УДК 681.3

Моделювання ієрархічних виробничих підприємств

Коробецький Ю.П., Дорошко В.В.
Східноукраїнський національний університет ім. В. Даля
MeisterLg@mail.ru

Abstract

Korobetsky Y., Doroshko V. Modeling of hierarchical production enterprises. *The features of hierarchical production enterprises modeling are considered in this work. It's proposed to build hierarchical models using hierarchical levels similar to each other. Using OOD and UML the tasks of modeling have been exposed; the modeling algorithms have been constructed and model structure have been defined. The features of model functioning are displayed in work using compound routes of processing. The next development paths are specified.*

Вступ

Підвищення ефективності роботи виробничих підприємств завжди було актуальною задачею. Рішення цієї задачі багато в чому визначається можливістю розробки інформаційних моделей діяльності виробничих підрозділів [1], що знаходяться один до одного у ієрархічній залежності, і відповідних програмних продуктів, за допомогою яких можна прогнозувати показники їхньої роботи. Розробка ефективних інформаційних моделей повинна виконуватися в обмежений строк, з невеликими витратами на розробку проектів та їхню реалізацію, із спроможністю адаптуватися до умов, що змінюються, з високими показниками надійності системи та точності розрахунків тощо [2]. У цій роботі розглядається проблема побудови моделей ієрархічних виробничих підприємств.

Проведений аналіз досліджень [3] показав, що методи, які зараз використовуються для моделювання подібних систем, на жаль, дозволяють будувати лише прості, тобто однорівневі моделі (наприклад, модель дільниці, або модель підприємства, що складається з цехів, але цехи розглядаються як найпростіші елементи, а не як структури, що мають свої складові елементи).

Існує значна різниця між побудовою однорівневих моделей обробки деталей на машинобудівному підприємстві та побудовою такихож багаторівневих моделей. Якщо побудова однорівневої моделі є досить зрозумілою та може вирішуватися різноманітними способами [4], [5], [6], то спроба перенести принципи такої побудови на багаторівневі моделі наражається на певні суперечності.

Оскільки виробничі системи є багаторівневими та ієрархічними, то розробка багаторівневої моделі є актуальною. У [7] показано подібність різних рівнів обробки один до одного, у [2] закладені головні принципи спрощення моделі, тобто відсікання другорядних аспектів. Згідно з наведеними принципами, вважаємо агрегати та транспортери цілком подібними одне до одного об'єктами, оскільки і ті, й інші виконують обробку, тобто витрачають час, характер обробки при цьому значення для нас не має. Назвемо ті й інші одним терміном обробних елементів, або, спрощено, елементів (Elem).

Згідно з методологією об'єктно-орієнтованого проектування (ООП) [8], [9], [10], [11], [12] та UML [13], [14], [15], [16] визначаємо основні цілі аналізу:

- виявлення задач моделювання;
- побудова алгоритмів моделювання;
- визначення структури моделі;
- організація функціонування моделі.

Виявлення задач моделювання

Інструментом аналізу та проектування будемо використовувати мову UML. Перший етап аналізу передбачає визначення вимог до системи, які описуються діаграмою прецедентів. На рис. 1. показано таку діаграму.

