УДК 681.3.06

ЖИТТЄЗДАТНІ ПРОГРАМНІ СИСТЕМИ. КОНЦЕПТУАЛІЗАЦІЯ ПІДХОДУ ДО АВТОМАТИЗАЦІЇ СИСТЕМ ОРГАНІЗАЦІЙНОГО КЕРУВАННЯ

П.П. Ігнатенко

Інститут програмних систем НАН України
03187, Київ, проспект Академіка Глушкова, 40,
тел.: +380 (44) 526 15 40; факс: 380 (44) 526 62 63; e-mail: ignat@isofts.kiev.ua

Розглянуто концепцію нового підходу до створення життєздатних прикладних програмних систем, що автоматизують системи організаційного керування, який базується на принципах та основних положеннях програмної інженерії.

It is considered new approach concept to design viable software for automated administration systems, that is based on principles of software engineering.

Вступ

Проблеми життєздатності складних систем є актуальними з часів виникнення кібернетики як науки. Початок їх розробки пов'язують з роботою Р. Ешбі [1] в якій він здійснив спробу формалізувати поняття життєздатності складних систем, ввівши Закон Необхідної Різноманітності (The Law of Reguisite Variety). При цьому життєздатність він розумів як здатність системи при функціонуванні зберігати свої характеристики в заданих межах.

Деякі теоретичні питання життєздатності, а саме адаптивності та координованості в ієрархічних багаторівневих системах розроблені та опубліковані в монографії авторів М. Месарович, Д. Мако і Т.Такахара [2]. Розглядувана авторами багаторівнева система включає ієрархію керуючих підсистем С, а також керований ними процес Р. Розглядається та досліджується два види керуючих взаємодій. Один – це вироблення та передача зверху вниз "командних " сигналів, інший – це вироблення та передача наверх інформаційних сигналів, або сигналів зворотного зв'язку. В монографії розглянуті проблеми забезпечення життєздатності ієрархічних багаторівневих систем через вирішення математичних задач адаптивності та координованості на основі сигналів зворотного зв'язку від процесу Р.

Практичний підхід до вирішення розглядуваних проблем містять роботи С. Біра [3 - 4]. У його основу покладено запропоновану автором формалізовану модель складної життєздатної системи організаційного керування (Viable System Model). Така життєздатна система повинна складатися із сукупності визначених автором п'яти типів життєздатних підсистем. С. Бір у своїй моделі зазначив шлях створення життєздатних систем у промисловості та суспільстві, які б найкращим чином відповідали закону необхідної різноманітності Р. Ешбі, та детально проаналізував механізми "зменшення" (attenuation) і "збільшення" (amplification) різноманітності в життєздатних системах для вирішення можливих протиріч у них. Важливим висновком із моделі життєздатної системи С. Біра є те, що основна мета типових підсистем більш високого порядку – не "заважати" підпорядкованим підсистемам , а основна мета їх "втручання" (intervention) є забезпечення гомеостатичного регулювання більш високого порядку.

Звичайно, названі підходи стосуються життєздатності складних систем взагалі й напряму не застосовні до ПС, які мають свої суттєві особливості.

1. Огляд стану справ щодо створення життєздатних ПС

Не підлягає сумніву, що і ПС в процесі функціонування та супроводження потребують внесення значних змін, природою яких може бути або уточнення моделі предметної області, що закладена в них, або зміна вимог користувача, що розглядаються як нефункціональні вимоги до ПС [5].

У зв'язку з цим проблемі життєздатності ПС в науковій літературі по програмній інженерії приділяється велика увага та присвячено значну кількість публікацій. Причому, в цих публікаціях можна виділити декілька напрямків.

Один із них – розвиває стосовно ПС підходи, розроблені для забезпечення життєздатності складних систем шляхом інтелектуалізації програмного забезпечення системи [6 - 7], інтелектуалізації архітектури ПС та системи її керування [8].

В основу інтелектуалізації програмного забезпечення покладено підхід до створення самоадаптивного програмного забезпечення. При цьому підхід розширює область застосування моделей програм настільки, що вони стають основою процесу проектування прикладних ПС. В подальшому ці моделі транслюються у виконувані прикладні застосування.

