УДК 681.3

# Концепція створення прикладних програмних систем з розвинутою функцією життєздатності

*Ігнатенко П.П., Стрєлов І.А., Ткаченко В. О., Дуднік Р. О.*
Інститут програмних систем НАН України
Україні, 03187, Київ, пр. Академіка Глушкова, 40
Тел.: +380 (44) 266 15 40
Факс: 380 (44) 266 62 63
e-mail: ignat@isofts.kiev.ua

Розглянуто основні аспекти створення прикладних програмних систем (ПС), що впливають на їх життєздатність на стадії функціонування та запропоновано концепцію технології створення ПС з розвинутою функцією життєздатності.

## Вступ

Практичні проблеми, що виникають на стадії супроводження функціонуючих ПС [1], викликають необхідність аналізу технології розробки та створення ПС щодо проблеми врахування їх еволюції на цій стадії, відзначаючи явні технологічні прогалини в цьому аспекті.

Не підлягає сумніву, що ПС в процесі функціонування та супроводження зазнають значних змін, природою яких може бути або уточнення моделі предметної області, що закладена в ПС, або зміна нефункціональних вимог користувача. Аналіз існуючих технологій створення ПС показує, що ці проблеми, як правило, не знайшли в них відображення або ж відображені лише частково.

Цю прогалину в технологіях останнім часом заповнюють численні публікації закордонних та вітчизняних наукових видань щодо відновлення знань про ПС та підходів до проведення вимушеного реінжинірингу ПС на стадії їх функціонування та супроводження [2-5].

Згідно пропонованого нами підходу [3-5] проблеми супроводження ПС необхідно вирішувати на стадії їх розробки та створення, доповнюючи середовище функціонування ПС засобами, що забезпечують їх життєздатність при функціонуванні та супроводженні.

При цьому засоби забезпечення життєздатності ПС при функціонуванні можуть бути двох типів:

- засоби настроювання ПС на зміни без перепроектування та перепрограмування їх компонентів;
- засоби забезпечення та підтримки реінжинірингу ПС, що реалізовує необхідні зміни.

Ми розглядаємо змінні (evolutional) – системи, в основі яких лежить модель предметної області - частина реального світу, які змінюються у відповідності з нею [1,6]. Ці системи дуже мінливі, оскільки їх проблема (предметна область) також може змінюватися. Зазнаючи активних функціональних змін, системи цього класу вимагають, щоб відповідні механізми були вмонтовані всередину системи для пристосування хоч б до деякої частини цих змін.

Крім змін з метою забезпечення точності функціонального рішення проблеми, розглядувані системи можуть також змінюватися, щоб забезпечити необхідні нефункціональні вимоги (*можливість повторного використання (reuseability), зручність підтримки (maintainability),* економічність *(economy),* надійність *(reliability),* ефективність *(performance)* тощо).

Найбільш розповсюдженими системами такого типу являються *автоматизовані системи організаційного керування* [7].

У найбільш повному обсязі системи організаційного керування формально описані в [8]. Програмні системи, що їх автоматизують, являють собою сукупність інформаційних комп'ютерних технологій, елементами яких є проектна та робоча документація, функціональні документи, бази даних, програми, повідомлення тощо [9].

Концепції технології створення (з забезпеченням життєздатності) саме таких прикладних програмних систем присвячена дана робота.

## 2. Вивчення предметної області та створення моделі класу ПС

Пропонований нами підхід передбачає класифікацію предметних областей, виділення для них класів ПС та створення для цих класів ПС їх UML-моделей.

Крім того, для класу ПС створюються *моделі простежування змін* та основні *сценарії* їх функуціонування [10,11].

Відповідно до архітектури процесів життєвого циклу *модель простежування змін* підтримує процес керування конфігурацією та, зокрема, процес внесення змін. Вона визначає зв'язок між артефактами ПС відповідно до різних етапів життєвого циклу та процесу розробки, а також механізми, що зумовлюють процес перетворень архітектури ПС.

Концепція сценарного підходу розглядає *сценарій* як базовий артефакт, що підтримує варіантність ПС та визначає механізми, що забезпечують варіантність проектних рішень, артефакти, що підлягають оцінюванню, конкретизує модель життєздатності на рівні мікропроцесу та макропроцесу об'єктно-орієнтованої розробки в рамках ітеративного життєвого циклу.

UML -модель ПС підтримує процес керування життєздатністю та процес її забезпечення. Вона визначає атрибути життєздатності ПС, відповідно до нефункціональних вимог до системи, внутрішні та зовнішні атрибути життєздатності ПС, а також зв'язок між ними, внутрішні та зовнішні метрики для вимірювання оцінок атрибутів життєздатності ПС тощо.

Наприклад, модель UML класу програмних систем CRM (Customer Relationship Management – керування взаємовідносинами з клієнтами [12,13]), над якою автори в даний час інтенсивно працюють, включає ***описи типових класів*** та ***описи типових діаграм взаємодій*** для *підсистеми продажів*; *підсистеми контролю роботи персоналу*; *підсистеми "Органайзер"*.

Названі підсистеми складають типову CRM-систему "початкового рівня", яка тим не менш є цілісною системою, аналоги якої є на ринку України.

Всі програмні системи для класу CRM-систем, що будуть створюватися в ближчий час в Україні, в основному, описуються цією UML –моделлю за винятком деяких особливостей, які виявляються при обстеженні конкретного об'єкту автоматизації.

## 3. Обстеження конкретного об'єкту автоматизації та створення моделі ПС

На перших етапах проектування ПС розроблювач і замовник взаємодіють між собою в обстеженні об'єкту автоматизації. Розроблювач зацікавлений у найбільш точному з'ясуванні процесу функціонування і вимог замовника до розроблюваної ПС, а замовник повинний бути зацікавлений у тому, щоб найбільше повно передати розроблювачу свої знання про предметну область і вимоги до майбутньої ПС.

Обстеження має бути направлене на конкретизацію моделі класу ПС для одержання моделі ПС, що проектується. Для вирішення цих проблем, та з метою підвищення якості створюваної ПС має бути розроблена спеціальна методика обстеження. Така методика обстеження є організаційно-методичним документом, що розробляється для досягнення наступних цілей:

- створення єдиної методологічної основи взаємодії розроблювача і замовника при обстеженні предметної області об'єкта автоматизації та створення його UML-моделі;
- забезпечення найбільш повного охоплення об'єкту автоматизації при обстеженні і відбитка результатів обстеження предметної області в змінах до UML-моделі класу ПС, змінах до моделей простежування змін класу ПС та змінах до сценаріїв функціонування класу ПС;
- забезпечення мінімальних тимчасових і трудових витрат замовника в процесі обстеження без шкоди для якості проведених робіт.