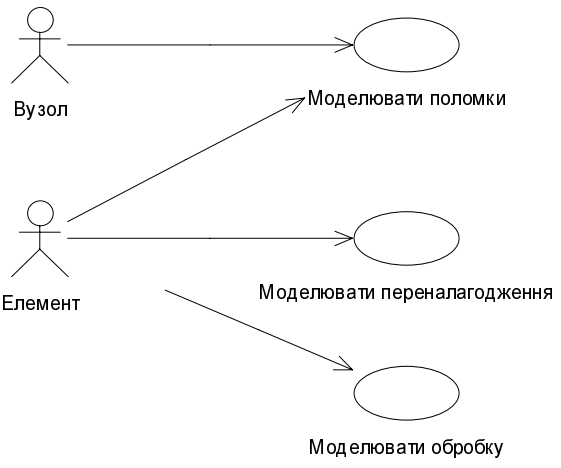

Рисунок 1 – Вимоги до системи

Як бачимо, тут виділяються два актори: елемент, про який, до речі, вже було сказано, та вузол. Під вузлом розуміється виробнича одиниця, що складається з більш простих одиниць: це може бути дільниця, що складається з агрегатів, або цех, що складається з дільниць, або все підприємство, що складається з цехів. На малюнку вказано, що актор елемент вимагає від системи змоделювати поломки, переналагодження та обробку. Вузол – тільки поломки, оскільки моделювання поломок вузлу можна уявити як сукупність окремо змодельованих поломок його складових, а із моделюванням переналагодження та обробки цього зробити не можна. Причиною цього є те, що крок обробляння деталі вузлом включає не обробку цієї деталі усіма його підвузлами (або елементами), а таке обробляння, що закріплене у маршруті обробки даної деталі (аналогічна ситуація з переналагодженням). Тому актор вузол і не вимагає від системи цих двох прецедентів.

Побудова алгоритмів моделювання

Тепер, коли окреслені прецеденти, настає час визначити алгоритм, що описує поведінку системи та її елементів, яка буде призводити до вирішення задач, що системі задаються (тобто прецедентів). У мові UML є спеціальна діаграма – діаграма активностей, що описує алгоритми поведінки. Нижче за допомогою цих діаграм будуть описані визначені прецеденти (див. рис. 2).

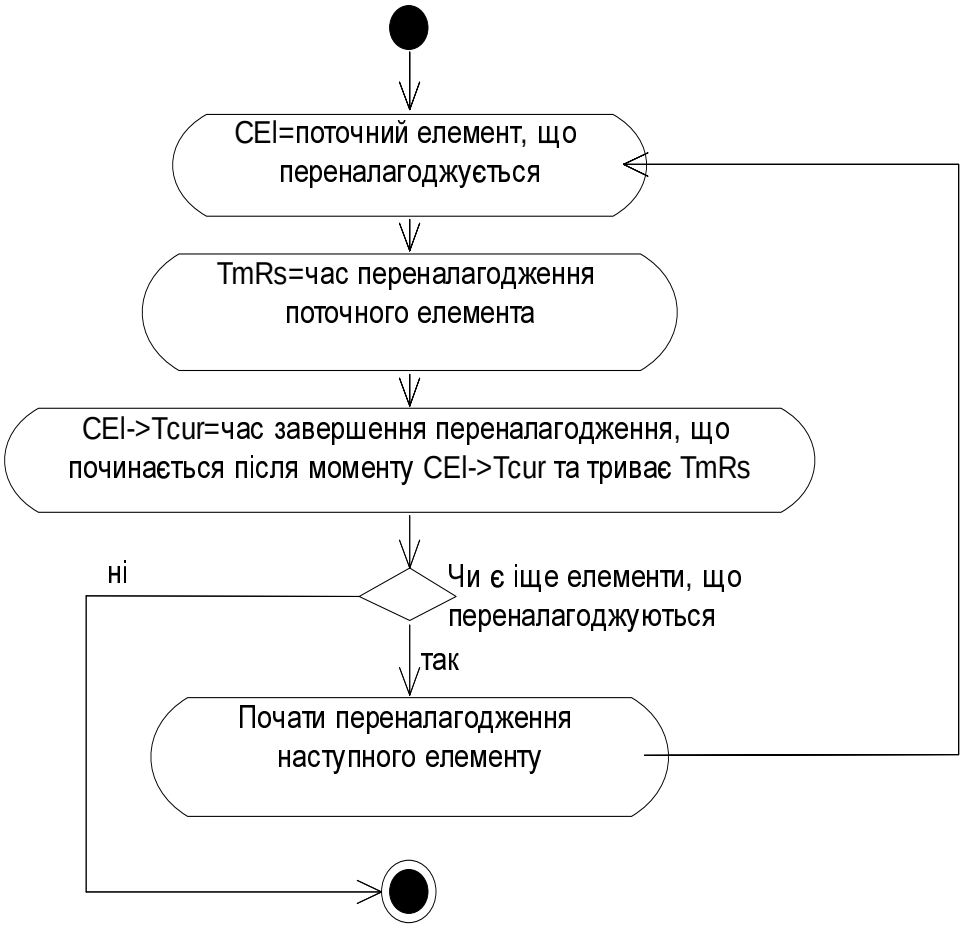

Рисунок 2 – Діаграма дій, що поширює прецедент «Моделювати переналаджування»

Як бачимо, алгоритм включає визначання поточного елементу, що переналагоджується, визначання часу переналагодження та, власне, самого переналагодження.

