Інформаційна модель машинобудівного підприємства

Дорошко В. В.

Вступ

Машинобудівні підприємства нашої країни змушені оптимізувати виробництво. Одним з інструментів такої оптимізації виступають інформаційні моделі. Машинобудівне підприємство є складною системою, що об'єднує багато підрозділів, які функціонують у зв'язку одне з одним, це означає необхідність розробки інформаційних моделей аналізу та проектування роботи підрозділів підприємства (ділянка, цех, підприємство у цілому) аж до реалізації програмного забезпечення. Подібні моделі використовуються для прогнозування показників роботи зазначених підрозділів, для аналізу та оптимізації їх роботи на підставі матеріалів, які збираються на машинобудівному підприємстві. Такі системи треба будувати за принципом „знизу-доверху", тобто починати з моделей роботи ділянки, потім на базі моделей ділянок будувати модель роботи цеху і так до рівня моделі роботи підприємства у цілому.

Отже, метою дослідження є побудова інформаційної моделі ділянки машинобудівного підприємства, як найпростішого підрозділу машинобудівного підприємства.

У [1] розглянуто побудову імітаційної моделі гнучких виробничих систем із використанням апарату мереж Петрі. Недоліком цієї моделі є вузькість її застосування. Цей підхід може застосовуватися для побудови імітаційних моделей, але для інформаційних його недостатньо. У [2] описане застосування імітаційної моделі виробничого процесу в управлінні дискретними виробничими системами та створення такої моделі. Модель виробничого процесу розроблено відповідно з принципами подійного моделювання. В імітаційній моделі підприємство розглядається як сукупність цехів – елементарних виробничих одиниць. Технологічний процес виготовлення виробу становить собою послідовність операцій, для кожної з яких зазначено цех, в якому вона виконується, тип необхідного устаткування, професія робітника, час виконання, вартість робіт. Для реалізації використано об'єктно-орієнтовану мову імітаційного моделювання EML подійного типу. Головним недоліком цієї роботи, на нашу думку, є її однорівневість, що зменшує точність відтворення виробничої системи.

Досягнення цілі передбачає виконання таких задач:

- формування моделі системи з точки зору виконуваних нею дій;
- формування моделі представлення даних;
- програмна реалізація сформованих моделей.

Основний матеріал

Інформаційні моделі виступають у ролі формалізованого відображення реального об'єкту виробництва з точки зору інформаційних явищ, що існують у цьому об'єкті. Забезпечує ж ефективність інформаційних моделей те, що будь-які явища навколишнього світу взагалі та виробничого процесу зокрема, можуть бути зображені з точки зору інформації, яку передають, отримують, обробляють, зберігають та відтворюють об'єкти, що приймають участь у ході цього явища. З іншого боку, інформаційні моделі корисні тим, що дослідник, відтворюючи ту чи іншу модель, може розглядати її крізь призму тих цілей, що встановлені у його дослідженні, винісши за межи моделі ті аспекти діяльності об'єкту, що його не цікавлять.

Інформаційні моделі є основою програмних продуктів. Таке їхнє втілення стає потужним і зручним інструментом у руках дослідників. Але щоб цей програмний продукт став саме зручним та потужним інструментом, він повинен бути вдало реалізованим. Процес реалізації програмного продукту є багатоаспектним, складним, неоднозначним і творчим. У зв'язку з цим постає проблема визначення якихось базових принципів, інструментів, методів, які б дозволили внести ясність у цей процес для фахівця, що розробляє такі системи. Останні тенденції у розвитку програмування винесли на перший план об'єктно-орієнтований підхід (до програмування та проектування). Одним із втілень об'єктно-орієнтованого проектування є графічна мова UML [3]. Крім того, нещодавно зародилися та дедалі поширюються CASE-технології [4], що за своєю суттю є реалізацією об'єктно-орієнтованого проектування у програмній системі. Одним з таких програмних комплексів є система Rational Rose, що використовує мову UML для опису моделей системи, яка проектується [5].