До напрямку інтелектуалізації програмного забезпечення належить також і підхід створення ПС на основі агентно-орієнтованої технології [9].

В основі підходу до інтелектуалізації архітектури ПС та системи її керування Ч.Е. Херінгом [8] покладено Viable System Model С.Біра, яка автором модифікована для застосування до систем цього типу.

Побудовані аналоги підсистем Viable System Model для ПС, виділені їх особливості та запропонована інтелектуальна парадигма керування архітектурою ПС, яка дозволяє системі самоадаптуватися до виникаючих змін.

Інтелектуалізація програмного забезпечення, а також інтелектуалізація архітектури ПС та системи її керування суттєво ускладнюють архітектуру прикладних ПС, їх програмне забезпечення та підвищують вартість розробки, а отже ускладнюють процеси індустріалізації створення та впровадження інформаційних технологій у суспільство. Крім того, як би ми не інтелектуалізували програмне забезпечення прикладної ПС та систему керування її архітектурою, врахувати всі виклики при її функціонуванні неможливо, а отже неможливо уникнути необхідності її модифікації при супроводженні, затратність якої у цьому випадку із-за складності ПС суттєво зростає. Найбільш доцільно такі підходи застосовувати до автоматизації *постійних (stable)* систем, в основі яких лежить структурна математична (формальна) модель предметної області, що має точне рішення проблеми або *наближених (practical)* систем, в основі яких лежить формальна модель предметної області, що практично має лише наближене рішення проблеми [5].

Інший із існуючих підходів, що належить до підходів програмної інженерії – робить наголос на розвитку методів реінжинірингу та рефакторингу [10 - 14], які дозволяють для любої функціонуючої ПС, навіть такої, що не має актуальної експлуатаційної документації, відновити її проектні рішення та провести необхідну її модифікацію. Для висвітлення методів та засобів реінжинірингу та рефакторингу в рамках цього підходу функціонують міжнародна та європейська конференції під назвою “Software Maintenance and Reengineering”, які проходять кожні два роки (дивись, наприклад, www.rcost.unisannio.it/csmr2003). Методи та засоби реінжинірингу та рефакторингу фактично вирішують проблеми спасіння раніше створених і фактично нежиттєздатних ПС для продовження терміну їх функціонування.

Необхідності застосування особливо трудомістких методів та засобів реінжинірингу та рефакторингу можна уникнути, якщо створювати прикладні ПС з розвиненими функціями життєздатності. Особливо це актуально для прикладних ПС, що автоматизують системи організаційного керування (в подальшому будемо позначати цей клас систем як ППС) . Концепція такого підходу пропонується в даній роботі.

2. Особливості пропонованого підходу до створення та супроводження життєздатних ППС

Програмна інженерія згідно звіту міжнародного співтовариства ISEN (Interdisciplinary Software Engineering Network) відноситься до групи дисциплін, орієнтованих на людину [15].

У рамках цих дисциплін для досягнення життєздатності систем використовуються прості еволюціонуючі архітектури [16] та створюються конструкції, явно налаштовані до змін при функціонуванні. Тобто, для програмної інженерії найбільш природним є підхід, що розглядає спеціалістів в якості невід’ємної частини ПС не лише під час її проектування та створення, а й під час її супроводження при функціонуванні.

Це дозволяє для забезпечення життєздатності ППС робити наголос не стільки на інтелектуалізації її програмного забезпечення та системи керування її архітектурою, скільки на інтелектуалізації технологічних засобів її супроводження, забезпечуючи їх механізмами зворотного зв’язку для реагування на зміни вимог користувача чи зміни навколишнього середовища та здійснення необхідних модифікацій ППС.

2.1. Основні поняття

Розглядаємо змінні (evolutional) – системи, в основі яких лежить модель предметної області - частина реального світу, які змінюються у відповідності з нею [5,17]. 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 (адаптації) хоча 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).

Визначимо основні поняття, що стосуються концепції забезпечення *життєздатності* (*viability*) ППС.