Використання методики дозволяє:

- систематизувати дані, отримані в результаті кожного етапу аналізу і проектування системи, подати їх у наочному і найбільше зручному для подальшого використання виді;
- спростити і полегшити передачу результатів роботи одних фахівців іншим;
- підвищити відповідальність розроблювачів будь-якої кваліфікації за якість виконуваної роботи;
- забезпечити порозуміння фахівців розроблювача і замовника;
- забезпечити взаімозамінюваність розроблювачів;
- впорядкувати планування і контроль за термінами виконання робіт.

Методика містить опис послідовності дій фахівців, що проводять обстеження об'єкту автоматизації, послідовності їхньої взаємодії з замовником, засоби і методи проведення обстеження, форми регламентних документів, що заповнюються при обстеженні і правила їхнього заповнення, а також вказівки по оформленні результатів обстеження. Дані обстеження фіксуються у вигляді змін до моделі класу ПС. На їх основі створюється UML-модель ПС об'єкту автоматизації.

Наприклад, при обстеженні конкретної CRM – системи, можуть бути відсутні деякі класи типової CRM – системи (наприклад, клас важливих дат; клас-нагадувач про важливі дати; клас електронний стікер), можуть бути добавлені деякі класи (наприклад, клас розкладу (на день, на період), клас, який будує розклади). Характеристики практично всіх класів типової CRM – системи будуть змінені у відповідністю з результатами обстеження. В результаті UML – модель конкретної CRM – системи набуває зміненого вигляду та служить основою для одержання оцінок щодо майбутньої ПС.

## 4. Моделювання та оцінювання характеристик життєздатності ПС

Як уже відмічалося, в зв'язку з великою мінливістю ПС, що розглядаються, виникає необхідність в їх пристосовності до цих змін з метою забезпечення довготривалості еволюційного етапу їх функціонування.

За класифікацією характеристик якості ПС, наведеної в ISO/IEC 9126-1 до характеристик життєздатності ПС можна віднести *супроводжуваність* (*maintainability*) – властивості ПС, що обумовлюють можливість її ефективної модифікації, включаючи коригування, удосконалення або адаптацію ПС до зміни середовища, вимог та функціональних специфікацій, а також *переносимість (portability)* – властивості ПС, що обумовлюють її здатність бути перенесеної з однієї середи до іншої.

Основними підхарактеристиками якості ПС, що забезпечують необхідну життєздатність при змінах функціональних та нефункціональних вимог, зокрема, являються наступні [14,15]:

- аналізованість;
- адаптованість (модернізованість);
- стабільність;
- тестованість.

При цьому, *аналізованість (analyzability)* - це властивості ПС, що обумовлюють здатність діагностування її недоліків або чинників відмов, а також ідентифікації частин, які потрібно модифікувати. Один з шляхів підвищення аналізованості програми –обробка всіх можливих виключних ситуацій з видачею зрозумілих користувачеві повідомлень та створення в ПС режиму логування, тобто ведення журналу подій в якому відображалися б усі збої ПС, час їхнього виникнення, їх зміст та, якщо це можливо, їх причина. *Адаптованість – модернізованість (changeability)* – це властивості ПС, що обумовлюють її здатність виконувати встановлені види модифікацій. Слід враховувати, що будь-яку ПС можливо змінити довільним чином, і питання модифікації є лише питанням часу (а тому й грошей), що витрачається на модернізацію. Систему слід вважати ідеально модернізованою, якщо будь-яку зміну (що не вимагає збільшення функціональності) можна провести маніпулюючи даними (або мета даними) за допомогою штатних засобів системи. В рамках концепції забезпечення життєздатності ПС дана система буде називатися системою з ідеальною адаптивністю. *Стабільність (stability)* – це властивості ПС, що обумовлюють її здатність мінімізувати неочікувані ефекти модифікацій. Для забезпечення стабільності ПС має мати якомога меншу взаємопов'язаність класів. *Тестованість (testability)* – властивості ПС, що обумовлюють її здатність допомагати перевірці ПЗ, що модифікується. Дана властивість залежить від системи тестування та ПЗ, що застосовується для тестування.

Зазначені характеристики життєздатності можуть бути оцінені на ранніх стадіях розробки ПС за допомогою деяких метрик (чи їх комбінацій) та UML – моделі майбутньої ПС.

Так, *аналізованість ПС*, може бути оцінена з допомогою метрики, яка вимірює відношення кількості методів (як тих процедур та функцій що є членами класів так і тих що не є ними), в яких відбувається обробка виключних ситуацій, до загальної кількості методів в ПС (*оброблюваність виключних ситуацій*). Різновидом цієї метрики можна вважати відношення кількості методів ПС в яких реалізовано можливість запису до журналу подій ПС до загальної кількості методів ПС (*показник логування*). *Адаптованість – модернізованість* можна оцінити за допомогою відношення кількості змінних та бізнес-правил предметної області які можна змінити за допомогою штатних засобів системи до загальної кількості змінних, констант та бізнес-правил предметної області (*модернізованість як адаптованість*) [16]. З визначення *стабільності ПС* витікає, що ПС з меншим рівнем взаємопов'язаності класів буде більш стабільною, ніж з більшим рівнем взаємопов'язаності. Звідси можна зробити висновок, що більш стабільними будуть ті ПС, які побудовані за допомогою шаблону GRASP Low Coupling. Для виміру зв'язаності використовуються наступні метрики: *CBO* (зв'язування між об'єктами), *CBOin* (зв'язування між об'єктами в ієрархії наслідування), *CBOout* (зв'язування між об'єктами не включаючи ієрархію наслідування) [17, 18]. При вимірюванні *тестованості* можливі наступні варіанти.

Якщо в якості системи тестування ПС прийняти систему тестування методом екстремального програмування (тобто тестування відбувається шляхом написання тестових класів в яких відбувається перевірка класів ПС) [18], то в якості метрики тестованості можна прийняти відношення кількості класів ПС до яких написані тестові класи до загальної кількості класів ПС. Дана метрика буде виміряти *загальну тестованість ПС*.