У даній статті розглянуто деякі аспекти побудови інформаційної системи роботи підрозділів машинобудівного підприємства. На сьогодні більш досконалою моделлю життєвого циклу програмного забезпечення (ЖЦПЗ) вважається спіральна (або ітеративна) модель [4]. Згідно з цією моделлю, розробка програмного забезпечення проходить декілька циклів (витків спіралі), на кожному з яких проходять усі етапи розробки: аналіз, проектування, реалізація, тестування, інтеграція. Тобто наприкінці кожного з етапів реалізується нова версія ПЗ, а кожна нова версія є вдосконаленням попередньої. Мета кожного циклу – отримати робочу версію, яка в той же час не обов'язково буде виконувати усі вимоги, що ставляться до кінцевого ПЗ. Тому при такій схемі логічно для першої версії обрати найпростішу модель, щоб добитися чіткого розуміння, стабільної роботи, та мати основу для подальшого поширення інформаційної моделі.

Об’єктом у нашому аналізі є ділянка механічного цеху. Спіральна модель ЖЦПЗ інформаційної системи підприємства, що починається з інформаційної системи ділянки, відповідає зазначеному принципові „знизу-доверху”. На ділянці відбувається обробка деталей по партіях, згідно з програмою обробки. Крім основних операцій (обробки виробничими агрегатами – верстатами), над деталями здійснюються допоміжні операції, як то транспортування, складування. Програма обробки у відношенні до ділянки є зовнішньою сутністю, тобто програма „спускається зверху”, а не розробляється на самій ділянці. Програма вказує, які партії деталі повинні бути оброблені ділянкою за певний період, яка послідовність їх обробки, тобто які агрегати будуть це робити.

Але з’ясовується, що однієї програми замало для того, щоб повністю описати обробку деталей на ділянці. Важливою є також схема послідовностей запуску деталей (СПЗД). СПЗД визначає, яким чином одночасно будуть оброблятися на ділянці декілька деталей однієї партії. На мал. 1 показана послідовність обробки однієї деталі агрегатами на ділянці згідно програми обробки.

Рисунок 1 - Послідовність обробки деталі агрегатами на ділянці

Тип виділення блоку тут відповідає агрегатові, що обробляє деталь, а довжина блоків – тривалості обробки. У наведеному прикладі обробка здійснюється трьома агрегатами: першим, другим, знов першим та третім.

Оскільки усі деталі оброблюються одними і тими ж агрегатами, часто деталі будуть змушені простоювати, очікуючи, поки потрібний їм агрегат закінчить обробку іншої деталі. Крім того, простої можуть бути викликані неможливістю вивантаження деталі з агрегату, оскільки агрегат, на якому вона повинна бути оброблена далі, іще не готовий до її завантаження, а стіл біля агрегату вже зайнятий (розглядається схема без нагромаджувачів); простій також може бути викликаний простоєм, з різних причин, попередніх за маршрутом агрегатів. Усе це потрібно враховувати під час побудови СПЗД. Складності у це питання також додають випадки, коли деталь оброблюється одним і тим же агрегатом декілька разів (на мал. 1 приведений саме такий випадок). Часто для однієї послідовності обробки деталі можливі декілька СПЗД. На мал. 2 показані різні СПЗД для послідовності обробки, наведеної на мал. 1.