Розглядувані програмні системи будемо називати *життєздатними*, якщо вони забезпечені *засобами пристосування* до змін у моделі предметної області (функціональні вимоги) та змін вимог користувача (нефункціональні вимоги) на стадії функціонування та супроводження ППС. Їх можна розглядати як засоби підтримки еволюції ППС в аспекті забезпечення життєздатності. Засоби пристосування знаходяться в середовищі функціонування та супроводження ППС і включають *засоби адаптації* та *засоби ефективної модернізації (реінжинірингу і рефакторингу*). Тобто комплексна властивість *життєздатності* ПС включає властивості *адаптивності* ПС та *здатності до ефективної модернізації (реінжинірингу і рефакторингу)* ПС.

Адаптивність ППС - це властивість, що характеризує здатність її складових елементів змінюватися шляхом переналагодження для збереження значень життєво важливих показників системи в заданих межах за певних змін вимог до неї на стадії функціонування [18].

Адаптивність ППС забезпечується розробкою ймовірно-можливих варіантів її складових елементів та розробкою інструментальних засобів їх вибору та переналагодження при необхідності внесення *передбачених проектом* змін, що виникають у процесі функціонування.

Під *реінжинірингом* ППС розуміємо проведення змін функціональності системи, не передбачених її проектом [10, 13]. Процес реінжинірингу при цьому включає:

- визначення областей ППС, над якими необхідно провести реінжиніринг;
- визначення реінжинірингу, який необхідно застосувати до виділених областей ППС;
- оцінка ефективності реінжинірингу (характеристик якості ППС та економічних характеристик);
- застосування реінжинірингу;
- перевірка ефективності реінжинірингу;
- узгодження всіх артефактів ППС.

Здатність до ефективного реінжинірингу ППС – це властивість, що характеризує здатність її складових змінюватися при необхідності внесення *непередбачених проектом* змін, що виникають в процесі її функціонування. *Ефективний реінжиніринг* ППС забезпечується наявністю в середовищі функціонування ППС засобів для його впровадження, які туди спеціально імплементуються на стадіях розробки та створення ППС, і не потребує затрат на відновлення знань про ППС.

Основними властивостями ППС, що забезпечують необхідну життєздатність при змінах нефункціональ-них вимог, є модифікуємість компонент системи без зміни їх функціональності.

Рефакторинг ППС [14] – це проведення змін ППС, не передбачених її проектом, що відображають зміну якісних (нефункціональних) вимог користувача.

Система супроводження ППС при цьому має забезпечувати функціонування ППС при таких змінах у вимогах користувача ППС.

Процес рефакторингу при цьому включає:

- визначення областей ППС, над якими необхідно провести рефакторинг;
- визначення рефакторингу, який необхідно застосувати до виділених областей ППС;
- перевірка, що застосований рефакторинг не змінить функціональності ППС;
- оцінка ефективності рефакторингу (характеристик якості ППС та економічних характеристик);
- застосування рефакторингу;
- перевірка ефективності рефакторингу;
- узгодження всіх артефактів ППС.

Здатність до ефективного рефакторингу ППС – це властивість, що характеризує здатність її складових змінюватися при необхідності внесення *непередбачених проектом* якісних (нефункціональних) вимог користувача, що виникають у процесі її функціонування. *Ефективний рефакторинг* ППС забезпечується наявністю у середовищі функціонування ПС засобів для його впровадження, які спеціально імплементуються на стадіях розробки та створення ППС, і не потребує затрат на відновлення знань про ППС.

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

Для забезпечення адаптивності ППС, здатності ППС до ефективного реінжинірингу та ефективного рефакторингу пропонується підхід, за яким названі властивості закладаються у процесі проектування і створення ПС та які забезпечуються відповідними технологічними засобами супроводження ППС.

2.2. Загальні положення

Механізми інтелектуалізації технології створення і супроводження ППС та здійснення зворотного зв'язку при функціонуванні ППС, згідно запропонованого підходу, включають:

- модель ППС, побудовану на основі існуючих шаблонів ППС;
- показники життєздатності ППС, механізми їх оцінки та підтримки прийняття рішень щодо досягнення їх необхідних значень (при створенні та при супроводженні ППС);
- економічні показники ППС, механізми їх оцінки та підтримки прийняття рішень щодо досягнення таких їх значень, які задовольняють вимоги користувача (при створенні та при супроводженні ППС);
- механізми простежування змін та виділення компонентів ППС, що змінюються;
- інструментальні засоби модифікації компонентів ППС, що вимагають змін.

