформації (EII), а також отримання перетворення і завантаження даних (ETL). Кожен з принципів визначає загальні методи інтеграції, не забезпечуючи конкретних рішень для певних видів інформаційних ресурсів та інформаційних систем. Тому на практиці під час вирішення конкретної

проблеми використовують поєднання цих принципів, відштовхуючись від вирішуваного завдання і найбільш відповідних технологій реалізації (DCOM, CORBA, OLAP, GIS, Sun RPC, XML, Web-сервіси) [21]. Також на даний момент прослідковується тенденція розробки програмних комплексів інформаційних систем як композиції сервісів (SAAS – Software As A Service) [20].

Проаналізувавши більшість існуючих CAD-систем і програмних засобів, доступних на світовому інформаційному ринку технологій автоматизованих систем проектування будівель і споруд, ми можемо відмітити значну проблему їх інтеграції. Основною проблемою є те, що кожна система намагається зосередити свою увагу на якомусь певному етапі життєвого циклу будівельного об’єкта, в результаті чого кожна система працює зі своєю, унікальною моделлю будівельного об’єкта. За таких умов інтеграція різних систем є доволі складною задачею, адже потрібно враховувати всі особливості проектування в тій чи інший CAD-системі.

Коли на початку сімдесятих років з’явилися пристрої для виводу графічної інформації (плотери), були створені передумови для об’єктної інтеграції автоматизованих процесів проектування. У цей період були розпочаті розробки інтегрованих систем, які для достатньо широкого класу будівельних об’єктів автоматизували основні процеси проектування: інженерно-технічні розрахунки, конструювання, проектування інженерного устаткування, випуск робочих креслень, кошторисно-фінансову частину [11].

Розробка інтегрованих систем йшла по різних напрямах. Як і раніше достатньо важливою залишалася проблема інтеграції. Одним з підходів інтеграції програмних комплексів був ряд розробок, заснованих на об’єднанні окремих самостійних підсистем під управлінням єдиної СУБД на основі єдиної бази даних. Прикладом такої розробки є технологічна лінія проектування (ТЛП) ЛІРІКА (ЛІРА і КАРКАС), розроблена фахівцями

НДІАСБ для автоматизованого проектування каркасних будівель по різних серіях із збірного залізобетону [11].

Щодо напрямку архітектурно-будівельного проектування, то тут нам потрібно спочатку розібрати усі етапи проектування, зробити це нескладно, оскільки процес проектування будівель та споруд в останні декілька десятиліть залишається незмінним. Приведемо перелік всіх ключових етапів проектування:

1 етап – вибір земельної ділянки для забудови. На даному етапі окрім, безпосередньо, вибору земельної ділянки відбувається геологічна розвідка ґрунту, для визначення його характеристик та правильного вибору фундаменту. Досліджується сейсмологічна активність місцевості, максимальна глибина промерзання ґрунту, товщина сніжного покрову, максимальний та мінімальний температурні режими;

2 етап – архітектурне планування будівельного об’єкта; Даний етап може проходити паралельно з першим етапом. На цьому етапі архітектор створює архітектурну модель будівельного об’єкта. На даному етапі може бути застосована низка програмних комплексів, які значно прискорюють процес проектування будівлі;

3 етап – конструювання та створення розрахункової схеми, розрахунок напружено-деформованого стану об’єкта. На даному етапі інженер-проектувальник розробляє комп’ютерну модель будівельного об’єкта із врахуванням даних, отриманих із архітектурної моделі та даних про місцевість;

4 етап – проектування інженерних мереж. Сюди входить проектування сантехнічних мереж (водопостачання, опалення, газопостачання, каналізації, вентиляції) та електротехнічних мереж (розрахунок та розстановка електроприладів). Для проектування інженерних мереж є також низка спеціальних програмних засобів, деякі з них у вигляді окремих програмних

комплексів, а деякі є додатковими модулями до тих чи інших систем проектування.

5 етап – розрахунок кошторисної вартості будівництва. На цьому етапі інженер розробляє масу різноманітних кошторисів, де враховується об’єм всіх робіт, які необхідно виконати, визначаються всі необхідні матеріали, машини, механізми та обладнання. Цей етап теж доволі добре автоматизований.

Ми привели основні етапи проектування будівельного об’єкта, але потрібно зауважити, що безпосередньо виконання робіт не починається відразу після видачі кошторисної документації, тобто після 5-го етапу. Перед цим є дуже важливий етап, а саме **календарне планування**. Цей етап дозволяє в подальшому ефективно управляти процесом будівництва. На даному етапі весь об’єкт будівництва розбивається на так звані «захватки», також створюються графіки виконання робіт.

Як бачимо, процес проектування є доволі комплексним, в ньому задіяно багато різних спеціалістів. Причому кожен з них використовує спеціалізований програмний застосунок для вирішення поставленої задачі. А оскільки процес проектування складається з етапів, то кожному спеціалісту для виконання своєї задачі, часто потрібні результати попереднього етапу, які виконані іншим спеціалістом в іншому програмному комплексі. Саме це все і складає проблему інтеграції та робить її першочерговим завданням для вирішення.

Приведемо схематичне зображення послідовності етапів проектування (рис. 1.6).

Рис. 1.6. Схема інтеграції етапів управління життєвим циклом будівельного об’єкта

Цифрова модель об’єкта (ЦМО, англійською BIM – Building Information Model) – дана модель будівельного об’єкта служить для вирішення різноманітних задач протягом усього його життєвого циклу. Детальніше про неї буде написано в наступному розділі.

Можемо зробити висновок, що проблема інтеграції в архітектурно-будівельному проектування не нова і досі не має повноцінного та комплексного рішення. Перспективним напрямком для вирішення цієї проблеми є розробка та доопрацювання ЦМО. Поняття ЦМО можна інтерпретувати, як «Віртуальну будівлю», тобто модель яка знаходиться у пам’яті комп’ютера і повністю відповідає реальній будівлі. Застосування ЦМО дозволить вирішити багато проблем передачі інформації між програмними комплексами. Оскільки інформація з різних етапів проектування буде знаходитись в одній моделі, це забезпечить повноту та коректність даних про будівельний об’єкт. Також кожен наступний програмний комплекс буде отримувати дані із ЦМО, що дозволить вирішити проблему із повторним вводом даних про будівельний об’єкт та відкине можливість виникнення низки помилок внаслідок некоректного вводу даних на якомусь із етапів проектування.

1.4. Дослідження сучасних методів і засобів інтеграції CAx-систем для будівельних об'єктів

Різноманітні CAD/CAM/CAE-системи зберігають дані про власну модель у різному вигляді, кожна система має свої певні технічні вимоги до даних та їх виду. Тому у випадку, коли ми хочемо передати дані із однієї системи в іншу, нам потрібно перетворити дані однієї системи з урахуванням усіх технічних вимог та обмежень у формат, відповідний до вимог та обмежень іншої системи. Виходить, що для обміну даними між двома системами, нам потрібно мати два конвертера, такі конвертери, для кожної пари систем називають прямими конверторами. Якщо у нас буде N систем, то в такому випадку нам потрібно буде розробити $\mathrm{N} * (\mathrm{N} - 1)$ конверторів, оскільки кількість пар систем дорівнює $\frac{\mathrm{N}*(\mathrm{N}-1)}{2}$. Наприклад, у нас буде 5 систем і нам потрібно буде забезпечити обмін даними між ними, застосуємо формулу описану вище і отримаємо 20, саме така кількість конверторів нам потрібна, для забезпечення обміну між кожною парою систем. Можна зробити висновок, що такий підхід є неефективним, так як потребує розробки занадто великої кількості конверторів. Окрім цього, додання однієї системи до N систем, потребує розробки $2N$ додаткових конверторів.

Можна забезпечити обмін даними, додавши нейтральну базу даних, або так званий «нейтральний файл», що буде незалежним від існуючих систем. Такий файл буде свого роду проміжним, між різними структурами баз даних систем. За такого підходу нам потрібно буде 2 конвертори для однієї системи, перший буде перетворювати дані із власного формату системи у нейтральний, такий конвертер називається **препроцесором**. А другий буде виконувати зворотне перетворення, такий конвертер називають **постпроцесором**. Тобто кількість конверторів для N систем буде становити $2N$, а при введенні нової системи потрібно буде розробити 2 конвертори. У порівнянні з першим варіантом даний підхід є набагато ефективнішим, адже не потрібно створювати постійно зростаючу кількість конверторів, саме це ї є основною причиною визнання

другого способу головним. Щодо переваг першого способу, то прямі конвертори працюють набагато швидше, а згенеровані ними файли мають значно менший розмір. Також у випадку коли ми переносимо дані за допомогою нейтрального файлу, як правило, деякі дані втрачаються.

Розглянемо наступні формати нейтральних файлів:

- IGES
- DXF
- STEP
- IFC

IGES був першим стандартним форматом обміну даними, розробленим для передачі між різноманітними САПР. Ранні версії формату булі розраховані на CAD/CAM – системи, тобто в основному для обміну кресленнями. Пізніше формат був розширений для передачі більш широкого спектру даних.

IGES-файл складається із шести розділів. Розділи повинні йти в чіткому порядку:

1) Flag (необов’язковий);
2) Start (Початок)
3) Global (Глобальні дані)
4) Directory Entry (DE – каталог даних)
5) Parameter Data (PD – параметричні дані)
6) Terminate (Кінець)

При використання препроцесорів та постпроцесорів з нейтральним форматом IGES виникають наступні проблеми:

- внутрішній спосіб подання елемента у системі може відрізнятися від способу подання в IGES. Наприклад, дуга кола в одній системі може бути визначена через центр, радіус, початковий та кінцевий кути, а в IGES вона визначається через центр, початкову точку та кінцеву.

Внаслідок цього, IGES конвертор повинен буде виконати перетворення з використанням параметричного рівняння дуги. Таке перетворювання повинно буде виконуватись двічі, при прямій та зворотній конвертації. Також слід враховувати, що кожного разу параметри дуги будуть мати певні похибки, за рахунок округлення даних;

- у випадку коли елемент явно не підтримується форматом, то його потрібно перетворити у найближчий по формі доступний елемент. Така проблема часто зустрічається, коли системи підтримують різні версії IGES. Наприклад, коли одна із систем із більш старою версію IGES не підтримує макроси і т.д.

DXF був вперше представлений у грудні 1982 року як частина AutoCAD 1.0. Цей обмінний файл надає ту ж саму інформацію, що і закритий внутрішній формат AutoCAD – DWG, специфікація на який ніколи не надавалася.

DXF (Drawing eXchange Format) – відкритий формат даних для обміну графічною інформацією між різними САПР. Спочатку цей формат розроблювався для того, щоб надати користувачам гнучкість в управлінні даними та перетворенні креслень програми AutoCAD у формати файлів, які могли б читатися та використовуватися іншими САПР. За рахунок популярності AutoCAD цей формат став практично стандартом обміну файлами CAD-креслень майже для всіх САПР. На даний момент, цей формат підтримується практично усіма CAD-системами на платформі PC.

DXF-файл – це текстовий ASCII-файл, що складається із 5 розділів:

1) Header (Заголовок) – описує середовище у якому був створений DXF-файл;
2) Table (Таблиця) – зберігає інформацію про типи ліній, шари, стилі текстів та види, які можуть бути задані у кресленні;
3) Block (Блок) – зберігає список графічних елементів, заданих як група;

4) Entity (Елемент) – основний розділ, в якому описуються усі елементи, присутні на кресленні;
5) Terminate (Кінець).

З появою нових версій AutoCAD список можливих елементів розширювався, але як і з IGES, файл створений більш пізньою версією AutoCAD не міг бути прочитаним іншими системами, які використовували більш старі версії формату DXF.

STEP (англ. STandard for Exchange of Product model data – стандарт обміну даними моделі виробу) – сукупність стандартів ISO 10303, що використовується в САПР. Формати IGES та DXF були розроблені для обміну даними технічних вимог, а не даними про виріб. Дані про виріб – це всі дані, які відносяться до всього життєвого циклу виробу (наприклад, проектування, контроль якості, виробництво і т.д.). Специфікації IGES та DXF були розширені з метою включення деяких даних про виріб, але цих даних не вистачає для опису всього життєвого циклу об’єкта. Це все викликало розробку нового стандарту під назвою PDES (Product Data Exchange Specification – специфікація для обміну даними про вироби), який почали розроблювати в США в 1983 році. Головною задачею PDES було виключення людської присутності при обміні даними про вироби. Іншими словами, ціллю PDES була ліквідація потреби в інженерних кресленнях та інших паперових документах при обміні інформацією про різні фази життєвого циклу виробу. Паралельно з цим у липні 1984 року у Міжнародній організації по стандартизації (ISO) був створений технічний комітет ТС 184 (Системи виробничої автоматизації) та його підкомітет SC4 (Зовнішнє подання даних про моделі виробу) для встановлення єдиного стандарту обміну даними про моделі виробу. Даний стандарт отримав назву STEP. Цілі PDES та STEP булу ідентичними, тому в червні 1985 року управляючий комітет IGES вирішив, що інтереси США у програмі STEP повинен представляти стандарт PDES. Результатом цього стала зміна значення акроніму PDES на «обмін даними про

виріб з використанням STEP» (Product Data Exchange using STEP). Це було зроблено з метою підкреслення ідентичності цілей PDES та STEP.

В основі розробки STEP лежать наступні принципи:

- Стандарт STEP повинен орієнтуватись на дані про виріб, які включають в себе інформацію про весь життєвий цикл виробу. Таким чином, у якості даних повинна розглядатися інформація про допуски, технологічні особливості форми, модель для кінематичного аналізу і т.д. Тобто, акцент робиться на формі виробу.
- Структури даних STEP повинні зберігати інформацію, що відноситься до застосунку у модулі рівня застосунку, окремо від загальної інформації про форму виробу. Такий підхід забезпечує те, що структура даних може підтримувати широкий спектр застосунків, уникаючи при цьому надлишковості в загальній структурі даних. Для визначення структури даних повинна використовуватися формальна мова. Специфікації IGES та DXF описують формат фізичного файлу, який зберігає в собі всі геометричні та інші дані. У STEP дані описуються на мові EXPRESS, а потім результат перетворюється у фізичний файл. Це дозволяє уникнути неоднозначності при інтерпретації даних про виріб.

STEP розробляється великою кількістю комітетів та робочих груп, кожна з яких займається різними частинами стандарту. Частини стандарту групуються по методам опису, інтегрованим інформаційним ресурсам, прикладним протоколам та методам реалізації. Статус кожної частини показаний поруч із її номером. Статут позначається буквами від «O» (попередня стадія ISO) до «I» (найвища стадія прийняття стандартів). Частини, які позначені буквами «E», «F» та «I» знаходяться на такому високому рівні, що дозволяють виробникам програмних застосунків приступити до їх реалізації. Група методів опису складає фундамент стандарту STEP. На наступному рівні знаходяться інтегровані інформаційні ресурси. Інтегровані інформаційні ресурси – це елементи, які за необхідністю використовуються прикладними

протоколами. Інтегровані інформаційні ресурси включають в себе загальні ресурси, прикладні ресурси та конструкції, інтерпретовані застосунком. На верхньому рівні ієрархії STEP знаходяться більш складні моделі даних, які використовуються для опису конкретних даних про виріб, такі частини називаються прикладними протоколами. Вони описують не тільки які дані повинні використовуватись при описі продукту, але і те, як ці дані повинні використовуватись у моделі. Також стандарт містить у собі групи методів реалізації, методології перевірки відповідності. У першій описується відповідність між формальними специфікаціями STEP та їх представленням, яке використовується для реалізації стандарту. У другій зберігається інформація про методи перевірки відповідностей програмних продуктів стандарту STEP.

На сьогоднішній день, стандарт STEP приваблює до себе багато уваги, оскільки очікується що цей формат увійде у систему стандартів технологій CALS (Computer-aided Acquisition and Logistics Support – Неперервні поставки і інформаційна підтримка життєвого циклу виробів) як стандарт обміну даними про вироби.

IFC (Industry Foundation Classes) – нейтральний формат даних з відкритою специфікацією, яка не контролюється ні однією компанією або групою компаній. Створення IFC почалося у 1994 році, коли компанія Autodesk створила галузевий консорціум для консультації компаній по розробці набору класів C++, які могли б підтримувати інтеграцію застосувань. До даного консорціуму приєдналися 12 американських компаній, серед них: At&T, HOK, Architects, Honeywell, Carrier, Tishman та Butler Manufacturing. Спочатку даний консорціум був названим індустріальним союзом для взаємодії, в 1997 році він змінив назву на Міжнародний альянс за інтероперабельність. Новий альянс був відтворений як некомерційна галузева організація з метою публікації классів IFC у якості нейтральної моделі, яка відповідає життєвому циклу будівельного об’єкта. В подальшому назва була змінена на BuildingSMART.

Специфікації IFC/ifcXML:

- IFC4 Add2 (2016)
- IFC4 Add1 (2015)
- IFC4 (березень 2013 р.)
- ifcXML2x3 (червень 2007 р.)
- IFC2x3 (лютий 2006 р.)
- ifcXML2 для IFC2x2 add1 (RC2)
- IFC2x2 Add1 (липень 2004 р.)
- ifcXML2 для IFC2x2 (RC1)
- IFC 2x2
- IFC 2x Add1
- ifcXML1 для IFC2x и IFC2x Addendum 1
- IFC 2x
- IFC 2.0
- IFC 1.5.1
- IFC 1.5

IFC стандарт визначає декілька форматів файлів, які можуть використовуватися:

- IFC-SPF - це текстовий формат, який визначається стандартом ISO 10303-21 («STEP-файл»), де кожен рядок зазвичай складається з одного запису об’єкта і має розширення файлу .ifc. Це найбільш поширений формат IFC, до його переваг можна віднести компактність та відносну читабельність.
- IFC-XML - це формат XML, визначений ISO 10303-28 («STEP-XML») з розширенням файлу .ifcXML. Цей формат підходить для взаємодії з інструментами XML і обміну частковими будівельними моделями. Через великий розмір типових моделей будівель цей формат на практиці менш поширений.

- IFC-ZIP - це стислий формат ZIP, що складається з вбудованого файлу IFC-SPF і файлу розширення «.ifcZIP».

IFC базується на моделі «сутності-відносини» на основі стандартної мови EXPRESS, що складається з декількох сотень об'єктів, організованих в ієрархію спадкування об'єктів на основі об'єктів. Приклади об'єктів включають в себе будівельні елементи, такі як IfcWall, геометрію, таку як IfcExtrudedAreaSolid, і базові конструкції, такі як IfcCartesianPoint.

Основною перевагою використання IFC в порівнянні з іншими файловими форматами є збереження при передачі даних всієї BIM-інформації. Це означає, що, наприклад, стіни і колони, створені в ARCHICAD, при відкритті IFC-файлу в іншому додатку, залишаться стінами і колонами, що містять всю пов'язану з ними інформацію, і не будуть перетворені в прості геометричні елементи. IFC формат підтримує не тільки всі геометричні 3D-характеристики, але і передає дані про розташування об'єктів, їх конструктивні функції і взаємозв'язки, а також - всі параметри кожного об'єкта. Процес взаємодії на основі IFC відповідає всім вимогам концепції опорної моделі, яка застосовується при спільному міждисциплінарному використанні 3D-моделей, і дає проектувальникам певні переваги:

• Доступ до даних об'єктів, що підвищує якість проектування. Наприклад, можна імпортувати систему трубопроводів, створену інженером розділу водопостачання і каналізації, у вигляді 3D-об'єктів.

• Можливість експорту моделей в додатки, які використовуються фахівцями суміжних розділів, наприклад, теплотехнічного аналізу, дозволяє краще зрозуміти якість проектних рішень з точки зору експлуатаційних характеристик будівлі.

• Можливість експорту моделей на предмет оцінки вартості будівництва і експлуатації. Передача інформації за допомогою IFC істотно підвищує

повноту, а значить, і цінність бази даних, використовуваної при подібних розрахунках.

У сучасних умовах часто використовують мову розмітки XML. Ця мова розмітки є добре читабельною для людини та є простою для використання на програмній стороні, адже на сьогоднішній день є дуже багато різних парсерів XML для різних мов (C++, C#, Java, Python і т.д.).

XML (англ. eXtensible Markup Language - розширювана мова розмітки - рекомендована Консорціумом Всесвітньої павутини мова розмітки, що фактично є зведенням загальних синтаксичних правил. XML - текстовий формат, призначений для зберігання структурованих даних (замість існуючих файлів баз даних), для обміну інформацією між програмами, а також для створення на його основі більш спеціалізованих мов розмітки (наприклад, XHTML), іноді званих словниками. XML є спрощеною підмножиною мови SGML.

Метою створення XML було забезпечення сумісності при передачі структурованих даних між різними системами обробки інформації, особливо при передачі таких даних через Інтернет. Словники, засновані на XML (наприклад, RDF, RSS, MathML, XHTML, SVG), самі по собі формально описані, що дозволяє програмно змінювати і перевіряти документи на основі цих словників, не знаючи їх семантики, тобто не знаючи смислового значення елементів. Важливою особливістю XML також є застосування так званих просторів імен.

Переваги XML:

- XML - це формат, одночасно зрозумілий і людині і комп’ютеру;
- XML підтримує Юнікод;
- у форматі XML можуть бути описані основні структури даних - такі як списки і дерева;

- XML - це самодокументований формат, який описує структуру і імена полів також як і значення полів;
- XML має строго певний синтаксис і вимоги до аналізу, що дозволяє йому залишатися простим, ефективним і несуперечливим;
- XML також широке використовується для зберігання і обробки документів;
- XML - формат, заснований на міжнародних стандартах;
- ієрархічна структура XML підходить для опису практично будь-яких типів документів;
- XML є простим текстом, вільним від ліцензування і яких-небудь обмежень;
- XML не залежить від платформи;
- XML є підмножиною SGML (який використовується з 1986 року). Вже накопичений великий досвід роботи з мовою і створені спеціалізовані додатки;
- XML не накладає вимог на розташування символів на рядку.

Недоліки XML:

- XML не містить вбудованої в мову підтримки типів даних. У ньому немає понять "цілих чисел", "рядків", "дат", "булевих значень" і т.д.;
- Ієрархічна модель даних, пропонована XML, обмежена в порівнянні з реляційною моделлю і об’єктно-орієнтованими графами;

При написанні конвертерів, будь то прямі конвертори чи постпроцесори та препроцесори, для доступу до баз даних систем використовуються спеціальні технології, серед них можна виділити наступні:

- COM (Component Object Model – Модель компонентного об’єкта);
- API (Application Program Interface – Прикладний програмний інтерфейс).

COM - це технологічний стандарт від компанії Microsoft, призначений для створення програмного забезпечення на основі взаємодіючих компонентів

об’єкта, кожен з яких може використовуватися в багатьох програмах одночасно. Стандарт втілює в собі ідеї поліморфізму і інкапсуляції об’єктно-орієнтованого програмування. Стандарт COM міг би бути універсальним і від платформи незалежним, але закріпився в основному на операційних системах сімейства Microsoft Windows. Основним поняттям, яким оперує стандарт COM, є COM-компонент. Програми, побудовані на стандарті COM, фактично не є автономними програмами, а являють собою набір взаємодіючих між собою COM-компонентів. Кожен компонент має унікальний ідентифікатор (GUID) і може одночасно використовуватися багатьма програмами. Компонент взаємодіє з іншими програмами через COM-інтерфейси - набори абстрактних функцій і властивостей.

Недоліки COM:

- необхідність використання двох мов програмування (.idl для опису інтерфейсів і, звичайно, C ++ для написання реалізацій). Необхідність виникає тільки при створенні власних інтерфейсів, і не виникає в разі, якщо розробник обмежив себе використанням готових інтерфейсів;
- необхідність «прокладки» коду (в його ролі зазвичай виступає ATL) для того, щоб створити COM-об’єкт на базі C ++ класу. Хоча цей код і тривіальний в використанні для досвідченої людини, він не дуже простий для початківців;
- необхідність реєстрації компонентів в реєстрі операційної системи, причому при цьому в якості ідентифікатора класу використовується інтуїтивно незрозумілий для людини GUID);
- інфраструктура remoting (віддаленого виклику методів) використовує бінарний формат запитів і відповідей, будучи розширенням DCE RPC. Це призводить до виникнення величезної уразливості з точки зору безпеки;

- інфраструктура remoting використовує за замовчуванням (слідом за DCE RPC) динамічно призначаються номери TCP- і UDP-портів, що робить її вкрай складної в налаштуванні при наявності міжмережевих екранів.
- обробка помилок. В COM прийнято використовувати 32-бітові коди помилки HRESULT, які мають значення на кшталт 0x80070123, і абсолютно не читані людиною.
- Простори імен XML складно використовувати і їх складно реалізовувати в XML парсерах.

Прикладний програмний інтерфейс (англ. Application Programming Interface, API) — набір визначень взаємодії різнотипного програмного забезпечення. API — це зазвичай (але не обов'язково) метод абстракції між низькорівневим та високорівневим програмним забезпеченням. API визначає функціональність, яку надає програма (модуль, бібліотека), при цьому API дозволяє абстрагуватися від того, як саме ця функціональність реалізована.

API широко використовується виробниками операційних систем. Практично всі операційні системи (UNIX, Windows, OS X і т. Д.) Мають API, за допомогою якого програмісти можуть створювати додатки для цієї операційної системи. В індустрії програмного забезпечення загальні стандартні API для стандартної функціональності мають важливу роль, так як вони гарантують, що всі програми, які використовують загальний API, будуть працювати однаково добре або типовим звичним чином. У разі API графічних інтерфейсів це означає, що програми будуть мати схожий користувальницький інтерфейс, що полегшує процес освоєння нових програмних продуктів.

З іншого боку, відмінності в API різних операційних систем істотно ускладнюють перенесення додатків між платформами.

На сьогоднішній день більшість виробників САПР мають свій власний API, це дозволяє прискорити розробку програмного забезпечення, дозволяє надати розробку сторонній компанії, адже API не дає відкритого доступу до вихідного

коду системи. Більшість виробників намагається викладати свій API у відкритий доступ, доповнюючи його детальною документацією. Це дозволяє значно збільшити кількість модулів та інших додатків, які будуть написані для САПР.

1.5. Розширена модель будівельного об’єкта на всіх етапах життєвого циклу

Модель будівельного об’єкта на кожному із етапів життєвого циклу, безпосередньо залежить від конкретного етапу життєвого циклу. Залежно від того які дані про будівлю потрібні на тому чи іншому етапі для його автоматизації, створюється модель подання будівельно об’єкта. Визначимо типи моделей будівельних об’єктів:

1. 3D модель – віртуальна тривимірна модель будівельного об’єкта. Ця модель є досить інформативною, вона дозволяє отримати інформацію як про всю будівлю, так і про окремі її елементи; 3D модель може бути розділена на два підтипи:
    a) 3D модель (архітектурна) – ця модель є дещо спрощеною відносно повної тривимірної моделі, адже в ній можуть бути відсутні дрібні конструктивні елементи, які не впливають на загальний вигляд будівлі;
    b) 3D модель (розрахункова) – ця модель містить лише несучі конструкції, всі інші елементи замінюються навантаженнями на них.
2. 2D модель – цей тип моделі використовується, для відображення креслень, планів, розрізів та інших 2D елементів.
3. Топологічна модель – модель подання будівельного об’єкта у вигляді ієрархічного дерева або графу.
4. Нормативно-кошторисна об’ємна модель – специфікація елементів з їх прямим та супутніми об’ємними показниками. Сюди також входять усі кошторисні норми.

5. Нормативно-кошторисна об’ємна модель з часовими характеристиками – ця модель дозволяє отримати інформацію про ступінь завершеності елемента будівельного об’єкта. Дану модель не можна виділити, як самостійну, адже для цього потрібно щоб кількісні характеристики були прив’язані до часових.
6. 4D модель – в цій моделі кожному елементу окрім геометричних характеристик надається часова характеристика, яка дозволяє в кожен момент часу отримати інформацію про ступінь завершеності цього елемента. Також часова характеристика впливає на розрахунок об’ємних характеристик будівельного об’єкта та його візуалізацію. В цьому випадку кількісні та часові характеристики пов’язані.
7. 5D модель – ця модель поєднує в собі 4D модель та кошторисно-нормативну модель, тобто вона зберігає в собі геометричні, часові та кошторисно-фінансові характеристики кожного елемента будівлі. Така модель забезпечує максимальну інтеграцію різних напрямів проектування.

Зобразимо взаємозв’язок між моделями у вигляді схеми (рис. 1.7)

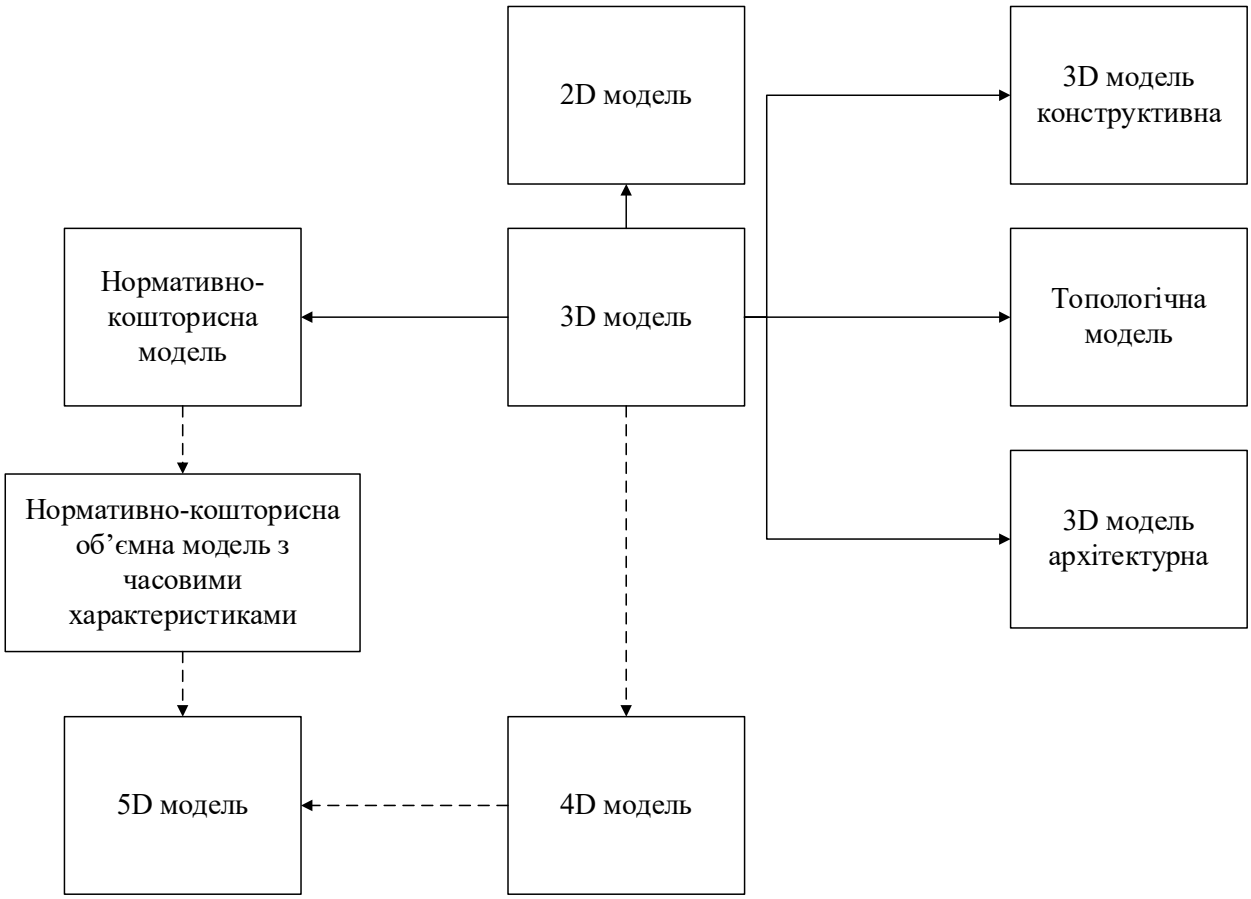


Рис. 1.7. Взаємозв'язок між моделями будівельного об'єкта

Як бачимо з рис. 1.7, існують проблеми переходу між деякими моделями, а саме:

- перехід від топологічною моделі до конструктивної;
- перехід від конструктивної моделі до топологічної;
- перехід від архітектурної моделі до топологічної;
- перехід від топологічної моделі до архітектурної;
- перехід від архітектурної моделі до конструктивної;
- перехід від конструктивної моделі до архітектурної.