Далі на рис. 3 показано алгоритм моделювання поломок.

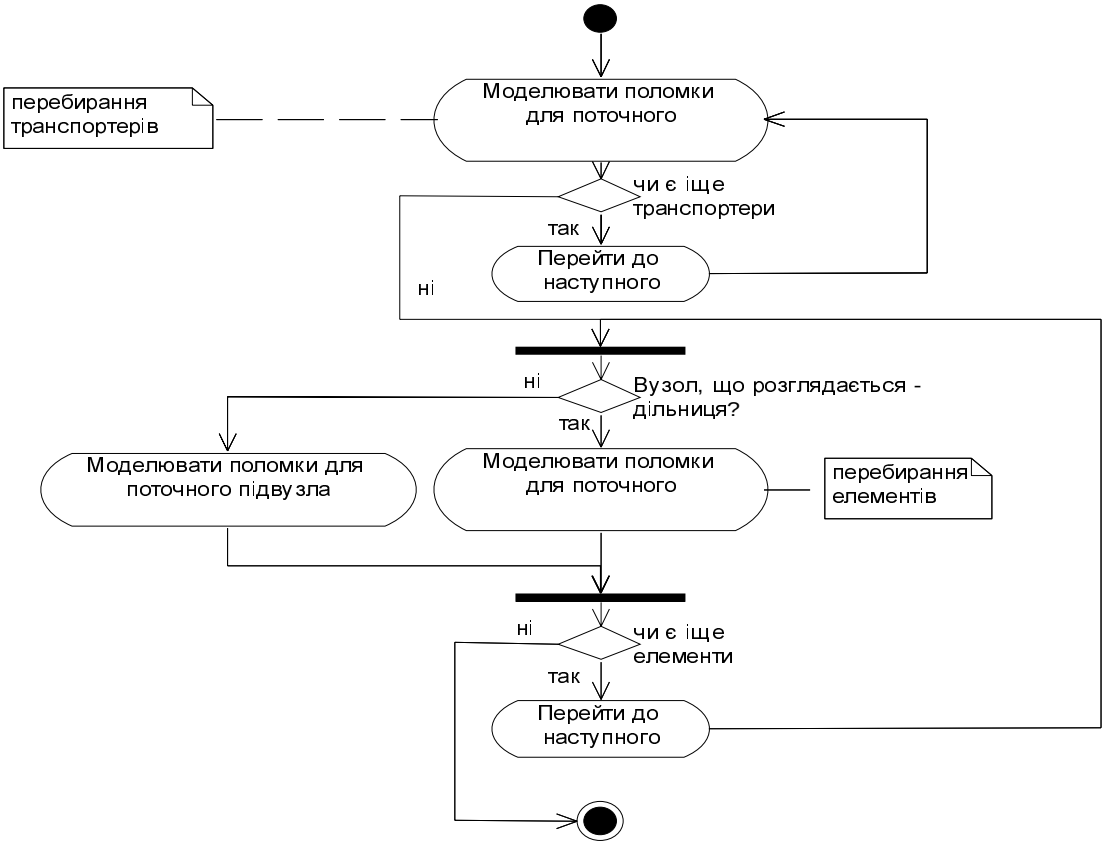

Рисунок 3 – Діаграма дій, що поширює прецедент «Моделювати поломки»

Як можна бачити з цієї діаграми, наведений алгоритм є рекурсивний, тобто моделювання поломок для вузлу є сукупністю моделювань поломок його підвузлів, або його елементів. Позначений вид діяльності «моделювати поломку для поточного» і виконує моделювання для елементу (поточного елементу). Далі наведена іще одна діаграма дій, що деталізує «моделювати поломку для поточного» (див. рис. 4).