Взаємопов'язану сукупність цих механізмів будемо називати моделлю життєздатності, а ППС, які мають у середовищі свого функціонування таку модель – життєздатними ППС.

2.2.1. Модель ППС

Запропонований нами підхід передбачає класифікацію предметних областей, виділення для них типів ППС та створення для цих типів ППС їх UML-моделей. UML-модель ППС підтримує процес керування життєздатністю та процес її забезпечення. Вона визначає атрибути життєздатності ППС, відповідно до

нефункціональних вимог до системи, внутрішні та зовнішні атрибути життєздатності ППС, а також зв'язок між ними.

Наприклад, модель UML класу програмних систем CRM (Customer Relationship Management – керування взаємовідносинами з клієнтами [19]) 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'єкта автоматизації.

2.2.2. Показники життєздатності ППС та методи їх оцінки

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

Основними показниками якості ППС, що забезпечують необхідну життєздатність при змінах функціональних та нефункціональних вимог, зокрема, є наступні [20]:

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

При цьому, *аналізованість (analyzability)* **–** це властивості ППС, що обумовлюють здатність діагностування її недоліків або чинників відмов, а також ідентифікації частин, які слід модифікувати. Один з шляхів підвищення аналізованості програми – обробка всіх можливих виключних ситуацій з видачею зрозумілих користувачеві повідомлень та ведення журналу подій з відображенням часу їхнього виникнення, їх змісту та, якщо це можливо, їх причин. *Адаптованість – модернізованість (changeability)* – це властивості ППС, що обумовлюють її здатність виконувати встановлені види модифікацій. Систему слід вважати ідеально модернізованою, якщо будь-яку зміну (що не вимагає збільшення функціональності) можна провести за допомогою штатних засобів системи. В рамках концепції забезпечення життєздатності ППС дана система буде називатися системою з ідеальною адаптивністю. *Стабільність (stability)* – це властивості ППС, що обумовлюють її здатність мінімізувати неочікувані ефекти модифікацій. Для забезпечення стабільності ППС має мати якомога меншу взаємопов'язаність класів. *Тестованість (testability)* – властивість ППС, що обумовлює її здатність допомагати перевірці програмного забезпечення, що модифікується.

Метрики та методи оцінки показників життєздатності ППС. Розглядувана програмна система на протязі свого життєвого циклу постійно еволюціонує.

У програмній інженерії метрики традиційно використовуються для попередження невірного проектуван-ня на ранній стадії життєвого циклу ППС. Знайдені невідповідності та дефекти можуть бути модифіковані та попереджені з меншими затратами коштів та зусиль ніж на пізніших етапах проектування або на стадії супроводу.

Метрики забезпечують розробнику швидкий зворотній зв'язок **–** шляхом аналізу зібраних даних можна прогнозувати життєздатність ППС. При відповідному використанні метрик можливе значне зменшення вартості всієї розробки та покращення якості кінцевого програмного продукту, що в свою чергу, веде до зменшення витрат на його підтримку. Однак інформація, доступна на ранній стадії проектування часто є неточною та неповною. Через це інформація про життєздатність ППС, отримана у вигляді метрик вимагає уточнення та доповнення на пізніших етапах життєвого циклу ППС.

Пропонується також застосовувати метрики в контексті сценаріїв, які реалізуються за допомогою представлення процесу (діаграм взаємодії у середовищі Rational Rose). Це дозволить оцінювати параметри системи не загалом, а в рамках деяких задач або процесів, що цікавлять нас особливо. Таким чином метрики будуть застосовуватися не до всіх класів та їх методів а лише до тих, що пов'язані спільним сценарієм.