В цій роботі акцентовано увагу на проблемі переходу від архітектурної моделі до конструктивної за рахунок створення проміжної пластинчато-стержневої моделі (ПСМ) (рис. 1.8).

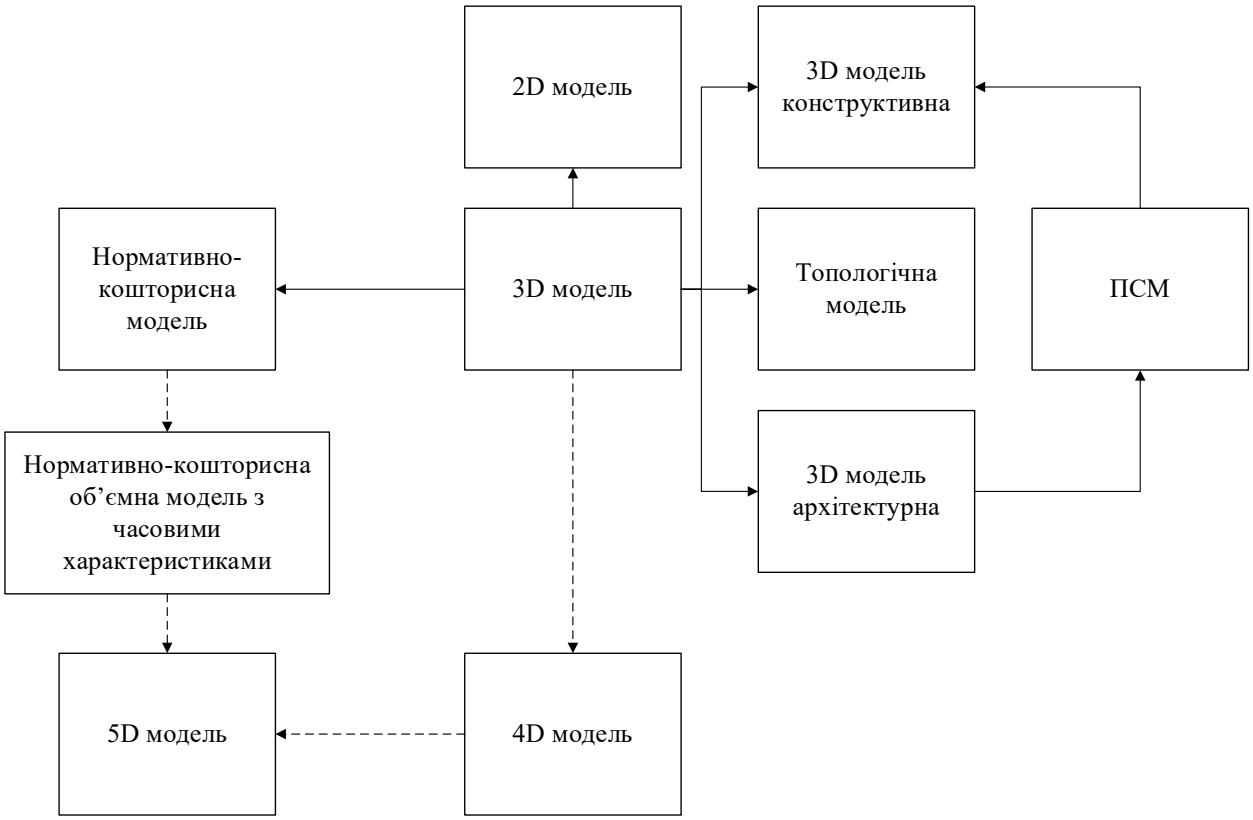


Рис. 1.8. Взаємозв'язок між моделями будівельного об'єкта з ПСМ

Використання ПСМ дозволить практично повністю автоматизувати перехід від архітектурної моделі до конструктивної. Конструктору більше не доведеться відтворювати конструктивні елементи моделі власноруч у спеціалізованому застосунку, натомість, він отримає вже базову конструктивну модель, зможе проаналізувати її та доповнити необхідними конструктивними даними.

ВИСНОВКИ ДО РОЗДІЛУ 1

1. Досліджене поняття будівельного об’єкта та наведені його класифікації в залежності від призначення. Досліджені основні конструктивні елементи будівель. Проведено аналіз життєвого циклу БО на всіх етапах проектування.
2. Досліджено основні сучасні САх-системи для моделювання БО, проведено аналіз та порівняння їх можливостей. Наведена їх загальна класифікація.
3. Проведено аналіз існуючих труднощів в інтеграції різних стадій життєвого циклу будівельного об’єкта. Це дозволило виокремети ключові проблеми інтеграції на кожному етапі проектування.
4. Виконано дослідження сучасних методів і засобів інтеграції САх-систем в БО. Досліджені основі нейтральні формати файлів, що використовуються для обміну між системами проектування, проведений аналіз кожного окремого формату, наведені їх недоліки та переваги.
5. Проведений аналіз розширеної моделі будівельного об’єкта на всіх етапах життєвого циклу. На базі поставлених задач, проведено удосконалення розширеної моделі.

РОЗДІЛ 2

ІНФОРМАЦІЙНА БІБЛІОТЕКА УНІФІКАЦІЇ БУДІВЕЛЬНИХ КОНСТРУКЦІЙ

У розділі будуть розглянуті моделі і методи, які необхідні для розробки бібліотеки уніфікації будівельних конструкцій. Бібліотека уніфікації необхідна для формалізації таких понять як: несуча стіна, зовнішня стіна, назви матеріалів і т.н. Адже використання цих понять доволі часто інтерпретується різними системами по різному. Наприклад, для одного модуля несуча стіна – це та стіна у якої присутній атрибут «несуча», а для іншого – це стіна у якої товщина більше 300 мм. Такий підхід абсолютно не є гнучким та інтуїтивно-зрозумілим. Використання бібліотеки уніфікації дозволить не лише уніфікувати певний набір понять, а й надасть можливість гнучко змінювати ці поняття.

Першочергове призначення бібліотеки, виступати проміжним шаром між вхідним набором даних моделі та інтерфейсом того чи іншого модуля (рис. 2.1).

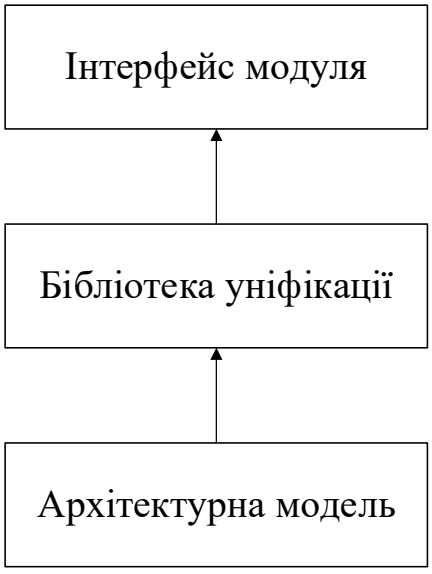


Рис. 2.1. Бібліотека уніфікації, як проміжний шар між модулем та архітектурною моделлю

Для більшості модулів використання бібліотеки уніфікації є необхідністю. Наприклад, для модуля генерації ПСМ, вхідні дані повинні бути чітко формалізовані та розбиті на групи (несучі стіни, перегородки і т.н.). Також використання бібліотеки позитивно впливає на швидкодію модулів, адже на вхід модуля надходять вже чітко формалізовані дані, що дозволяє відкинути необхідність повторної формалізації даних безпосередньо у самому модулі.

2.1. Інформаційна модель об’єкта у Allplan

У зв’язку з тим, що вирішувати поставлену проблему будемо на прикладі системи Allplan, приведемо інформаційну модель об’єкта цієї системи у вигляді схеми (рис. 2.2)

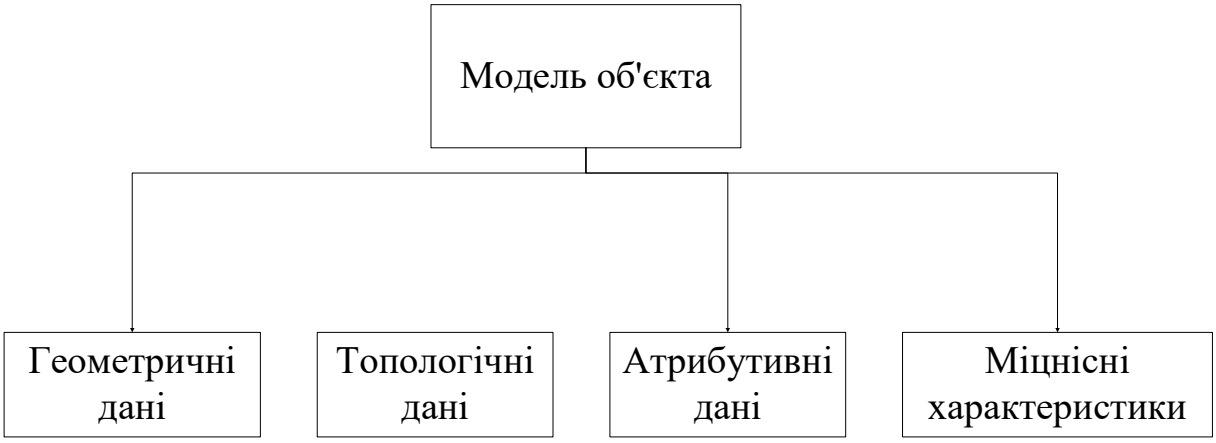


Рис. 2.2. Інформаційна модель об’єкта Allplan

Як бачимо, на стороні бібліотеки уніфікації ми зможемо оперувати геометричними, топологічними, атрибутивними даними та міцнісними характеристиками, чого достатньо для вирішення всіх поставлених задач.

2.2. Загальна концепція інформаційної бібліотеки уніфікації

Опишемо загальну концепцію інформаційної бібліотеки уніфікації, розглянемо всі необхідні функції, їх параметри та варіанти роботи (табл. 2.1).

Таблиця 2.1

Функції інформаційної бібліотеки уніфікації

Функція інформаційної бібліотеки уніфікації	**Опис**
IsBearingWall	Визначає несуча стіна чи ні. Ця функція отримує на вхід архітектурну стіну та обраний параметр, за яким буде визначатись, несуча стіна чи ні. В залежності від результату роботи функція повинна повертати «**true**» або «**false**». Несуча стіна може бути визначена за наступними параметрами: 1. атрибут «Несуча»; 2. кількість шарів; 3. товщина; 4. матеріал; 5. матеріал + товщина.
IsOuterWall	Визначає зовнішня стіна чи ні. Отримує на вхід архітектурну стіну та обраний параметр. Повертає – «**true**» або «**false**». Зовнішня стіна може бути визначена за наступними параметрами: 1. кількість шарів; 2. товщина; 3. матеріал; 4. матеріал + товщина.

CorrectMaterialName	Повертає рядок із уніфікованою назвою матеріалу. Отримує на вхід рядок із назвою матеріалу. Параметри відсутні.
ShowSettingsForm	Викликає діалогове вікно налаштувань для функцій бібліотеки уніфікації.
GetRoomType	Визначає тип приміщення. Отримує на вхід об’єкт – приміщення. Повертає кодову назву приміщення, наприклад," «ЖП» – житлове приміщення. Тип приміщення може бути визначений за допомогою наступних параметрів: 1. атрибут «Функція»; 2. атрибут «Тип площі».
GetFloorCount	Повертає кількість поверхів для поточного проекту. Кількість поверхів може бути визначена за наступними параметрами: 1. атрибут «Кількість поверхів»; 2. зчитана кількість поверхів із файлу зі структурою проекту.
GetObjectFloorNumber	Визначає номер поверху, на якому розташований об’єкт. Отримує на вхід архітектурний об’єкт. Номер поверху може бути визначений за наступними параметрами: 1. за допомогою висотних відміток проекту; 2. за допомогою файлу зі структурою проекту.

Враховуючи одну із вимог до бібліотеки, а саме «можливість оновлення бібліотеки, незалежно від модуля чи плагіна» можемо зробити висновок, що бібліотеку потрібно реалізувати у вигляді DLL.

DLL – це бібліотека, яка містить код і дані, що можуть використовуватися кількома програмами одночасно. Таким чином, кожна програма може використовувати функції, що містяться в цій бібліотеці DLL. Це забезпечує повторне застосування коду й ефективне використання пам'яті. Завдяки використанню DLL програму можна розділити на кілька компонентів. Наприклад, бухгалтерська програма може продаватися окремими модулями. Кожен модуль може завантажуватися в основну програму під час виконання, якщо він інстальований. Оскільки модулі відокремлені один від одного, програма завантажується швидше, і модуль завантажується, лише якщо запитується його функціонал. Крім того, стає легше застосовувати оновлення до кожного модуля, не впливаючи на інші компоненти програми.

Функціям уніфікації для роботи з об'єктами Allplan необхідний доступ до NOI API. Nemetschek Object Interface API – набір бібліотек з готовими функціями та описаними класами які дають змогу керувати та працювати з об'єктами в Allplan. З точки зору NOI API Allplan виступає в ролі потужної графічної бази даних. Внаслідок того, що NOI API написаний на C++, бібліотека уніфікації також буде розроблена на цій мові. Виклик функцій із DLL в C++ є доволі громіздким та незручним, для зручності користування бібліотекою пропонується розробити обгортку «**LibraryWrapper**». Дана обгортка буде реалізована у вигляді статичної бібліотеки LIB.

Статичні бібліотеки LIB – можуть бути у вигляді початкового тексту, що підключається програмістом до своєї програми на етапі написання (наприклад, для мови Fortran існує величезна кількість бібліотек для вирішення різних завдань саме в початкових текстах), або у вигляді об'єктних файлів, що приєднуються (лінкуються) до виконуваної програми на етапі компіляції (у Microsoft Windows такі файли мають розширення .lib, у UNIX-подібніх ОС —

зазвичай .a). В результаті програма включає всі необхідні функції, що робить її автономною, але збільшує розмір.

Для зберігання конфігурації функцій уніфікації використана мова розширюваної розмітки XML (eXtensible Markup Language). Під час розробки структури файлу конфігурації, потрібно врахувати і те, що цей файл повинен бути максимально простим для зчитування. Враховуючи те, що файл конфігурації буде зчитуватись як кожною функцією уніфікації, так і формою налаштувань «**LibraryUI**», а форма налаштувань зможе його ще й змінювати, пропонується винести можливість зчитування та запису файлу конфігурації в окрему DLL «**LibraryParser**». Для більшої гнучкості, форма налаштувань буде динамічною, тобто всі елементи форми будуть створюватись в залежності від змісту файлу конфігурації.

Зобразимо роботу із бібліотекою уніфікації у вигляди схеми (рис. 2.3)

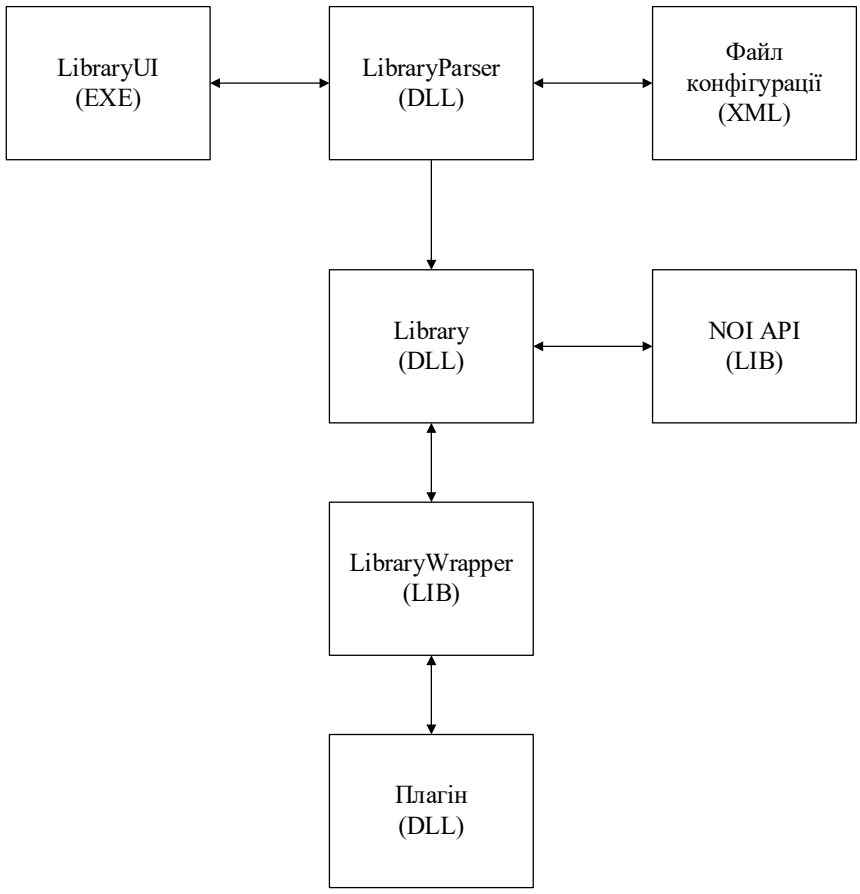


Рис. 2.3. Схема роботи бібліотеки уніфікації

2.3. Структура файлу конфігурації

Для збереження конфігураційних даних у XML файлі пропонується використовувати наступні секції.

uLibrary	Коренева секція
uLocale	Секція, яка відповідає налаштуванням для конкретної локалізації. Є дочірньою до **uLibrary.**
uElement	Секція яка зберігає налаштування для конкретної функції. Є дочірньою до **uLocale.**
uVariant	Секція, яка є дочірньою до **uElement**, використовується для зберігання переліку властивостей для конкретного варіанту використання функції.
uProperty	Секція, яка може бути дочірньою як до **uVariant**, так і до **uElement**. В ній зберігається перелік відповідностей, які використовуються безпосередньо в функції уніфікації.
uPropertyItem	Секція, яка є яка є дочірньою до **uProperty**, зберігає дані відповідності.

Зобразимо структуру файлу конфігурації у вигляді схеми (рис. 2.4)

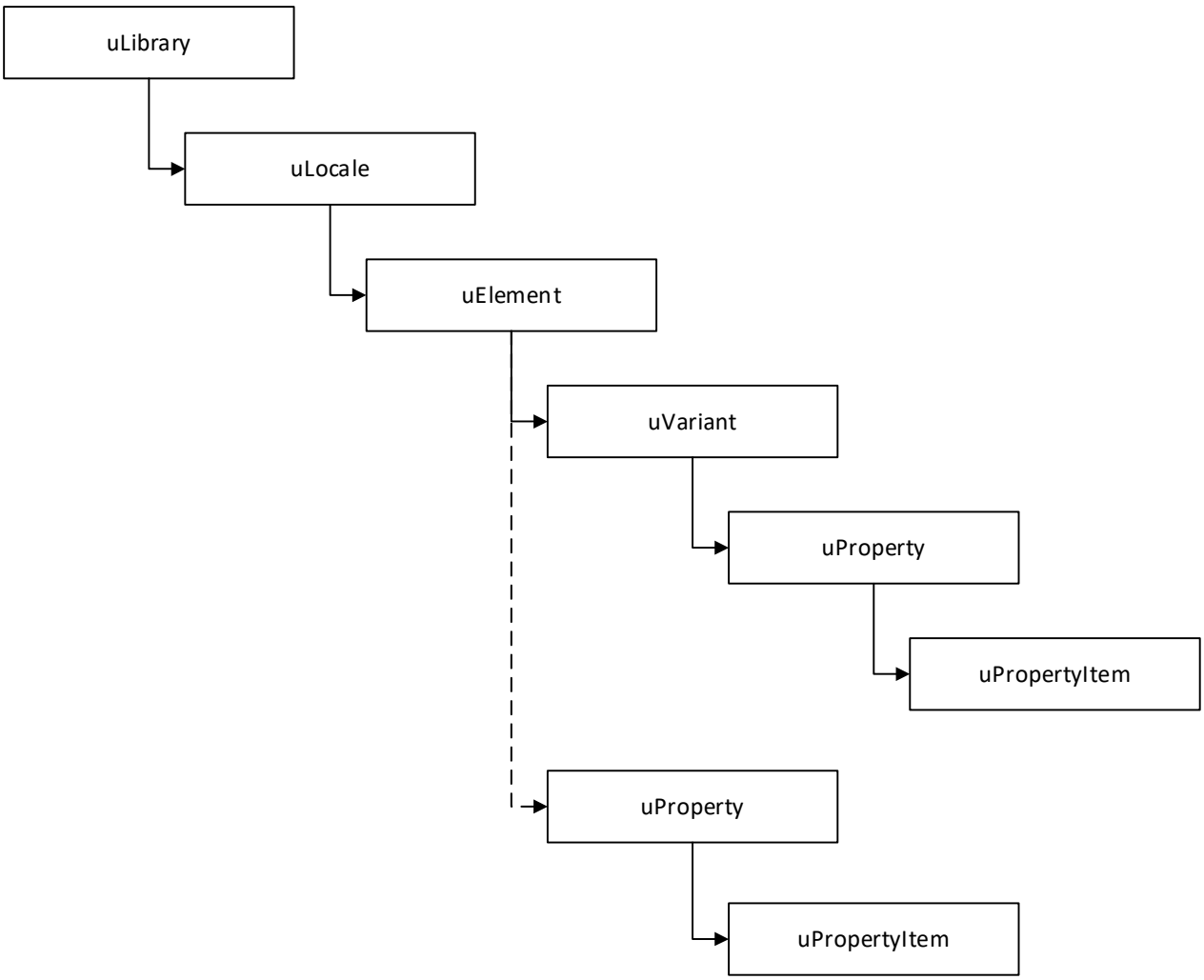


Рис. 2.4. Структура файлу конфігурації

Налаштування для функції **IsBearingWall** виглядають наступним чином:

```
<uElement code="bearingwall" control="ComboBoxTable_Linked">
      <uVariant code="material+thickness" selected="false">
        <uProperty>
          <uPropertyItem code="material" type="string">бетон</uPropertyItem>
          <uPropertyItem code="thickness" type="range">300|400</uPropertyItem>
        </uProperty>
      </uVariant>
      <uVariant code="material" selected="true">
        <uProperty>
```

```
        <uPropertyItem code="material" type="string">бетон</uPropertyItem>
      </uProperty>
    </uVariant>
    <uVariant code="thickness" selected="false">
      <uProperty>
        <uPropertyItem code="thickness" type="range">300|400</uPropertyItem>
      </uProperty>
    </uVariant>
    <uVariant code="layercount" selected="false">
      <uProperty>
        <uPropertyItem code="layercount" type="range">3|4</uPropertyItem>
      </uProperty>
    </uVariant>
  </uElement>
```

В наведеному прикладі, ми заповнили дані для кожного з варіантів виклику функції. Наприклад, якщо ми хочемо дізнатись чи несуча наша стіна за кількістю шарів, то ми викличемо функцію **IsBearingWall**, яка звернеться до секції `<uVariant code="layercount">` і побачить, що стіна вважається несучої в тому випадку, коли кількість її шарів складає від 3 до 4.

Для того, щоб знати як відображати налаштування на формі, введені наступні атрибути:

control	Цей атрибут належить секції **uElement**. За допомогою цього атрибута, на стороні форми зрозуміло які елементи потрібно використати аби забезпечити відображення та можливість редагування налаштувань. Можливі варіанти: 1. **ComboBox** – випадний список 2. **Table** – таблиця 3. **ComboBoxTable_Linked** – випадний список із зв’язаною таблицею. Тобто, при зміні у випадному списку, змінюються дані у таблиці.

	4. **ComboBoxTable_UnLinked** – випадний список із таблицею. В даному випадку, зміни у випадному списку ніяк не впливають на дані у таблиці.
type	Цей атрибут належить секції **uPropertyItem**. Він дозволяє обмежити тип даних, які може ввести користувач при редагуванні того чи іншого поля в файлі конфігурації. Можливі варіанти: 1. **string** – рядок 2. **range** – діапазон дійсних чисел
selected	Цей атрибут належить секції **uVariant**. Він потрібен у тому випадку, коли функція уніфікації містить декілька варіантів виклику, наприклад як **IsBearingWall**. Атрибут зберігає дані, про те, вибраний даний варіант користувачем на формі чи ні.

Для того, аби користувачу в подальшому при налаштуванні було зрозуміло, яку саме функцію він налаштовує, до файлу конфігурації прикладається файл **.lng** – файл локалізації. В якому кожному значенню атрибута **code** файлу конфігурації відповідає інтуїтивно зрозуміле значення на обраній мові.

```
<Languages>
    <Language name="en">
        <Localized keyword="bearingwall"
            message="Criterion for selection of bearing walls" />
    </Language>
    <Language name="ru">
        <Localized keyword="bearingwall"
            message="Критерий отбора несущих стен" />
```

```
    </Language>
</Languages>
```

2.4. Діалогове вікно налаштувань

Як було сказано вище, діалогове вікно буде генеруватись динамічно, в залежності від змісту файлу конфігурації. Наприклад, для функції **IsBearingWall** будуть згенеровані наступні елементи діалогового вікна (рис. 2.5).

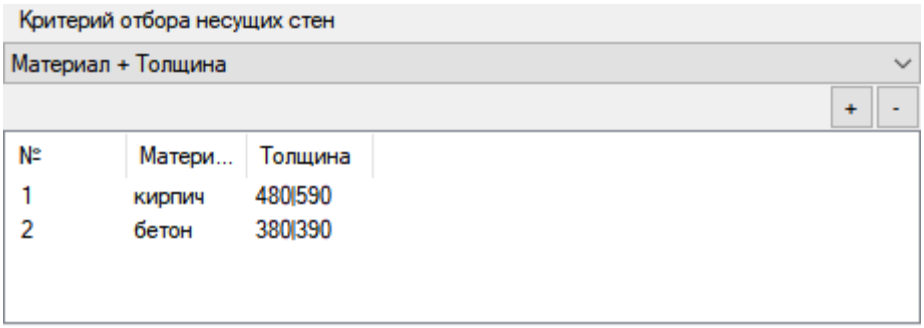


Рис. 2.5. Елементи діалогового вікна для функції **IsBearingWall**

Діалогове вікно налаштувань може бути викликане як з плагіна, так і як зовнішній застосунок. Налаштування пропонується зберігати у двох каталогах:

1) **local** – даний каталог є унікальним для кожного плагіна, тому користувачу дозволено змінювати конфігураційний файл за допомогою діалогового вікна у даному каталозі;
2) **extern** – даний каталог є спільним для всіх плагінів, тому редагування конфігураційного файлу в даному каталозі можливе лише при відкритті діалогового вікна налаштувань у вигляді зовнішнього застосунку, конкретним користувачем.

У випадку коли діалогове вікно викликається з плагіна, користувач може обрати з випадного списку потрібний конфігураційний файл (**local**, **extern**), локалізація у даному випадку задається автоматично, в залежності від мови, встановленої в AllPlan. У випадку коли вікно відкривається як зовнішній

застосунок, користувач повинен обрати потрібний конфігураційний файл самостійно, також у нього буде змога переключатись між локалізаціями.

Зобразимо роботу форми у вигляді схеми (рис. 2.6)

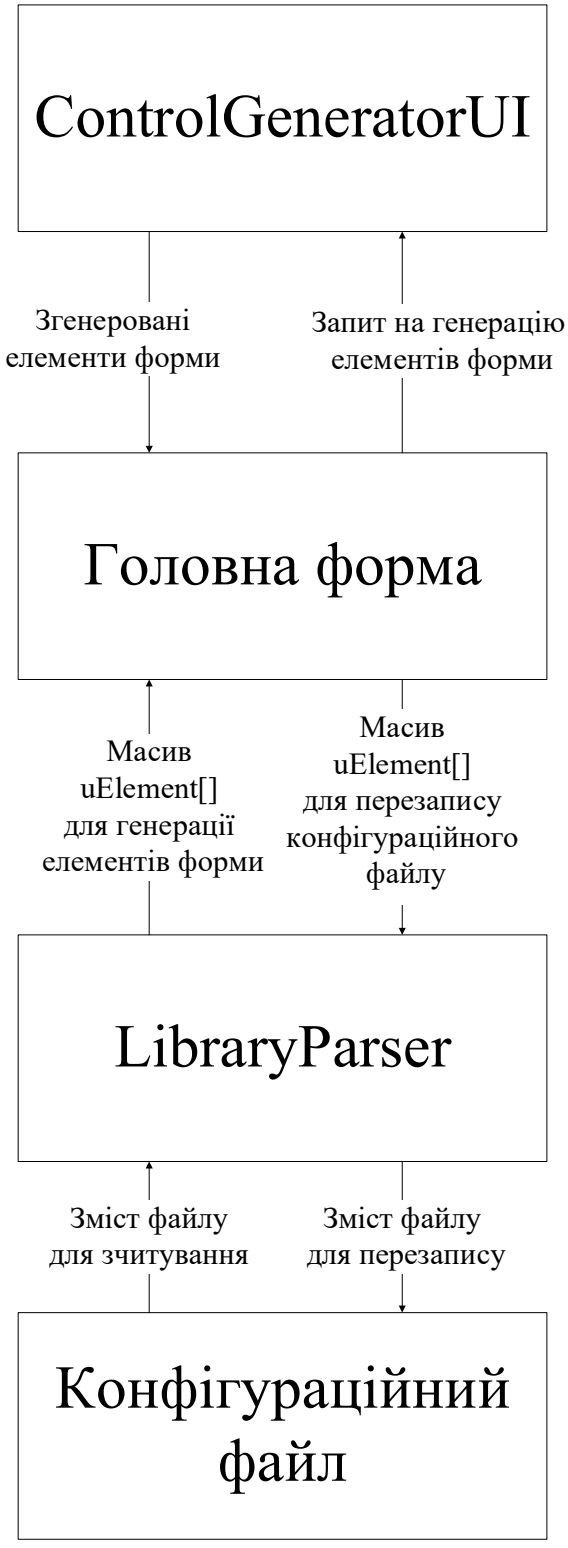


Рис. 2.6. Схема роботи форми налаштувань бібліотеки уніфікації

2.5. Приклад виклику функції уніфікації

За приклад візьмемо функцію **IsBearingWall**, яка допоможе визначити несуча стіна чи ні. Для цього нам потрібно звернутись до **LibraryWrapper** та викликати з нього необхідну функцію, передати в неї стіну та обраний варіант перевірки на несучість. Для зручності використання, всі варіанти були обгорнуті в спеціальні переліки **Enum**. Перелік для функції **IsBearingWall** буде виглядати наступним чином:

```
enum EnumBearingWallSelect
{
        e_SelectedVariant = 0, //варіант обраний у конфігураційному файлі
        e_Material_Thickness = 1, //матеріал + товщина
        e_Material = 2, //матеріал
        e_Thickness = 3, //товщина
        e_LayerCount = 4 //кількість шарів
};
```

Для того аби перевірити несуча стіна чи ні, за кількістю шарів, опишемо наступний виклик:

```
if(LibraryWrapper::IsBearingWall(archWall, EnumBearingWallSelect::e_LayerCount))
      cout << "Стіна несуча";
   else
       cout << "Стіна не несуча";
```

Якщо не вказувати варіант перевірки на несучість, за замовчуванням буде використаний варіант, який був обраний користувачем на формі налаштувань.