Як бачимо, моделювання поломки для поточного елементу – це циклічний процес, що поширюється на якийсь наперед визначений термін (попередній час моделювання), що, безумовно, буде довшим за реальний час моделювання. По-перше, визначається початок поломки, а потім визначається її тривалість, початок та тривалість у сумі дають закінчення. Початок наступної поломки рахується від закінчення попередньої, і так далі.

Далі розглянемо алгоритм найскладнішого з аспектів наведеного тут моделювання – моделювання обробки. На рис. 5. показана діаграма дій, що описує моделювання обробки.

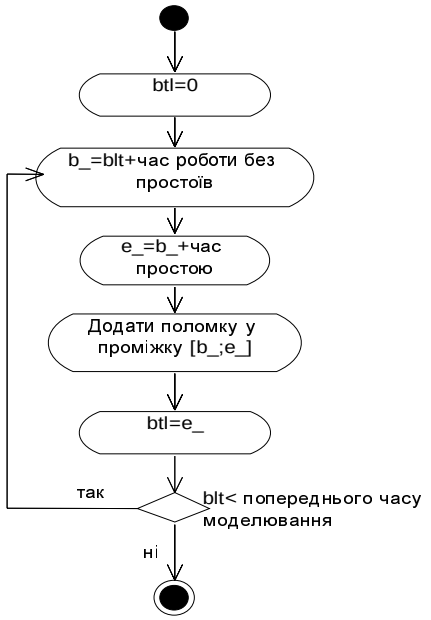

Рисунок 4 – Діаграма дій, що поширює дію «Моделювати поломки для поточного»

.