Так, *аналізованість ППС*, може бути оцінена з допомогою метрики, яка вимірює відношення кількості методів (як тих процедур та функцій, що є членами класів, і тих що не є ними), в яких відбувається обробка виключних ситуацій, до загальної кількості методів в ППС (*оброблюваність виключних ситуацій*). Різновидом цієї метрики можна вважати відношення кількості методів ППС 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*). З визначення *стабільності ППС* витікає, що ППС з меншим рівнем взаємопов'язаності класів буде більш стабільною, ніж з більшим рівнем взаємопов'язаності. Звідси можна зробити висновок, що більш стабільними будуть ті ППС, які побудовані за допомогою шаблону GRASP Low Coupling [21]. 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) [22]. При вимірюванні *тестованості* можливі наступні варіанти. Якщо в якості системи тестування ППС прийняти систему тестування методом екстремального програмування (тобто тестування відбувається шляхом написання тестових класів у яких відбувається перевірка класів ППС), то в якості метрики тестованості можна прийняти відношення кількості класів ППС до яких написані тестові класи до загальної кількості класів ППС.

2.2.3. Економічні показники ППС та методи їх оцінки

Основні економічні показники ППС. 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* ППС.

Метрики та методи оцінки економічних показників ППС. Нарівні з оцінюванням характеристик життєздатності ППС дуже важливим аспектом розробки та супроводження ППС є оцінювання економічних характеристик, пов'язаних з її якістю.

В області управління проектами з розробки ППС гостро стоїть проблема оцінювання проектів з розробки, модифікації чи реінжинірингу ППС в аспекті необхідних для їх виконання ресурсів, бюджету, та строків їх виконання. Існування цієї проблеми пов'язане з тим, що процес створення ППС є процесом перетворення специфікацій та вимог до системи у програмні коди готової ППС. Складність управління такими процесами викликана складністю вимірювання характеристик цих процесів для отримання кількісних характеристик вхідної та результуючої інформації та складністю моделювання цих процесів. За відсутності засобів підтримки прийняття цих рішень управління проектами часто ведеться методом проб та помилок, а успіх залежить від досвіду та інтуїції керівника.

Оскільки з чотирьох основних економічних характеристик проекту ППС (*розмір проекту, трудомісткість, час, необхідний для виконання та вартість)* три останні у великій мірі залежать від першої характеристики – *розміру* ППС – 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 [23], оскільки вона розроблялася саме з метою оцінки бюджету, вартості та тривалості проектів та є нині найбільш широко застосовуваною моделлю. Модель COCOMO використовує як вхідний параметр *розмір* ПС у рядках вихідного коду (SLOC [24]). 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. Водночас найкращим вибором для оцінювання розміру проекту є метод FPA [25]: він має об'єктивний характер, простий у застосуванні, є найбільш широко застосованим у індустрії розробки програмного забезпечення, має статус міжнародного стандарту та є загально доступним. Розмір ППС, отриманий за допомогою FPA, можна використати у моделі COCOMO, оскільки для багатьох мов програмування існують залежності між розміром ППС у одиницях UFP та одиницях SLOC [26].

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

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 ППС;

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.

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.

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

Особливості застосування моделі COCOMO. Калібрування параметрів. Модель COCOMO дає змогу отримати більш точні результати, якщо відкалібрувати її за даними за історичними проектами, тобто визначити

такі значення параметрів моделі, які б відображали особливості виконання проектів за умов конкретної організації.

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

Примінивши запропонований алгоритм [26], можна отримати значення калібровочних констант для певних умов виконання проектів.

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

Оцінювання характеристик, важливих в економічному аспекті, при виконанні проектів за розробкою та модифікацією ППС є необхідною складовою ***процесу керування*** розробкою та супроводженням життєздатних ППС. Застосування методів оцінювання економічних характеристик дозволяє більш точно прогнозувати як матеріальні ресурси, необхідні для реалізації проекту, так і людські та часові ресурси. Це дає можливість приймати обгрунтовані рішення щодо формування пакету нових розробок ППС в організації, а також щодо доцільності проектів модифікацій супроводжуваних ППС.

2.2.4. Механізми простежування змін в ПС

Відповідно до архітектури процесів життєвого циклу ПС [27] *модель простежування змін* підтримує процес керування конфігурацією та, зокрема, процес внесення змін. Вона визначає зв'язок між артефактами ППС відповідно до різних етапів життєвого циклу та процесу розробки, а також механізми, що зумовлюють процес перетворень архітектури ППС. У відповідності з цим, для класу ППС створюються *моделі простежування змін* та основні *сценарії* їх функціонування [28].

Концепція сценарного підходу розглядає *сценарій* як базовий артефакт, що підтримує варіантність ППС 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.