ВИСНОВКИ ДО РОЗДІЛУ 2

1. Досліджена інформаційна модель об’єкта Allplan.
2. Проведений аналіз існуючих проблем уніфікації таких понять як: несуча стіна, зовнішня стіна, назви матеріалів і т.н.
3. Сформована концепція загальної бібліотеки уніфікації на прикладі інформаційної системи автоматизованого проектування Allplan. Показаний її взаємозв’язок з модулями системи.
4. Досліджена структура конфігураційних даних, необхідних для функціонування бібліотеки уніфікації.
5. Наведені приклади роботи з бібліотекою уніфікації.

РОЗДІЛ 3

МЕТОДИ СТВОРЕННЯ ПЛАСТИЧАТО-СТЕРЖНЕВОЇ МОДЕЛІ

3.1. Схема пластинчато-стержневої моделі

Виконавши дослідження сучасних проблем інтеграції, було встановлено, що система автоматизованого проектування Allplan від компанії Nemetschek не має готового рішення для автоматичної генерації конструктивної моделі та потрібних інструментів для роботи з нею. Оскільки комплексний розрахунок конструктивної моделі в Allplan виконати неможливо, було вирішено розробити проміжну модель для подальшої її передачі у розрахункові комплекси, цю модель в подальшому будемо називати ПСМ.

Пластинчато-стержнева модель (ПСМ) - це уніфікована модель, склад якої зводиться до пластин та стержнів, основною задачею якої є збереження конструктивних характеристик архітектурних елементів моделі. ПСМ є свого роду проміжною моделлю між архітектурною моделлю і конструктивною (див. рис. 3.1).

Рис. 3.1. Перехід від архітектурної моделі до конструктивної

Створена ПСМ служить лише основою для подальшої побудови розрахункової схеми, адже в ПСМ відсутні чисто розрахункові елементи які моделюють ті чи інші конструктивні вузли (жорсткі вузли шарнірні вузли, платформні стики та ін.), відсутня кінцево-елемента сітка, відсутня необхідна інформація для підготовки розрахунку на сейсмічні та вітрові навантаження, немає інформації для формування умов спирання і т.д.

Складається ПСМ з пластин та стержнів, створених на основі геометричних, атрибутивних та міцнісних характеристик архітектурних елементів. У загальному, ПСМ можна подати у вигляді формули:

$$\text{ПСМ} = \{\boldsymbol{P}\} \cup \{\boldsymbol{B}\} \quad (3.1)$$

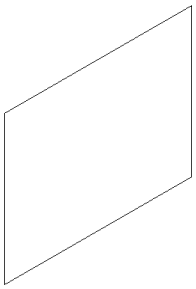
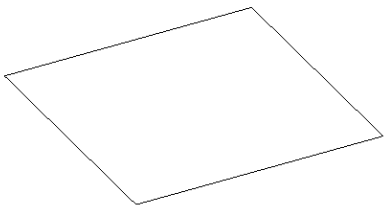
де $\boldsymbol{P} = \{\boldsymbol{P_1}, \boldsymbol{P_2} \ldots \boldsymbol{P_n}\}$ – множина всіх пластин;

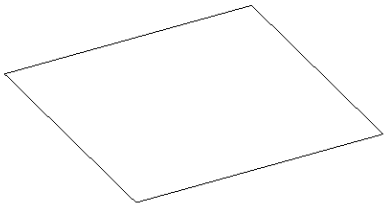
$\boldsymbol{B} = \{\boldsymbol{B_1}, \boldsymbol{B_2} \ldots \boldsymbol{B_n}\}$ – множина всіх стержнів.

Наведомо таблицю відповідностей між архітектурною моделлю та ПСМ (табл. 3.1).

Таблиця 3.1

Подання архітектурних елементів у конструктивній моделі

Архітектурний елемент		**Елемент ПСМ**	
W	Стіна	Пластина	
S	Перекриття	Пластина	
C	Колона	Стержень	

B	Балка	Стержень	
TF	Стрічковий фундамент	Стержень	
CF	Стовпчастий фундамент	Стержень	
SF	Суцільний фундамент	Пластина	

Враховуючи наведену таблицю, запишемо ПСМ у вигляді наступної формули:

$$\text{ПСМ} = \{\boldsymbol{W}\} \cup \{\boldsymbol{S}\} \cup \{\boldsymbol{C}\} \cup \{\boldsymbol{B}\} \cup \{\boldsymbol{TF}\} \cup \{\boldsymbol{CF}\} \cup \{\boldsymbol{SF}\} \qquad (3.2)$$

$\boldsymbol{C} = \{\boldsymbol{C_1}, \boldsymbol{C_2} \ldots \boldsymbol{C_n}\}$ – множина стержнів колон;

$\boldsymbol{B} = \{\boldsymbol{B_1}, \boldsymbol{B_2} \ldots \boldsymbol{B_n}\}$ – множина стержнів балок;

$\boldsymbol{TF} = \{\boldsymbol{TF_1}, \boldsymbol{TF_2} \dots \boldsymbol{TF_n}\}$ – множина стержнів стрічкового фундаменту;

$\boldsymbol{CF} = \{\boldsymbol{CF_1}, \boldsymbol{CF_2} \dots \boldsymbol{CF_n}\}$ – множина стержнів стовпчастого фундаменту;

$\boldsymbol{SF} = \{\boldsymbol{SF_1}, \boldsymbol{SF_2} \dots \boldsymbol{SF_n}\}$ – множина пластин суцільного фундаменту;

Для передачі ПСМ у розрахункові комплекси, вона повинна містити також дані про архітектурну модель, які в подальшому будуть використовуватися у конструктивних розрахунках. Для цього кожен елемент ПСМ, а саме пластина та стержень, повинні містити в собі необхідний набір даних (табл. 3.2).

Таблиця 3.2

Необхідні дані елементів ПСМ для подальших конструктивних розрахунків

Пластина	**Стержень**
Тип архітектурного елементу (стіна, перекриття, колона і т. ін.)	
Товщина	Базова форма елементу (круглий, прямокутний)
	Діаметр, або ширина та довжина

У випадку з пластиною, висота та довжина елемента міститься в геометричних характеристиках пластини. Додаткові дані про елемент записуються в атрибути пластини або стержня.

3.2. Загальна концепція генерації ПСМ

Для генерації ПСМ потрібно спочатку проаналізувати вхідну архітектурну модель і на основі цього згенерувати пластини та стержні. Тобто, пройдемося по всіх елементах, визначимо що за тип у елемента і на основі цього згенеруємо його подання у ПСМ. Зобразимо схематично цей метод (рис. 3.2).

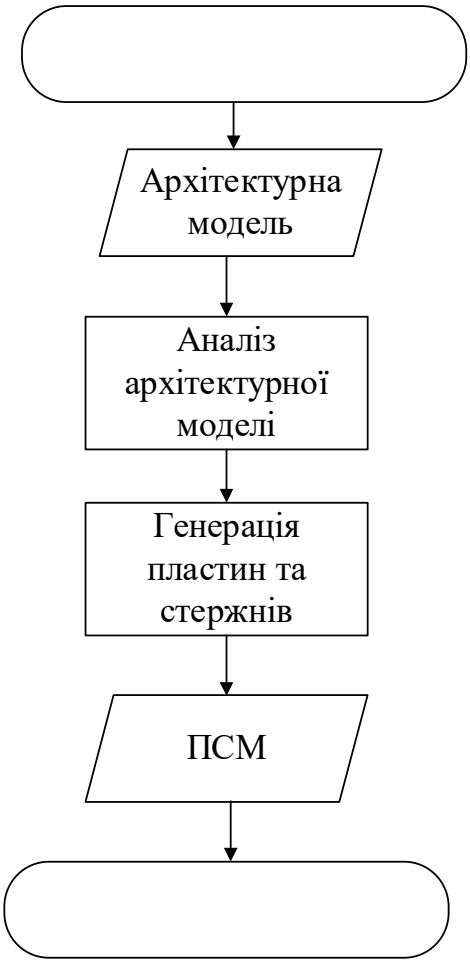


Рис. 3.2. Схема генерації ПСМ

Але як зазначалося вище, після генерації ПСМ, потрібно обов’язково «дотягнути» створену модель, для того щоб позбавитись від колізій, які виникли внаслідок обробки. Прикладом, найпростішої колізії можуть бути дві стіни, що примикають одна до одної (рис. 3.3).

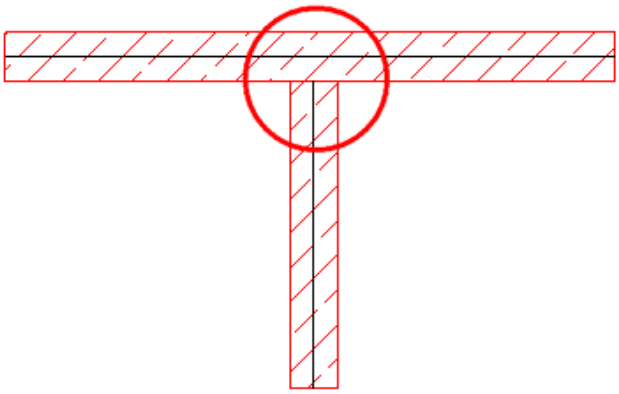


Рис. 3.3. Колізія між пластинами двох стін в згенерованій ПСМ

Для уникнення, подібних колізій, зобразимо схему алгоритму з урахуванням використання методів дотягувань (рис. 3.4).

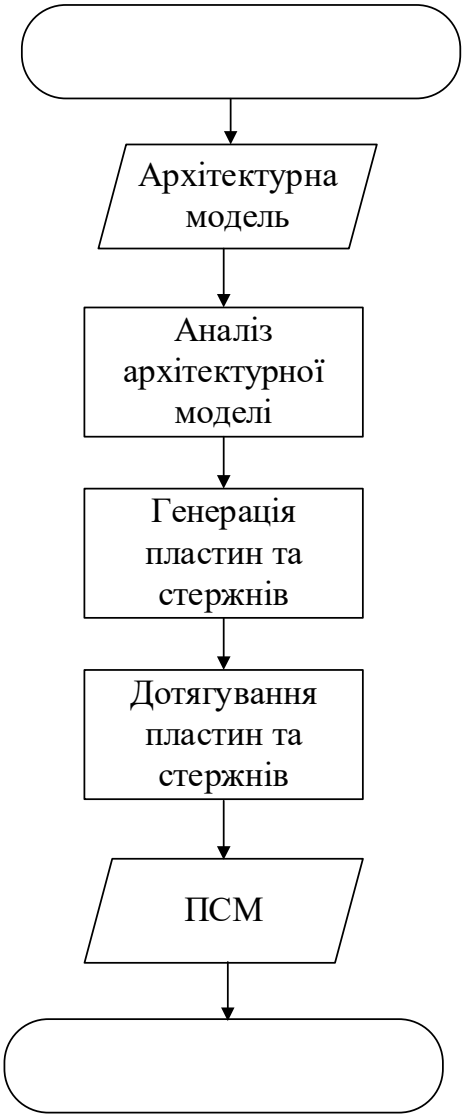


Рис. 3.4. Схема генерації ПСМ з урахуванням необхідності усунення колізій

За такого підходу ми не втрачаємо цілісність пластинчато-стержневої моделі та можемо передавати її у розрахункові комплекси.

Але, в той же час, виникає ряд недоліків, оскільки у нас 7 типових елементів, а саме: стіна, перекриття, балка, колона, стрічковий фундамент, суцільний фундамент та стовпчастий фундамент, то нам потрібно врахувати всі випадки можливих колізій та написати велику кількість алгоритмів дотягування, для кожного конкретного випадку. Наприклад, алгоритм для дотягування стін до стін, алгоритм для дотягування стін до перекриття і т.д. Плюс до цього всього бувають не типові випадки дотягувань, які також треба враховувати. Також перед нами постає проблема послідовності дотягування, адже невідомо в якому порядку виконувати ці дотягування, аби не спричинити нових колізій внаслідок виконання дотягування. Маючи набір методів дотягування та застосувавши їх у різному порядку ми будемо отримувати різні результати. До *основних недоліків такого підходу* можемо віднести:

- потребує написання великої кількості алгоритмів дотягування;
- послідовність виконання алгоритмів дотягування впливає на результат;
- можлива поява нових колізій внаслідок виконання дотягувань не в тому порядку;
- повільна робота застосунку, внаслідок перебору всіх елементів, які беруть участь у дотягуванні.

Переваги:

- Простота в реалізації

Проаналізувавши результати роботи такого підходу, можемо зробити висновок, що цей підхід не є оптимальним і потрібно знаходити більш дієвий спосіб генерації пластинчато-стержневої моделі.

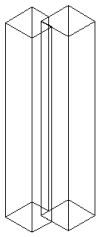
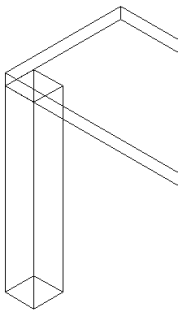
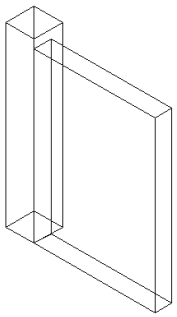
Спробуємо підійти з іншого боку, та аналізувати модель на основі стиків архітектурних об’єктів, а в подальшому усувати колізії для кожного

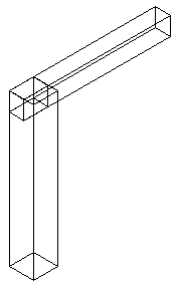
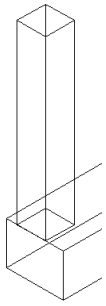
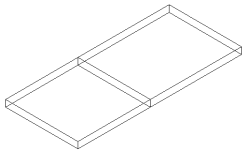
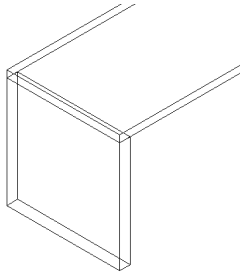
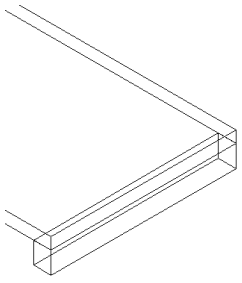
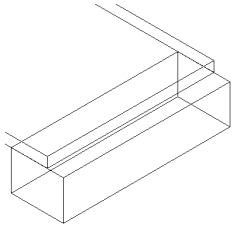
конкретного стику, наприклад стик (стіна-стіна), стик (стіна-перекриття) і т.д. Це дозволить на етапі аналізу виявити можливі колізії, та підібрати необхідні методи для їх усунення. Для подальшого аналізу, потрібно дати визначення поняттю стик. **Стик** – це торкання (дотик) двох архітектурних елементів, яке може бути виражено у вигляді точки, лінії або площини.

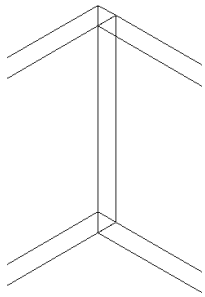
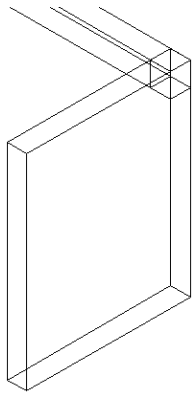
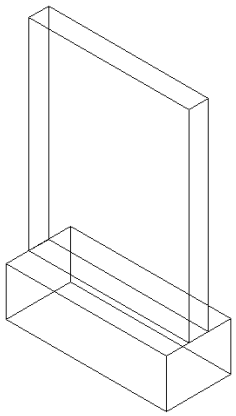
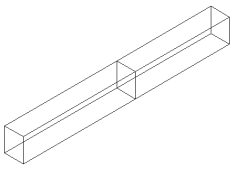
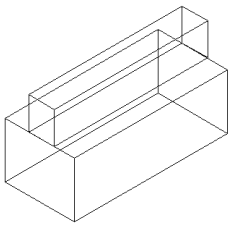
Наведемо перелік можливих стиків між елементами архітектурної моделі. Для простоти відображення не будемо перелічувати кожен тип фундаменту (табл. 3.3).

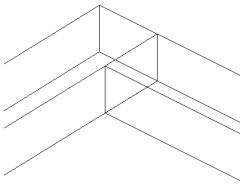
Таблиця 3.3

Можливі типи стиків між елементами архітектурної моделі

Перелік можливих стиків			
CC	Колона	Колона	
CS	Колона	Перекриття	
CW	Колона	Стіна	

CB	Колона	Балка	
CF	Колона	Фундамент	
SS	Перекриття	Перекриття	
SW	Перекриття	Стіна	
SB	Перекриття	Балка	
SF	Перекриття	Фундамент	

WW	Стіна	Стіна	
WB	Стіна	Балка	
WF	Стіна	Фундамент	
BB	Балка	Балка	
BF	Балка	Фундамент	

FF	Фундамент	Фундамент	

$$\text{ПСМ} = \{\boldsymbol{CC}\} \cup \{\boldsymbol{CS}\} \cup \{\boldsymbol{CW}\} \cup \{\boldsymbol{CB}\} \cup \{\boldsymbol{CF}\} \cup \{\boldsymbol{SS}\} \cup \{\boldsymbol{SW}\} \cup \{\boldsymbol{SB}\} \cup \{\boldsymbol{SF}\} \cup \{\boldsymbol{WW}\} \cup \{\boldsymbol{WB}\} \cup \{\boldsymbol{WF}\} \cup \{\boldsymbol{BB}\} \cup \{\boldsymbol{BF}\} \cup \{\boldsymbol{FF}\} \quad (3.3)$$

де $\boldsymbol{CC} = \{\boldsymbol{CC_1}, \boldsymbol{CC_2} \dots \boldsymbol{CC_n}\}$ – множина стиків «колона-колона»;

$\boldsymbol{CS} = \{\boldsymbol{CS_1}, \boldsymbol{CS_2} \dots \boldsymbol{CS_n}\}$ – множина стиків «колона-перекриття»;

$\boldsymbol{CW} = \{\boldsymbol{CW_1}, \boldsymbol{CW_2} \dots \boldsymbol{CW_n}\}$ – множина стиків «колона-стіна»;

$\boldsymbol{CB} = \{\boldsymbol{CB_1}, \boldsymbol{CB_2} \dots \boldsymbol{CB_n}\}$ – множина стиків «колона-балка»;

$\boldsymbol{CF} = \{\boldsymbol{CF_1}, \boldsymbol{CF_2} \dots \boldsymbol{CF_n}\}$ – множина стиків «колона-фундамент»;

$\boldsymbol{SS} = \{\boldsymbol{SS_1}, \boldsymbol{SS_2} \dots \boldsymbol{SS_n}\}$ – множина стиків «перекриття-перекриття»;

$\boldsymbol{SW} = \{\boldsymbol{SW_1}, \boldsymbol{SW_2} \dots \boldsymbol{SW_n}\}$ – множина стиків «перекриття-стіна»;

$\boldsymbol{SB} = \{\boldsymbol{SB_1}, \boldsymbol{SB_2} \dots \boldsymbol{SB_n}\}$ – множина стиків «перекриття-балка»;

$\boldsymbol{SF} = \{\boldsymbol{SF_1}, \boldsymbol{SF_2} \dots \boldsymbol{SF_n}\}$ – множина стиків «перекриття-балка»;

$\boldsymbol{WW} = \{\boldsymbol{WW_1}, \boldsymbol{WW_2} \dots \boldsymbol{WW_n}\}$ – множина стиків «стіна-стіна»;

$\boldsymbol{WB} = \{\boldsymbol{WB_1}, \boldsymbol{WB_2} \dots \boldsymbol{WB_n}\}$ – множина стиків «стіна-балка»;

$\boldsymbol{WF} = \{\boldsymbol{WF_1}, \boldsymbol{WF_2} \dots \boldsymbol{WF_n}\}$ – множина стиків «стіна-фундамент»;

$\boldsymbol{BB} = \{\boldsymbol{BB_1}, \boldsymbol{BB_2} \dots \boldsymbol{BB_n}\}$ – множина стиків «балка-балка»;

$\boldsymbol{BF} = \{\boldsymbol{BF_1}, \boldsymbol{BF_2} \dots \boldsymbol{BF_n}\}$ – множина стиків «балка-фундамент»;

$\boldsymbol{FF} = \{\boldsymbol{FF_1}, \boldsymbol{FF_2} \dots \boldsymbol{FF_n}\}$ – множина стиків «фундамент-фундамент»;

Кожен стик може мати свої колізії після генерації ПСМ. Розбиття всієї моделі на стики дозволяє нам для кожного виду стику розробити певний набір методів, який дозволить на етапі генерації ПСМ автоматично, або ж за участі користувача, усунути ті чи інші колізії.

Приведемо загальну схему генерації ПСМ (рис. 3.5).

Рис. 3.5. Схема генерації ПСМ

Як бачимо з рисунка, на вході у нас архітектурна модель, далі ми повинні провести аналіз архітектурної моделі і виділити типові стики, описанні у табл.

3.3. Після цього ми можемо згенерувати попередню ПСМ та виявити всі колізії які виникли внаслідок генерації. Виявивши колізії, ми повинні їх усунути, аби в подальшому передати у розрахункові комплекси цілісну модель.

Недоліки. Хоч при цьому підході ми й застосовуємо методи дотягувань окремо до кожного стику, враховуючи те, що елемент з одного стику може входити до іншого, ми знову отримаємо проблему з накладанням дотягувань одне на одного. У випадку з великою моделлю, у нас буде занадто велика кількість стиків, що ще більше ускладнить поставлену задачу та поставить під сумнів коректність результуючої пластинчато-стержневої моделі.

Переваги. Розбиваючи модель на стики, ми спрощуємо задачу із пошуком колізій між елементами і дотягуванням пластин та стержнів елементів у подальшому.

Підведемо підсумки даного підходу. Швидкість роботи порівняно з першим не зростає, адже дуже багато часу піде на аналіз моделі та її розбиття на стики, економія буде лише за рахунок дотягувань окремих стиків. Можемо зробити висновок, що даний підхід вирішує поставлену задачу не оптимально.

Базуючись на попередньому підході введемо поняття **складного стику або архітектурного вузла** – це система, яка складається з певної взаємопов’язаної кількості архітектурних об’єктів.

За такого підходу ми зможемо створювати спеціальні методи дотягувань для конкретних випадків будь-якої складності. Головною задачею є реалізація механізму аналізу вхідної архітектурної моделі, виділення архітектурних вузлів та реалізація методів дотягувань.

Опишемо сутність «вузол» разом з його параметрами для подальшої реалізації (табл. 3.4).

Таблиця 3.4

Параметри архітектурного вузла

Архітектурний вузол	
Вхідні архітектурні елементи	Всі елементи, що входять у вузол
Геометричні параметри	Набір геометричних залежностей для визначення типу вузла
Додаткові параметри	Набір додаткових залежностей, наприклад, атрибути вхідних елементів

За такого підходу, ми абстрагуємося, та не використовуємо поняття стіна, пластина стіни чи стик стін і т.д., а оперуємо лише вузлами. Цей підхід дозволяє описати вузли будь-якої складності, а вже в подальшому розбити вузол на конкретні пластини та стержні і усунути всі колізії, що з’явилися внаслідок генерації ПСМ.

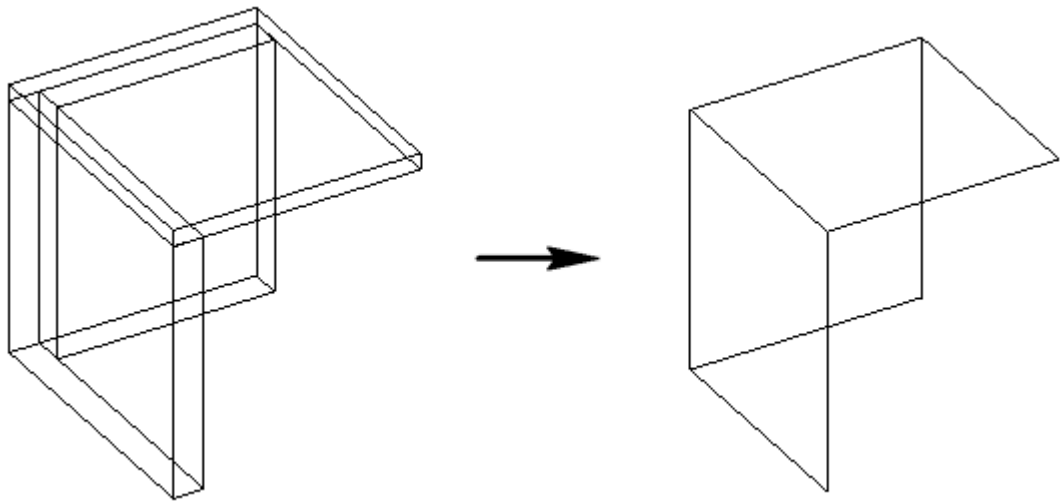


Рис. 3.6. ПСМ архітектурного вузла

Можемо розглядати ПСМ, як сукупність «дотягнутих» вузлів.

$$\text{ПСМ} = \bigcup_n T \tag{3.4}$$

T – архітектурний вузол;

n – кількість архітектурних вузлів.

Зобразимо роботу даного підходу у вигляді схеми.

Рис. 3.7. Схема генерації ПСМ

Кожен вузол буде містити в собі певний набір пластин, стержнів та набір відповідних методів дотягувань. Але навіть при такому підході залишається проблема порядку виконання методів дотягувань, адже кожен метод працює лише з відповідною парою об’єктів. Розглянемо детальніше цю проблему на прикладі архітектурного вузла, який складається із 3-х стін (рис. 3.8).

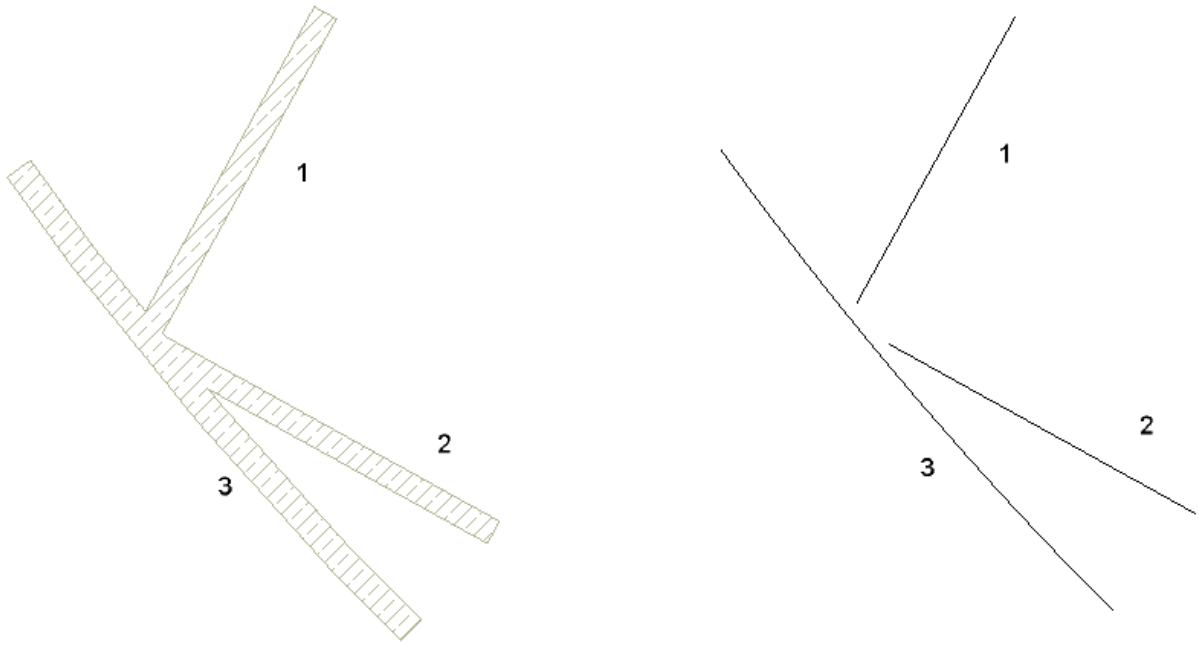


Рис. 3.8. Архітектурний вузол на базі 3-х стін

Для того аби порахувати необхідну кількість дотягувань, потрібно скористатися формулою для розрахунку кількості комбінацій:

$$C_n^k = \frac{n!}{k!(n-k)!} \tag{3.5}$$

n – загальна кількість елементів;

k – кількість елементів у підмножині.

Виконаємо розрахунок кількості дотягувань для нашого прикладу:

$$C_3^2 = \frac{3!}{2!(3-2)!} = \frac{6}{2} = 3 \tag{3.6}$$

У даному випадку кількість дотягувань буде дорівнювати трьом, а саме:

1. дотягування [1-стіна, 3-стіна];
2. дотягування [2-стіна, 3-стіна];
3. дотягування [1-стіна, 2-стіна].

Порядок стін при дотягуванні неважливий. Детально розглянемо результат за такого порядку дотягувань [1, 2, 3] (рис. 3.9).

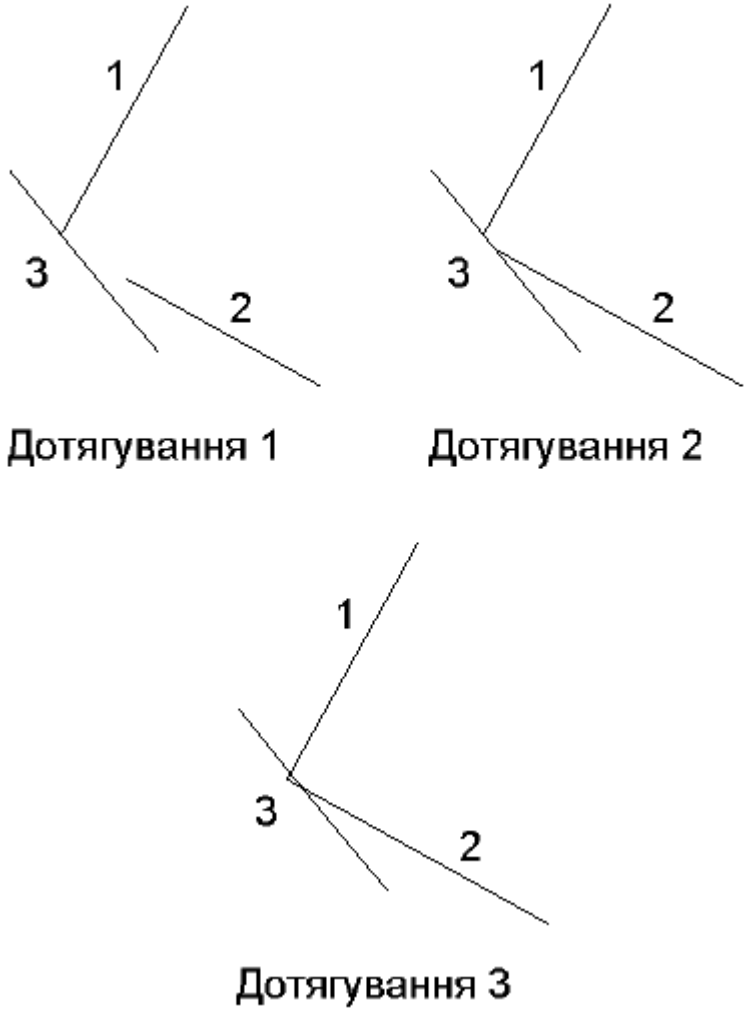


Рис. 3.9. Комбінація дотягувань

Як бачимо, в результаті такого порядку дотягувань, ми отримали нові колізії. Для того аби модель була цілісною нам потрібно або позбутися 3-го дотягування, або поставити його на перше місце. Коректний результат ми отримаємо при наступних порядках дотягувань:

- [3, 1, 2];
- [3, 2, 1];
- [1, 2];
- [2, 1].