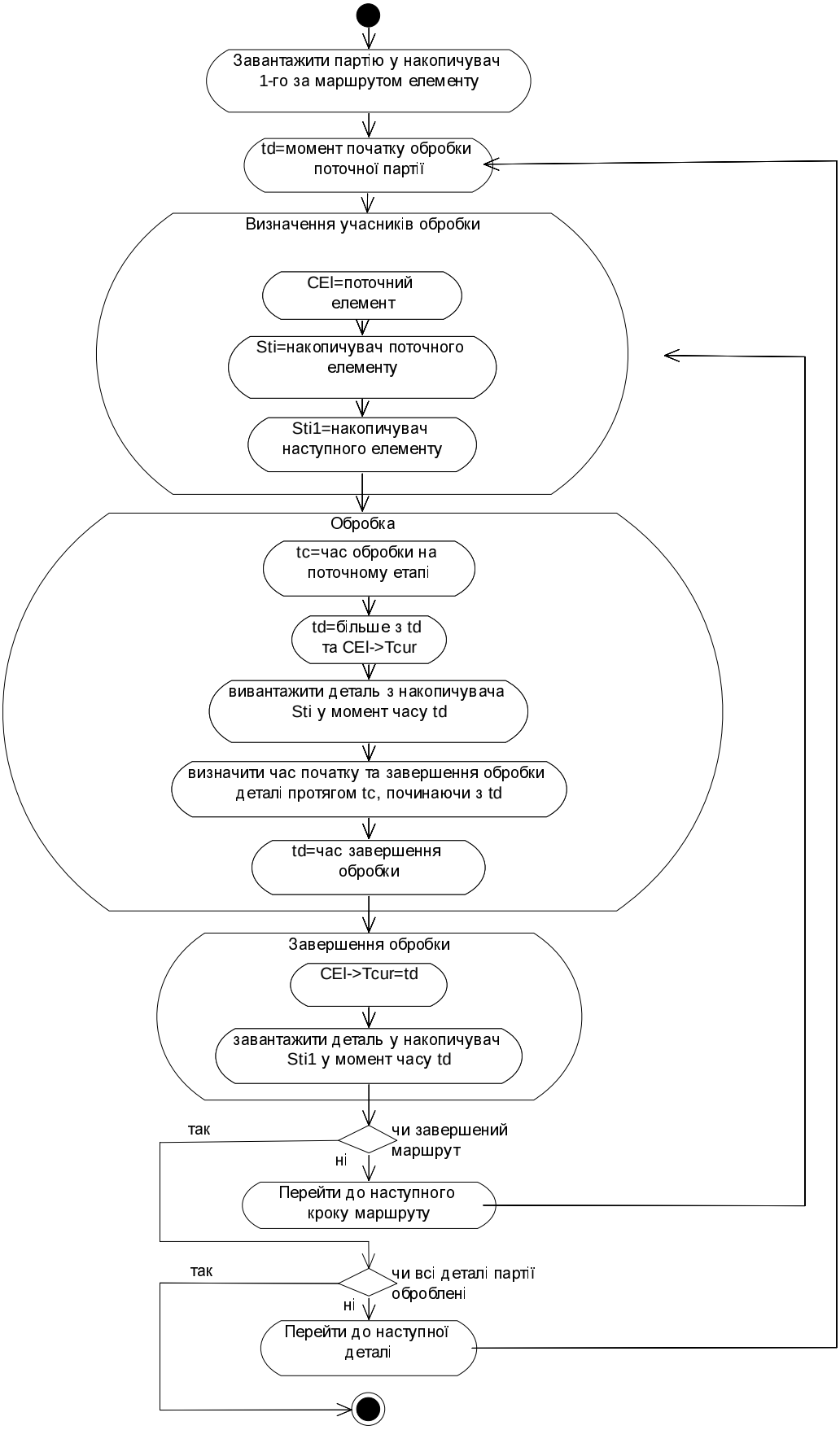

Рисунок 5 – Діаграма дій, що поширює прецедент «Моделювати обробку»

Як бачимо, моделювання складається з трьох основних етапів: визначання учасників обробки, власне обробки та закінчення обробки. Під час окреслення учасників обробки визначається елемент, який буде виконувати обробку, накопичувач поточного елементу, тобто накопичувач, звідки деталь буде забиратися для обробки, накопичувач наступного елементу, тобто накопичувач, куди деталь буде вивантажуватися після закінчення обробки. Сама обробка включає визначення часу обробки, визначення початку обробки, вивантаження деталі з поточного накопичувача, обробки деталі та визначення часу закінчення обробки. Завершення включає фіксацію часу звільнення елементу та вивантаження обробленої деталі до накопичувача наступного елементу.

Визначення структури моделі

Дослідника також цікавить послідовність виконання дій під час реалізації прецедентів, обмін повідомленнями між об'єктами, що беруть участь у цій реалізації. Для цього у мові UML використовується діаграма послідовностей. Нижче наведено діаграму послідовностей для прецеденту «Моделювати обробку» (див. рис. 6).

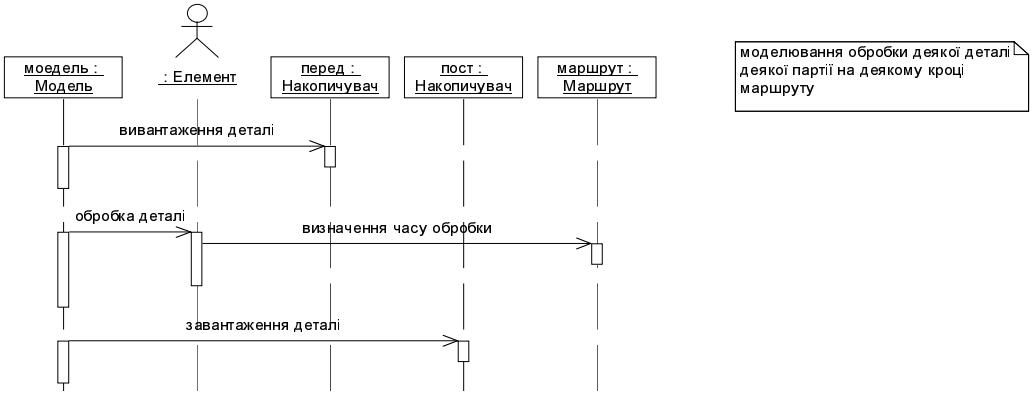

Рисунок 6 – Діаграма послідовностей. Процес обробки деталі

Тут пред – це накопичувач поточного елементу, пост – накопичувач наступного елементу, а маршрут – об'єкт, що містить послідовність обробних елементів для певної партії деталей та час їхньої обробки.