Але оскільки багато класів ПС виконують допоміжні функції і може виникнути потреба оцінити тестованість ПС насамперед з точки зору предметної області (наприклад нас може не цікавити тестованість графічного інтерфейсу користувача, але цікавить правильність виконання бізнесових обрахунків, чи інших функцій системи з переліку функціональності ПС), то в цьому випадку використовується така метрика як *тестованість ПС в термінах предметної області*: відношення кількості функцій системи (з загального переліку функціональності ПС), які перевіряються тестовими класами, до загальної кількості функцій ПС.

Схема моделювання та одержання варіанту проекту ПС, який має оцінки підхарактеристик життєздатності, що задовольняють вимоги користувача ПС зображена на рис.1.

На *стадії розробки* пропонований підхід передбачає:

- проведення моделювання ПС в основних (необхідних) аспектах та вибір кращого варіанту моделі ПС для подальшої реалізації на основі вхідних даних від предметної області, створеної UML - моделі, вимог користувача;

- одержання оцінок характеристик життєздатності ПС та їх аналіз в аспекті виконання нефункціональних вимог користувача, а за їх прийнятності - вироблення проекту ПС;
- реалізація проекту ПС;
- передача ПС в експлуатацію разом із засобами її моделювання.

На *стадії функціонування та супроводження* пропонований підхід передбачає:

- визначення змін в UML -моделі на основі змінених вимог користувача та (або) змін в предметній області;
- проведення аналізу необхідних змін в ПС, за допомогою моделей простежування змін визначаються необхідні зміни в UML -моделі, проводиться їх оцінка (за схемою), реалізація та імплементація в проект ПС та саму ПС з використанням технології та засобів ефективного реінжинірингу.

Основою запропонованого підходу до забезпечення можливості одержання оцінок варіантів реінжинірингу ПС є одержання оцінок на відповідність шаблонам GRASP, а також класифікація та специфікація спеціальних ситуацій. Шаблони GRASP містять базові принципи щодо розподілення обов'язків між об'єктами при створенні діаграм взаємодій стосовно певної проблеми, яка повинна вирішуватися. Ці проблеми та їх рішення відповідають п'яти шаблонам GRASP, які вирішують основні питання проектування [19].

Аналіз шаблонів GRASP вказує на те, що застосування цих шаблонів під час розробки діаграм взаємодій складає певну задачу їх оптимального використання та приводить до створення ПС, для яких характерна наявність наступних властивостей:

- підтримка інкапсуляції, надійність та пристосованість до обслуговування (Expert, Low Coupling);
- простота розуміння та підтримка визначень класів, спрощення підтримки та внесення змін (Expert, High Cohesion).
- зменшення витрат на супроводження та можливість повторного використання (Creator, Low Coupling);
- покращення умов для повторного використання компонентів щодо логіки обробки процесів предметної області на рівні реалізації, спрощення підтримки та доробки (Controller, High Cohesion);
- можливість контролю стану прецеденту щодо виконання системних операцій у визначеній послідовності (Controller, Expert, High Cohesion).

Кваліфіковане розподілення обов'язків між компонентами ПС за допомогою шаблонів GRASP приводить до створення ПС, компоненти яких відповідають необхідним вимогам до їх підтримки, повторного використання, розширення, розуміння, а також ефективності та надійності. Ці вимоги можна розглядати як такі, що відносяться до нефункціональних атрибутів компонентів ПС в контексті розробки сценарію та відповідних діаграм взаємодій. Нефункціональні атрибути можна характеризувати можливим набором значень та обмеженнями цих значень. Такі обмеження визначаються із загальних обмежень на розробку програмної системи. Інакше кажучи, існує проблема оптимізації використання шаблонів відповідно до існуючих вимог та обмежень на розробку як компонентів програмної системи, так і системи в цілому. Ця проблема має розв'язуватися виходячи з наявності альтернативних варіантів розробки компонентів програмної системи, бази знань про параметри оцінки сценарію, які зберігаються у репозитарії та накопичуються відповідно до ітеративного процесу розробки системи та внесення змін.

Параметри оцінки сценарію визначаються залежно від типу об'єктів та їх взаємодій відповідно до бізнес-логіки сценарію в контексті певного шаблону GRASP. Питання визначення та класифікації параметрів оцінки сценарію складають окрему задачу і значною мірою відповідають проблемі створення і використання метрик для оцінки компонентів ПС з урахуванням якості програмного продукту та характеристик процесу розробки й реінжинірингу ПС.

Для оцінки сценарію на відповідність шаблонам GRASP кожному шаблону необхідно співставити певні параметри оцінки сценарію та спеціальні ситуації, створення яких супроводжується обробкою цих параметрів на етапі функціонування ПС.

Аналіз та оцінка базових артефактів ПС, зокрема варіантів сценаріїв, яким відповідають певні діаграми взаємодій, повинен завершуватись встановленням зв'язку між їхніми оцінками (на основі внутрішніх метрик ПС) та зовнішніми атрибутами ПС (функціональними та нефункціональними). Це приводить до створення інтегральних оцінок сценарію та ПС у цілому, які повинні відповідати, зокрема, таким нефункціональним вимогам до ПС, як відмовостійкість, час відгуку, ефективність транзакцій бази даних, вартість, потужність потрібних обчислювальних ресурсів тощо.

Актуальними задачами вирішення цієї проблеми для об'єктно-орієнтованих ПС є створення засобів одержання таких оцінок на основі аналізу відповідності ПС шаблонам проектування із використанням діаграм UML-моделей та відповідних інструментальних засобів, таких, як Rational Rose [20]. Для всіх інших ПС, що розробляються не методом компонентного програмування, задача значно складніша. Для цього необхідно

спочатку побудувати їх модель, використовуючи відповідні інструментальні засоби моделювання, а потім створити засоби одержання таких оцінок.

## 5. Моделювання та оцінювання економічних характеристик ПС

Нарівні з оцінюванням характеристик життєздатності ПС дуже важливим аспектом розробки та супроводження ПС є оцінювання економічних характеристик, пов'язаних з її якістю.

Найбільш важливими економічними характеристиками проекту ПС для управління є *розмір проекту, трудомісткість, час, необхідний для виконання та вартість*. Серед цих характеристик три останні у великій мірі залежать від першої характеристики – *розміру* ПС. Тому оцінювання цих характеристик можна розбити на два етапи:

- оцінювання розміру;
- оцінювання трудомісткості, вартості та тривалості з використанням вже отриманого значення розміру.