У подібних випадках потрібно дати користувачу можливість вибрати правильний порядок та можливість відключити те чи інше дотягування.

Підведемо підсумки щодо цього варіанту вирішення проблеми генерації пластинчато-стержневої моделі. До недоліків можемо віднести складний та довгий аналіз моделі. До переваг: мінімізація накладань дотягувань одна на одну. Для практичної реалізації буде обраний саме цей варіант.

3.3. Методи генерації ПСМ

Як зазначалося вище – ПСМ складається з пластин та стержнів. Тобто необхідно проаналізувати архітектурну модель, та на її основі неї згенерувати відповідні пластини та стержні. Розглянемо методи генерації на прикладі базових архітектурних елементів.

3.3.1. Генерація пластин стін

Розберемо цей метод на прикладі наступної архітектурної стіни (рис. 3.10)

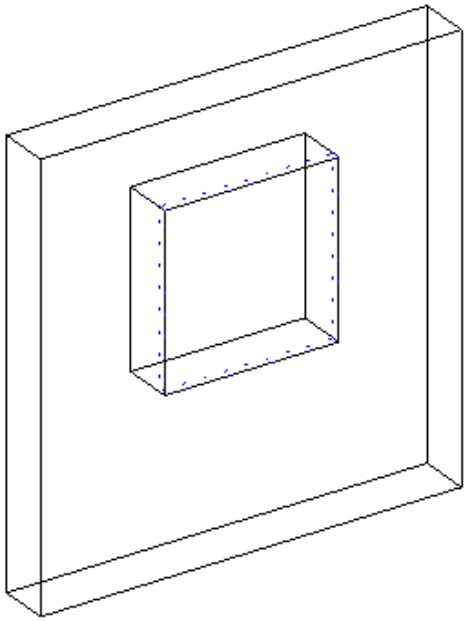


Рис. 3.10. Архітектурна стіна у Allplan

Як бачимо з рисунка 3.10 – в основі стіни лежить паралелепіпед. Що стосується віконного отвору, то у цьому випадку в його основі також лежить паралелепіпед, але він може бути створений більш складним геометричним тілом. У Allplan для відображення отворів 3D тіло має булевий атрибут «орієнтація», у випадку, якщо він негативний – система вважає його отвором та намагається вирізати його об’єм з 3D тіла, до якого входить отвір. У нашому випадку ми маємо два 3D тіла, в основі яких лежать паралелепіпеди, одне тіло є позитивним, інше – негативним. Також, стіна має архітектурну вісь, яка задає напрямок побудови стіни під час її генерації. (рис. 3.11).

Рис. 3.11. Архітектурна вісь стіни у Allplan

Саме архітектурну вісь ми будемо використовувати для побудови базової пластини стіни. Оскільки архітектурна вісь – це 2D відрізок, нам не достатньо координат його точок для побудови пластини. Для побудови пластини нам потрібна висота стіни та координата Z низу стіни. Для цього нам

потрібно отримати **BoundingBox** стіни (рис. 3.12). **BoundingBox**— це паралелепіпед зі сторонами, паралельними осям координат, що обмежує деякий геометричний об’єкт в просторі. При обертанні об’єкта паралелепіпед зберігає свою орієнтацію, однак може змінювати свої розміри. Активно використовується в програмуванні (наприклад в фізичних движках різних ігор) для пошуку перетинів, зіткнень різних об’єктів між собою.

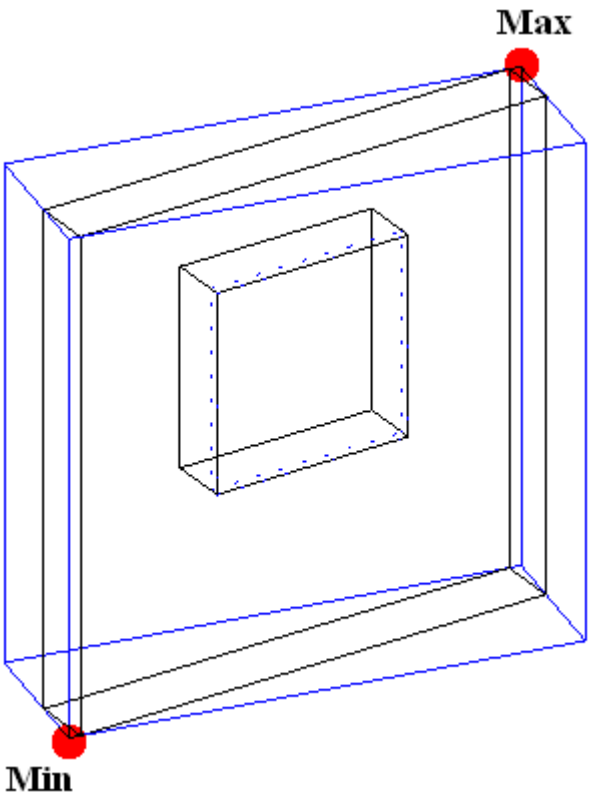


Рис. 3.12. BoundingBox стіни

BoundingBox містить в собі точки **Min** та **Max**, саме їх ми будемо використовувати для отримання максимальної та мінімальної координати Z. В результаті, маючи архітектурну вісь, значення максимальної та мінімальної координат Z можемо побудувати базову пластину стіни (рис. 3.13).

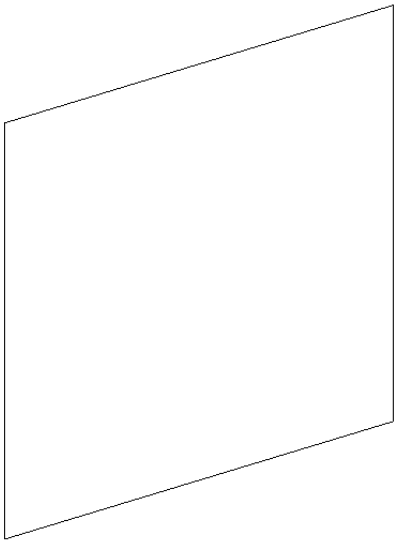


Рис. 3.13. Пластина архітектурної стіни

Далі потрібно перенести на пластину всі отвори, які є в стіни. Для цього скористаємося функціями NOI API «**Intersection**» та «**Union**». Функція **Intersection** – отримує на вхід два 3D тіла та повертає 3D тіло побудоване внаслідок перетину вхідних тіл. Функція **Union** – отримує на вхід два 3D тіла та повертає 3D тіло побудоване внаслідок об’єднання вхідних тіл. Для початку нам потрібно взяти 3D тіло стіни та об’єднати його з усіма 3D тілами отворів, в результаті отримаємо одне 3D тіло, з якого будуть вирахувані об’єми всіх отворів.

$$UB = WB \cup OB \tag{3.7}$$

WB – 3D тіло стіни з позитивним об’ємом;

OB – 3D тіло отвору з негативним об’ємом;

UB – результуюче 3D тіло стіни з вирахуваними негативними об’ємами.

Отримавши результуюче тіло стіни з вирахуваними об’ємами, знайдемо його перетин з базовою пластиною стіни, яка була побудована на основі архітектурної осі та BoundingBox.

$$IB = UB \cap WP \tag{3.8}$$

WP – базова пластина стіни;

ІВ – результуюче 3D тіло.

В результаті розрахунку отримаємо потрібну пластину із необхідними отворами (рис. 3.14).

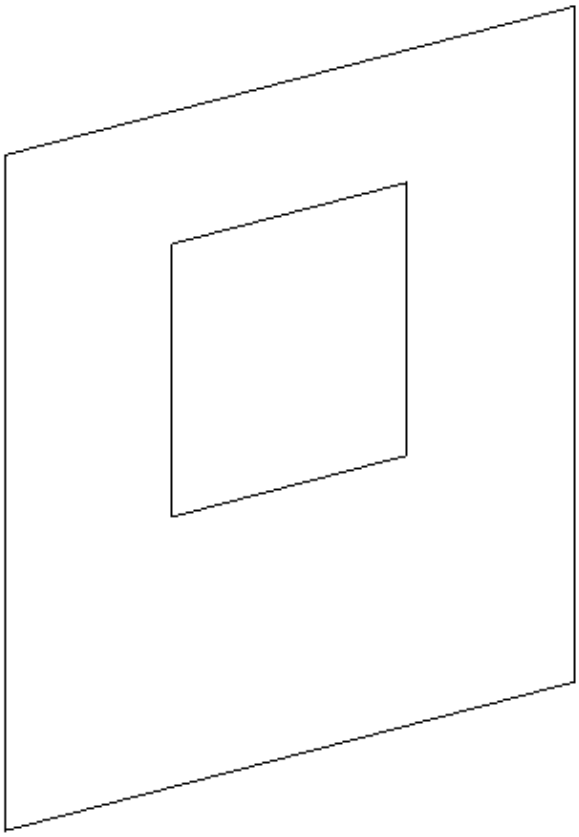


Рис. 3.14. Пластина архітектурної стіни з отвором

Більш складним є випадок з радіальними стінами (рис. 3.15).

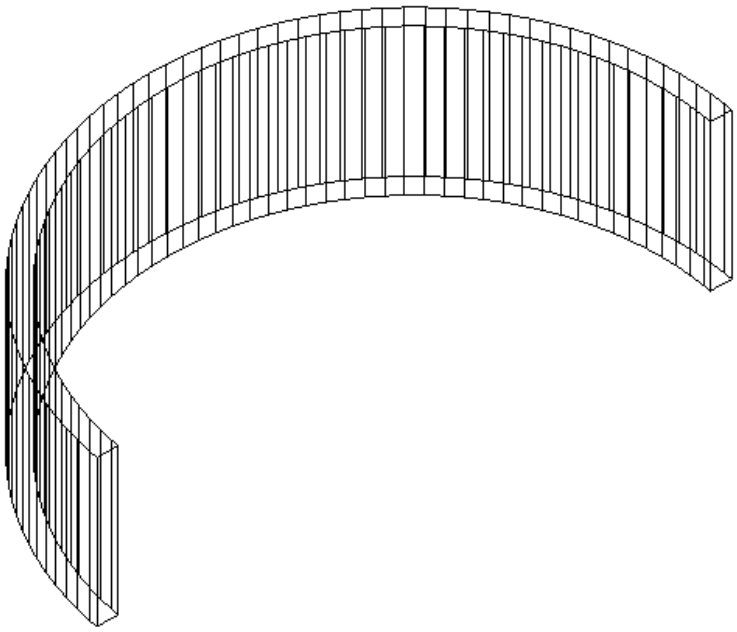


Рис. 3.15. Радіальна стіна у Allplan

В даному випадку стіна складається зі складного 3D тіла. Архітектурною віссю у такої стіни буде не відрізок, а полілінія (рис. 3.16)

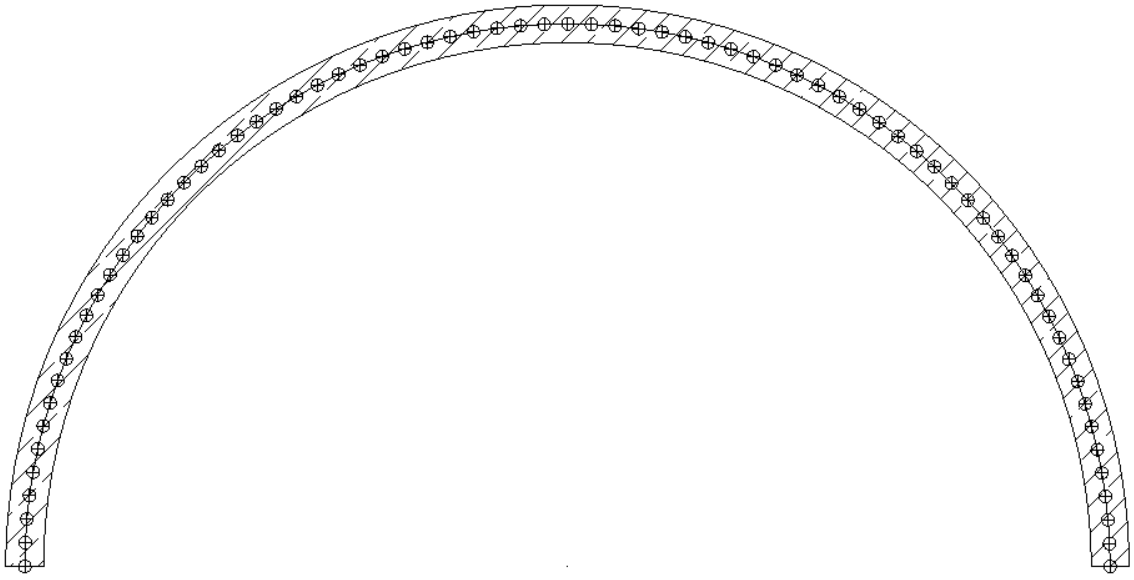


Рис. 3.16. Архітектурна вісь радіальної стіни у Allplan

Для звичайної стіни ми генерували базову пластину, для радіальною потрібно генерувати набір базових пластин на основі точок архітектурної вісі (полілінії) та **BoundingBox**. По суті генерація кожної базової пластини для радіальної стіни така ж сама, як і генерація базової пластини для звичайної стіни. Особливістю є те, що при генерації набору пластин потрібно пам’ятати,

що кожна точка полілінії, окрім першої та останньої, належить зразу двом відрізкам. В результаті генерації, отримаємо наступні пластини (рис. 3.17.)

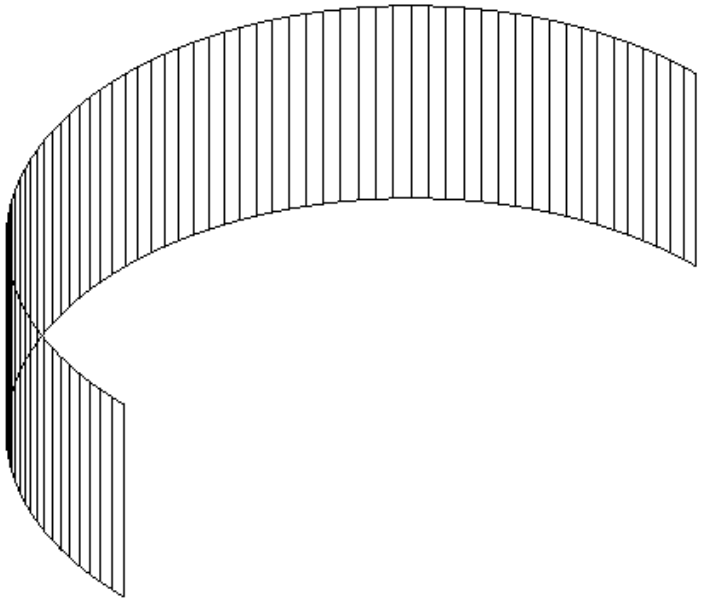


Рис. 3.17. ПСМ радіальної стіни

Що стосується отворів у радіальних стінах, то можна їх отримати аналогічно до звичайної стіни. У випадку радіальної стіни, нам потрібно зібрати набори позитивних та негативних 3D тіл стіни, об'єднати їх та шукати перетини з кожною базовою пластиною.

$$UB = \bigcup WB \cup \bigcup OB \tag{3.9}$$

WB – 3D тіло стіни з позитивним об'ємом;

OB – 3D тіло отвору з негативним об'ємом;

UB – результуюче 3D тіло стіни з вирахуваними негативними об'ємами.

Результуючий набір пластин отримаємо за наступною формулою:

$$IBs = UB \cap \bigcup_i WP_i \tag{3.10}$$

WP – базова пластина фрагменту стіни;

IBs – результуючий набір пластин.

3.3.2. Генерація пластин перекриття

Розглянемо цей метод, на прикладі наступного перекриття (рис. 3.18).

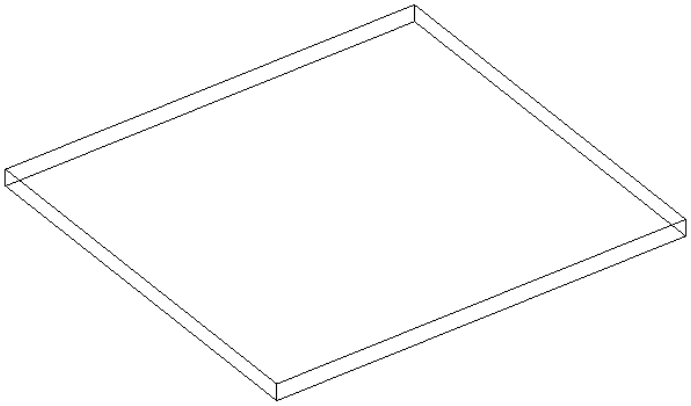


Рис. 3.18. Перекриття у Allplan

На відміну від стіни, перекриття не має архітектурної вісі у системі Allplan. Але кожне 3D тіло у системі подане у вигляді набору точок, ребер та граней. У випадку з перекриттями, ми обираємо верхню грань і вважаємо її нашою базовою пластиною. Для того аби розмістити нашу пластину по центру перекриття, отримуємо **BoundingBox**. Оскільки перекриття завжди паралельне площині XOY, беремо координати Z у мінімальної та максимальної точок **BoundingBox** та знаходимо середнє значення. Замінюємо координату Z у всіх точок базової пластини перекриття (рис. 3.19).

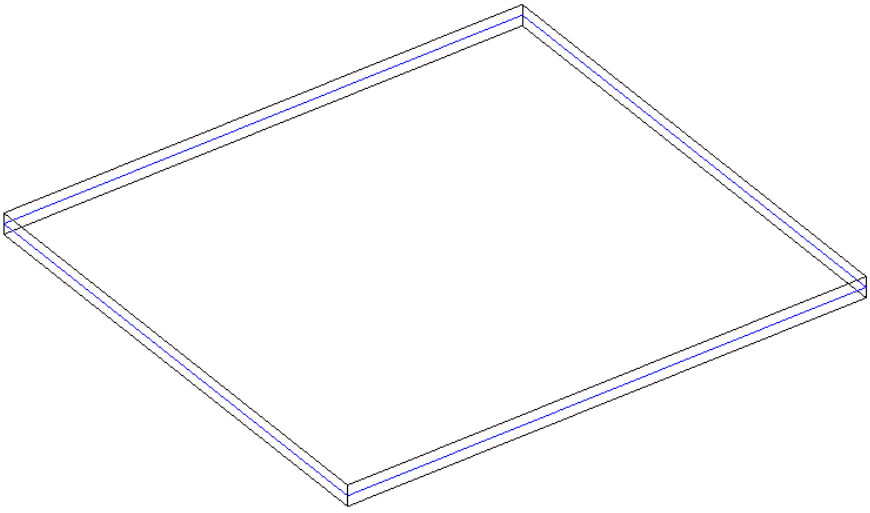


Рис. 3.19. ПСМ перекриття у Allplan.

У випадку коли перекриття містить отвори, використовуємо методи **Intersection** та **Union**, аналогічно методу генерації пластин для стін.

3.3.3. Генерація стержнів колон та балок

Колони, балки, стрічковий та стовпчастий фундаменти представляються в ПСМ у вигляді стержнів. Розглянемо метод на прикладі двох колон: круглої та прямокутної (рис. 3.20).

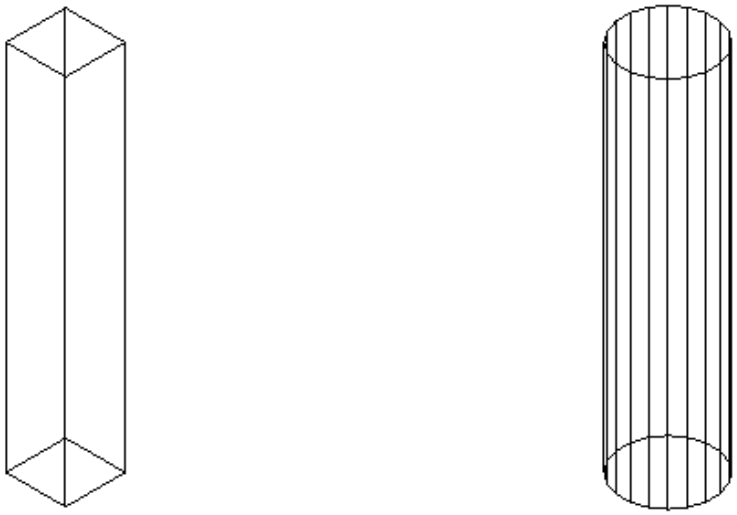


Рис. 3.20. Колони у Allplan

Як бачимо з рисунка, в основі кожної з колон лежить багатокутник. Для того аби побудувати стержні, нам потрібно виділити нижні та верхні грані колон (рис. 3.21).

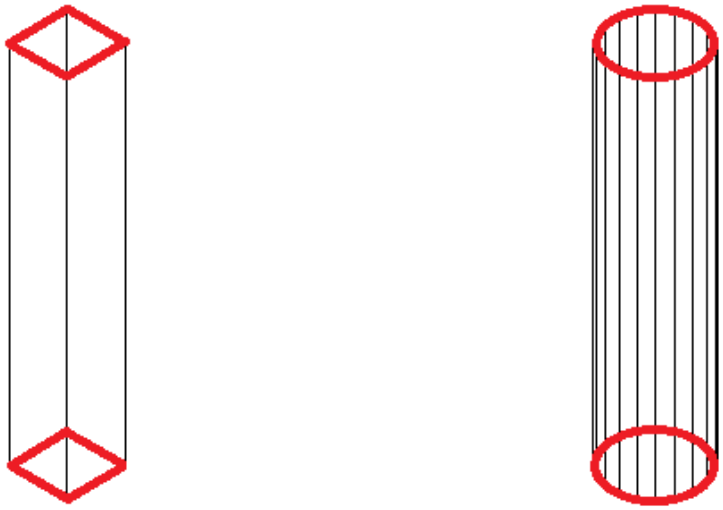


Рис. 3.21. Грані основ колон у Allplan

Щоб знайти координати центру багатокутника використаємо наступні формули:

$$O_x = \frac{\sum_i x_i}{i}$$

$$O_y = \frac{\sum_i y_i}{i} \qquad (3.11)$$

$$O_z = \frac{\sum_i z_i}{i}$$

Застосувавши наступні формули до багатокутників, які лежать у верхній та нижній гранях колони, ми отримаємо дві точки, які дозволять нам побудувати стержень (рис. 3.22). Дане рішення також справедливе для балок, стрічкового та стовпчастого фундаментів.

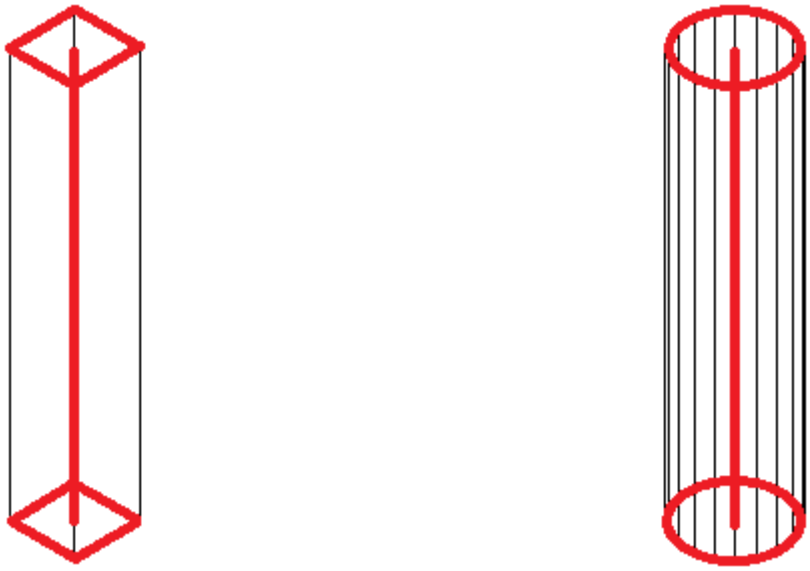


Рис. 3.22. ПСМ колон у Allplan

3.4. Методи «дотягувань»

Під час генерації ПСМ виникають колізії, внаслідок перетворення об’ємних 3D тіл в пластини та стержні. Для усунення цих колізій, необхідно розробити наступні методи:

- дотягування пластин до пластин;
- дотягування стержнів до пластин;
- дотягування стержнів до стержнів;
- дотягування пластин до стержнів.

3.4.1. Метод дотягування пластин до пластин

Розглянемо цей метод на прикладі двох архітектурних стін (рис. 3.23)

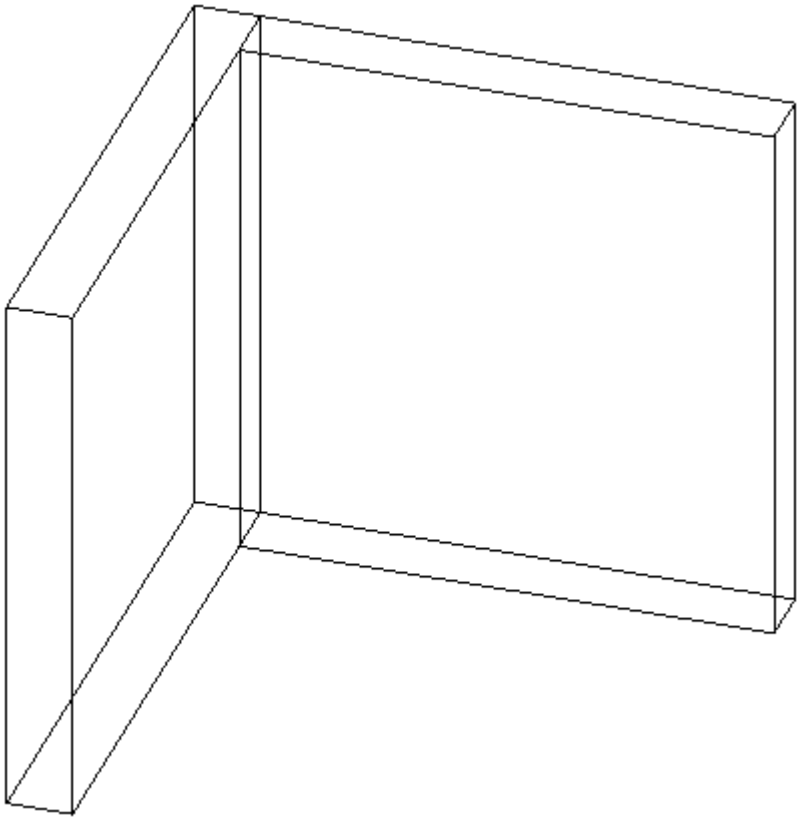


Рис. 3.23. Стик двох стін у Allplan

Як бачимо стіни стикуються і ніяких колізій між ними не має. Побудуємо пластини для цих стін, використовуючи методи, що описані вище (рис. 3.24).

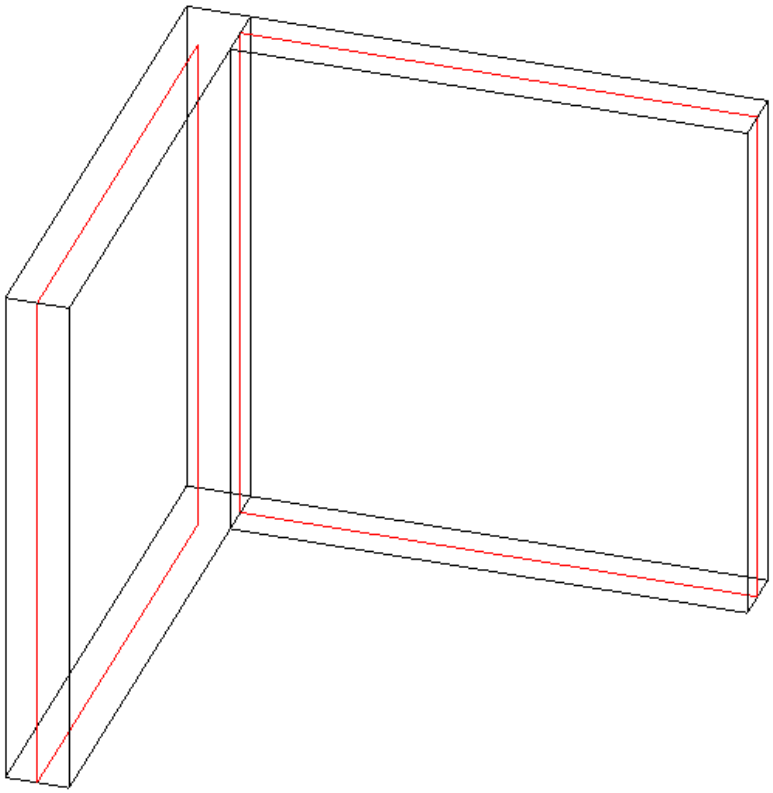


Рис. 3.24. ПСМ стику двох стін

З рисунку 3.24 видно, що пластини стін недотягнулись і між ними виник проміжок, довжина якого дорівнює половині товщини стіни. Для усунення цієї колізії потрібно знайти пряму перетину між цими пластинами. Знайшовши пряму перетину, потрібно знайти найближчі ребра пластин, спроектувати їх точки на пряму та замінити точки, що проектували на спроектовані. Для того аби знайти пряму перетину, потрібно подати наші пластини у вигляді рівнянь площини. Спочатку потрібно знайти нормалі до цих площин. Для опису площини потрібно як мінімум 3 точки у просторі, в нашому випадку їх по 4 для кожної площини (рис. 3.25).