Тут, як бачимо, введено такий об'єкт, як модель. Під об'єктом «модель» розуміються взагалі всі дані про модель – структура, вузли, елементи, маршрути обробки, партії деталей тощо. Такий підхід дозволить використовувати декілька моделей, що допомагає, наприклад змінювати існуючу структуру (виробництва або замовлень), розраховуючи поведінку підприємства у нових умовах. Так от, саме сутність «модель» і ініціює моделювання обробки.

Далі зберемо усю інформацію про класи до однієї діаграми (рис. 7).

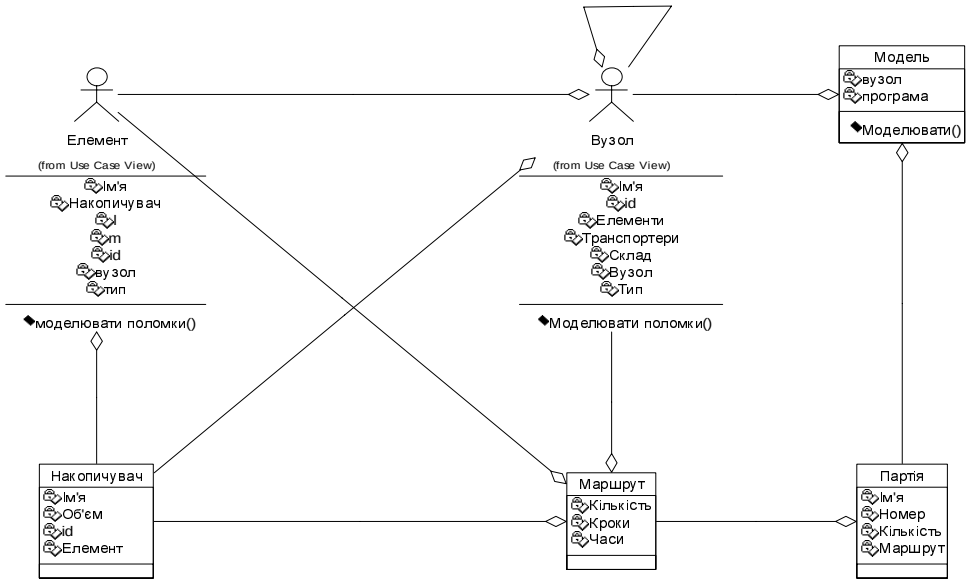

Рисунок 7 – Діаграма класів. Виділення класів у моделі

Організація функціонування моделі

Однак однієї схеми моделі недостатньо для її функціонування, потрібна також організація цієї моделі, тобто визначення логіки, згідно з якою вона буде функціонувати. У нашому випадку це означає визначення маршрутів обробки деталей різними рівнями підприємства та їхнього сумісного взаємовідношення.

Приклад найпростішого одноріневого маршруту обробки деталей певної партії показаний на рис. 8.

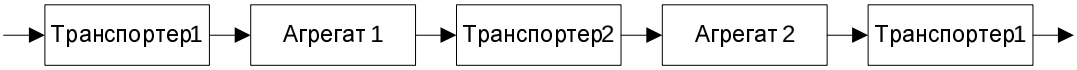

Рисунок 8 – Приклад одноріневого маршруту обробки партії деталей

Якщо ми розглянемо маршрут обробки партії деталей на другому рівні (тобто рівні цеха), то побачимо таку картину (див. рис. 9).