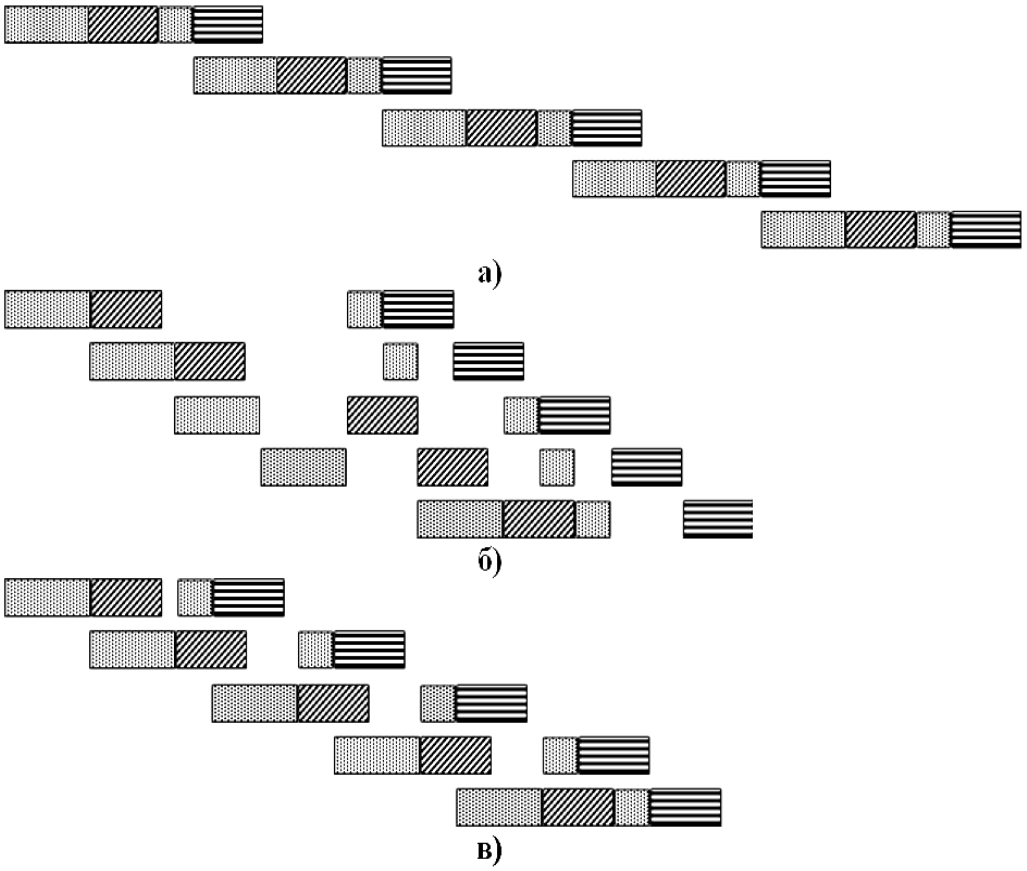

Рисунок 2 - Різноманітні СПЗД для однієї послідовності обробки деталей

Тут наведена СПЗД для партії з п’яти деталей. На малюнку можна побачити простої деталей у зв’язку із зазначеними причинами. Крім того, бачимо, що СПЗД, показана на мал. 2 а), вимагає найбільше часу, схема б) більш оптимальна, та схема в) є найкращою з трьох схем і дозволяє обробити партію швидше. Безумовно, подані схеми не враховують випадковий характер величини часу обробки деталей, не враховують також витрат на допоміжні операції (перш за все, транспортування), але головною метою аналізу СПЗД є не визначення часу обробки партії деталей, а визначення саме кращої схеми, тому такі фактори, як випадковість та наявність допоміжних операцій, тут є другорядними.

Першим кроком аналізу за допомогою мови UML є аналіз з точки зору прецедентів (варіантів використання, use case), що відображається за допомогою діаграм прецедентів. Ці діаграми служать для моделювання контексту системи, тобто відокремлення та виділення акторів, як зовнішніх сутностей, що взаємодіють із системою, від самої системи. Крім того, діаграма допомагає моделювати вимоги до системи, тобто бажаної її поведінки, розглядаючи саму систему як „чорний ящик” [3].

На мал. 3 зображена діаграма прецедентів UML для досліджуваної системи.