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

Аналіз та оцінка базових артефактів ППС, зокрема варіантів сценаріїв, яким відповідають певні діаграми взаємодій, повинен завершуватись встановленням зв'язку між їхніми оцінками (на основі внутрішніх метрик ПС) та зовнішніми атрибутами ППС (функціональними та нефункціональними). Це приводить до створення інтегральних оцінок сценарію та ППС у цілому.

2.2.5. Модель життєздатності ППС та її застосування при їх створенні та супроводженні

Вищевизначені на концептуальному рівні всі основні компоненти моделі життєздатності ППС. Схема моделювання та одержання варіанта проекту (варіанта модифікації проекту), що задовольняє вимоги користувача ППС включає:

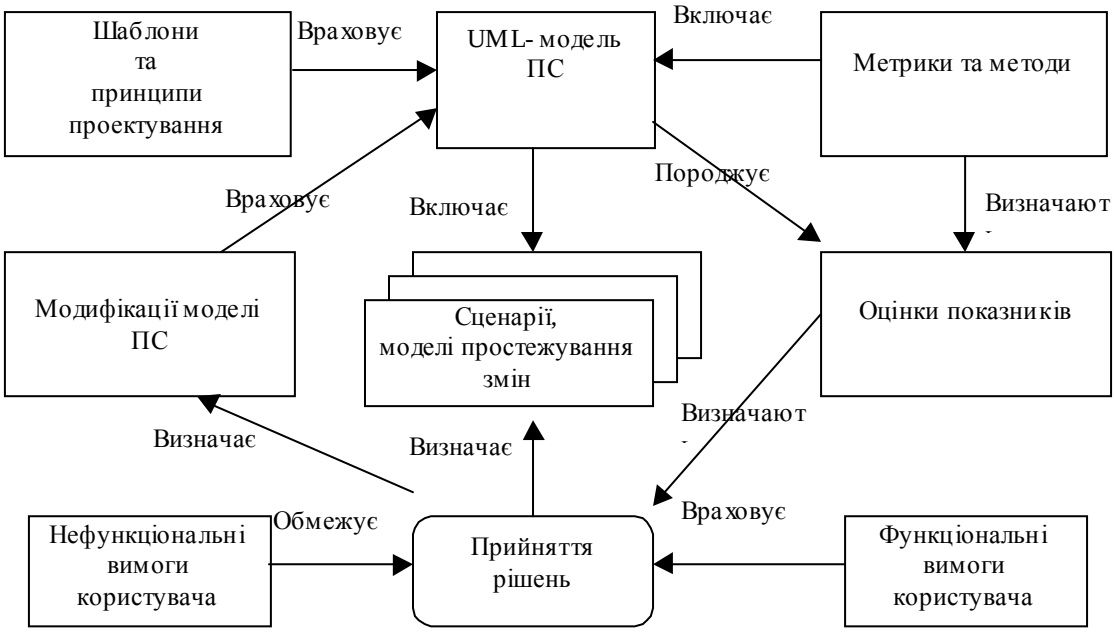
на *стадії розробки*

- ♦ проведення моделювання ППС в основних (необхідних) аспектах та вибір кращого варіанта моделі ПС для подальшої реалізації на основі вхідних даних від предметної області, створеної UML - моделі, вимог користувача;
- ♦ одержання оцінок характеристик життєздатності ППС та їх аналіз в аспекті виконання нефункціональних вимог користувача, а за їх прийнятності – перехід до одержання оцінок економічних характеристик;
- ♦ одержання оцінок економічних характеристик ППС та їх аналіз в аспекті виконання вимог користувача, а за їх прийнятності – вироблення за моделлю ППС, проекту ППС;
- ♦ реалізація проекту ППС;
- ♦ передача ППС в експлуатацію разом із засобами її моделювання.

На *стадії функціонування та супроводження (при реінжинірингу та рефакторингу)*

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

Модель життєздатності ППС , схема моделювання та одержання у рамках цієї моделі варіанта проекту ППС, який має оцінки, що задовольняють вимоги користувача, показана на рисунку.