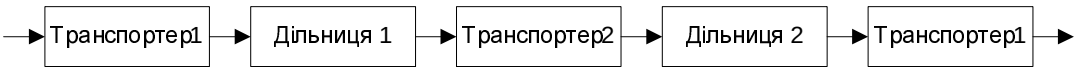

Рисунок 9 – Приклад маршруту обробки партії деталей рівня цеху

Але при більш детальному аналізі ми зрозуміємо, що вказана на дев'ятому малюнку «Ділянка 1» для однієї партії деталей оброблятиме деталі в одній послідовності, а для іншої та ж «Ділянка 1» може обробляти їх у зовсім іншій послідовності. Тому тут вірніше буде говорити не про ділянки, а про маршрут обробки рівня ділянки, тобто той, приклад якого наведений на восьмому малюнку. Загальну картину представлено на рис. 10.

Рисунок 10 – Уточнений маршрут обробки партії деталей рівня цеху

Тобто ми спостерігаємо укладеність маршрутів меншого рівня у маршрут більшого рівня, подібно до вкладання самих виробничих одиниць один до одного. Можемо уявити себе усю складну структуру багаторівневого маршруту. На рис. 11. показано структуру трирівневого маршруту обробки партії деталей.

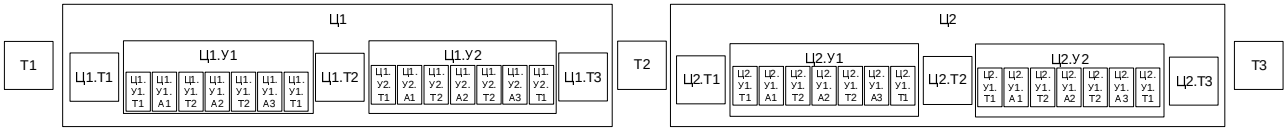

Рисунок 11 – Структура трирівневого маршруту обробки партії деталей

У [2] також вказувалося, що кожний з обробних елементів має накопичувачі для збереження деталей. Важливим моментом є також визначення послідовності переміщення не тільки між обробними елементами, але й між їхніми накопичувачами, тому що, наприклад, агрегат нерухомий, тому забирає та вивантажує деталі в один і той же накопичувач, тоді, як транспортер може переміщуватися між ними.

Знов почнемо з рівня дільниці (див. рис. 12), використавши приклад, наведений на рис. 8.

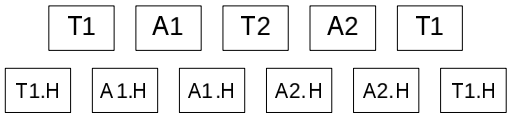

Рисунок 12 – Приклад одноріневого маршруту з накопичувачами

Тут Ті – і-й транспортер, Аі – і-й агрегат, відповідно Ті.Н, Аі.Н – їхні накопичувачі. Як бачимо, накопичувачів завжди на один більше, ніж обробних елементів. Тепер наочно наведемо, яким чином співвідносяться між собою маршрути різних рівнів, коли там також присутні накопичувачі (див. рис. 13).

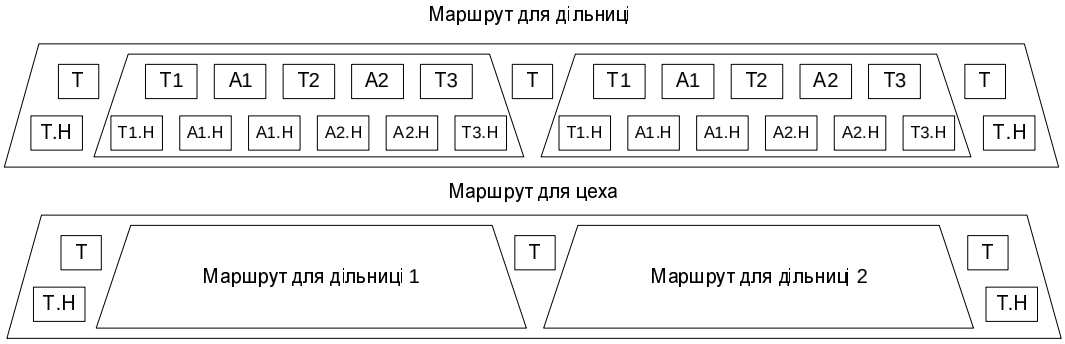

Рисунок 13 – Багаторівневі маршрути з урахуванням накопичувачів

Висновки

Отже, мова UML дозволяє виконати аналіз та проектування системи, описати основну структуру моделі та її поведінку.