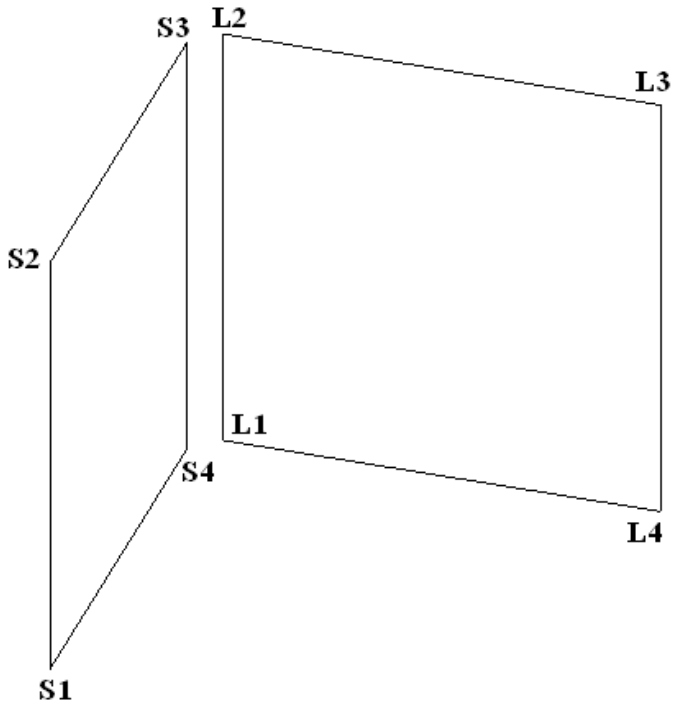


Рис. 3.25. Дві площини у просторі

Маємо площини S (S1, S2, S3) та L (L1, L2, L3), які задані відповідними точками. Знайдемо нормалі до площини S та L:

$$NS = \overrightarrow{S2S1} \times \overrightarrow{S2S3} \tag{3.12}$$

$$NL = \overrightarrow{L2L1} \times \overrightarrow{L2L3} \tag{3.13}$$

Канонічне рівняння площини у просторі має наступний вигляд:

$$Ax + By + Cz + D = 0 \tag{3.14}$$

Отримавши нормалі, можемо представити площини у вигляді канонічних рівнянь:

$$S \rightarrow NS_x + NS_y + NS_z - (NS_x * S1_x + NS_y * S1_y + NS_z * S1_z) = 0 \tag{3.15}$$

$$L \rightarrow NL_x + NL_y + NL_z - (NL_x * L1_x + NL_y * L1_y + NL_z * L1_z) = 0 \tag{3.16}$$

Склавши систему рівнянь, отримаємо 3 невідомих. Але знаючи, що пластини стін паралельні вісі OZ, то і пряма перетину цих пластин також буде паралельна вісі OZ. Врахувавши це, розв’яжемо систему рівнянь при Z = 0. Розв’язавши систему, отримаємо точку **P1** на прямій перетину. Для того аби

знайти ще одну точку на прямій **P2**, можемо підставити інше значення Z у систему рівнянь та розв’язати її або знайти напрямний вектор прямої та додати його до точки **P1**. Для того аби отримати напрямний вектор прямої **U**, знайдемо векторний добуток нормалей:

$$\vec{U} = NS \times NL \tag{3.17}$$

Знайдемо точку **P2**:

$$\begin{aligned} \boldsymbol{P2_x} &= \boldsymbol{P1_x} + \vec{\boldsymbol{U}}_{\boldsymbol{x}} \\ \boldsymbol{P2_y} &= \boldsymbol{P1_y} + \vec{\boldsymbol{U}}_{\boldsymbol{y}} \\ \boldsymbol{P2_z} &= \boldsymbol{P1_z} + \vec{\boldsymbol{U}}_{\boldsymbol{z}} \end{aligned} \tag{3.18}$$

В результаті вищеописаних обчислень отримаємо пряму, що задана двома точками **P1** та **P2**.

Далі нам потрібно спроектувати найближчі точки пластин стін на пряму перетину площин (рис. 3.26), отримання проекції точки на пряму детально розглядається у розділі **3.4.3.**

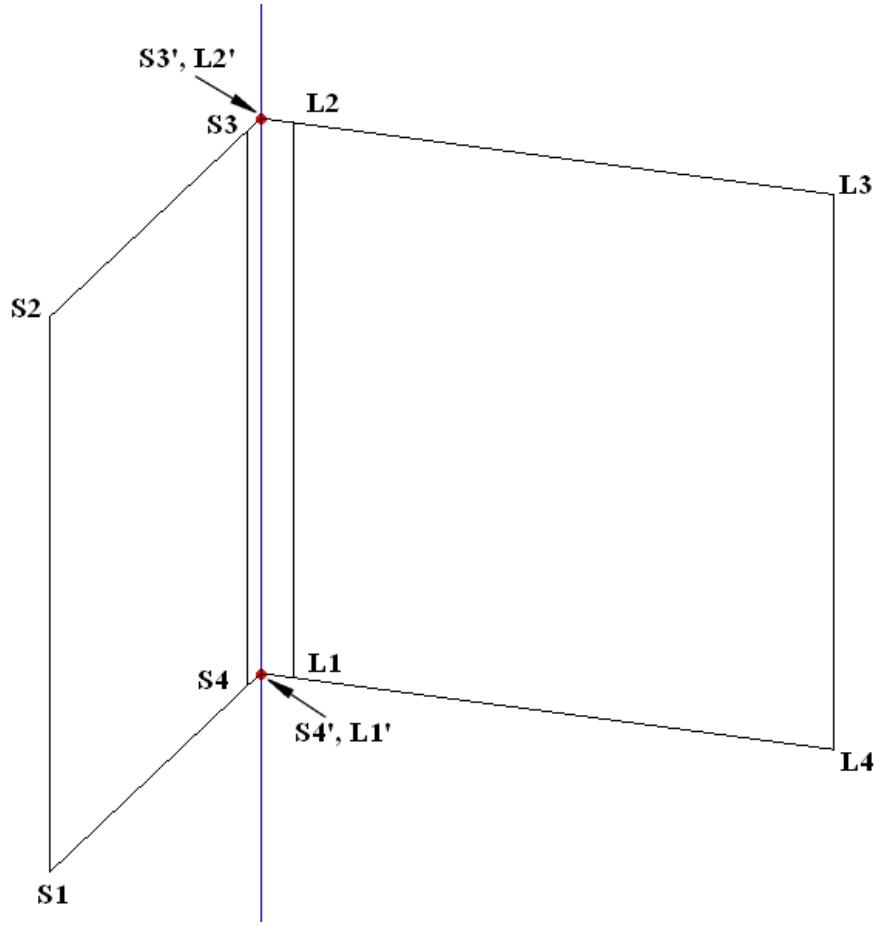


Рис. 3.26. Пряма перетину двох площин

Після того як спроектували найближчі точки пластин на пряму їх перетину, заміняємо точки на їх проекції:

$$L1 = L1'$$
$$L2 = L2' \qquad (3.19)$$
$$S3 = S3'$$
$$L4 = L4'$$

В результаті ми отримаємо 2 дотягнуті пластини стіни, які стикуються одна з одною без колізій.

Описаний вище випадок, за якого виникає потреба дотягування, можна віднести до найпростішого, адже дві архітектурні стіни стикуються в одному куті.

3.4.1.1. Дотягування стін у випадку коли стіни не стикуються в одному куті

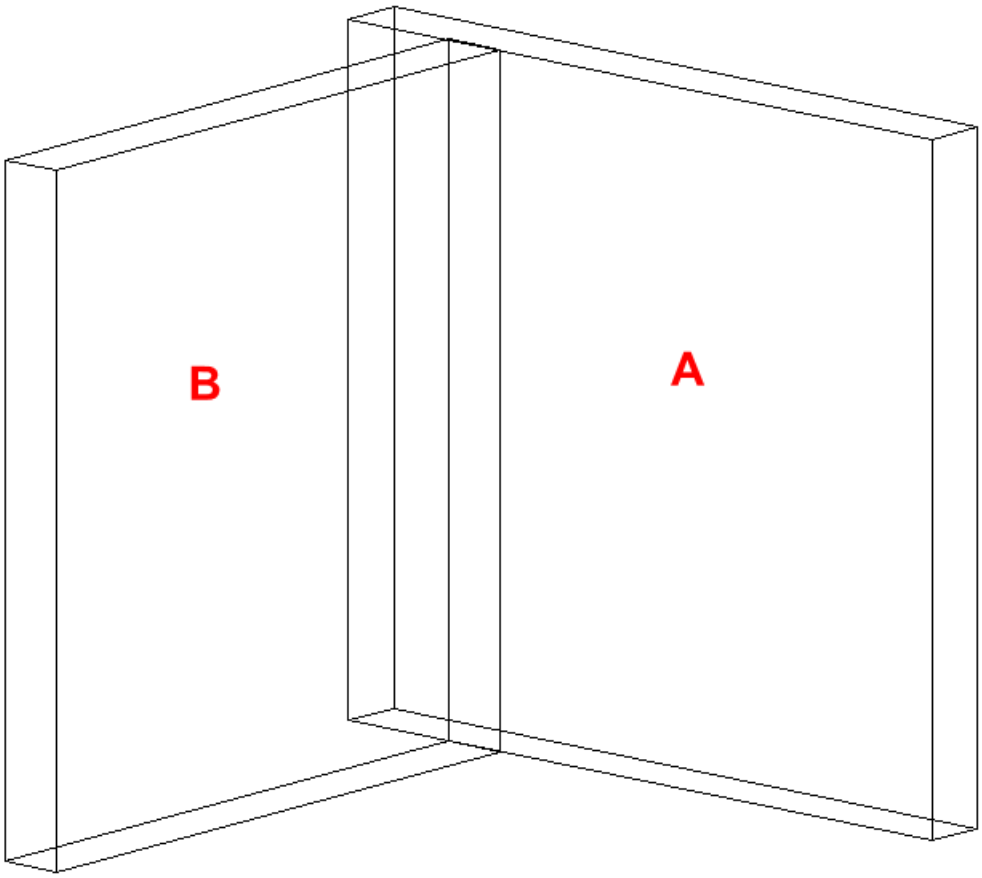


Рис. 3.27. Примикання двох стін

В такому випадку ми не можемо використати метод описаний вище, адже в результаті пластина стіни А буде мати некоректний розмір і дотягнеться до перетину з пластиною стіни В (рис. 3.28).

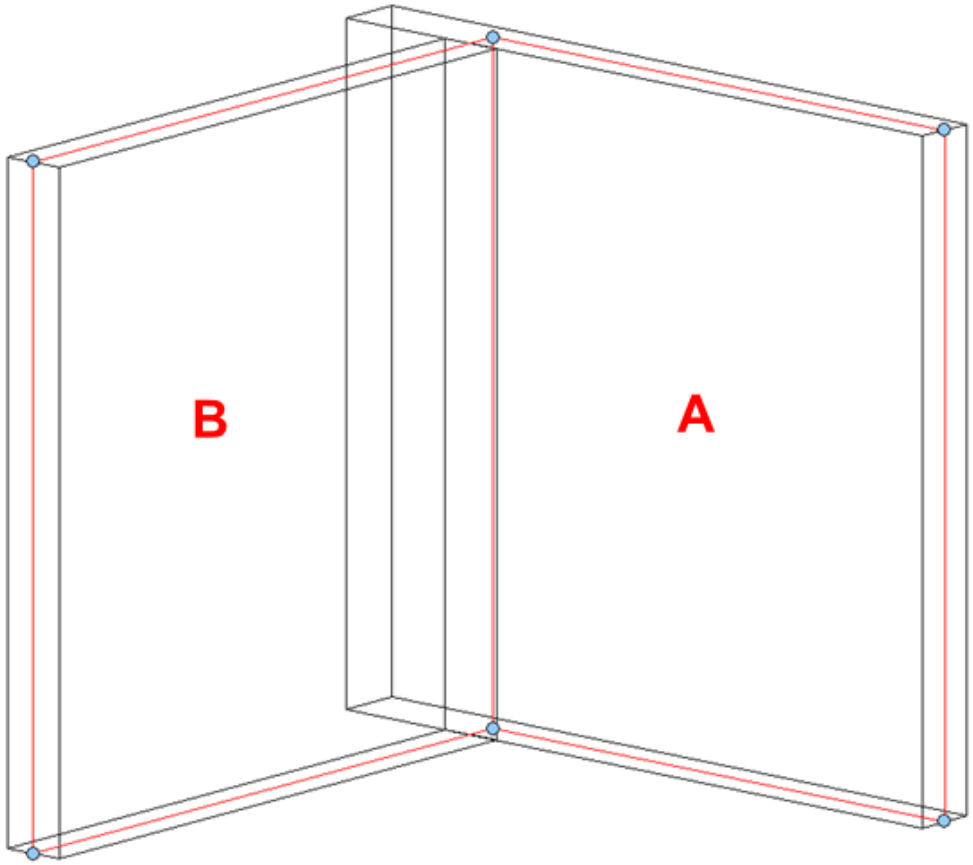


Рис. 3.28. ПСМ на базі звичайного дотягування

Для вирішення цієї проблеми пропонується додати додаткову умову при дотягуванні пластин стін, а саме, перевіряти відстань від спроектованої точки до оригінальної точки пластини і виконувати заміну точки лише в тому випадку, якщо відстань менша ніж половина товщини стіни.

$$Distance(P,\ P') < \frac{Thickness(A)}{2} \rightarrow\ P = P'\text{ , де:} \qquad (3.22)$$

$Distance(point1, point2)$ – метод для пошуку відстані між двома точками у просторі;

A – архітектурна стіна А;

P – точка на пластині стіни В;

P' – проекція точки Р на пластину стіни А;

Thickness – метод для отримання товщини архітектурного елемента.

Ця умова буде справедлива в тому випадку коли вісь архітектурної стіни буде проходити по центру стіни. Аби врахувати інші можливі варіанти, пропонується додати можливість налаштування умовної відстані.

$$Distance(P,\ P') < ManualDistance \rightarrow\ P = P'\ ,\ \text{де:} \qquad (3.23)$$

$ManualDistance$ – умовна відстань, яку можна налаштувати під той чи інший випадок дотягування.

Також це дозволить зробити метод більш універсальним і врахувати навіть той випадок, коли попередньо розглянутий результат дотягування (рис. 3.28) буде коректним. Врахувавши цю умову, отримаємо коректний результат (рис. 3.29).

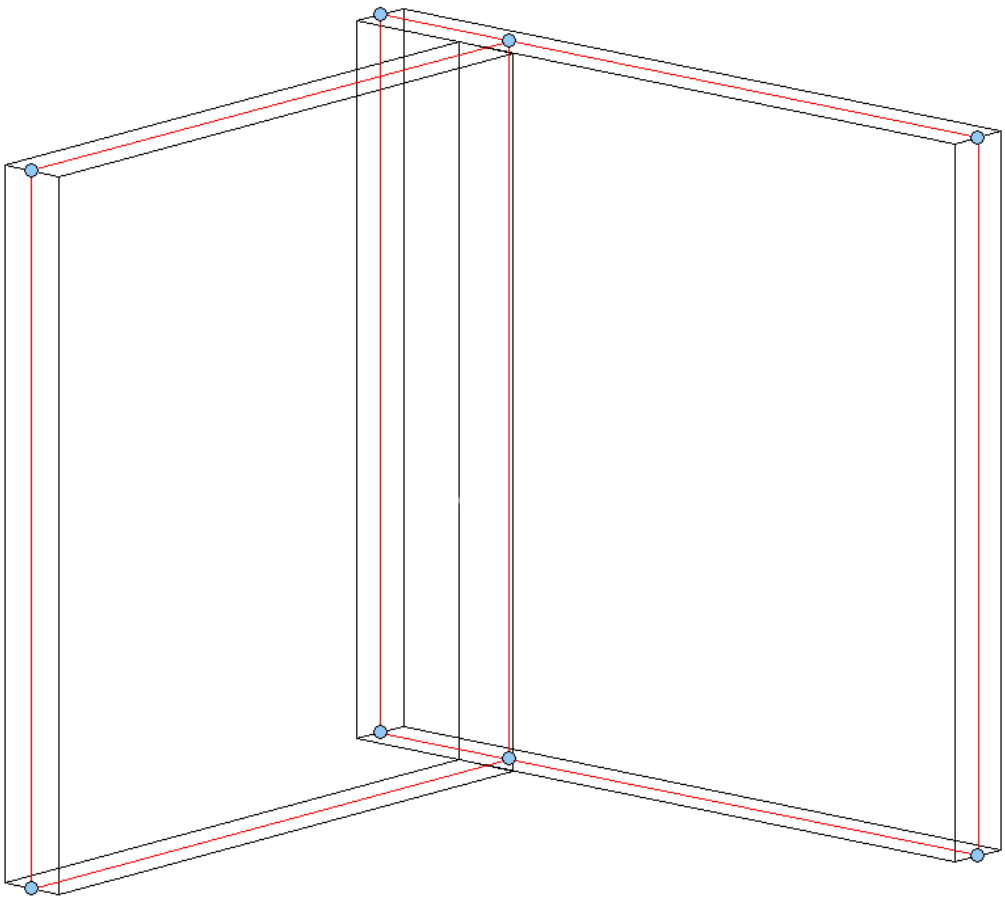


Рис. 3.29. Дотягнута ПСМ з додатковою перевіркою відстані

3.4.1.2. Дотягування радіальних стін

Окремим складним випадком є дотягування пластин радіальних стін, адже кожна стіна складається не з однієї пластини, а з декількох.

Дотягування радіальних стін можна розбити на наступні 2 підпункти:

- дотягування звичайних стін до радіальних;
- дотягування радіальних стін до звичайних;
- дотягування радіальних стін до радіальних.

3.4.1.2.1. Дотягування звичайних стін до радіальних стін

Опишемо метод на наступному прикладі (рис. 3.29)

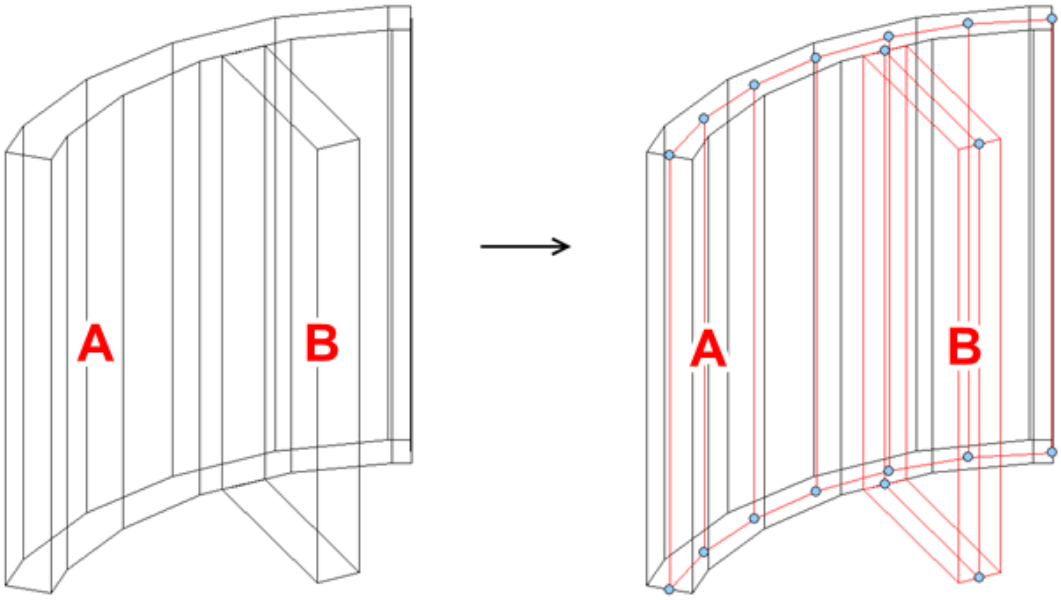


Рис. 3.29. Примикання звичайної стіни до радіальної

В аналітичній моделі, модель радіальної архітектурної стіни складається з набору пластин, тому першою задачею перед дотягування є пошук пластини радіальної стіни, до якої нам необхідно дотягнути пластину звичайної стіни. Для цього пропонується пройтись по всім пластинам радіальної стіни та знайти найближчі пластини до пластини звичайної стіни. В результаті отримаємо певний набір пластин радіальної стіни.

$$NearestPlates = B \rightarrow GetNearestPlates(A), \text{ де:} \quad (3.24)$$

B – звичайна архітектурна стіна;

A – радіальна архітектурна стіна;

GetNearesPlates(x) – метод для пошуку найближчих пластин, до пластини x;

$NearestPlates$ – результуючий набір найближчих пластин.

Для того аби обрати конкретну пластину, що нас цікавить, проектуватимемо найближчі точки пластини звичайної стіни на кожну знайдену пластину радіальної стіни та перевірятимемо, чи потрапляють спроектовані точки в той чи інший полігон (пластину радіальної стіни), таким чином ми зможемо локалізувати конкретну пластину радіальної стіни. Зобразимо це за допомогою псевдокоду.

$$\boldsymbol{for\ each}\ NearestPlate\ \boldsymbol{in}\ NearestPlates\ \{$$

$$B\_NearestPoints = B \rightarrow Plate \rightarrow GetNearestPoints(NearestPlate)$$

$$B\_ProjectPoints = ProjectToPlate(B\_NearestPoints, NearestPlate)$$

```
If (IsPointsInPlate(B_ProjectPoints, NearestPlate))
{
    FindPlate = NearestPlate;
    Break;
}
```

}, де:

$B \rightarrow Plate$ – пластина архітектурної стіни B;

$GetNearestPoints(x)$ – повертає найближчі точки пластини до пластини x;

$ProjectToPlate(points, x)$ – повертає набір спроектованих точок $points$ на пластину x;

IsPointsInPlate(points, plate) – повертає **True** в тому разі, якщо всі точки *points* належать полігону *plate;*

FindPlate – найближча до пластини стіни В пластина радіальної стіни А.

Знаючи пластину, до якої нам потрібно дотягти пластину звичайної стіни, дотягнемо її за методом описаним вище. В результаті отримаємо коректно дотянуті пластини стін (рис. 3.30).

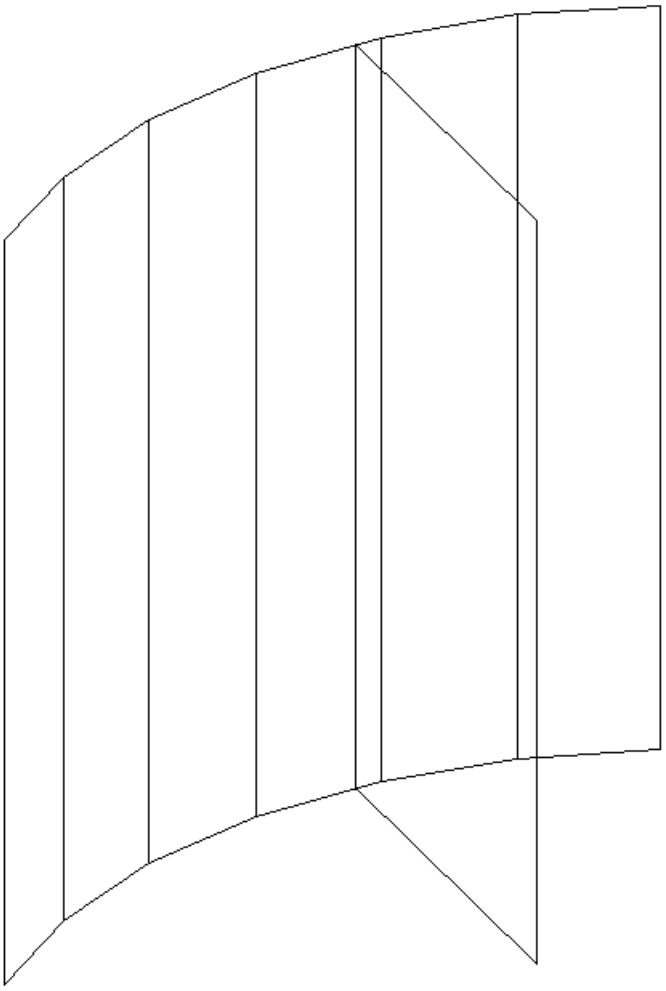


Рис. 3.30. ПСМ примикання звичайної стіни до радіальної

3.4.1.2.2. Дотягування радіальних стін до звичайних

Розглянемо метод на наступному прикладі (рис. 3.31)

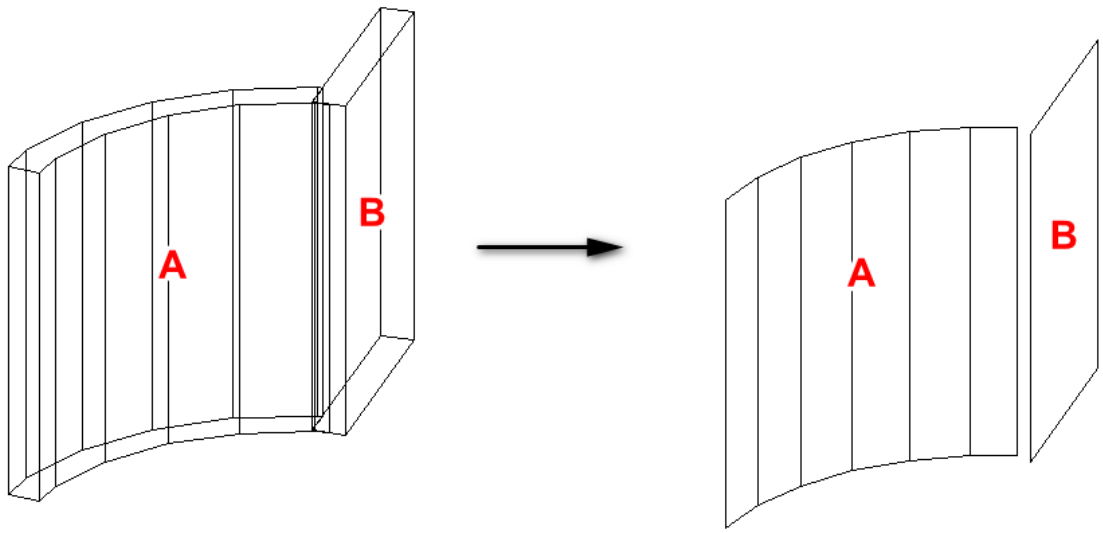


Рис. 3.31. Примикання радіальної стіни до звичайної

В цьому випадку для виконання коректного дотягування потрібно визначитись з пластиною, яку ми будемо тягнути та врахувати декілька перевірок. Пластина радіальної стіни А, яку потрібно дотягувати – це найближча пластина до пластини стіни В. Також аби не зіпсувати аналітичну (пластичнато-стержневу) модель радіальної стіни, потрібно ввести обмеження на пластини які ми можемо дотягувати, а саме: останню та першу пластини радіальної стіни.

$NearestPlate = B \rightarrow GetNearest(A \rightarrow FirstPlate, A \rightarrow LastPlate)$, де: (3.25)

$B \rightarrow GetNearest(plate1, plate2)$ – повертає найближчу пластину серед двох переданих до пластини архітектурної стіни В;

$NearestPlate$ – найближча пластина радіальної стіни А до пластини стіни В.

Знайшовши найближчу пластину радіальної стіни $NearestPlate$, можемо застосувати такий самий метод дотягування, як для двох звичайних стін.

В результаті отримаємо коректну аналітичну модель (рис 3.32).

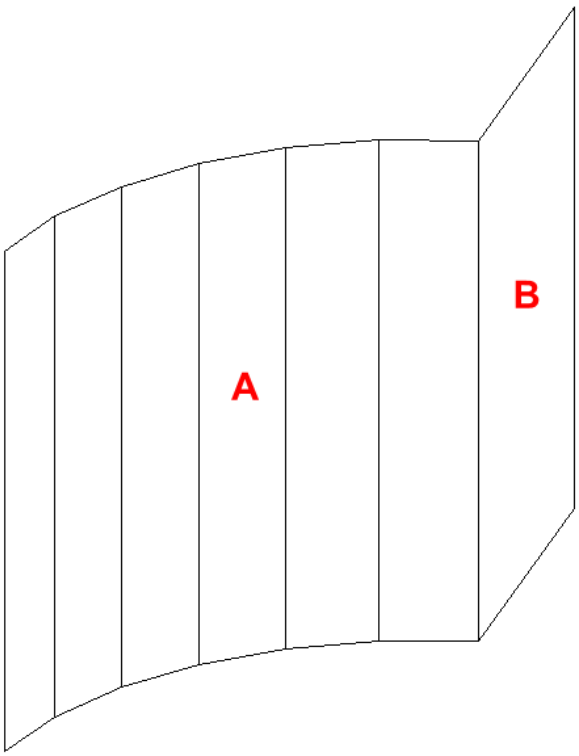


Рис. 3.32. ПСМ примикання радіальної стіни до звичайної

3.4.1.2.3. Дотягування радіальних стін до радіальних

Розглянемо метод на наступному прикладі (рис. 3.33)

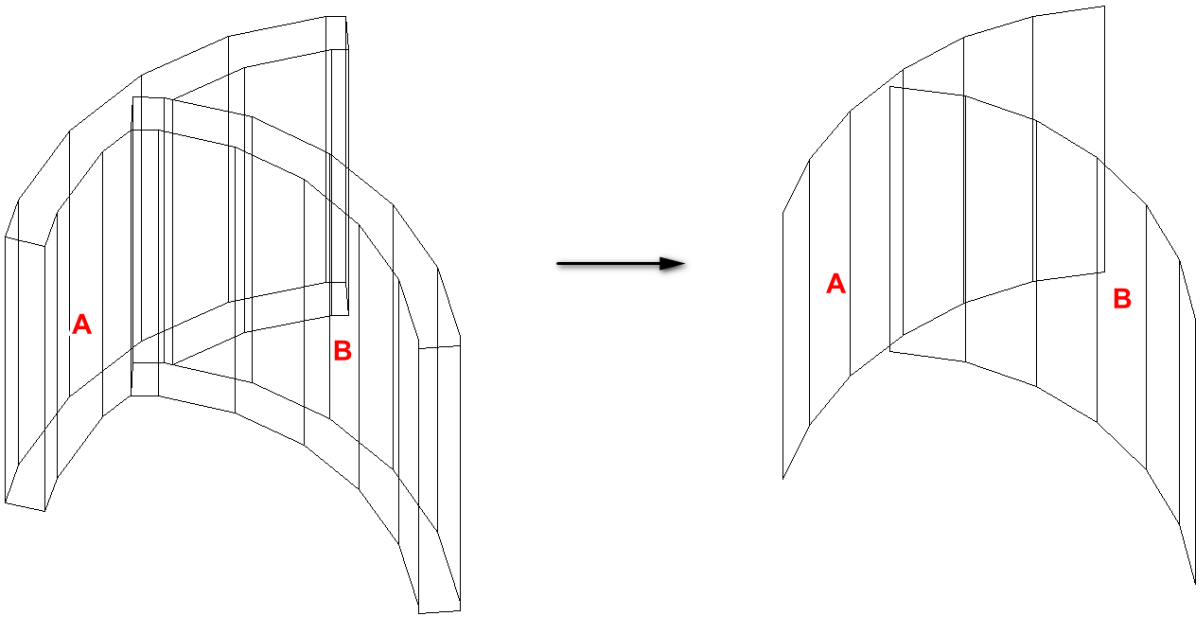


Рис. 3.33. ПСМ двох радіальних стін

Дотяжка пластин двох радіальних стін подібна до дотяжки звичайної стіни до радіальної, відмінність полягає лише в тому, що потрібно знайти одну пластину на радіальній стіні, яку ми будемо дотягувати. Як і в попередньому випадку зі стіни яку дотягуємо беремо лише першу та останню пластини, знаходимо найближчу та дотягуємо за методом дотягування звичайної стіни до радіальної. В результаті отримаємо коректну аналітичну модель двох радіальних стін (рис. 3.34).