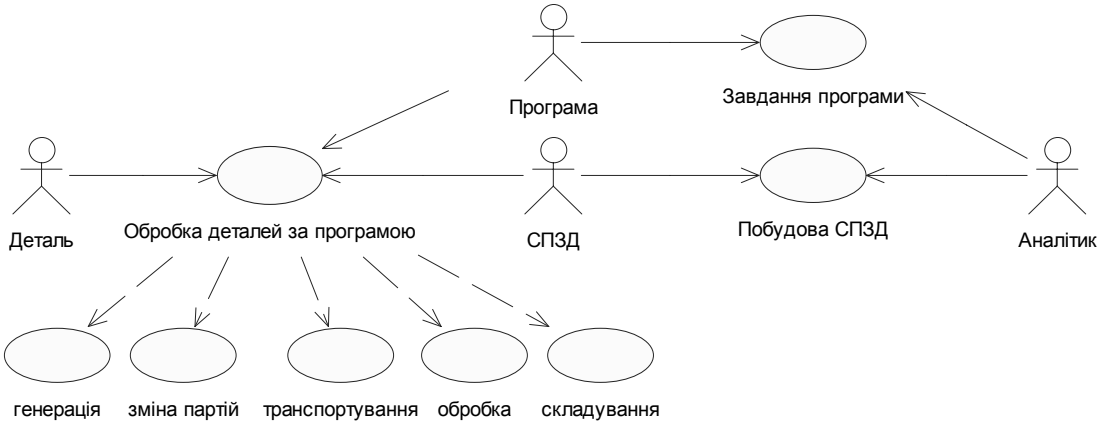

Рисунок 3 - Діаграма прецедентів для ділянки механічного цеху

Отже, три головних задачі, що постають перед системою: обробка деталей згідно програми обробки, завдання програм обробки та побудова СПЗД. Актор „аналітик” у даному випадку відповідає експертові, що буде надавати системі необхідну інформацію про програми та СПЗД. Крім того, найбільш складну з задач, задачу обробки деталей за програмою, можна поділити на більш прості: обробка, транспортування, складування, зміна партій, генерація. І кожну з цих задач будуть виконувати певні елементи системи. Генерація тут – це дещо штучна дія, пов’язана з тим, що система обмежена лише однією дільницею. Насправді система має зовнішнє середовище (інші ділянки, цехи), які будуть передавати у систему необроблені деталі та забирати оброблені. Щоб замінити зовнішнє середовище, замість надходження деталей у систему вводиться їх генерація, а замість вилучення – ліквідація (обидва ці поняття „генерація” та „ліквідація” тут включаються у термін „генерація”).

У системі можемо виділити сутності, що будуть виконувати поставлені перед нею задачі: генерацію деталей виконує генератор, транспортування – транспортні засоби, обробку – агрегати, складування – склад. Усі ці виділені сутності є прототипами майбутніх класів. Оскільки усі вони будуть функціонувати за подібними принципами, виконуючи операції над деталлю, логічно виділити сутність-предка, що вбиратиме в себе усі їхні подібні ознаки. На мал. 4 показана схема сутностей, про які казалося.

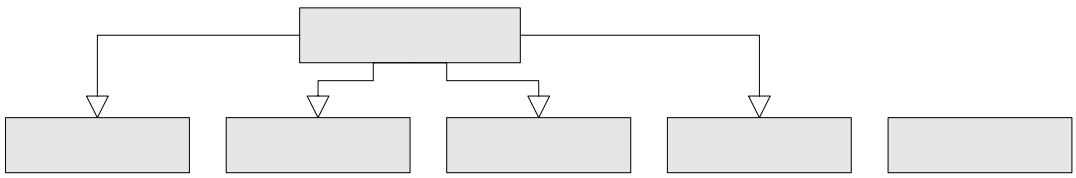

Рисунок 4 - Схема сутностей

Як бачимо, між генератором та об’єктами немає зв’язку спадкування. Це викликано зазначеною штучністю цієї сутності (генератора) та її тимчасовим характером.

CASE-технології передбачають поряд з розробкою самої системи розробку бази даних цієї системи. На мал. 5 показана структура бази даних ділянки.