Рисунок

3. Технологічні аспекти методики створення життєздатних ППС

Актуальними задачами вирішення цієї проблеми для об’єктно-орієнтованих ППС є створення засобів одержання таких оцінок на основі аналізу відповідності ППС шаблонам проектування із використанням діаграм UML-моделей та відповідних інструментальних засобів, таких, як Rational Rose [29]. Для всіх інших ППС, 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.

Можливість створення та застосування засобів забезпечення життєздатності ППС, наприклад, в контексті об'єктно-орієнтованої технології розробки ППС і створення відповідного середовища розробки визначається наступними факторами. Це застосування *шаблонів проектування* [21] таких, що визначають загальні принципи розробки ППС, використання *мови моделювання UML* [29], яка реалізує метод об'єктно-орієнтованого аналізу й проектування та інструменту *Rational Rose* [26], 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 ППС на основі UML вироблені та сформульовані загальні принципи та стандартні рішення, які удосконалюють розробку ППС. 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 ППС) і GoF (Gang of For – союз чотирьох) [30].

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 у процесі розробки об'єктно-орієнтованих ППС. Такі види діяльності при розробці ППС, 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 ППС. Їх успішна реалізація визначає успіх проекту в цілому. При цьому стратегія внесення змін на етапі як розробки, так і впровадження та супроводження ППС 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.

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

Забезпечення підтримки прийняття рішень у пропонованому підході базується на використанні сучасних аналітичних засобів для оцінок показників життєздатності та економічних показників моделюючих

варіантів ППС та динамічному багатовимірному їх аналізі за допомогою OLAP-систем та сховищ даних, що забезпечують можливість реалізації широкого спектра аналітичних задач [31 - 32].

Проведення досліджень та розробка підходу щодо забезпечення життєздатності автоматизованих систем організаційного керування здійснювалось за програмою фундаментальних досліджень Інституту програмних систем НАН України в рамках комплексної теми "Розробка теоретичних основ та методологічних засад компонентного програмування". Автор висловлює щиру вдячність керівнику комплексної теми доктору фізико-математичних наук, професору Е.М. Лавріщевій за постановку проблем та поради щодо їх вирішення, аспірантам та співробітникам відділу "Математичного та програмного забезпечення АС" ІПС НАН України, що брали участь за темою, без наукових зусиль яких прогрес в її вирішенні був би неможливий.

Висновки

Проблеми створення життєздатних ППС є актуальними та важливими для забезпечення їх довготривалого функціонування та зменшення витрат на стадії супроводження. Відсутність моделі життєздатності в ППС, що передаються в експлуатацію, призводить до значних втрат при спробах врахування виникаючих змін на стадії супроводження, або навіть до припинення їх експлуатації.

Розглядувана в роботі концепція створення ППС, що враховує проблеми забезпечення їх життєздатності, дозволяє доповнити технологію розробки та супроводження ППС формалізованими моделями та засобами, що в сукупності забезпечують життєздатність ППС.