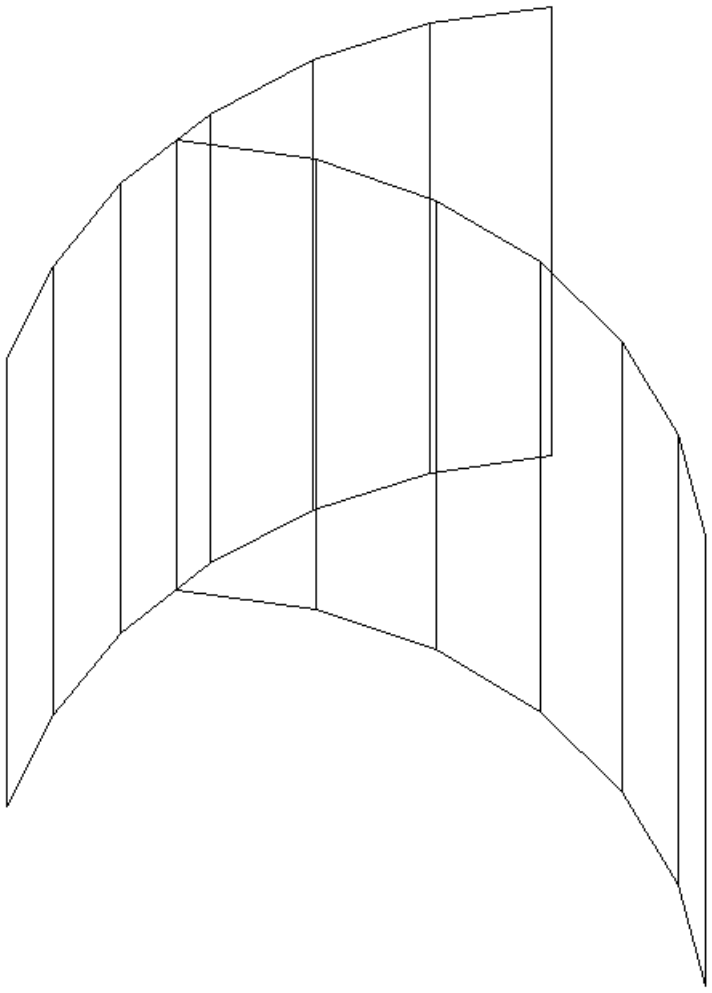


Рис. 3.34. Коректна ПСМ двох радіальних стін

3.4.2. Дотягування стержнів до пластин

Розглянемо цей метод на наступному прикладі (рис. 3.35).

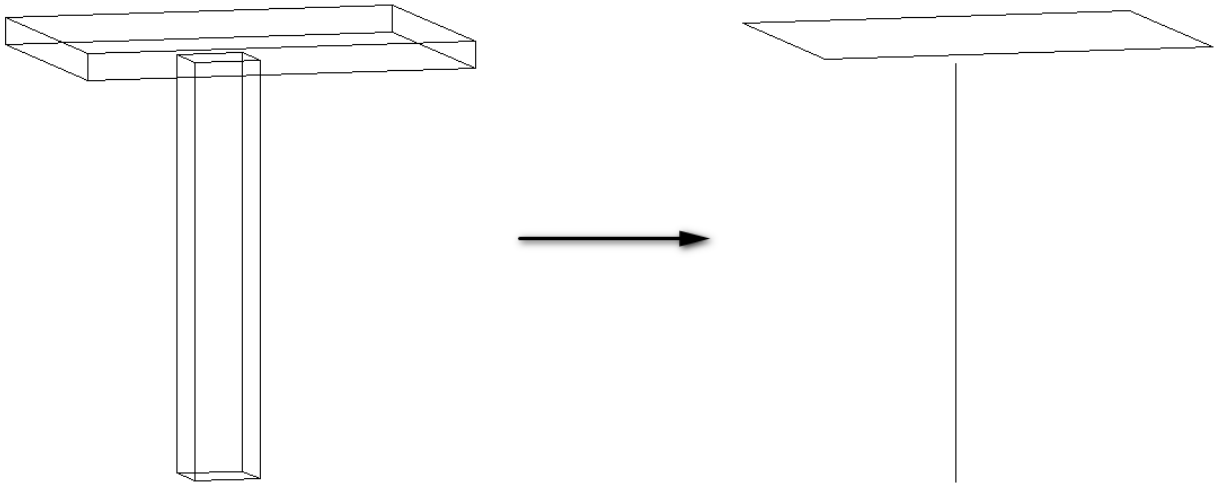


Рис. 3.35. ПСМ перекриття та колони

За рахунок того, що перекриття має свою товщину, а пластину перекриття ми можемо побудувати по верхній його площині, або по середній, виникає проміжок між стержнем колони та перекриттям (рис. 3.36).

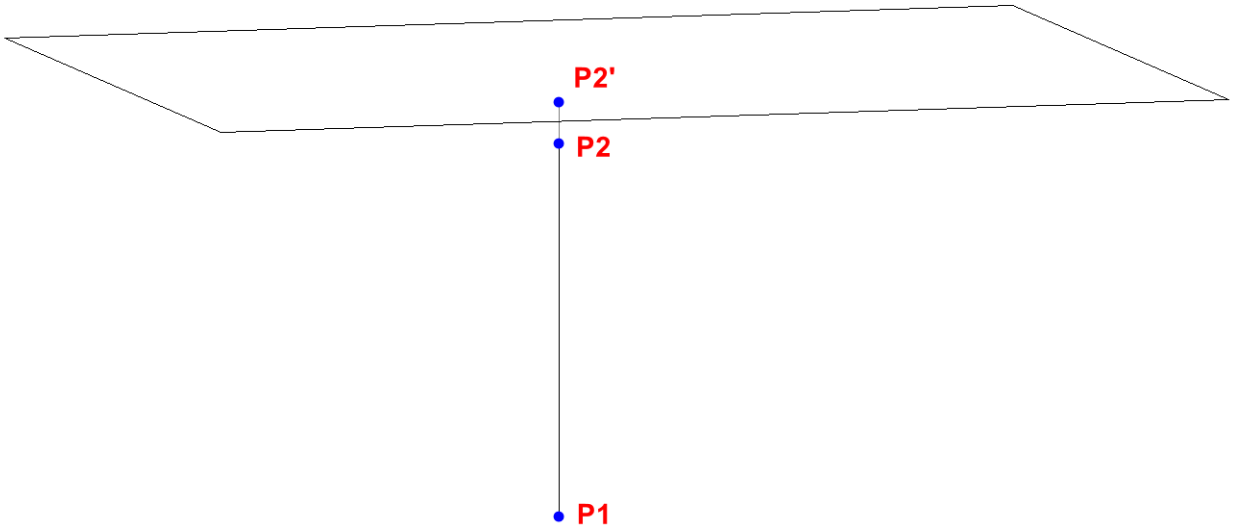


Рис. 3.36. Пластина перекриття та стержень колони

Для усунення цієї колізії, потрібно спроектувати найближчу точку стержня на площину пластини перекриття та замінити її на спроектовану.

Для того аби знайти координати спроектованої точки, скористаємось методом описаним нижче.

Маємо площину задану трьома точками A, B, C та точку P, яку потрібно спроектувати на площину ABC (рис. 3.37.).

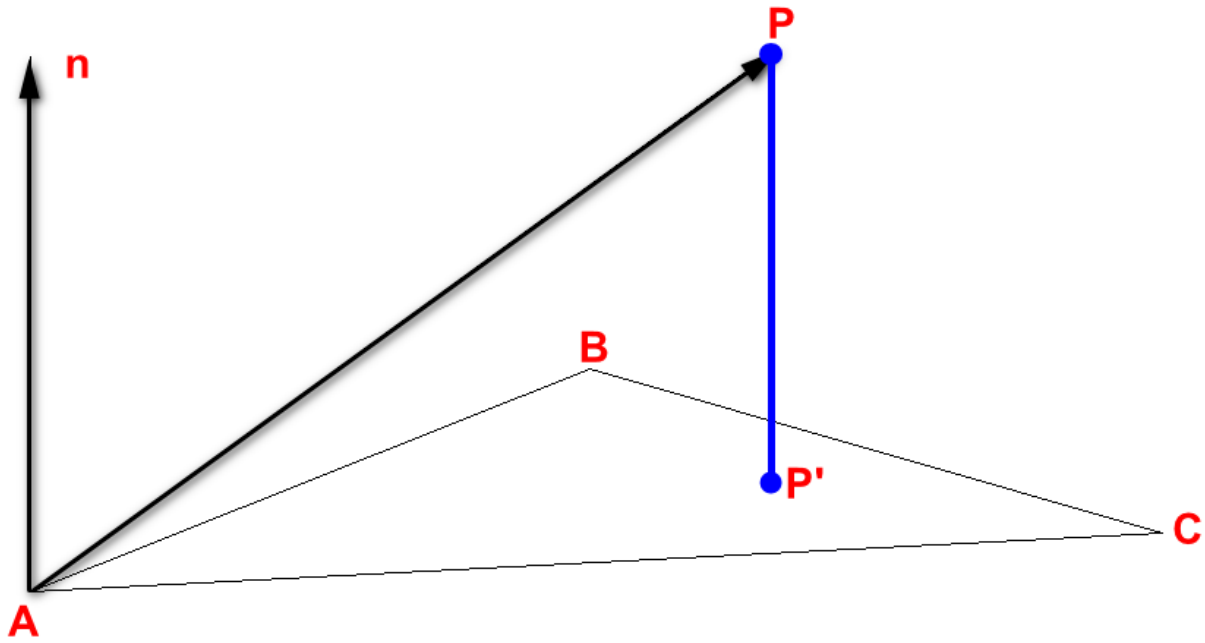


Рис. 3.37. Площина на яку необхідно спроектувати точку

Спочатку необхідно знайти вектор $\overrightarrow{AP}$:

$$\overrightarrow{AP} = (P_x - A_x;\ P_y - A_y; P_z - A_z);$$

Далі знаходимо нормаль $\vec{n}$ до площини ABC:

$$\overrightarrow{AB} = (B_x - A_x;\ B_y - A_y; B_z - A_z);$$

$$\overrightarrow{AC} = (C_x - A_x;\ C_y - A_y; C_z - A_z);$$

$$\vec{n} = \overrightarrow{AB} \times \overrightarrow{AC};$$

Знаходимо відстань від точки P до площини ABC вздовж нормалі $\vec{n}$, для цього знайдемо скалярний добуток нормалі та вектора $\overrightarrow{AP}$:

$$distance = \overrightarrow{AP} * \vec{n};$$

Для того аби знайти точку P', віднімаємо від точки P її відстань до площини ABC помножену на одиничний вектор нормалі:

$$P' = P - distance * Normalize(\vec{n})$$

Використавши описаний метод, отримаємо координати точки $P2'$. Замінивши точку $P2$ на $P2'$ отримаємо коректно дотягнутий стержень.

3.4.3. Дотягування стержнів до стержнів

Розглянемо цей метод на наступному прикладі (рис. 3.38).

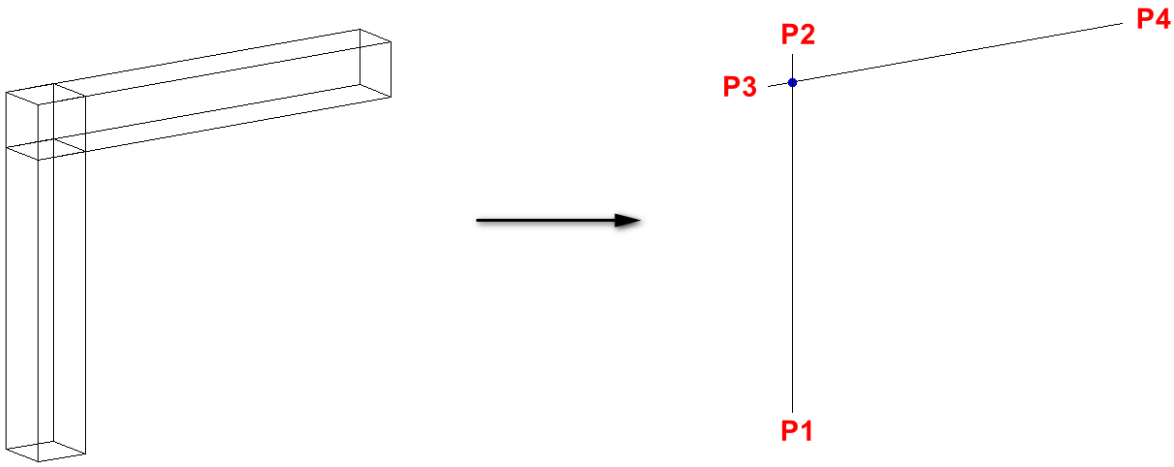


Рис. 3.38. ПСМ балки та колони

Як бачимо на рис. 3.38, під час дотягування стержня колони до стержня балки, виникає колізія. Для її усунення потрібно знайти точку перетину прямих, що задані відрізками P1P2 та P3P4. Для цього скористаємось наступним методом:

Знайдемо направляючі вектори прямих:

$l = (P2_x - P1_x,\ P2_y - P1_y,\ P2_z - P1_z);$ – напрямний вектор 1-ї прямої;

$m = (P4_x - P3_x,\ P4_y - P3_y,\ P4_z - P3_z);$ – напрямний вектор 2-ї прямої;

Складемо параметричні рівняння прямих:

$$\begin{cases} x = l_x * t1 + P1_x \\ y = l_y * t1 + P1_y \\ z = l_z * t1 + P1_z \end{cases} \quad (3.26)$$

$$\begin{cases} x = m_x * t2 + P3_x \\ y = m_y * t2 + P3_y \\ z = m_z * t2 + P3_z \end{cases} \quad (3.27)$$

Перед тим як шукати точку перетину прямих, потрібно перевірити чи лежать прямі в одній площині та чи не паралельні вони.

$$\begin{vmatrix} P3_x - P1_x & P3_y - P1_y & P3_z - P1_z \\ l_x & l_y & l_z \\ m_x & m_y & m_z \end{vmatrix} = 0 \quad (3.28)$$

Якщо умова (3.28) виконується, то прямі лежать в одній площині. Перевіримо чи не паралельні прямі.

$$\frac{l_x}{m_x} = \frac{l_y}{m_y} = \frac{l_z}{m_z} \quad (3.29)$$

Якщо не виконується умова (3.29), то прямі не паралельні. У випадку якщо одне з чисел $m_x, m_y, m_z = 0$, то в нуль перетворюємо і відповідне йому l_x, l_y, l_z.

У випадку коли умова (3.28) виконується та умова (3.29) не виконується, то прямі перетинаються. Оскільки прямі перетинаються, вони мають спільну точку, тому прирівняємо системи рівнянь (3.26) та (3.27).

$$\begin{cases} l_x * t1 + P1_x = m_x * t2 + P3_x \\ l_y * t1 + P1_y = m_y * t2 + P3_y \\ l_z * t1 + P1_z = m_z * t2 + P3_z \end{cases} \quad (3.30)$$

Маємо систему з 3-х рівнянь (3.30) та два невідомих $t1$ та $t2$. Вибравши 2 будь-яких рівняння та розв’язавши систему з цих рівнянь, знайдемо невідомі

$t1$ та $t2$. Далі підставивши знайдені значенні $t1$ та $t2$ у систему (3.26) або (3.27) знайдемо точку перетину двох прямих.

Для того аби обмежити дотяжку, введений обмежувальний радіус пошуку точки перетину, у випадку якщо точка перетину прямих знаходиться по за цим радіусом, заміна найближчої точки стержня на точку перетину не відбувається.

В результаті застосування описаного методу отримаємо коректну конструктивну модель (рис. 3.39).

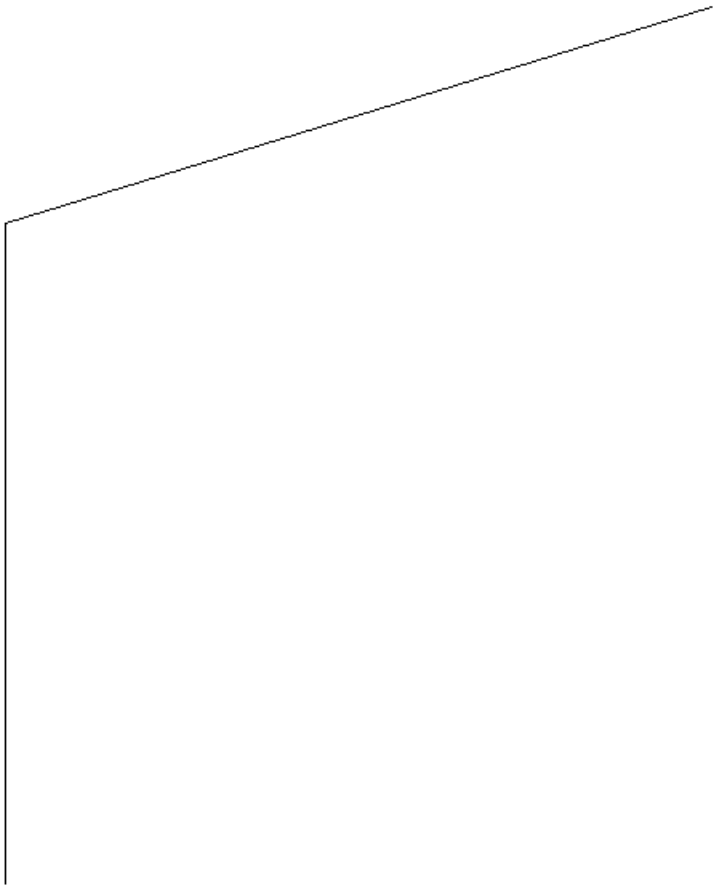


Рис. 3.39. Коректна ПСМ балки та колони

3.4.4. Дотягування пластин до стержнів

У випадку коли колона несуча, то пластина стіни повинна дотягуватись до стержня колони (рис. 3.40).

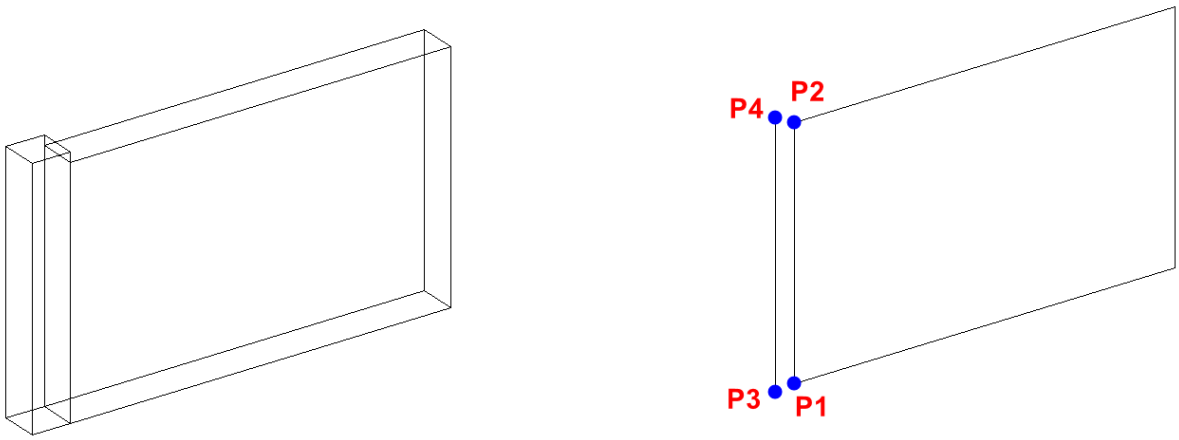


Рис. 3.40. ПСМ колони та стіни

Для того аби усунути зображену колізію та дотягнути пластину стіни до стержня колони, потрібно спроектувати найближчі точки пластини, в нашому випадку P1 та P2 на пряму задану точками P3 та P4. Для цього скористаємось наступним методом.

Нехай задана точка $M(x, y, z)$ та пряма L задана двома точками $A1$ та $A2$. Необхідно знайти проекцію точки M на пряму L.

Складемо канонічне рівняння прямої L. Оскільки пряма проходить через точки A1 та A2 то її рівняння матиме наступний вигляд:

$$\frac{x - P1_x}{m} = \frac{y - P1_y}{p} = \frac{z - P1_z}{l}$$

або (3.31)

$$\frac{x - P2_x}{m} = \frac{y - P2_y}{p} = \frac{z - P2_z}{l}, \text{ де}$$

$\vec{n} = (m, p, l) -$ напрямний вектор прямої.

Знайдемо координати направляючого вектора прямої L:

$$m = P2_x - P1_x;$$

$$p = P2_y - P1_y; \quad (3.32)$$

$$l = P2_z - P1_z$$

Складаємо рівняння площини α, яка проходить через точку $M(x, y, z)$ перпендикулярно прямій *L.* Для цього приймаємо координати направляючого вектора прямої, як відповідні координати нормального вектора площини α:

$$m(x - M_x) + p(y - M_y) + l(z - M_z) = 0$$

Відкриємо дужки:

$$mx + py + lz - mM_x - pM_y - lM_z = 0 \quad (3.33)$$

Запишемо параметричне рівняння прямої *L*:

$$\begin{cases} t = \dfrac{x - P1_x}{m} \\ t = \dfrac{y - P1_y}{p} \\ t = \dfrac{z - P1_z}{l} \end{cases}$$

$$\begin{cases} x = mt + P1_x \\ y = pt + P1_y \\ z = lt + P1_z \end{cases} \quad (3.34)$$

Підставимо значення x, y, z в (3.33).

$$m(mt + P1_x) + p(pt + P1_y) + l(lt + P1_z) - mM_x - pM_y - lM_z = 0$$

$$t = \frac{mM_x + pM_y + lM_z - mP1_x - pP1_y - lP1_z}{m^2 + p^2 + l^2} \quad (3.35)$$

Ми знайшли таке t, при якому координати x, y, z точки на прямій *L* задовольняють рівняння площини (3.33). Підставивши значення t в (3.34) знайдемо координати проекції точки $M'(x, y, z)$ на пряму *L*.

Застосувавши даний метод, отримаємо коректну конструктивну модель (рис. 3.41).

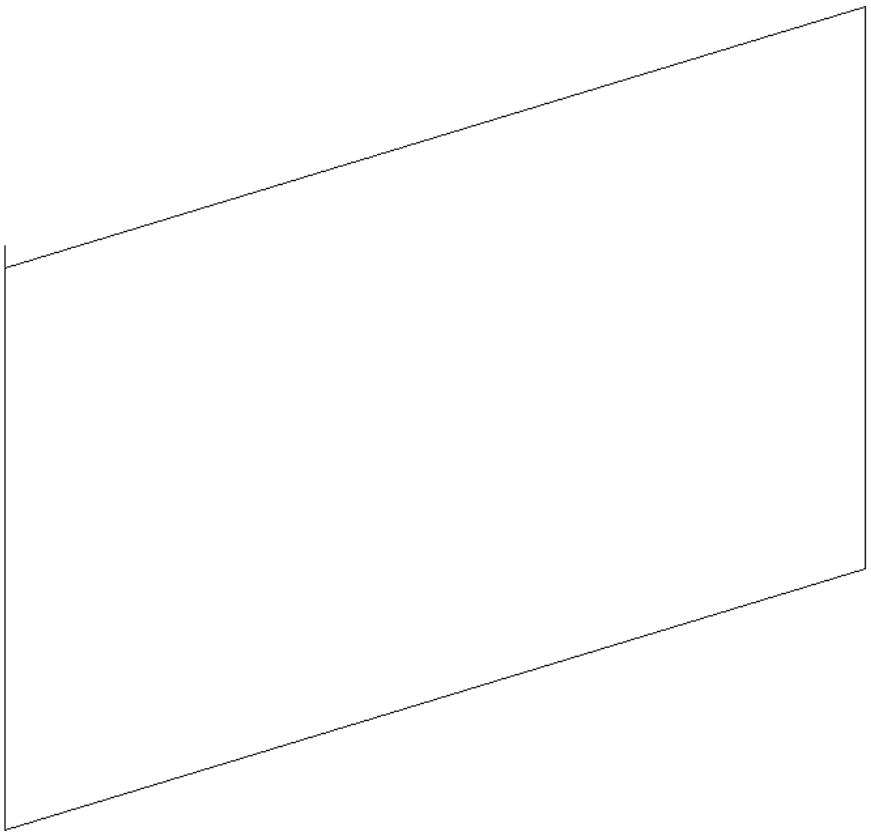


Рис. 3.41. Коректна ПСМ колони і стіни

ВИСНОВКИ ДО РОЗДІЛУ 3

1. На базі проаналізованих проблем інтеграції виокремлено поняття пластинчато-стержневої моделі (ПСМ). Проведено аналіз проблем переходу від архітектурної моделі до конструктивної.
2. Виконано побудову загальної концепції генерації ПСМ. Наведені основні концепції генерації ПСМ та відповідні їм моделі. Виявлено та виокремлено проблему пост-генерації ПСМ – усунення колізій, що виникли.
3. Досліджені та сформовані основні методи генерації ПСМ.
4. На базі аналізу колізій, що виникли, розроблений комплекс методів, які дозволяють їх усунути.

РОЗДІЛ 4

ПРАКТИЧНЕ ЗАСТОСУВАННЯ РОЗРОБЛЕНИХ МОДЕЛЕЙ І МЕТОДІВ, ДОСЛІДЖЕННЯ ЇХ ЕФЕКТИВНОСТІ. МЕТОДИ ІНТЕГРАЦІЇ З СИСТЕМАМИ УПРАВЛІННЯ ПРОЕКТАМИ

4.1. Узагальнена модель архітектури Allplan

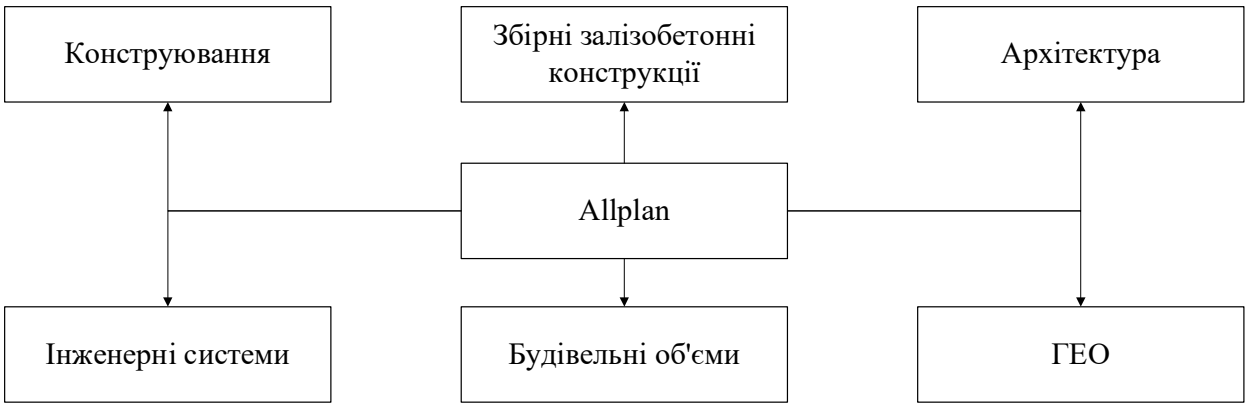


Рис. 4.1. Узагальнена модель архітектури Allplan

Allplan Архітектура – цей модуль орієнтований на створення об’єктів для проектування, в якому можна створити всі рівні проектів – від простого 2D-ескізу, 3D-моделювання, до віртуальної моделі будівлі. При цьому, користувач сам визначає, як бажає працювати. Allplan вміє перемикатися між 2D і 3D-простором, в залежності від потреби. Отже, проектувальник може в реальному часі комбінувати методи роботи, в залежності від того, як для нього буде більш зручно. Разом з тим, під час створення віртуальної моделі будівлі, виникає структура, яка використовується в подальших програмах.

Allplan Конструювання – цей модуль розроблений та оптимізований для робочих процесів інженерів-будівельників при будівництві будівель.

Особливості модуля:

- арматура "знає" свої специфічні характеристики, такі як діаметр стрижня, форма вигину і довжина анкерування;

- автоматизоване створення видів і розрізів;
- укладання арматури в один вид, уявлення у всіх;
- індивідуальне керування видимістю стрижнів: один, кілька, все;
- параметрично керована простановка позицій і розмірів;
- автоматичне керування призначенням позицій;
- автоматичне формування специфікації стрижнів, відомості деталей, відомості сталей по ДСТУ;
- оновлення всіх видів і розрізів, специфікацій, розмірів і позиційних міток, картограм розкрою сіток після змін в армуванні;
- перевірка арматури на перетин: автоматично, або візуально.

Allplan Інженерні системи – модуль для інтегрованого проектування систем опалення, вентиляції, водопостачання, водовідведення та електрики.

Особливості модуля:

- розрахунок тепловтрат і освітленості безпосередньо за архітектурним проектом;

- зручне автоматичне і ручне конструювання систем вентиляції, опалення, водопостачання, каналізації та електрики;

- автоматична генерація асоціативних видів, перспектив і розрізів;

- наочна робота на довільному вигляді;

- допомога при конструюванні, що полегшує роботу: велика база даних, інтелектуальні символи, автоматичне розпізнавання систем і т.п.;

- візуальний контроль колізій, по всіх майданчиках одночасно, в т.ч. з будівельними конструкціями;

- асоціативне або довільне надписування;

- функції зміни і копіювання;

- подання в 2D / 3D;

- інтелектуальні графічні елементи зі специфічними властивостями;
- швидке внесення змін і створення варіантів проекту завдяки централізованому зміни властивостей систем;
- легко перевіряються розрахунки, в т.ч. по СНіП;
- автоматична генерація відомостей і специфікацій по ДСТУ;
- різні варіанти специфікацій, наприклад, специфікація розкрою коробів, специфікація фасонних частин з графічним зображенням, протоколи розрахунків;
- формування аксонометричної схеми по ДСТУ.

Allplan ГЕО – модуль дозволяє планувати забудову земельних ділянок з урахуванням їх реального рельєфу, проектувати дороги і схеми озеленення, з яких можна в будь-який момент отримати дані навіть про окремі рослинах.

Особливості модуля:

- імпорт точок з тахеометра або польового журналу;
- тріангуляційна модель;
- обсяг переміщуваних мас;
- цифрова модель місцевості;
- автоматичне отримання горизонталей і розрізів;
- картограма земляних робіт по ДСТУ.

Allplan Збірні залізобетонні конструкції (Precast) – модуль для створення залізобетонних конструкцій, керуванням їх виробництвом та логістикою.

Особливості модуля:

- безпосереднє управління плоттером / лазером, роботами і верстатами з ЧПУ, роздруківку технологічних карт, або передачу даних на центральний комп’ютер управління виробництвом;

- підтримка відкритої бібліотеки типових арматурних елементів, вузлів з’єднань і заставних деталей;

- легкість внесення змін до конструкторську частину і підготовку виробництва при змінах в архітектурі;

- свободу запозичення архітектурних даних;

- наочну диспетчеризацію процесів виробництва, транспорту і монтажу.

Allplan Будівельні об’єми – модуль для видачі будівельних об’ємів та оцінки вартості.

Особливості модуля:

- інтерактивний зв’язок між кресленням і відомістю робіт;

- можливість підрахунку не пророблених детально або відсутніх на кресленні позицій;

- достовірні графічні обсяги для будь-яких кошторисних класифікаторів;

- аналіз та оцінки для підтримки прийняття рішень користувача, пов’язаних з вартістю;

- інструмент для ранньої оцінки витрат;

- перехід від конструктивних обсягів до кошторисних з технічних частин;

- оновлюваний розрахунок обсягів відомості робіт при змінах креслення;
- Опис будівництва (склад робіт);
- завдання на тендер підрядників / субпідрядників;
- каталог розцінок і фрагментів всіх країн СНД;
- асистенти для Allplan, налаштовані на фрагменти класифікаторів СНД;
- показ на кресленні розглядається кошторисного рядка;
- протокол передачі обсягів до кошторису з автоматичним контролем помилок;
- ранжування розцінок за вкладом в загальну вартість проекту;
- розрахунок обсягів за спеціальними формулами, в т.ч. контрольним**.**

Повної картини щодо використаних матеріальних ресурсів та затрачених людино-годин даний модуль нам не дає. Оскільки його можливостей недостатньо для відображення необхідних проектних даних, пропонується розробити методи для інтеграції даного модуля з відомою системою управління проектами Microsoft Project.