Для розрахунку оцінок трьох останніх характеристик добре підходить модель COCOMO [21], оскільки вона розроблялася саме з метою оцінки бюджету, вартості та тривалості проектів та є зараз найбільш широко застосовуваною моделлю. Модель COCOMO використовує як вхідний параметр *розмір* ПС у рядках вихідного коду (SLOC [22]). Отримати точний розмір у таких одиницях можна лише після завершення проекту. На будь-якому попередньому етапі розмір має визначатися за допомогою експертних оцінок або за допомогою деякої методики оцінювання розміру проекту, яка може дати оцінку в одиницях SLOC. В той же час найкращим вибором для оцінювання розміру проекту є метод FPA [23]: він має об'єктивний характер, простий у застосуванні, є найбільш широко застосованим у індустрії розробки ПЗ, має статус міжнародного стандарту та є загально доступним. Розмір ПС, отриманий за допомогою FPA, можна використати у моделі COCOMO, оскільки для багатьох мов програмування існують залежності між розміром ПС у одиницях UFP та одиницях SLOC [24].

Існуючі методи експертних оцінок також включені в схему, оскільки застосування більш формалізованих методів у певних випадках може бути неможливим або невигідним.

Схема поєднання згаданих методів приведена на рис. 2.

**Організаційні аспекти підходу до оцінювання проектів.** Схема пропонованої організаційної схеми процесу оцінювання представлена на рис. 3.

За цією схемою, процес оцінювання ділиться на чотири етапи:

а) *підготовка* - вибір методу оцінювання, планування процесу оцінювання;

б) *виконання* - підготовка вхідних даних, оцінювання розміру, оцінювання зусиль та вартості, оцінювання термінів виконання проекту, підготовка звіту про оцінювання;

в) *перевірка* - перевірка отриманих результатів, збір даних про процес оцінювання;

г) *аналіз та покращення процесу оцінювання* - зміна процесу оцінювання.

**Особливості застосування методу FPA.** Для успішного використання в пропонованому підході методу FPA необхідно попередньо оцінити такі його характеристики:

*стабільність оцінок*, які дає цей метод;

*узгодженість оцінок із статистикою по індустрії ПС;*

*узгодженість оцінок між проектами*.

Підхід використовує припущення про існування співвідношення розмірів проектів у одиницях SLOC та одиницях UFP. Причому, величина цього співвідношення для різних проектів є однаковою. Ці припущення були підтверджено експериментально.

Оцінки функціонального розміру можуть бути отримані з UML-діаграми системи, оскільки мова UML містить засоби для відображення всіх ключових елементів системи, які розглядаються у методі FPA. Єдиним елементом, для якого немає прямої відповідності, є межа системи, але вона необхідна лише для ідентифікації інших елементів, тому немає потреби зображувати її у явному вигляді. Для всіх інших елементів існує пряма відповідність. Так, файли системи, як зовнішні, так і внутрішні, відповідають пакетам (package) UML. Типи записів та інтерфейси відповідають класам (class), транзакції (ввід, вивід, та запит) відповідають взаємодії між класами. Отже, на діаграмі класів (class diagram) можна зобразити як класи системи, які реалізують функціональність системи, так і файли, зовнішні інтерфейси та окремі типи записів (рис. 4), причому класи, які описують типи записів, містяться всередині пакету, який зображує відповідний файл (рис. 5). Для опису транзакцій (ввід, вивід, та запит) використовується діаграма послідовності (sequence diagram) або діаграма взаємодії (collaboration diagram) (рис. 6).

Для того, щоб повністю відобразити елементи системи за FPA, необхідно дотримуватись певного рівня деталізації та формату іменування та позначення цих елементів. Це дещо обмежує ту свободу вираження, яку дає UML, але позитивними рисами такого підходу є більша структурованість діаграм, більш чітке та повне зображення системи як у аспекті її функціональності, так і в аспекті даних, з якими вона працює. Наявність певної структури діаграм дає змогу отримувати оцінки розміру системи за FPA автоматично та неперервно у ході створення та супроводження ПС. Відповідно, можуть бути отримані значення й інших економічних

характеристик. У поєднанні з методами визначення характеристик якості системи за UML-діаграмами це дасть нові можливості для керування проектами з розробки, модифікації та реінжинірингу ПС.

**Особливості застосування моделі COCOMO. Калібрування параметрів.** Модель COCOMO дає змогу отримати більш точні результати, якщо відкалібрувати її за даними по історичним проектам, тобто визначити такі значення параметрів моделі, які б відображали особливості виконання проектів в умовах конкретної організації.

У рівнянні COCOMO для оцінки бюджету проекту присутні дві калібровочні константи. Допустимо, у організації є дані по *n* проектах.

Примінивши запропонований алгоритим [24], можна отримати значення калібровочних констант для певних умов виконання проектів.

Комплексну схему виконання пропонованого підходу відображає рис. 7.

При одержанні оцінок економічних характеристик ПС, що не задовольняють користувача із-за їх великих значень, необхідно повернутися до етапу моделювання та оцінювання характеристик життєздатності ПС, понизити вимоги до характеристик життєздатності, одержати компромісний проект та одержати задовільні оцінки його економічних характеристик.

## 6. Створення та супроводження ПС з розвинутою функцією життєздатності

Можливість створення та застосування засобів забезпечення життєздатності ПС в контексті об'єктно-орієнтованої технології розробки ПС і створення відповідного середовища розробки визначається наступними факторами. Це застосування *шаблонів проектування* [19] як таких, що визначають загальні принципи розробки об'єктно-орієнтованих програмних систем, використання *мови моделювання UML* [20], яка реалізує метод об'єктно-орієнтованого аналізу й проектування та інструменту *Rational Rose* [20], який є інструментом аналізу та проектування об'єктно-орієнтованих програмних систем на основі мови UML, використання сценарного підходу [10] для формування та вибору альтернативних рішень щодо проекту ПС.

