УДК 381.3.06

ПІДХІД ДО ОЦІНЮВАННЯ ДЕЯКИХ ХАРАКТЕРИСТИК ЯКОСТІ ТА ПРИЙНЯТТЯ РІШЕННЯ ЩОДО ВИБОРУ ВАРІАНТУ ЖИТТЄЗДАТНИХ АВТОМАТИЗОВАНИХ СИСТЕМ

В.М. Ткаченко
Інститут програмних систем НАНУ
e-mail: vovant@isofts.kiev.ua

Часто практичні проблеми, що виникають на стадії розробки або супроводження автоматизованих систем (АС) [1], пов'язані з прийняттям рішення щодо вибору одного з декількох варіантів реалізації системи (при створенні нової системи) або одного з варіантів зміни системи при додаванні нової функціональності (на стадії супроводження).

Критеріями для оцінювання варіантів моделей можуть будти: ціна системи, трудоємність створення системи (дана характеристика тісно пов'язана з ціною) або оцінка важливих для розробника чи замовника характеристик якості. Частіше за все вимоги до характеристик якості пов'язані зі стадією супроводження обмежуються вимогами користувача відносно ціни та терміну розробки як вказано на рис. 1.

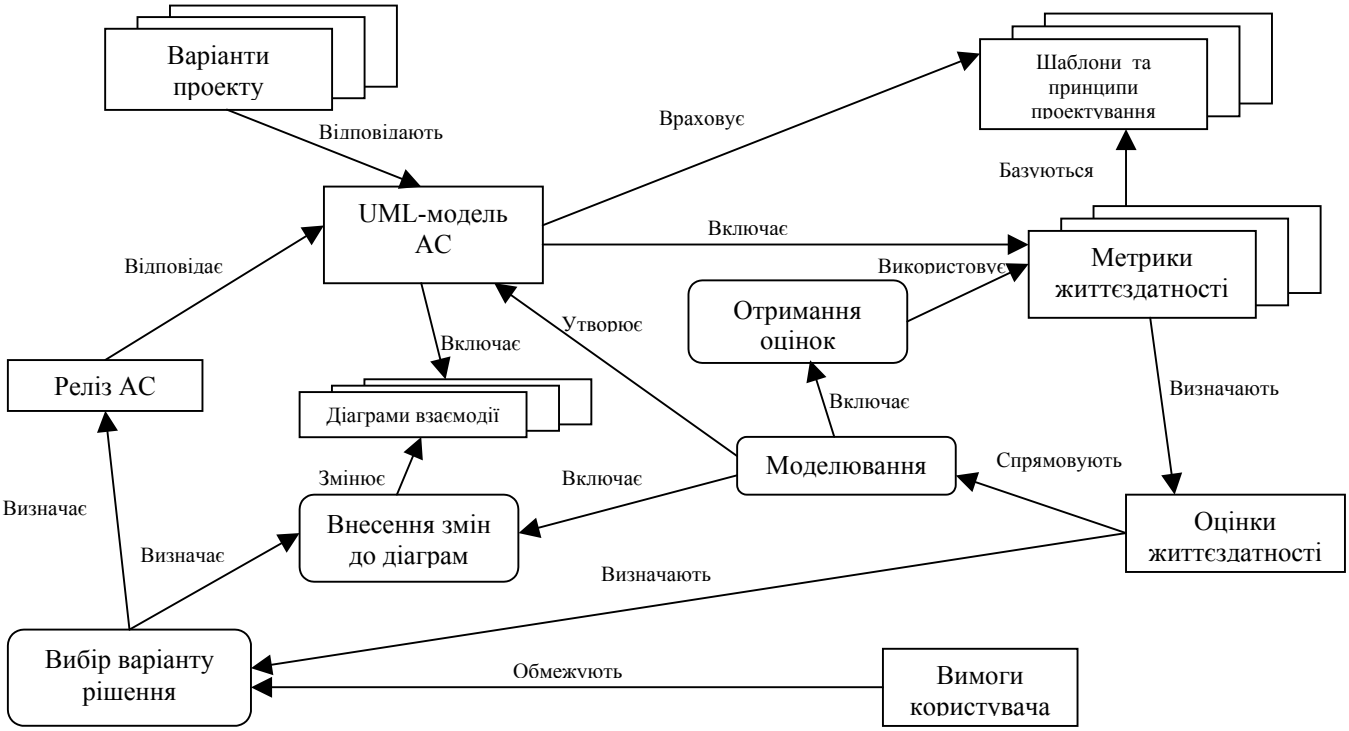

Рис. 1. Схема моделювання та оцінювання характеристик життєздатності об'єктно-орієнтованих АС при їх створенні, модифікації та реінжинірингу

Однією з інтегральних характеристик якості АС є життєздатність АС [1]. Програмні системи будуть називатися життєздатними, якщо вони забезпечені засобами пристосування до змін в моделі предметної області (функціональні вимоги) та змін вимог користувача (нефункціональні вимоги) на стадії функціонування та супроводження АС. Такі засоби можна розглядати як засоби підтримки еволюції АС у аспекті забезпечення їх життєздатності. Засоби пристосування знаходяться в середовищі функціонування та супроводження АС і включають засоби адаптації та засоби ефективного реінжинірингу. Тобто, комплексна властивість життєздатність АС включає властивості адаптивність АС та здатність до ефективного реінжинірингу АС [2-5]. Саме оцінюванню даної характеристики й присвячено цю доповідь.