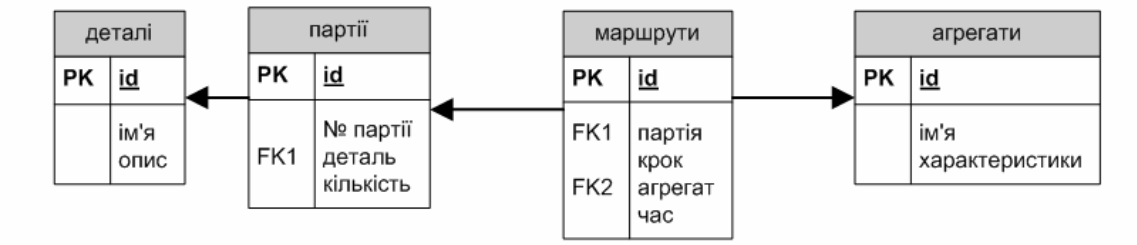

Рисунок 5 - Структура бази даних ділянки

Крім вже зазначених таких сутностей, як деталь та агрегат, тут з’являються партії деталей. Партію як сутність виділяє те, що усі деталі однієї партії оброблюються однаково (за однією програмою та СПЗД). Крім того, виділена сутність „маршрути” визначає маршрути обробки також саме для партій деталей. Маршрут обробки деталей включає партії та послідовність кроків, кожний з яких містить обробляючий агрегат та час обробки цим агрегатом. Приклад маршруту наведений у табл. 1.

Таблиця 1. Приклад маршруту обробки деталей

№ кроку	**Перша партія**	1	2	3	**Друга партія**	1	2	3
Обробляючий агрегат		Агрегат1	Агрегат2	Агрегат3		Агрегат4	Агрегат2	Агрегат3
Час обробки (хв.)		1,2	0,7	2,4		2,1	0,8	2,4

Така структура у таблиці бази даних подається чотирма полями (а також первинний ключ id).

Зображені на мал. 3 основні прецеденти (обробка деталей за програмою, завдання програми, побудова СПЗД), що відповідають основним завданням, визначають поділення системи на частини – підсистеми. Маючи БД, яка зберігає усі дані про систему, треба мати і інструмент її обробки, отже для підтримки інформаційної моделі потрібна також система обробки бази даних.

Система обробки бази даних будується за принципом максимального універсалізму. Ідея полягає в тому, що процес формування найкращої інформаційної системи – дуже довгий процес, а база даних при цьому – лише інструмент роботи системи, тому структура її (бази даних)

змінюватиметься. Система ж обробки бази даних повинна мати можливість працювати з будь-якою конфігурацією бази даних, бути при цьому якомога зручнішою та мати можливість швидко прилаштовуватися до зміненої бази даних, без зміни самої програми (системи обробки бази даних). Редагуючи базу даних, зазвичай приходиться працювати з парами таблиць, пов'язаних між собою. Наприклад, пара деталі–партії, тобто тут партії будуть відповідати деталям. Якщо редагувати таку пару у стандартній СУБД, складно відразу зрозуміти, до якої деталі належить та чи інша партія, бо приналежність до неї визначається значенням унікального коду id, та для того, щоб зрозуміти, який код відповідає якій деталі, треба зайвий раз зазирати у таблицю деталей. Зручний варіант рішення такої проблеми: відображати таблиці парою, при цьому у другій таблиці вибирати тільки ті записи, що відповідають виділеному рядку першої. Але для того, щоб відображати таким чином пару таблиць, треба заздалегідь знати структуру бази даних, та при такому підході неможливе відображення таблиць при зміні структури БД, а це суперечить зазначеному принципу універсалізму та взагалі зводить нанівець усю зручність системи обробки бази даних. Аби виконати поставлені перед підсистемою умови, був обраний проміжний підхід, при якому користувач спочатку визначає, з якою парою таблиць він хоче працювати, та зовнішній ключ, яким вони поєднуються, а потім зручно працює з цією парою таблиць. На мал. 6 показано, яким чином це здійснюється.