1. *Эшби У.Р.* Введение в кибернетику. – М.: Наука, 1975. – 427 с.
2. *Месарович М., Мако Д., Такахара И.* Теория иерархических многоуровневых систем. – М.: Мир, 1973. – 344 с.
3. *Бир С.* Кибернетика и управление производством. – М.: Наука, 1965. – 390 с.
4. *Бир С.* Мозг фирмы. – М.: Радио и связь, 1993. – С. 218
5. *Ігнатенко П.П.* Проблеми забезпечення життєздатності програмних систем та підходи до їх вирішення // Пробл. Програм. – 2002. – №3-4. – С. 58–73.
6. *Karsai G., Sztipanovits J.* A Model-Based Approach to Self-Adaptive Software // IEEE Intelligent Systems. –1999, Maj/June. – P. 46 – 62.
7. *Черняк Л.* Адаптируемость и адаптивность// Журн. Открытые системы. 2004 -№ 9 (www.osp.ru).
8. *Herring C.E.* Viable software (www.charles-herring.com).
9. *Wooldridge M.* Agent–based software engineering.– IEE Proccedings Software Engineering, 1997.–v. 144.–№ 01.– P. 2-10.
10. *Software* Reengineering // Ed. Robert S. Arnold // IEEE Comput. Soc. Press, 1994. — 676 P.
11. *Ігнатенко П.П., Неумоїн В.М., Бистров В.М.* Аспекти реінжинірингу програмних систем // Проблемы программирования, 2000. – № 1-2. – С. 367–375.
12. *Ігнатенко П.П., Неумоїн В.М., Бистров В.М.* Про забезпечення ефективного реінжинірингу прикладних програмних систем // Там же. – 2001. – № 1-2. – С. 42–52.
13. *Підхід* до забезпечення реінжинірингу об'єктно-орієнтованих програмних систем/ П.П. Ігнатенко, В.М.Бистров, І.О. Заєць, О.П. Ігнатенко // Там же. – 2002. – № 1-2. – С. 98–108.
14. *Mens T., Tourwe T.* A Survey of Software Refactoring // IEEE Software Engineering, 2004,-Februar. – P. 126 – 139.
15. *http*://www.servise-oriented.com
16. *Kruchten P.B.* The 4+1 Vien Model of Architecture // IEEE Software. Nov. –1995. –12. – P. 42–50.
17. *Pfleeger S.L.*The Nature of Sustem Change // IEEE Software. –1998, Maj/June. – P. 87–90.
18. *Неумоїн В.М., Ігнатенко П.П.* Аспекты адаптивности программных систем // Пр. Першої міжнар.науково-практичної конф. УкрПРОГ"98. –К.: Ін-т кібернетики ім В.М.Глушкова НАН України,1998. – с.542–546.
19. *Ігнатенко П.П., Ткаченко В.М., Стрєлов І.А., Дуднік Р.О.* Підхід до моделювання та проектування CRM- систем // УСИМ. – 2005. – № 2. – С. 57–65.
20. *Ігнатенко П.П., Стрєлов І.А., Ткаченко В.М., Дуднік Р.О.* Концепція створення моделі прикладної програмної системи з розвинутою функцією життєздатності // Проблемы программирования, 2004. – № 2-3. – С. 163–172.
21. *Ларман К.* Применение UML и шаблонов проектирования. – М.: Вильямс, 2001. – 496 с.
22. *Chidamber S. and Kemerer C.* Towards a metrics suite for object-oriented design. Conference on Object-Oriented Programming System, Languages and Applications (OOPSLA'91), 1991 – P. 197-211.
23. CSE, 1999 - Center for Software Engineering, " COCOMO II Reference Manual," Computer Science Department, USC Center for Software Engineering, 1999. – 86p.
24. *Park. R* Software Size Measurement: A Framework for Counting Source Statements // Tech. Report CMU/SEI-92-TR-020. - Software Eng. Inst., Pittsburg, 1992. – 242 p.
25. *COSMIC*, 2003 – The Common Software Measurement International Consortium, The COSMIC FFP Measurement Manual. Version 2.2. – 81p.// www.cosmicon.com.
26. *Стрєлов І.А., Ігнатенко П.П.* Підхід до оцінювання економічних характеристик проектних рішень при розробці, модифікації та реінжинірингу програмних систем// Проблемы программирования, 2004. – № 1. – С.38–51.
27. *Scenarios* in System Development: Current Practice / K. Weidenhaupt, K. Polh, M. Jarke, Haumer P // IEEE Software. - March/April. – 1998. – P. 34–35.
28. *Ігнатенко П.П., Бистров В.М., Ігнатенко О.П., Ткаченко В.М.* Задачі та засоби моделювання і оцінювання життєздатних програмних систем // Проблемы программирования, 2003. - № 3. – С .59– 70.
29. *Боггс У., Боггс М.* UML и Rational Rose. – М.: "Лори", 2000. – 582 с.
30. Gamma E., Helm R., Johnson R., and Vlissides J *Design Patterns, Elements of Reusable Object-oriented Software,- N.- Y.: Addison-Wesley, 1995. – 345 p.*
31. Архипенков С., Голубев Д., Максименко О. *Хранилища данных. От концепции до внедрения// Под общ. ред. С.Я. Архипенкова. – М.: ДИАЛОГ-МИФИ, 2002. – 528 с.*
32. Архипенков С. *Аналитические системы на базе Oracle Express OLAP. Проектирование, создание, сопровождение. – М.: ДИАЛОГ-МИФИ, 1999.-320 с.*