На даний час у галузі застосування об'єктно-орієнтованої технології розробки ПС на основі UML вироблені та сформульовані загальні принципи та стандартні рішення, які удосконалюють розробку ПС. Ці принципи та ідіоми систематизовані і структуровані у вигляді *шаблонів* (patterns). Серед широко відомих шаблонів можна назвати такі, як GRASP (General Responsibility Assignment Software Patterns - загальні шаблони розподілення обов'язків у ПС) і GoF (Gang of For - союз чотирьох) [25].

Шаблони GRASP використовуються у процесі створення діаграм взаємодій при розподіленні обов'язків між об'єктами та розробці способів їх взаємодії. Діаграма взаємодій є найважливішим видом артефактів в процесі розробки об'єктно-орієнтованих ПС. Такі види діяльності при розробці ПС, як створення прецедентів і відповідних сценаріїв, що знаходяться у відношенні “клас/об'єкт”, реалізація сценаріїв у вигляді кооперацій і відповідних діаграм взаємодій, що знаходяться у відношенні “інтерфейс/реалізація”, є основними на різних етапах створення об'єктно-орієнтованих ПС. Їх успішна реалізація визначає успіх проекту у цілому. При цьому стратегія внесення змін на етапі як розробки, так і впровадження та супроводження ПС повинна коригуватися відповідно до існуючих вимог до системи та існуючих обмежень, які можуть змінюватися під час ітеративного та інкрементного процесу розробки системи, а також її впровадження та супроводження.

На основі аналізу та простежування відповідності ПС існуючим шаблонам запропоновано підхід до створення засобів забезпечення життєздатності ПС, який детально описано в [11].

## Висновки

Проблеми створення життєздатних ПС є актуальними та важливими для забезпечення їх довготривалого функціонування та зменшення витрат на стадії супроводження..

В існуючих технологіях основна увага приділяється розробці ПС з фіксованим варіантом їх функціональних та нефункціональних вимог. Проблеми еволюції систем` на стадії функціонування практично не розглядаються. Отже, ПС передаються в експлуатацію, як правило, без засобів забезпечення їх життєздатності.

Відсутність розвинутої функції забезпечення життєздатності ПС, що передаються в експлуатацію, призводить до значних втрат при спробах врахування виникаючих змін на стадії функціонування та супроводження, або навіть до припинення їх експлуатації.

Концепція технології створення ПС, що враховує проблеми забезпечення життєздатності, дозволить доповнити технології розробки ПС формалізованними моделями, CASE-засобами та інструментарієм, що в сукупності забезпечують створювані ПС розвиненою функцією життєздатності.

1. Ігнатенко П.П. Проблеми забезпечення життєздатності програмних систем та підходи до їх вирішення // Пробл. Програм. – 2002. - №3-4. – С .58-73/
2. Software Reengineering// Ed. Robert S. Arnold // IEEE Comput. Soc. Press, 1994. — 676 P.
3. Ігнатенко П.П., Неумоїн В.М., Бистров В.М. Аспекти реінжинірингу програмних систем // Пробл. программирования. – 2000. - №1-2. – С. 367-375.
4. Ігнатенко П.П., Неумоїн В.М., Бистров В.М. Про забезпечення ефективного реінжинірингу прикладних програмних систем // Там же. – 2001. - №1-2. – С. 42-52.
5. Підхід до забезпечення реінжинірингу об'єктно-орієнтованих програмних систем/ П.П. Ігнатенко, В.М.Бистров, І.О. Заєць, О.П. Ігнатенко // Пробл. программирования. – 2002. - №1-2. – С.98-108.
6. Pfleeger S.L.The Nature of Sustem Change // IEEE Software. –1998,- Maj/June. - P. 87-90.
7. Глушков В.М. Введение в АСУ. - Киев: Техніка, 1974 - 317 с.
8. Месарович М., Мако Д., Такахара И. Теория иерархических многоуровневых систем. - М: Мир, 1973.-344 с.
9. Глушков В.М. Основи безбумажной информатики.-Москва: Наука, - 1982. - 552 с.
10. Scenarios in System Development: Current Practice/ K. Weidenhaupt, K. Polh, M. Jarke, Haumer P // IEEE Software. - March/April.- 1998.- P. 34–35.
11. П.П. Ігнатенко, В.М. Бистров, О.П. Ігнатенко, В.М. Ткаченко Задачі та засоби моделювання і оцінювання життєздатних програмних систем// Пробл. Програм. – 2003. - №3. – С .59-70/
12. Дуднік Р.О. CRM –стратегія та її місце в управлінні компанією. Збірник наукових праць. Матеріали Міжнародної конференції "Інформаційні технології в банківській та страховій справі". Київ, 2002. С. 28-32.
13. Шрайбман І. Новые технологии взаимоотношениями с клиентами или как купить любовь за деньги (CRM и e-CRM). Збірник наукових праць. Матеріали Міжнародної конференції "Інформаційні технології в банківській та страховій справі". Київ, 2002. С. 33-40.
14. Андон Ф.И., Коваль Г.И., Коротун Т.М., Суслов В.Ю. Основы инженерии качества программных систем. – К.: Академпериодика, 2002. – 504с.
15. Бабенко Л.П., Лаврищева Е.М. Основи програмної інженерії. – "Знання" –Київ 2001, 269с.
16. S. Chidamber and C. Kemerer. A metrics suite for object-oriented design. IEEE Transaction Software Engineering, 20(6): 476-493, 1994.
17. S. Chidamber and C. Kemerer. Towards a metrics suite for object-oriented design. Conference on Object-Oriented Programming System, Languages and Applications (OOPSLA'91), pages 197-211, 1991.
18. Extreme programming applied: playing to win / Ken Auer, Roy Miller. Pearson Education, 2002.
19. Ларман К. Применение UML и шаблонов проектирования. – М.: Вильямс, 2001. – 496с.
20. Боггс У., Боггс М. UML и Rational Rose. – М.: "Лори", 2000. – 582с.
21. CSE, 1999 - Center for Software Engineering, " COCOMO II Reference Manual," Computer Science Department, USC Center for Software Engineering, 1999. – 86с.
22. Park.R Software Size Measurement: A Framework for Counting Source Statements // Tech. Report CMU/SEI-92-TR-020. - Software Eng. Inst., Pittsburg, 1992. – 242p.
23. COSMIC, 2003 – The Common Software Measurement International Consortium, The COSMIC FFP Measurement Manual. Version 2.2. // www.cosmicon.com. – 81с.
24. Стрєлов І.А., Ігнатенко П.П. Підхід до оцінювання економічних характеристик проектних рішень при розробці, модифікації та реінжинірингу програмних систем// Пробл. Програм. – 2004. - №1. – С .
25. Gamma E., Helm R., Johnson R., and Vlissides J Design Patterns, Elements of Reusable Object-oriented Software,- N.- Y.: Addison-Wesley, 1995. – 345p.