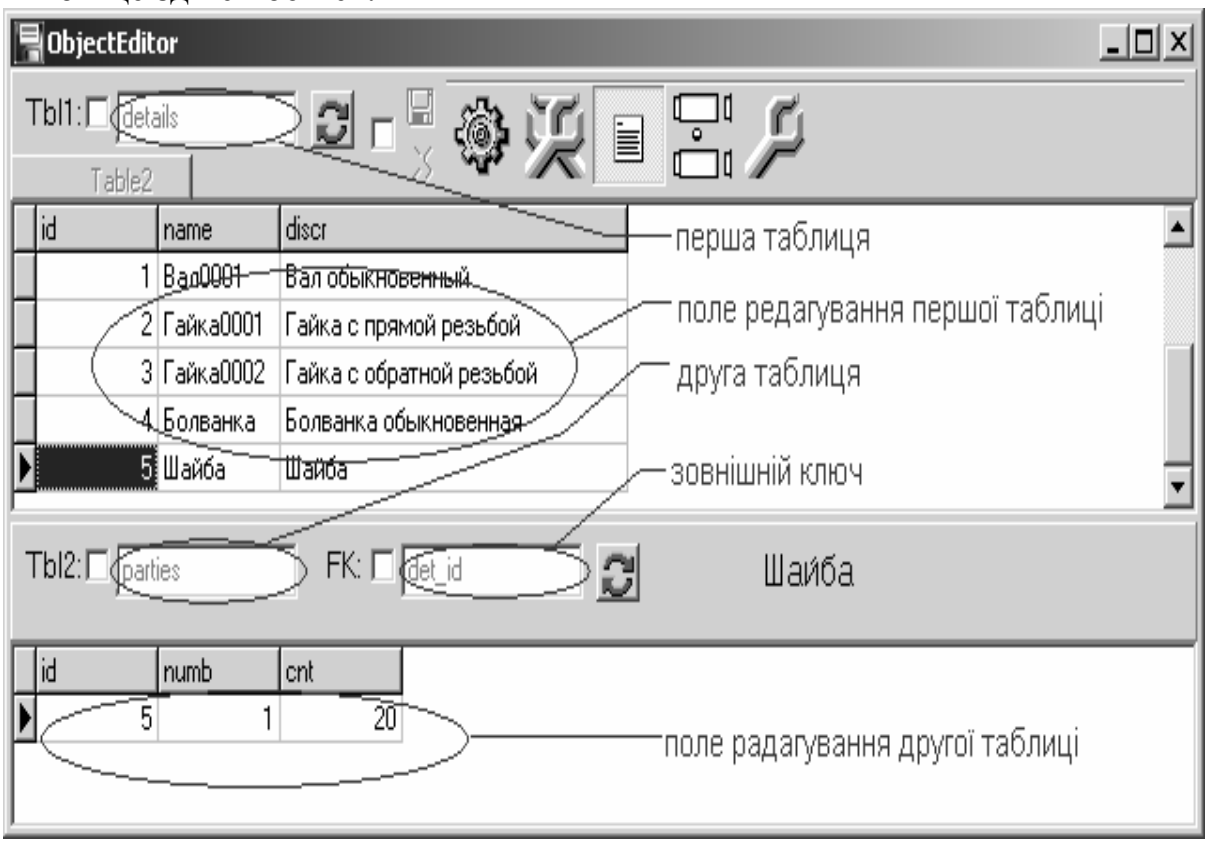

Рисунок 6 - Робота з двома таблицями у підсистемі обробки бази даних

Крім цих можливостей, підсистема має інструменти для зберігання установок, швидкого виклику певних установок, та також є можливість обробляти одну, а не пару таблиць.

Генератор програми обробки деталей – це програма, що формує програму обробки деталей на базі маршрутів обробки деталей. Зв'язок між ними дуже тісний. Маршрути обробки задаються у базі даних та визначають загальні принципи обробки кожної партії деталей, але вони не визначають конкретних агрегатів (а лише типи агрегатів), що оброблятимуть деталі партій. Для визначення конкретних обробників і потрібна програма обробки деталей, яку генерує ця підсистема. Згенерована нею програма обробки деталей зберігається у вигляді файлу, який можуть використовувати інші підсистеми.