Виходячи з моделі, описаної UML, можна перейти до програмної реалізації. Для програмної реалізації може бути застосована будь-яка об'єктно-орієнтована мова програмування, наприклад C++ [17], [18], [19], [20]. Але моделювання – не єдина задача програми, що розробляється. Вона повинна вирішувати такі задачі:

- безпосередньо моделювання;
- графічне відображення результатів;
- ведення статистики.

Надана схема організації моделей дає можливість мати чітку послідовність обробки кожної партії деталей кожним обробним елементом на всіх рівнях виробництва. У такому вигляді дані про маршрути повинна отримувати модель, а база даних повинна надати можливість отримати їх.

Література

1. Коробецький Ю. П., Рамазанов С. К. Імітаційні моделі у гнучкому виробництві. Монографія. – Луганськ: Вид-во СНУ ім. В. Даля, 2003. – 280 с.
2. Коробецкий Ю. П., Дорошко В. В. Особенности построения информационной модели работы производственного подразделения – Праці Луганського відділення Міжнародної академії інформатизації. – 2008. - № 1. – С. 59-61.
3. Дорошко В.В., Коробецкий Ю.П. Використання методів об'єктно-орієнтованого проектування для побудови моделей машинобудівного підприємства – Вісник Східноукраїнського національного університету імені Володимира Даля. – 2008. - № 6 (124). Частина 2. – С. 109-113.
4. ИММОД-2003: Материалы Всеросс. науч. конф. в г. Санкт-Петербурге, 23-24 октября 2003 года / УГАТУ; Загидуллин Р. Р. – СПб, 2003.
5. ИММОД-2003: Материалы Всеросс. науч. конф. в г. Санкт-Петербурге, 23-24 октября 2003 года / ОрелГПУ; Лазарев. – СПб, 2003.
6. Карташов Леонід Євгенович. Моделювання гнучких виробничих систем із часовим резервуванням: Автореф. дис... канд. техн. наук (05.13.20) / Нац. техн. ун-т України "Київ. політех. ін-т". - К., 2005. – 20 с.
7. Коробецкий Ю. П., Дорошко В. В. – Праці Луганського відділення Міжнародної академії інформатизації. – 2006. - № 1. – С. 59-61.
8. E. Gamma, R. Helm, R. Johnson, J. Vlissides. Design Patterns. – Addison-Wesley Publishing Company, 1994.
9. J. Garzas M. Piattini. Object-oriented Design Knowledge: Principles, Heuristics And Best Practices. – Igi Global, Hershey, 2006.
10. G. Butch. Object-Oriented Analysis and Design with Applications. – Pearson Education, 1993.
11. Трофимов С.А. CASE-технологии: практическая работа в Rational Rose. Изд. 2-е. – М.: Бином-Пресс, 2002. - 288 с.
12. Вендров А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. – М.: Финансы и статистика, 1998. – 176 с.
13. G. Booch, J. Rumbaugh, I. Jacobson. Unified Modeling Language User Guide. – Addison-Wesley Publishing Company, 2005.
14. D. Pilone, N. Pitman. UML 2.0 in a Nutshell. – O'Reilly Media, Inc, 2005.
15. Scott W. Ambler. The Elements of UML 2.0 Style. – Cambridge University Press, 2005.
16. Quatrani T. Visual Modeling with Rational Rose 2000 and UML. – Addison-Wesley Publishing Company, 1999.
17. Архангельский А. Я. Программирование в C++ Builder 6. – М., Бином, 2003. – 1152 с.
18. Bjarne Stroustrup. The C++ programming Language. – Addison-Wesley Publishing Company, 1997.
19. Bruce Eckel. Thinking in C++. – MindView. Inc., 2000.
20. J. Hollingworth, D. Butterfield, B. Swart, J. Allsop. C++ Builder 5: Developer's Guide. – Sams, 2000.

Надійшла до редколегії 10.03.2009