**Про авторів**
Ігнатенко Петро Петрович, кандидат технічних наук, завідувач відділом ІПС НАНУ;
Стрєлов Ігор Анатолієвич, аспірант ІПС НАНУ;
Ткаченко Володимир , аспірант ІПС НАНУ;
Дуднік Раїса Олексіївна, пошукач ІПС НАНУ.

Розглянуто основні аспекти створення прикладних програмних систем (ПС), що впливають на їх життєздатність на стадії функціонування та запропоновано концепцію технології створення ПС з розвинутою функцією життєздатності.

Рассмотрено основные аспекты создания прикладных программных систем (ПС), влияющие на их жизнеспособность на стадии функционирования и предложено концепцию технологии создания ПС с развитой фунцией жизнеспособности.

An overview is given for major aspects of software systems development that affect viability of the systems during usage and maintenance phase. A concept of technology for creation of advanced viability systems is proposed.

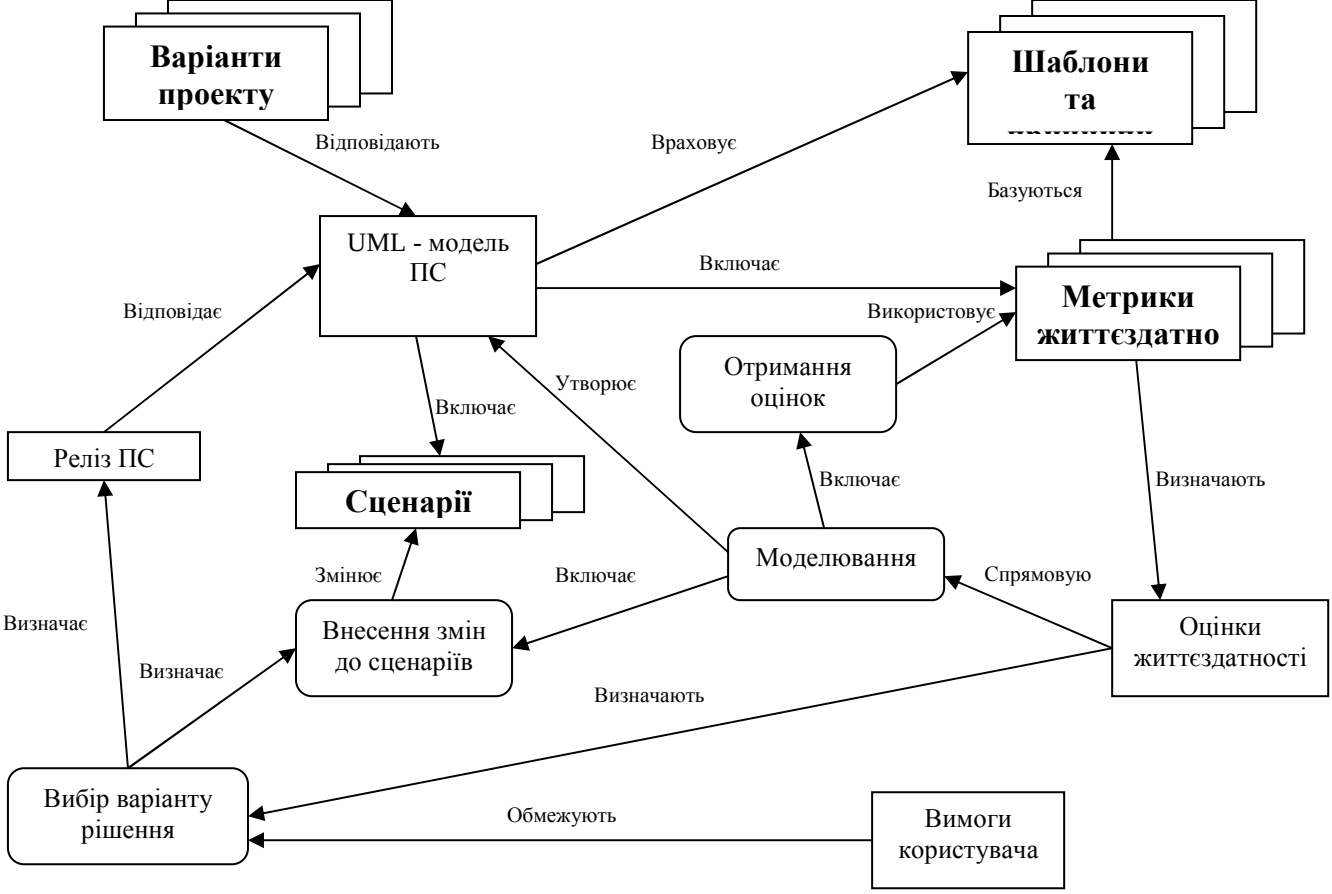


Рис. 1. Схема моделювання та оцінювання характеристик життєздатності об’єктно-орієнтованих ПС при їх створенні, модифікації та реінжинірингу