4.2. Інформаційна модель інтеграції Allplan та Microsoft Project

Перш за все необхідно проаналізувати Allplan та Microsoft Project, щоб виявити точки дотику в їх інформаційних моделях. За результатами дослідження було запропоновано інформаційну модель інтеграції Allplan та Microsoft Project (рис. 4.2).

Основними сутностями інформаційної моделі є:

- Mo – модель об’єкта;
- G – геометричні дані;
- T – топологічні дані;
- C – міцнісні характеристики;
- A – атрибутивні дані;
- P – параметри об’єкта;
- V – об’ємні дані;
- F – фрагменти;
- N – розцінка;
- Mp – модель об’єкта в MS Project;
- З – задача в MS Project.

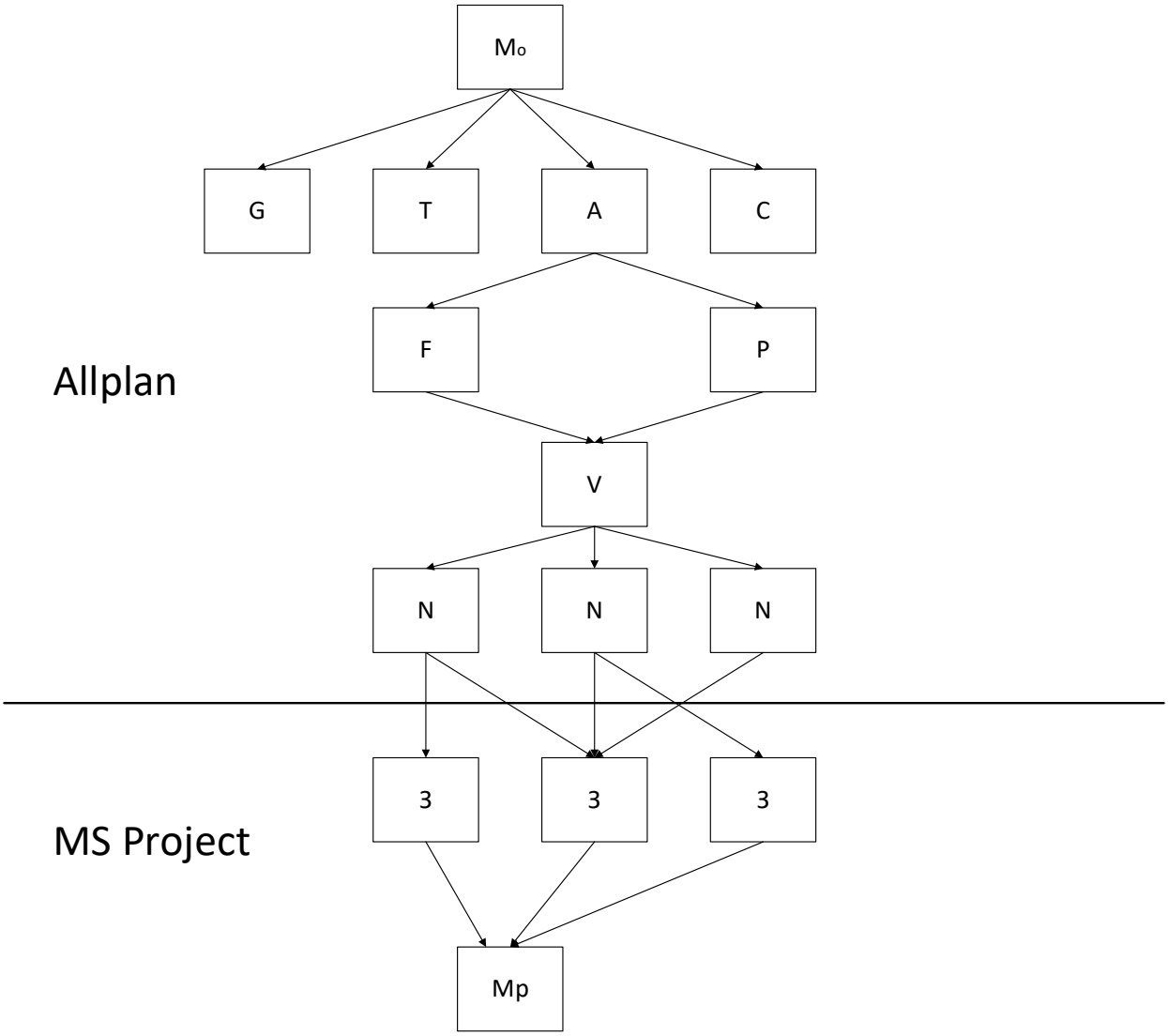


Рис. 4.2. Інформаційна модель інтеграції між Allplan та Microsoft Project

Детально дослідивши запропоновану інформаційну модель, було виділено декілька підзадач, що необхідно вирішити для реалізації автоматизованої інтеграції Allplan та Microsoft Project:

1) отримання об’ємів із Allplan;
2) зчитування даних із Microsoft Project;
3) аналіз отриманої інформації;
4) запис даних у Microsoft Project;
5) запис даних у об’єктну модель Allplan.

Розглянемо детальніше деякі пункти.

4.2.1. Отримання об’ємів з Allplan

Розрахунок об’ємів в Allplan реалізований через програмний комплекс під назвою BCM (Building Cost Management). За допомогою цього програмного комплексу користувач може згенерувати базу даних, яка міститиме в собі набір фрагментів, які, в свою чергу, містять у собі набори розцінок. Набори фрагментів формують собою «Журнали Фрагментів».

Кожен фрагмент відповідає за той чи інший тип об’єкта в САПР. Розцінка це формула з умовою, яка може містити в собі такі параметри як площа, довжина, ширина об’єкта тощо. В залежності від параметрів об’єкта, BCM знаходить відповідну розцінку у вказаному фрагменті і використовуючи формулу, яка вказана у знайденій розцінці, розраховує об’єм для цього об’єкта. Зв’язком між BCM і Allplan служить призначений атрибут «Код Фрагменту», який і вказує об’єкту, в якому фрагменті шукати розцінки. Для одного проекту в Allplan можна назначити декілька журналів фрагментів (рис. 4.3).

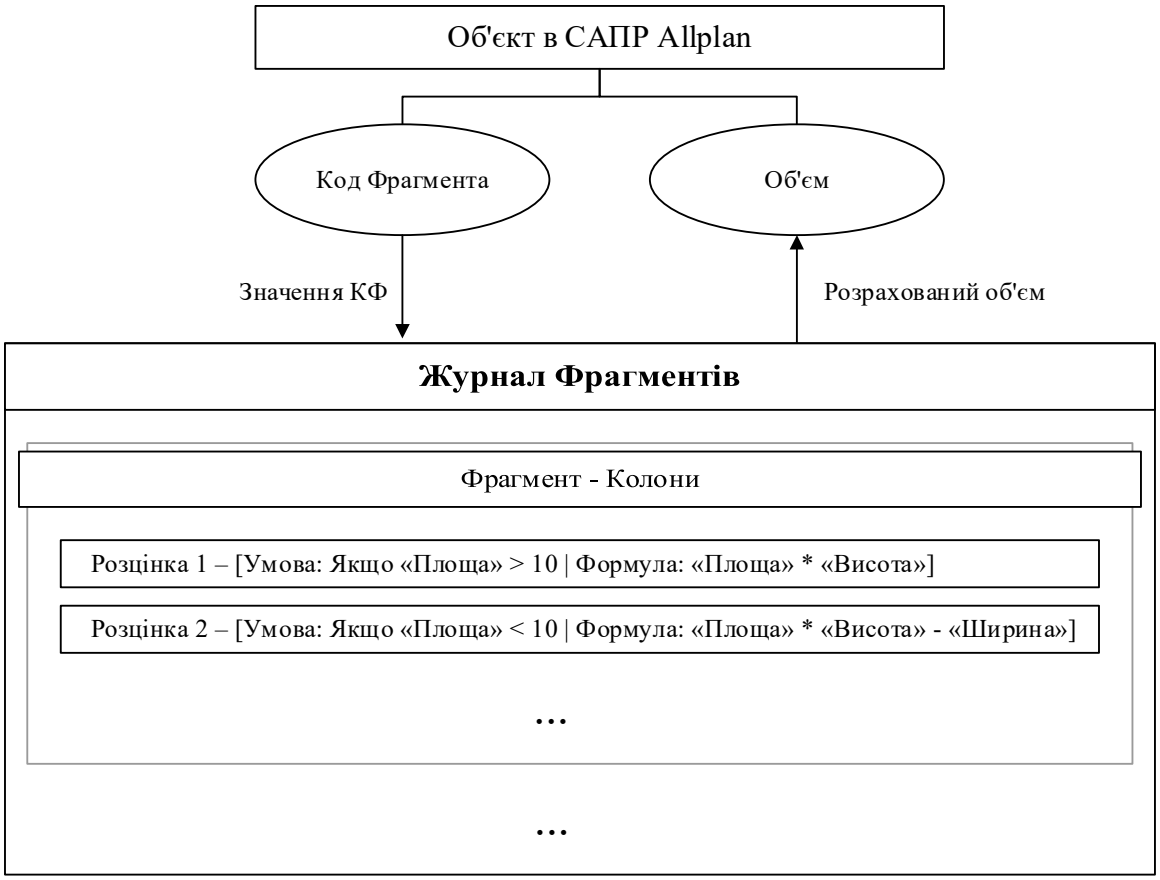


Рис. 4.3. Взаємозв’язок Allplan з BCM

4.2.2. Зчитування даних з Microsoft Project

Для зчитування та запису даних в Microsoft Project використовується стандарт COM. COM (англ. Component Object Model – об’єктна модель компонентів) - це технологічний стандарт від компанії Microsoft, призначений для створення програмного забезпечення на основі взаємодіючих компонентів, кожен з яких може використовуватися в багатьох програмах одночасно. Стандарт втілює в собі ідеї поліморфізму та інкапсуляції об’єктно-орієнтованого програмування.

Основною проблемою, яку потрібно було вирішити, була проблема автоматизованого співставлення об’ємів будівельних елементів та розцінок. Для вирішення цієї проблеми було запропоновано використати систему правил автоматичного зв’язування об’ємів та розцінок. Користувач має можливість модифікувати існуючі та створювати власні правила зв’язування з використанням запропонованої метамови.

В якості параметрів для правил пропонується використовувати наступні атрибути об’єктів та атрибути проекту Microsoft Project:

- Код фрагмента (КФ)
- Код розцінки (КР)
- Топологія Allplan (АТ)
- Тип елемента (ТЕ)
- Вид робіт (ВР)
- Топологія Microsoft Project (РТ)
- Назва задачі в Microsoft Project (TN)

Також для створення більш універсальних правил реалізована можливість використання спеціальних символів. Наприклад, будь-яке значення можна позначити через «*», або всі значення які починаються на літеру А – «А*».

Для наочності наведемо приклад правила:

IF (КФ = «КОЛОНИ» && КР = «*» && АТ = «Будівля/1 поверх» && ТЕ = «КОЛОНА» && ВР = «Бетонні роботи») → (РТ = «Вся будівля/1-П», TN = «Всі колони»)

Наведене правило вказує, що потрібно обрати всі об’єми з моделі AllPlan, у яких «Код фрагменту» дорівнює «КОЛОНИ», «Код розцінки» може бути будь-яким, топологія в Allplan – «Будівля/1 поверх», тип елементу – «КОЛОНА», вид робіт – «Бетонні роботи», та записати в задачу MS Project, яка знаходиться за топологією файлу – «Вся будівля/1-П» та має назву «Всі колони».

Маючи доступ до об’єктної моделі Allplan та даних Microsoft Project можемо побудувати схему роботи модуля (рис. 4.4).

Рис. 4.4. Схема роботи модуля

В результаті роботи модуля ми отримаємо проект Microsoft Project з заповненими об’ємами по конкретним задачам. Для зворотнього зв’язку, під час запису значень об’ємів у Microsoft Project, в кожен об’єкт Allplan записується ID задачі в Microsoft Project (рис. 4.5).

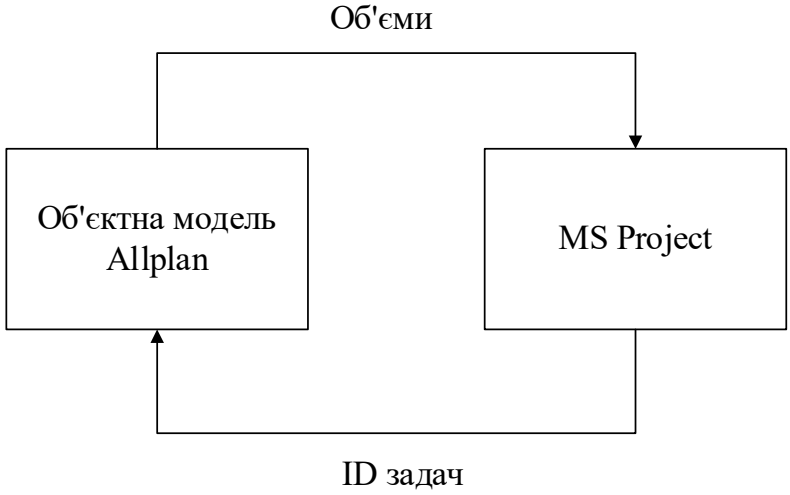


Рис. 4.5. Процес обміну даними між Allplan та Microsoft Project

Після того як користувач вивантажив об’єми з Allplan в Microsoft Project, він може розставити потрібні строки виконання, розпланувати хід робіт тощо.

Після налаштування проекту Microsoft Project потрібно передати терміни робіт та статуси задач в Allplan для того, щоб в подальшому оперувати цими даними в ньому. Саме для цієї мети використовується зв’язка по ID задачі в Microsoft Project (рис. 4.6).

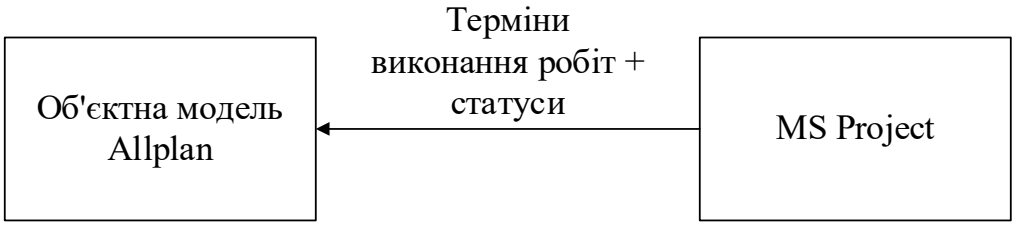


Рис. 4.6. Зворотній процес обміну даними між Allplan та Microsoft Project

В результаті передачі даних в кожен об’єкт AllPlan буде записано статус та термін виконання кожної роботи (дата завершення).

В подальшому планується доробити модуль під інші програмні комплекси для управління проектами, і у зв’язку з тим, що статуси робіт в

різних програмних комплексах можуть відрізнятися, доцільно буде уніфікувати набір статусів. Наразі в модулі використовується наступний перелік статусів:

1) робота не почата;
2) робота знаходиться в процесі виконання;
3) робота завершена.

Отримавши терміни та статуси із програми управління проектами залишається відобразити хід робіт та дати можливість користувачеві прослідкувати за виконанням робіт: побачити прострочені роботи і т. д. Для цього модуль має всі необхідні дані, а саме – дати завершення виконання робіт та статуси, збережені в шарі атрибутів кожного об’єкта.

Для гнучкого контролю за виконанням робіт модуль включає в себе 3 режими:

1) планове відображення (відображення всіх робіт, які потрапили у відповідний діапазон дат та мають статуси «робота не почата» або «робота знаходиться у процесі виконання»);

2) фактичне відображення (відображення всіх робіт, які потрапили у відповідний діапазон дат та мають статус «робота завершена»);

3) контроль (відображення всіх робіт, які потрапили у відповідний діапазон дат та мають дату завершення менше поточної). Даний режим дозволяє відобразити прострочені роботи безпосередньо на моделі в САПР Allpan.

4.3. Застосування інформаційної бібліотеки уніфікації

Розглянемо застосування бібліотеки уніфікації на наступному прикладі. Дано модель будинку, необхідно отримати специфікацію всіх стін та їх класифікацією на несучі та не несучі стіни (рис. 4.7).

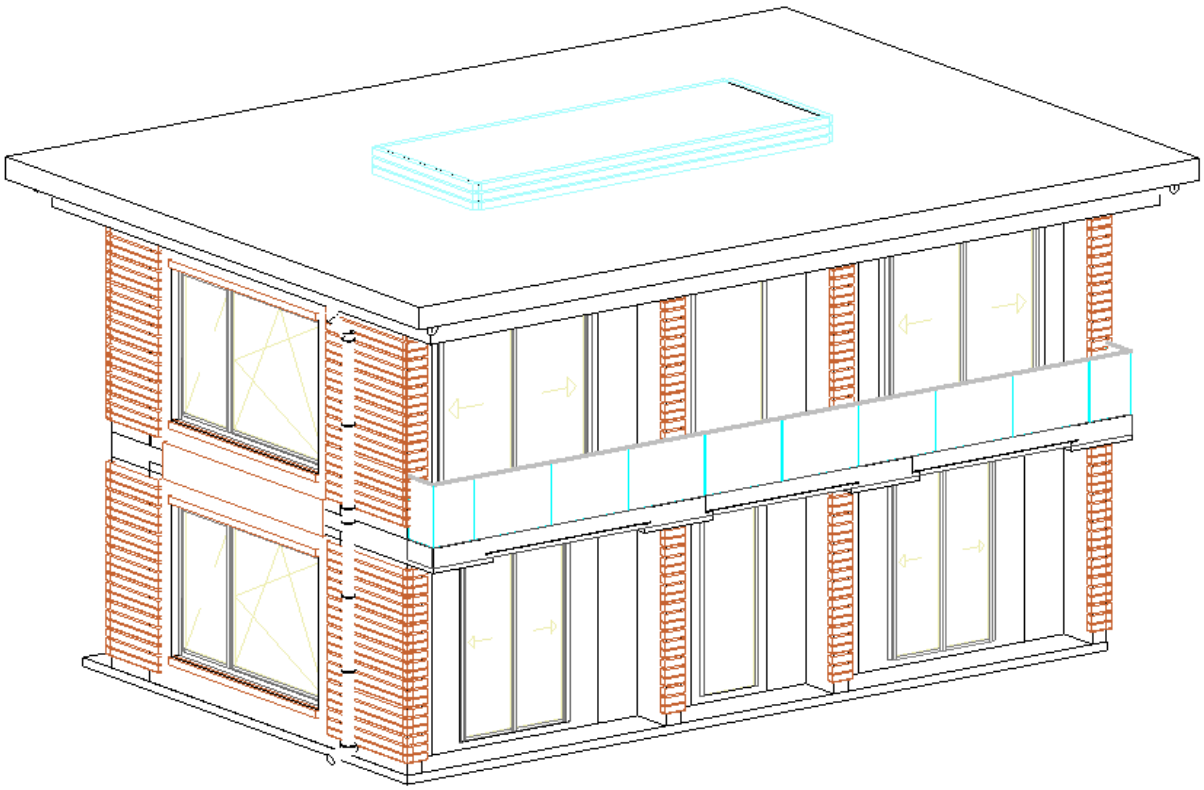


Рис. 4.7. Модель будинку у Allplan

В Allplan передбачено розподілення стін на несучі та не несучі за допомогою використання булевого атрибуту об’єкта «Несуча». Але при даному підході, не враховується ситуація, коли архітектор на етапі проектування не визначив які стіни будуть несучими і не встановив необхідний атрибут для тих чи інших стін. В такому разі доведеться переглядати всю модель і вручну задавати даний атрибут для кожної стіни. Також Allplan не має розділення на зовнішні та внутрішні стіни. Сформуємо специфікацію без використання бібліотеки уніфікації (рис. 4.8).

Імя об'єкта	Тип стіни	Об'єм
Стіна	-	0,01 м3
Стіна	-	0,03 м3
Стіна	-	0,04 м3
Стіна	Несуча	0,30 м3
Стіна	Несуча	0,45 м3
Стіна	-	0,52 м3
Стіна	Несуча	0,65 м3
Стіна	-	0,67 м3
Стіна	Несуча	0,80 м3
Стіна	Несуча	0,82 м3
Стіна	-	1,00 м3
Стіна	Несуча	1,17 м3
Стіна	Несуча	1,51 м3
Стіна	Несуча	1,52 м3
Стіна	Несуча	2,17 м3
Стіна	-	2,46 м3
Стіна	Несуча	4,29 м3
Стіна	Несуча	5,66 м3

Резюме	шт	м3
Стіна (Несуча)	11	19,34
Стіна (-)	7	4,73

Рис. 4.8. Специфікація стін у Allplan без використання бібліотеки уніфікації

Як бачимо зі специфікації, ми можемо згрупувати стіни на несучі та не несучі, лише за рахунок того, що архітектор на етапі проектування позначив несучі стіни відповідним атрибутом. Але дана специфікація не дає нам гарантії на те, що все позначені стіни є дійсно несучими, або не несучими, можливо була допущена помилка під час проектування. Також не має інформації про те, зовнішня ця стіна чи ні. Використання бібліотеки уніфікації дозволяє нам опиратись не лише на думку архітектора, а й на реальні геометричні характеристики тої чи іншої стіни. Майже в 90% ми можемо визначити тип стіни, знаючи її геометричні та фізичні характеристики, наприклад, якщо стіна складається з 3-х шарів, а її товщина 500 мм і матеріал «цегла», то ця стіна зовнішня і несуча. Сформувавши певний набір закономірностей для нашого проекту, можемо отримати наступний результат (рис. 4.9).

Імя об’єкта	Тип стіни	Об’єм
Стіна	Не несуча, Внутрішня	0,01 м3
Стіна	Не несуча, Внутрішня	0,03 м3
Стіна	Не несуча, Внутрішня	0,04 м3
Стіна	Несуча, Зовнішня	0,30 м3
Стіна	Несуча, Зовнішня	0,45 м3
Стіна	Не несуча, Внутрішня	0,52 м3
Стіна	Несуча, Зовнішня	0,65 м3
Стіна	Не несуча, Внутрішня	0,67 м3
Стіна	Несуча, Внутрішня	0,80 м3
Стіна	Несуча, Внутрішня	0,82 м3
Стіна	Не несуча, Внутрішня	1,00 м3
Стіна	Несуча, Зовнішня	1,17 м3
Стіна	Несуча, Зовнішня	1,51 м3
Стіна	Несуча, Зовнішня	1,52 м3
Стіна	Несуча, Зовнішня	2,17 м3
Стіна	Не несуча, Внутрішня	2,46 м3
Стіна	Несуча, Зовнішня	4,29 м3
Стіна	Несуча, Зовнішня	5,66 м3

Резюме	шт	м3
Стіна (Несуча)	11	19,34
Стіна (Не несуча)	7	4,73
Стіна (Внутрішня)	9	6,35
Стіна (Зовнішня)	9	17,72
Стіна (Несуча, Зовнішня)	9	17,72
Стіна (Не несуча, Зовнішня)	0	0
Стіна (Несуча, Внутрішня)	2	1,62
Стіна (Не несуча, Внутрішня)	7	4,73

Рис. 4.9. Специфікація стін у Allplan з використанням бібліотеки уніфікації

Результуюча специфікація є більш формалізованою і дозволяє нам виділити окремі групи стін та отримати дані по ним.

Найбільш корисною бібліотека уніфікації є на етапі обробки набору даних тим чи іншим модулем, адже її використання дозволяє одразу формалізувати вхідні дані та відкинути зайві, що значно впливає на швидкість роботи модулів, особливо на великих об’ємах даних.

4.4. Створення ПСМ та усунення колізій

Створення пластинчато-стержневої моделі (ПСМ) дозволяє нам вирішити проблему передачі архітектурної моделі конструктору за рахунок автоматизації створення конструктивної моделі на базі архітектурної. У випадку, якщо не використовувати генерацію ПСМ, то конструктивну модель доведеться формувати вручну, що займе дуже багато часу у конструктора.

Розглянемо генерацію ПСМ на базі наступного фрагмента будівлі (рис. 4.10)

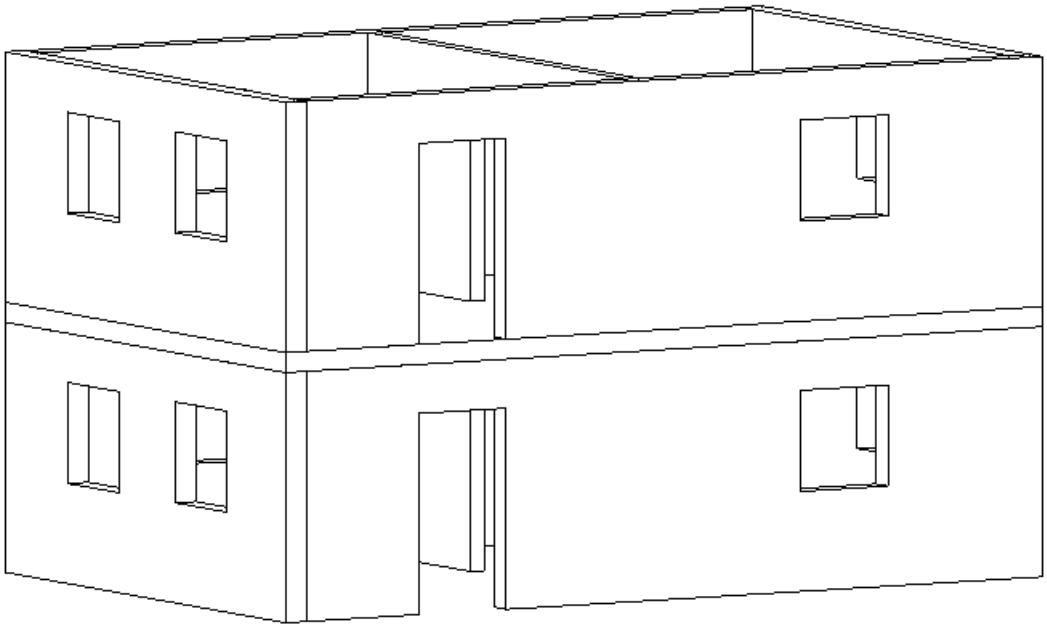


Рис. 4.10. Фрагмент будівлі у Allplan

Застосувавши методи генерації описані у розділі 3, отримаємо в результаті наступну конструктивну модель (рис. 4.11).

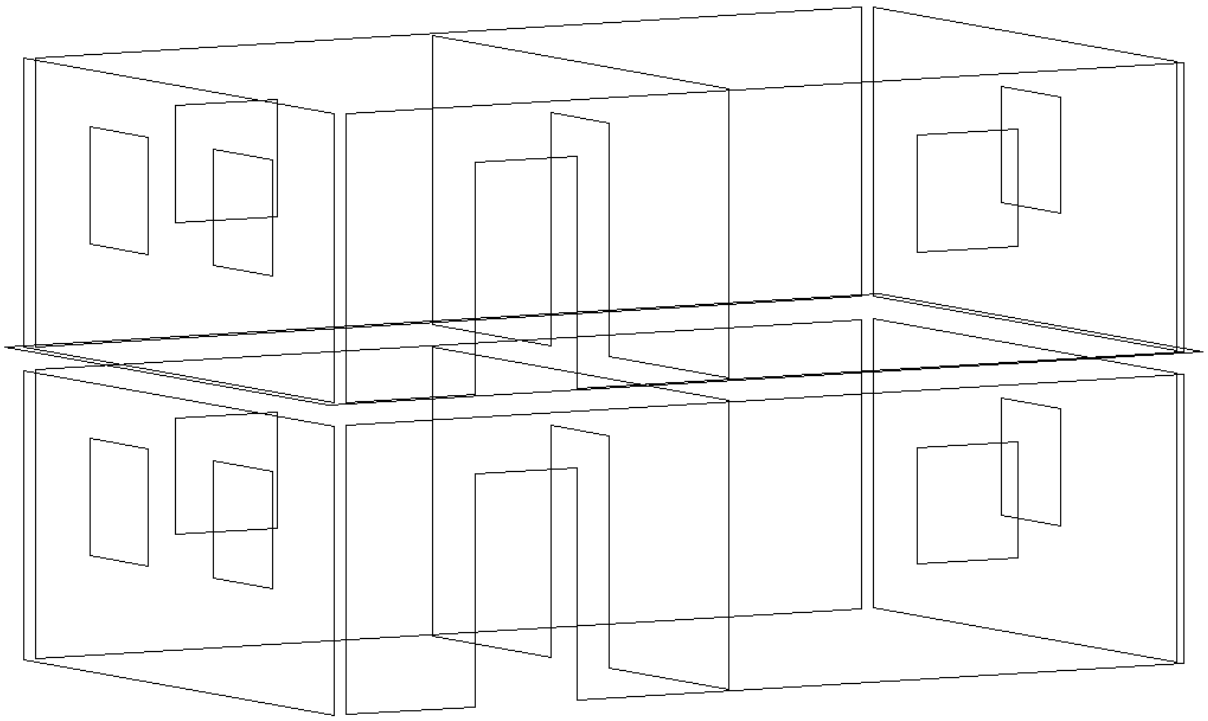
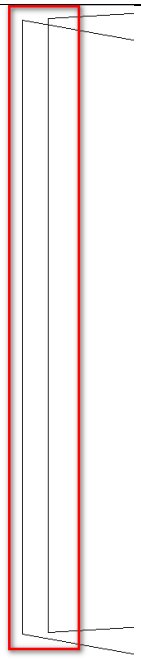
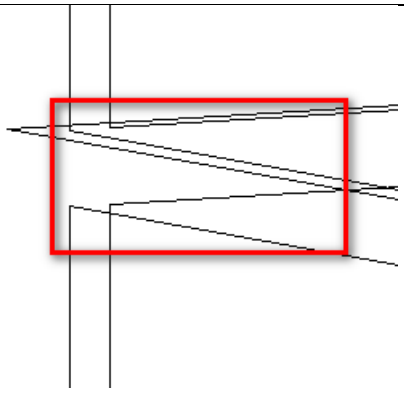
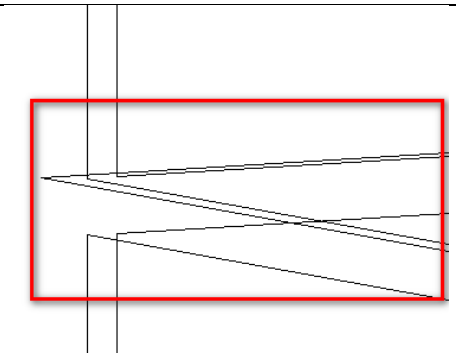


Рис. 4.11. Конструктивна модель без використання методів дотягування

Як бачимо з рисунка, простої генерації ПСМ недостатньо, адже виникає велика кількість колізій. Виділимо типові колізії та відобразимо їх у вигляді таблиці 4.1.