Підсистема генерування схеми послідовностей запуску деталей будується на таких принципах:

- наочність (під час побудови схеми вона віддзеркалюється СПЗД графічно);
- автоперіодизація: система автоматично добудовує послідовність, якщо знайде закономірність у введених користувачем даних (тобто користувач не повинен задавати однотипні дані для всіх 50 деталей партії, а може ввести їх для 3-х чи 4-х);
- автогенерація: система автоматично може генерувати послідовності за якимись базовими, закладеними у ній, принципами.

Вихідні дані видаються підсистемою у вигляді послідовності запуску деталей на кожному з агрегатів.

Підсистема моделювання обробки деталей здійснює моделювання за принципом особливих становищ. На основі прийнятої інформаційної моделі було побудовано структуру класів та розподілено між ними функції, що покладені на підсистему відповідно до їх ролей.

Об'єкти взаємодіють за наступним принципом: є активний об'єкт, що викликає взаємодію (той об'єкт, для якого настало особливе становище) – цей об'єкт виконує певні дії відповідно до особливого становища, що настало; а також пасивний об'єкт чи об'єкти, що також можуть виконувати дії, але ці дії ініційовані повідомленнями від активного об'єкта. Уся логіка роботи підсистеми криється у цих діях, що виконуються під час особливих становищ та у реакціях на повідомлення інших об'єктів. На мал. 7. показані об'єкти, що працюють на дільниці у спрощеному її уявленні, та їхня взаємодія.

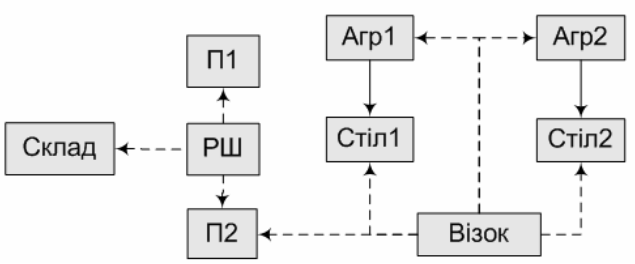

Рисунок 7 - Об’єкти імітаційної моделі та їхня взаємодія

Тут деталі на ділянку надходять через першу позицію (П1), готові деталі також забираються з П1, робот-штабелер (РШ) обслуговує склад, П1 та П2, візок обслуговує П2, агрегати (Агр1, Агр2) та колоагрегатні столи (Стіл1, Стіл2). При такій схемі взаємодій найбільше навантаження припадає на транспортні засоби, які найчастіше є активними об’єктами. Але запорукою успішної її реалізації є зрозумілість схем взаємодії та однозначність дій, що пов’язана з наперед зазначеними схемами запуску (програма обробки деталей, СПЗД).

Висновки

Таким чином, було описано побудову однорівневої інформаційної системи машинобудівного підприємства. Зазначено аспекти системи, з точки зору виконуваних дій, представлення даних та програмної реалізації. Побудована система стає об’єктом експериментального дослідження, що допоможе перевіряти теоретичні ідеї на практиці.

Наступним кроком цієї роботи повинен стати перехід до багаторівневої моделі (ділянка-цех-підприємство).

Література

1. ИММОД-2003: Материалы Всеросс. науч. конф. в г. Санкт-Петербурге, 23-24 октября 2003 года / УГАТУ; Загидуллин Р. Р. – СПб, 2003.
2. ИММОД-2003: Материалы Всеросс. науч. конф. в г. Санкт-Петербурге, 23-24 октября 2003 года / ОрелГПУ; Лазарев. – СПб, 2003.
3. Буч Г., Рамбо Дж., Якобсон А. Язык UML: руководство пользователя./ Пер. с англ. – С-Пб.: Питер, 2003. - 432с.
4. Вендров А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. – М.: Финансы и статистика, 1998. – 176 с.
5. Трофимов С.А. CASE-технологии: практическая работа в Rational Rose. Изд. 2-е. – М.: Бином-Пресс, 2002. - 288 с.
6. Коробецький Ю. П., Рамазанов С. К. Імітаційні моделі у гнучкому виробництві. Монографія. – Луганськ: Вид-во СНУ ім. В. Даля, 2003. – 280 с.Дата надходження до редакції 15.12.2007 р.

Дата надходження до редакції 20.12.2007 р.