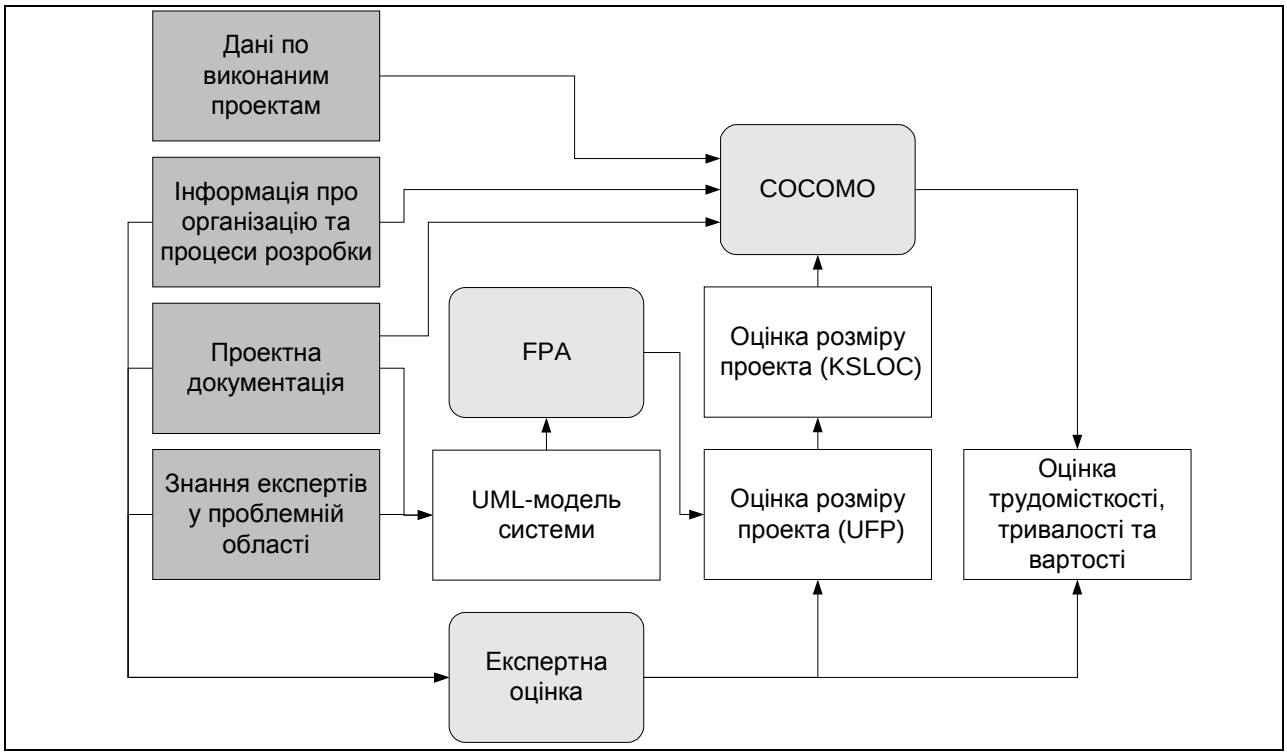


**Рис. 2. Оцінювання за допомогою методів FPA та COCOMO.**

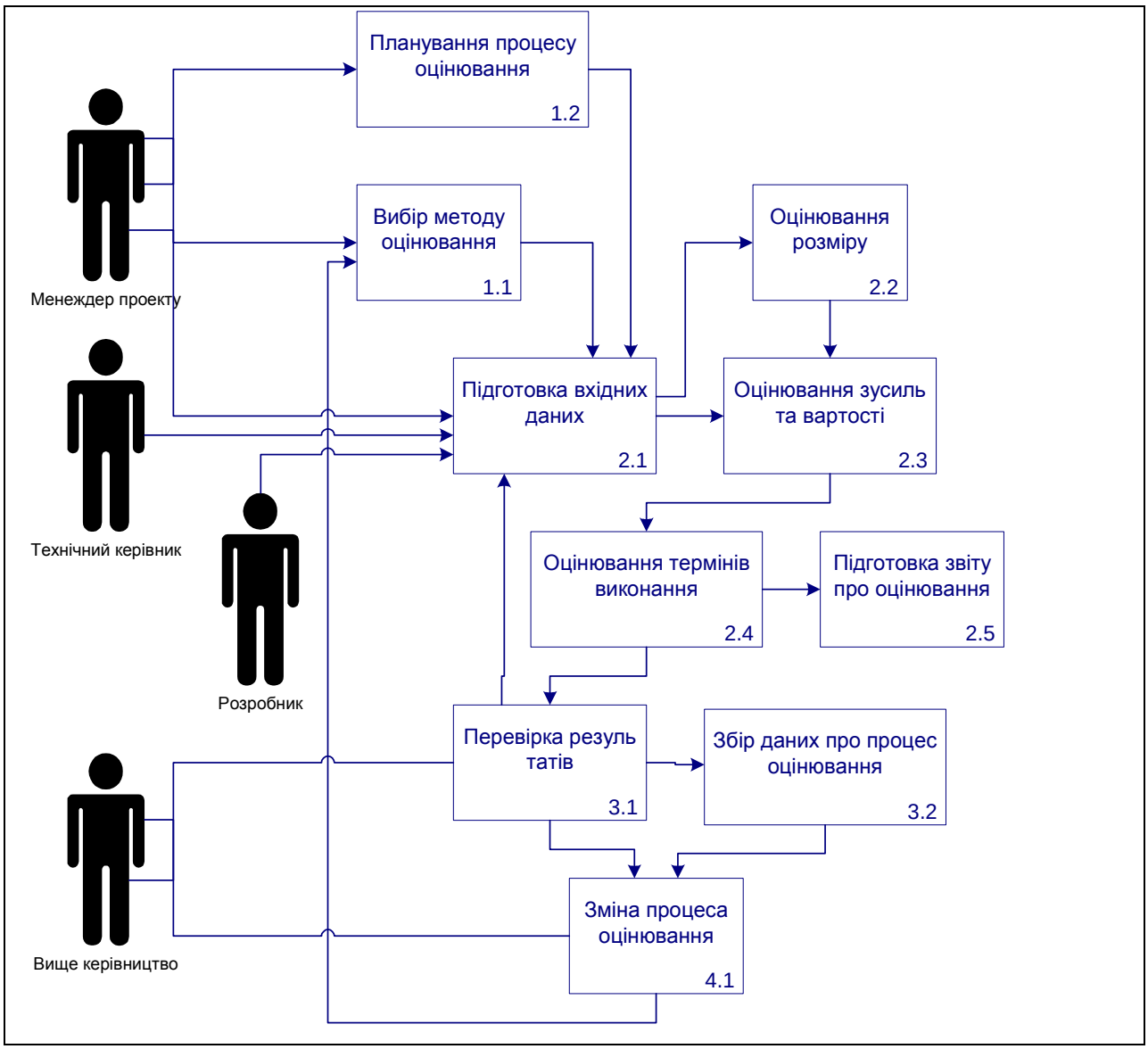


**Рис. 3. Схема процесу оцінювання проекту**


**Рисунок 4. Зображення внутрішніх логічних файлів (ВЛФ) та зовнішніх інтерфейсних файлів (ЗІФ) на UML-діаграмі.**

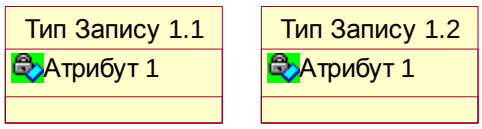