Таблиця 4.1

Колізії, що виникли в результаті генерації ПСМ

Опис колізії	Колізія
Стіна не дотягнулась до стіни	
Стіна не дотягнулась до перекриття	
Перекриття не дотягнулось до стін	

Застосувавши методи дотягувань, описані в розділі 3, отримаємо наступну пластинчато-стержневу модель (рис. 4.12)

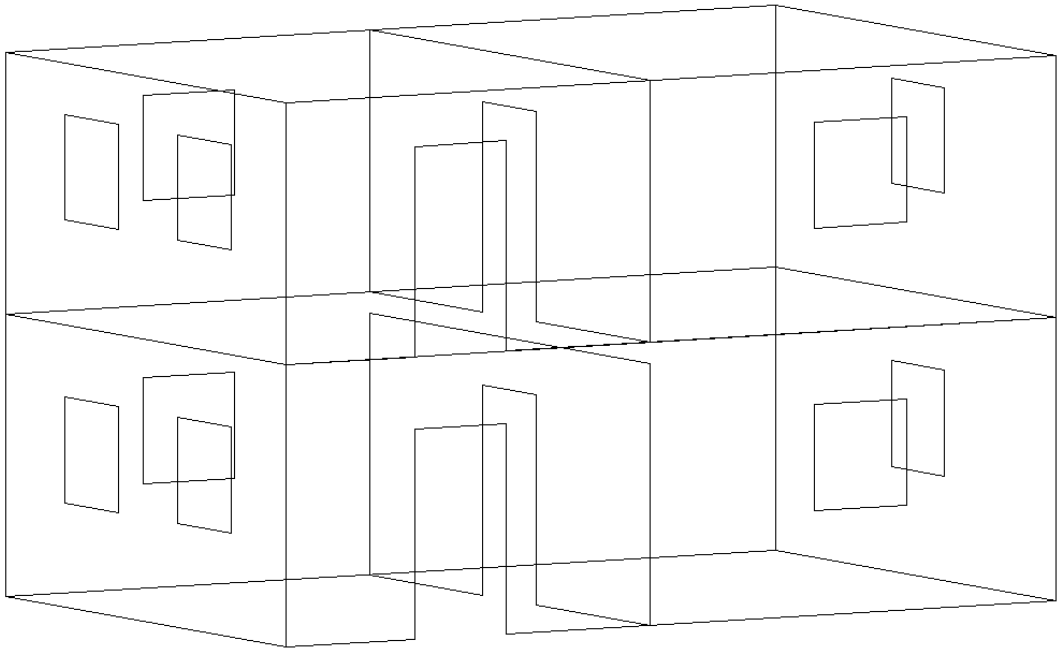


Рис. 4.12. Конструктивна модель з використанням методів дотягування

Як бачимо, недотягнута залишилась лише стіна у центрі (рис. 4.13)

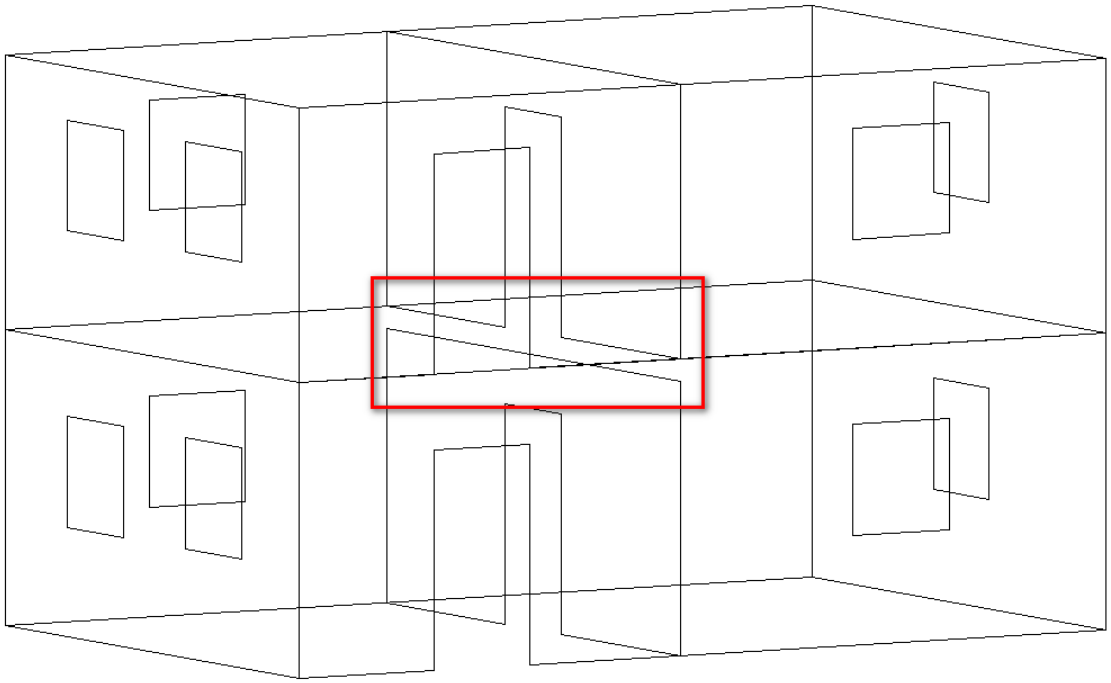


Рис. 4.13. Колізія на конструктивній моделі будівлі

Але ця недотяжка залишилась тому, що центральні стіни є перегородками, а пластини перегородки на конструктивній моделі не повинна дотягуватись до пластини перекриття. Перевірка стіни, на те чи є вона перегородкою, досягається за рахунок використання розробленої бібліотеки уніфікації. Результуючу конструктивну модель можна вважати коректною та експортувати у спеціалізовані розрахункові комплекси.

ВИСНОВКИ ДО РОЗДІЛУ 4

1. Досліджена узагальнена модель архітектури інформаційної системи автоматизованого проектування Allplan. Окремо розглянуті її модулі, наведені їх характеристики. Виокремлена проблема недостатньої деталізації процесу календарно-об’ємного планування. На базі цього запропоновані методи для інтеграції Allplan та комплексу для управління проектами MS Project.
2. Запропонована інформаційна модель інтеграції Allplan та MS Project. Наведені основні схеми та методи інтеграції.
3. Проведено аналіз результатів застосування інформаційної бібліотеки уніфікації на прикладі типової архітектурної моделі.
4. Проведено аналіз результатів використання моделей та методів генерації ПСМ на типовій архітектурній моделі у Allplan.

ЗАГАЛЬНІ ВИСНОВКИ

В дисертаційній роботі вирішено актуальну науково-технічну задачу розробки методів та моделей для автоматизації процесу моделювання об’єктів в інформаційних системах автоматизації життєвого циклу БО. В результаті виконаної роботи було розроблено модель та методи створення інформаційної бібліотеки уніфікації будівельних об’єктів, модель та методи створення пластинчато-стержневої моделі (ПСМ), методи усунення колізій на створеній ПСМ та модель і методи для передачі календарно-планувальної інформації у системи управління проектами. Всі створені методи та моделі були використані під час розробки різноманітних модулів для інформаційної системи Allplan, що дозволило збільшити кількісні та якісні показники ефективності проектувальних робіт.

У дисертації одержані такі основні теоретичні та практичні результати:

1. Досліджені методи інтеграції комп’ютерних застосунків. Виявлені та досліджені основні проблеми інтеграції спеціалізованих комплексів автоматизованого проектування. Проаналізовані проблеми існуючих форматів даних, які використовуються для обміну даними між інформаційними системами автоматизованого проектування.
2. Проведено аналіз основних етапів проектування будівель та споруд. Удосконалено розширену модель життєвого циклу будівельного об’єкта, виділено проблему переходу від архітектурної моделі до конструктивної.
3. Досліджені основні CAD-системи, що використовуються на основних етапах процесу проектування будівель та споруд, проведено їх порівняння. На базі аналізу існуючих CAD-систем виокремлені наступні: Allplan, ArchiCAD та Revit, як найбільш перспективні.
4. Досліджені проблеми уніфікації будівельних об’єктів. Розроблені моделі та методи створення бібліотеки уніфікації, як загального

інтерфейсу для інших модулів, що з практичної точки зору значно покращує якісні характеристики всіх інших модулів інформаційної системи автоматизованого проектування. Практична реалізація та впровадження запропоновані на прикладі інформаційної системи Allplan.

5. Проведено аналіз проблеми переходу від архітектурної моделі до конструктивної. Вперше запропонована проміжна модель між архітектурною та конструктивною – пластинчато-стержнева модель (ПСМ). Розроблено методи для генерації ПСМ.
6. Проведено аналіз всіх можливих колізій після створення ПСМ. Розроблено комплекс методів для усунення колізій на вже створеній ПСМ.
7. Досліджена проблема передачі календарно-планувальних даних із інформаційної системи Allplan у комплекси управління проектами на прикладі Microsoft Project. Розроблені моделі та методи, які дозволяють автоматизувати передачу календарно-планувальних даних із Allplan у Microsoft Project.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДБН Д.1.1-1-2000. Наказ Про затвердження Правил визначення вартості будівництва.
2. А. Валиков. Технология XSLT. – СПб: БХВ – Петербург, 2002. – 274 с.
3. Державний класифікатор будівель та споруд ДК 018–2000.
4. Артемьев В.И. Разработка САПР. В 10 кн. Кн. 5. Организация диалога в САПР / В.И. Артемьев, В.Ю. Строганов – М.: Высшая школа, 1990 – 348 с.
5. Норенков И.П. Основы автоматизированного проектирования: Учебник для вузов. 2-е издание, перераб. и доп. – М.: Изд. МГТУ им. Н.Э. Баумана, 2002. – 336 с.
6. Архангельский А.Я. Решение типовых задач в C++ Builder 6. – М.: ЗАО «Издательство БИНОМ», 2003. – 426 с.
7. Архангельский А.Я. Стандартная библиотека STL языка C++ / А.Я. Архангельский, М.А. Тагин – М.: ЗАО «Издательство БИНОМ», 2002.– 486 с.
8. Архангельский А.Я. Стандартная библиотека функций C / А.Я. Архангельский, М.А. Тагин – М.: ЗАО «Издательство БИНОМ», 2002.– 348 с.
9. Ахо А. Теория синтаксического анализа, перевода и компиляции / А. Ахо, Дж. Ульман – М.: Мир, 1978. – Т.1 – 614 с., Т.2 – 488 с.
10. Барабаш М.С. Нова концепція автоматизації проектування об’єктів будівництва на основі цифрової моделі / М.С. Барабаш, С.Д. Коба // Будівництво України. – 2004. – №5. – С. 25–30.
11. Компьютерное моделирование процессов жизненного цикла объектов строительства Барабаш
12. Інформационные технологии интеграции на основе програмного комплекса САПФИР

13. Бекаревич Ю.Б. Технология разработки баз данных от проектирования до создания приложений / Ю.Б. Бекаревич, Н.В. Пушкина – СПб.: БХВ-Петербург, 2001. – 512 с.
14. Боггс У. UML и Rational Rose / У. Боггс, М. Боггс: Пер. с англ. – М.: ЛОРИ, 2000. – 582 с.
15. Бородавка Є. В. Логічна і фізична організація структури даних цифрової моделі об'єкта. Концептуальна модель ЦМО / Є. В. Бородавка // Східноєвропейський журнал передових технологій. – 2006. – №3/3(21). – С. 47–49.
16. Бородавка Є. В. Цифрова модель об'єкта як засіб інтеграції архітектурно-будівельних програмних комплексів / Є. В. Бородавка // Східноєвропейський журнал передових технологій. – 2006. – №2/2(20). – С. 1–4.
17. Бьерн Страуструп. Язык программирования C++. Специальное издание. – М.: БИНОМ, 2006 – 624 с.
18. Вейнеров О.М. Разработка САПР. В 10 кн. Кн. 4. Проектирование баз данных САПР / О.М. Вейнеров, Э.Н. Самохвалов – М.: Высшая школа, 1990. – 478 с.
19. Вендров А.М. Один из подходов к выбору средств проектирования баз данных и приложений. – М.: «СУБД», 1995. – 348 с.
20. Вин Зо. Моделирование процессов интеграции информационных систем. 05.13.11 // Московский инженерно-физический институт. – М., 2007.– 24 с.
21. Воробьева М.С. Математическое моделирование и технологии интеграции данных в учетных информационных системах. 05.13.18 // Тюменский государственный университет. – Т., 2006. – 20 с.
22. Гамма Э. Приемы объектно-ориентированного проектирования / Э. Гамма, Р. Хелм, Р. Джонсон – Питер, 2001 – 368 с.
23. Гилой В. Интерактивная машинная графика. – М.: Мир, 1982. – 196 с.

24. Горев А. Эффективная работа с СУБД / А. Горев, Р. Ахаян, С. Макашарипов – СПб.: Издательский дом «ПИТЕР», 1997. – 446 с.
25. Городецький О. С. Засоби підтримки процесу проектування будівель і споруд з використанням уніфікованої цифрової моделі об'єкта / О. С. Городецький, Є. В. Бородавка // Будівництво України. – 2007. – №4. – С. 36–39.
26. ГОСТ 34.003–90. Информационная технология. Автоматизированные системы. Термины и определения // Информационная технология. Комплекс стандартов и руководящих документов на автоматизированные системы. М.: Комитет стандартизации и метрологии СССР, 1991. – 144с.
27. Гради Буч. Объектно-ориентированный анализ и проектирование с примерами приложений на С++. 2-е изд. М. «Издательство Бином». 1999. – 560 с.
28. Грейди Буч. UML. Руководство пользователя / Грейди Буч, Джеймс Рамбо, Айвар Джекобсон – М. ДМК 2000. – 432 с.
29. Д. Э. Федотова. CASE-технологии. Практикум / Д. Э. Федотова, Ю. Д. Семенов, К. Н. Чижик – М.: Горячая линия – Телеком, 2005. – 468 с.
30. Дейт К.Дж. Введение в системы баз данных.: Пер. с англ. – 6-е изд. – К.: Диалектика, 1998. – 784 с.
31. Демченко В.В. Функціональна модель графічних застосувань на основі OpenGL / В.В. Демченко, В.О. Анпілогова – Збірка праць міжнародної науково-практичної конференції «Сучасні проблеми геометричного моделювання». – Харків, 2001. – с.194–196.
32. Демченко В. В. Формальний опис і практичне використання уніфікованої цифрової моделі об'єкта будівництва / В. В. Демченко, Є. В. Бородавка // Східноєвропейський журнал передових технологій. – 2007. – №2/2(26). – С. 64–69.
33. Демченко В. В. Комплексна цифрова модель об'єктів будівництва / В. В. Демченко, Є. В. Бородавка // Матеріали VIII Міжнар. наук.-тех. конф. «АВІА-2007». – К. : НАУ, 2007. – Т. 3. – С. 42.38–42.40.

34. Демченко В. Самоучитель ArchiCAD 8 / В. Демченко, А. Михайленко, Е. Бородавка – СПб. : Питер; К.: Издательская группа BHV, 2005. – 432 с.
35. Демченко В. Самоучитель ArchiCAD 9 / В. Демченко, А. Михайленко, Е. Бородавка – СПб.: Питер; К.: Издательская группа BHV, 2006. – 448 с.
36. Дмитриев Л.Г. Системы автоматизированного проектирования объектов гражданского строительства / Л.Г. Дмитриев, Н.Г. Лихогруд, В.В. Штабовенко – К.: Будівельник, 1988. – 190 с.
37. Здоренко В.С. Технологическая линия проектирования конструкций многоэтажных зданий (ТЛП КАЛИПСО). / В.С. Здоренко, М.А. Захаров, В.И. Сикорская // Система автоматизированного проектирования объектов строительства: Сборник трудов вип.. 4 – К., Будівельник., 1987. – С 41–48.
38. Зиндер Е.З. Бизнес-реинжиниринг и технологии системного проектирования. Учебное пособие. – М.: Центр Информационных Технологий, 1996. – 188 с.
39. Иванов В.П. Трехмерная компьютерная графика / В.П. Иванов, А.С. Батраков – М.: Радио и связь, 1994. – 424 с.
40. Калиниченко Д.А. Методы и средства интеграции неоднородных баз данных. – И.: Наука, 1983. – 424 с.
41. Калиниченко Л.А. Машины баз данных и знаний / Л.А. Калиниченко, В.М. Рывкин – М.: Наука. Гл. ред. физ-мат. лит., 1990. – 296 с.
42. Калянов Г.Н. CASE. Структурный системный анализ (автоматизация и применение). – М.: ЛОРИ, 1996 – 268 с.
43. Климов В.Е. Разработка САПР. В 10 кн. Кн. 7. Графические системы САПР. – М.: Высшая школа, 1990. – 348 с.
44. Когаловский М. Энциклопедия технологий баз данных. – М.: ЛОРИ, 2003. – 800 с.
45. Козачевский А.И. Проблемы кадрового обеспечения автоматизированного проектирования / А.И. Козачевский, В.В. Демченко // Конструкции гражданских зданий: Сборник научных трудов. – К.: КиевЗНИИЭП. – 2003. – С. 156–160.

46. Конолли Т. Базы данных: проектирование, реализация и сопровождение. – М.: Диалектика, 2000. – 1120 с.
47. Кораблев В. С и С++. – СПб.: Питер; Киев: Издательская группа BHV, 2002 – 432 с.
48. Крамаренко В.В. Організація баз даних та знань: Навчальний посібник. ДДТУ. – Д.: Системні технології, 2000. – 214с.
49. Красковский Д. PLM/AECO – новая панацея // САПР и графика. – 2003. – №2. – С. 4–9.
50. Краснов М.В. OpenGL. Графика в проектах Delphi. – СПб.: БХВ, 2000. – 352 с.
51. Крис Паппас. Программирование на С и С++ / Крис Паппас, Уильям Мюррей – СПб.: Питер; Киев: Издательская группа BHV, 2000. – 320 с.
52. Крэг Ларман. Применение UML и шаблонов проектирования. – Вильямс, 2001. – 496 с.
53. Кэнту М. Delphi 4 для професионалов – СПб.: издательство «Питер», 1999. – 1120 с.
54. Лелюк В.А. Концептуальное проектирование систем с базами знаний – Х.: Изд-во «Основа» при Харьк. Ун-те, 1990. – 144 с.
55. Ли К. Основы САПР (CAD/CAM/CAE). – СПб.: Питер, 2004. – 560 с.
56. Литвин П. Разработка настольных приложений в Access 2002. Для профессионалов / П. Литвин, К. Гетц, М. Гунделой – СПб.: Питер; Киев: Издательская группа BHV, 2002. – 1008 с.
57. Лященко А.А. Структура и функции создания цифровой модели объекта в интеллектуальном проектирующем комплексе «МОНОМАХ» для автоматизированного проектирования железобетонных конструкций / А.А. Лященко, Д.А. Городецький // Коммунальное хозяйство городов: Науч.-техн. Сб. ХГАГХ. – Х.:Техніка, 1998. – Вып.14. – С. 33–39.
58. М. Фаулер. UML в кратком изложении / М. Фаулер, К.Скотт – М. Мир. 1999. – 191 с.

59. Майкл Кей. XSLT. Справочник программиста. – СПб.: «Символ-Плюс», 2002. – 364 с.
60. Майкл, Ласло. Вычислительная геометрия и компьютерная графика на C++. – М.: Бином, 1997. – 364 с.
61. Малыхина М.П. Базы данных: основы, проектирование, использование. – – СПб.: BHV-СПб, 2004. – 512 с.
62. Марка Д.А. Методология структурного анализа и проектирования / Д.А. Марка, К. МакГоуэн – М.: «МетаТехнология», 1993. – 486 с.
63. Международные стандарты, поддерживающие жизненный цикл программных средств. – М.: МП «Экономика», 1996. – 344 с.
64. Михайленко В.Е. Геометрическое моделирование и машинная графика в САПР / В.Е. Михайленко, В.Н. Кислоокий, А.А. Лященко: Учебник – К.: Вища школа, 1991. – 374 с.
65. Мюллер Р. Базы данных и UML Проектирование. – М.: ЛОРИ, 2003. – 432 с.
66. Наумов А.Н. Системы управления базами данных и знаний / А.Н. Наумов, А.М. Вендров, В.К. Иванов: Справ. изд. // Под ред. А.Н. Наумова. – М.: Финансы и статистика; 1991. – 352 с.
67. Нестеров Ю.Г. Разработка САПР. В 10 кн. Кн. 6. Выбор состава программно-технического комплекса САПР / Ю.Г. Нестеров, И.С. Папшев – М.: Высшая школа, 1990. – 234 с.
68. Новоженов Ю.В. Объектно-ориентированные технологии разработки сложных программных систем. – М.: 1996. – 268 с.
69. Норенков И.П. Информационная поддержка наукоемких изделий. CALS-технологии / И.П. Норенков, П.К. Кузьмик – Издательство МГТУ им. Н. Э. Баумана, 2002. – 320 с.
70. Ньюмен У. Основы интерактивной графики / У. Ньюмен, Р. Спрулл – М.: Мир, 1985. – 326 с.
71. Павлидис У. Алгоритмы машинной графики и обработка изображений. – М.: Радио и связь, 1988. – 196 с.

72. Петров А.В. Разработка САПР. В 10 кн. Кн. 1. Проблемы и принципы создания САПР / А.В. Петров, В.М. Черненький – М.: Высшая школа. – 1990. – 468 с.
73. Райордан Р. Основы реляционных баз данных. Пер. с англ. – М.: Издательско-торговый дом «Русская Редакция» – 2001. – 384 с.
74. Рихтер Джефри. WINDOWS для профессионалов, Microsoft, 1998. – 680с.
75. Роберт Сигнэр. Использование ODBC для доступа к базам даннях / Роберт Сигнэр, Михаэль О. Стегион – Binom Publishers, 1995. – 568 с.
76. Роджерс Д. Математические основы машинной графики / Д. Роджерс, Дж. Адамс – М.: Машиностроение, 1980. – 274 с.
77. Ролланд Фред. Основные концепции баз данных. – М.: Издательский дом «Вильямс», 2002. – 256 с.
78. Романов В. Н. Системный анализ для инженеров. – СПб: СЗГЗТУ – 2006. – 186 с.
79. Рубин А.Г. Пользовательский интерфейс для прикладных задач / А.Г. Рубин, В.К. Смирнов, В.П. Тульский. – М.:2000. – 29 с. (Препринт // Ин-т прикладной математики им. М.В. Келдыша РАН, №74)
80. С. А. Трофимов. CASE-технологии. Практическая работа в Rational Rose. – М.: Бином-Пресс, 2002. – 248 с.
81. Системы автоматизированного проектирования: Учеб. пособие для втузов // Под ред. И. П. Норенкова, в 9-ти кн. Трудоношин В. А., Пивоварова Н. В. Математические модели технических объектов. – М.: Высшая школа, 1986. – Кн. 4. – 160 с.
82. Системы автоматизированного проектирования: Учеб. пособие для втузов // Под ред. И. П. Норенкова, в 9-ти кн. Кузьмик П. К., Маничев В. Б. Автоматизация функционального проектирования. – М.: Высшая школа, 1986. – Кн. 5. – 144 с.
83. Скляров В.А. Автоматизация проектирования ЭВМ. Учеб. пособие для вузов. – М.: Высшая школа. 1990. – 468 с.

84. Смирнов С.В. Онтологический анализ: определения и алгоритмы // Проблемы управления и моделирования в сложных системах. Труды III международной конференции. Самара: СНЦ РАН, 2001. – С. 24–28.
85. Смирнов Н.Н. Программные средства персональных ЭВМ. – Л.: Машиностроение. Ленингр. отд-ние, 1990. – 272с.
86. Стивенс Р. Программирование баз данных. – М.: Бином, 2003. – 384 с.
87. Терри Катрани. Визуальное моделирование с помощью Rational Rose 2002 и UML. – М.: Вильямс, 2003. – 272 с.
88. Тимоти Бадд. Объектно-ориентированное программирование в действии. – Питер, 1997. – 634 с.
89. Тихомиров Ю. Программирование трехмерной графики. – СПб.: BHV, 1998. – 298 с.
90. Тодорой Д. Н. Расширяемые средства машинной графики. – Москва: «Радио и связь», 1983. – 208 с.
91. Т.Я. Лазарева. Интегрированные системы проектирования и управления. Структура и состав / Т.Я. Лазарева, Ю.Ф. Мартемьянов, А.Г. Схиртладзе – М.: «Издательство машиностроение-1» , 2006. – 88 с
92. Уэнди Боггс. UML и Rational Rose / Уэнди Боггс, Майкл Боггс – М.: ЛОРИ, 2000. – 342 с.
93. Федоров Б.С. Разработка САПР. В 10 кн. Кн. 3. Проектирование программного обеспечения САПР / Б.С. Федоров, Н.Б. Гуляев – М.: Высшая школа, 1990. – 284 с.
94. Филипп Крачтен. Введение в Rational Unified Process. – М.: Вильямс, 2002. – 474 с.
95. Бородавка Є.В. Квасневський В.М. Методи сортування геометричних об’єктів та їх реалізація на прикладі плагіна автонумерації для САПР Allplan/ Є.В. Бородавка, В.М. Квасневський// Управління розвитком складних систем. – 2014. № 18. – С. 128 - 132.

96. Бородавка Є.В. Квасневський В.М. Геометричні методи побудови отворів та гільз для інженерних мереж в САПР Allplan / В.М. Квасневський, Є.В. Бородавка // Управління розвитком складних систем. – 2015. № 22.
97. Бородавка Є.В. Квасневський В.М. Методи побудови об'єктів в комп'ютерній графіці / Є.В. Бородавка, В.М. Квасневський // Управління розвитком складних систем. – 2015. № 23.
98. Квасневський В.М. Базові принципи побудови УРМ (Універсальної розрахункової моделі). //The Scientific Heritage. VOL 2, No 6 (6) (2016)
99. Бородавка Є.В. Квасневський В.М. Implementation of Data Structure for Digital Representation of Building Model // International Journal of Computer Science and Telecommunications. 01.11.2017
100. Бородавка Є.В. Квасневський В.М. Informational unified library of construction structures // The Scientific Heritage. VOL 2, No 24 (2018)
101. Фокс Ф. Вычислительная геометрия. Применение в проектировании и на производстве / Ф. Фокс, М. Пратт – М.: Мир, 1982. – 214 с.
102. Фолли Дж. Основы интерактивной машинной графики / Дж. Фолли, Ф. ван Дэм – М.: Мир, 1985. – 428 с.
103. Фролов А.В. Базы данных в Интернете. Практическое руководство по созданию Web-приложений с базами даннях / А.В. Фролов, Г.В. Фролов – М.: Издательско-торговый дом «Русская Редакция» – 2000. – 448 с.
104. Харрингтон Д. Проектирование реляционных баз данных. Просто и доступно. – М.: ЛОРИ, 2005. – 230 с.
105. Чекалов А. Базы данных: от проектирования до разработки приложений. – СПб.: BHV-СПб, 2003. – 384 с.
106. Шикин Е.В. Компьютерная графика. Динамика, реалистичные изображения / Е.В. Шикин, А.В. Боресков – М.: Диалог-МИФИ, 1995. – 348 с.
107. Шикин Е.В. Компьютерная графика. Полигональные модели / Е.В. Шикин, А.В. Боресков – М.: ДИАЛОГ-МИФИ, 2001. – 464 с.

108. Шикин Е.В. Начала компьютерной графики / Е.В. Шикин, А.В. Боресков, А.А. Зайцев – М.: Диалог-МИФИ, 1993. – 432 с.

109. Шикин Е.В. Кривые и поверхности на экране комп'ютера / Е.В. Шикин, А.И. Плис – М.: Диалог-МИФИ, 1996. – 344 с.

110. Шлеер С. Объектно-ориентированный анализ: моделирование мира в состояниях / С. Шлеер, С. Меллор – Киев, «Диалектика», 1993.– 452 с.

111. Энкарначо Ж. Автоматизированое проектирование. Основные понятия и архитектура систем / Ж. Энкарначо, Э Шпехтендаль, пер. с англ. – М.: Радио и связь, 1986. – 288 с.

112. Эрик Рэй. Изучаем XML. – Символ-Плюс, 2001. – 234 с.

113. Barsky B. Computer graphics and geometric modeling using Beta-Splines. – Springer Verlag, 1998. – 354 с.

114. Elmasri R. Fundamentals of Databased Systems (2nd ed.) / R. Elmasri, S.B. Navathe – Redwood City, Calif.: Benjamin/Cummings, 1994. – 246 с.

115. Farin G. Curves and surfaces for computer aided geometric design. A practical guide. – Academic Press, 1990. – 392 с.

116. Grady Booch. The Unified Software Development Process / Grady Booch, James Rumbaugh, Ivar Jacobson – Addison-Wesley. 1999 – 462 с.

117. Grady Booch. UML. Reference manual / Grady Booch, James Rumbaugh, Ivar Jacobson – Addison-Wesley. 1999. – 342 с.

118. Jos Warmer. The Object Constraint Language: Precise Modeling with UML / Jos Warmer, Anneke Kleppe – Addison-Wesley. 1999. – 252 с.

119. McGoveran D. Nothing from Nothing // Database Programming & Design. – 1993. – 6, № 12. – Part I; 1994. – 7, № 1. – Part II; № 2. – Part III; № 3. – Part IV.

ДОДАТКИ

Додаток А

Товариство з обмеженою відповідальністю
«Аллбау Софтвер»

ДОВІДКА

про впровадження результатів кандидатської дисертації на тему:

«МОДЕЛІ І МЕТОДИ ІНФОРМАЦІЙНОГО МОДЕЛЮВАННЯ ОБ'ЄКТІВ У ALLPLAN»

Квасневського Владислава Михайловича

Результати кандидатської дисертації Квасневського В.М. отримали практичне застосування під час розробки цілого комплексу модулів для інформаційної системи Allplan, серед них: «Генерація пластинчато-стержневої моделі (ПСМ)», «Передача ПСМ в SCAD», «Передача ПСМ в САПФИР», «Передача ПСМ в ЛІРА», «Експорт ПСМ в IFC». Розроблена концепція пластинчато-стержневої моделі та модулі розроблені на базі неї, дозволили автоматизувати процес переходу від архітектурної моделі до конструктивної, адже раніше це виконувалось конструктором власноруч, що викликало масу незручностей та породжувало низку помилок які обумовлені людським фактором. Розроблений комплекс модулів на базі ПСМ забезпечив можливість інтеграції інформаційної системи Allplan з цілою низкою конструктивних програмних застосунків.

Комплекс програмних застосунків на базі технології генерації ПСМ, що розроблена автором дисертації, використовується нашими співробітниками під час виконання внутрішніх задач та постачається нашим клієнтам, як у вигляді окремих модулів, так і як частина програмного пакету «RusAddon від Allbau».

Директор ТОВ «Аллбау Софтвер» Смірнов Ю.О.

Додаток Б

Товариство з обмеженою відповідальністю
«Аллбау Софтвер»

ДОВІДКА

про впровадження результатів кандидатської дисертації на тему:

«МОДЕЛІ І МЕТОДИ ІНФОРМАЦІЙНОГО МОДЕЛЮВАННЯ ОБ'ЄКТІВ У ALLPLAN»

Квасневського Владислава Михайловича

Результати кандидатської дисертації Квасневського В.М. отримали практичне застосування під час розробки комплексу застосунків «Design2Time». Використання методів та моделей описаних у дисертаційній роботі дозволили автоматизувати передачу будівельних об'ємів з інформаційної системи Allplan у комплекс для керування проектами Microsoft Project та забезпечити їх синхронізацію. Розроблений комплекс застосунків дозволив інтерактивно відображати хід будівництва, супроводжуючи це необхідною документацією у MS Project.

Комплекс застосунків «Design2Time», що розроблений автором дисертації з використанням методів та моделей описаних у кандидатській роботі, використовується нашими співробітниками під час виконання внутрішніх задач та постачається нашим клієнтам.

Директор ТОВ «Аллбау Софтвер» Смірнов Ю.О.