**Рисунок 5. Зображення типів записів всередині пакету**

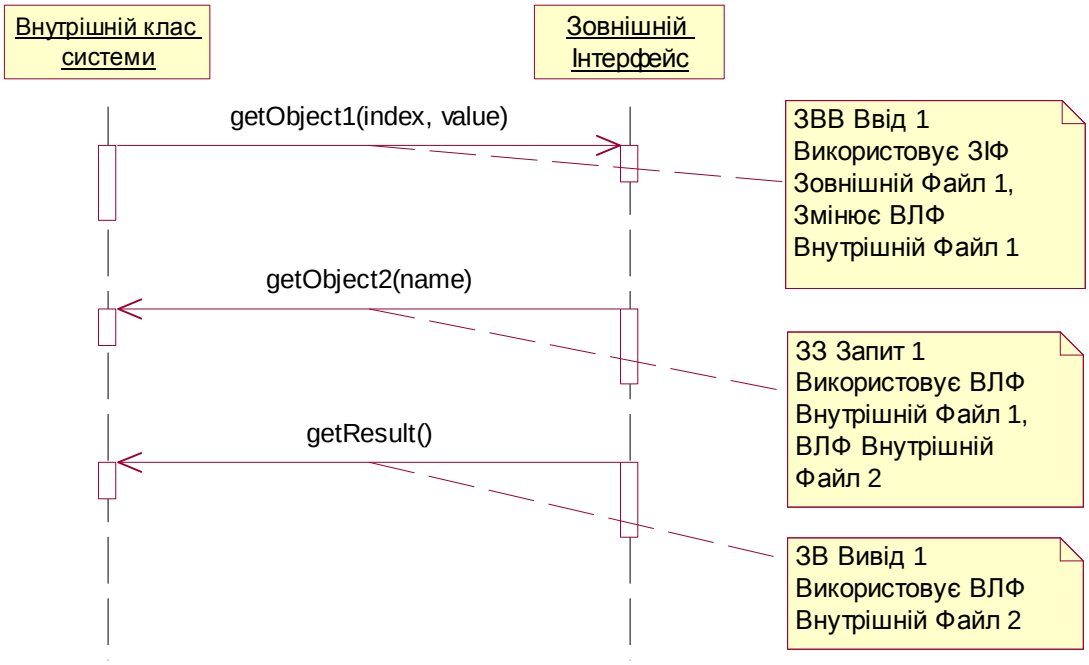


**Рисунок 6. Відображення трансакцій зовнішнього вводу (ЗВВ), зовнішнього виводу (ЗВ) та зовнішнього запиту (ЗЗ) на діаграмі UML**

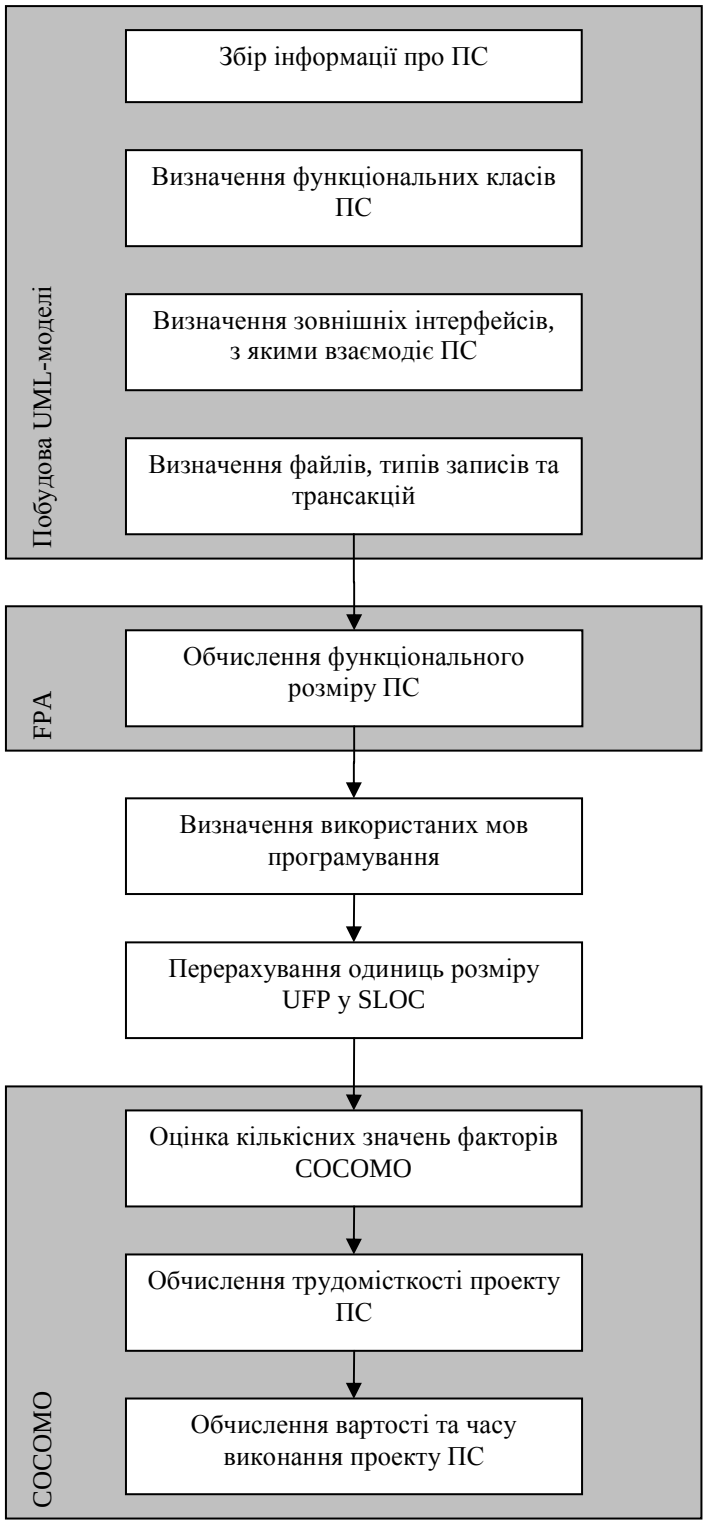


**Рис. 7. Комплексна схема оцінювання економічних характеристик ПС**