Підхарактеристики життєздатності АС

За класифікацією характеристик якості АС, наведеною в ISO/IEC 9126-1, до характеристик життєздатності АС можна віднести *супроводжуваність* (*maintainability*) – властивості АС, що обумовлюють її ефективну модифікацію, включаючи коригування, удосконалення або адаптацію до зміни середовища, вимог та функціональних специфікацій.

Основними підхарактеристиками якості АС, що забезпечують необхідну життєздатність при змінах функціональних та нефункціональних вимог, зокрема, є наступні [6, 7]:

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

При цьому *аналізованість помилок (error analyzability)* - це властивості АС, що обумовлюють здатність діагностування її недоліків або чинників відмов, а також ідентифікації частин, які потрібно модифікувати.

Аналізованість коду (code analyzability) - це властивості АС, що обумовлює можливість проаналізувати програмний код. Для більшої аналізованості коду програми (в рамках парадигми об'єкто-орієнтованого програмування) класи мають виконувати пов’язані між собою задачі та не виконувати робіт непомірних об’ємів.

Адаптованість – модернізованість (changeability) – це властивості АС, що обумовлюють її здатність виконувати встановлені види модифікацій. Систему слід вважати ідеально модернізованою, якщо будь-яку зміну (що не вимагає збільшення функціональності) можна провести, маніпулюючи даними (або метаданими) за допомогою штатних засобів системи. В рамках концепції забезпечення життєздатності АС дана система буде називатися системою з ідеальною адаптивністю. *Стабільність (stability)* – це властивості АС, що обумовлюють її здатність мінімізувати неочікувані ефекти модифікацій. Для забезпечення стабільності АС має мати якомога меншу взаємопов’язаність класів. *Тестованість (testability)* – властивості АС, що обумовлюють її здатність допомагати перевірці ПЗ, що модифікується. Дана властивість залежить від системи тестування та ПЗ, що застосовується для тестування.

Метрики, за допомогою яких можна виміряти життєздатність АС

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

Так, *аналізованість помилок (ЕА) АС* 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), де відбувається обробка виключних ситуацій, до загальної кількості методів в АС (*оброблюваність виключних ситуацій*). Різновидом цієї метрики можна вважати відношення кількості методів АС, в яких реалізовано можливість запису до журналу подій АС (|Ml|), до загальної кількості методів АС (|M|) (*показник логування*).

$$EA = \frac{|Ml|}{|M|} \quad (1)$$

З визначення *аналізованості коду (СА) АС*, випливає, що АС з більш аналізованим кодом, буде АС класи якої мають більший рівень функціонального зчіплення (функції якого більш пов’язані між собою). Звідси можна зробити висновок, що з більш

аналізованим кодом будуть ті АС, які побудовані за допомогою шаблону GRASP [8] High Cohesion. Для виміру зв'язаності використовуються наступна метрика: *PCOM* (Prosperity of Cohesion), яка визначається для кожного класу системи як середнє значення відношення кількості методів класу які використовують змінну v_i класу (Nm_i) до загальної кількості методів класу (n), k – кількість змінних класу. Аналізованість коду АС визначається як середнє значення *PCOM* для всіх класів системи (c – кількість класів в системі)

$$PCOM = \frac{1}{k}\sum_{i=1}^{k}\frac{Nm_i}{n} (2) \qquad CA = \frac{1}{c}\sum_{i=1}^{c} PCOM_i (3)$$

Адаптованість – модернізованість (AM) можна оцінити за допомогою відношення кількості змінних та бізнес-правил предметної області, які можна змінити за допомогою штатних засобів системи (|BRc|), до загальної кількості змінних, констант та бізнес-правил предметної області (|BR|) (*модернізованість як адаптованість*) [6]. $AM = \frac{|BRc|}{|BR|} (4)$

З визначення *стабільності (S) АС* витікає, що АС з меншим рівнем взаємопов'язаності класів буде більш стабільною, ніж з більшим рівнем взаємопов'язаності. Звідси можна зробити висновок, що більш стабільними будуть ті АС, 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 без ієрархії наслідування) [9, 10]. Для виміру стабільності виберемо *CBOout* яка визначається як:

$$CBO_{out}(c) = |Mc - Ic| (5)$$

Де Mc – набір класів, які залежать (викликають методи або звертаються до змінних) від класу C, Ic – набір класів які знаходяться з класом C у відношенні підклас-суперклас. Тоді стабільність системи можна визначити як середнє CBOout за всіма класами системи.

$$S = \frac{1}{c}\sum_{i=1}^{c} CBO_{out} (6)$$

При вимірюванні *тестованості* (T) можливі наступні варіанти. Якщо в якості системи тестування АС прийняти систему тестування методом екстремального програмування (тобто тестування відбувається шляхом написання тестових класів, в яких відбувається перевірка класів АС) [10], то в якості метрики тестованості можна прийняти відношення кількості класів АС, до яких написані тестові класи, до загальної кількості класів АС. Дана метрика буде виміряти *загальну тестованість АС*.

Але оскільки багато класів АС виконують допоміжні функції і може виникнути потреба оцінити тестованість АС 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 АС), то в цьому випадку використовується така метрика, як *тестованість АС в термінах предметної області*: відношення кількості сценаріїв системи (UCTN – Use Case Tested Number), які перевіряються тестовими класами, до загальної кількості сценаріїв АС (UCN – Use Case Number). $T = \frac{UCTN}{UCN} (7)$

Шкали для оцінки обраних метрик

В даному розділі наведено оціночні шкали для метрик кожної з п'яти підхарактеристик життєздатності. Всі оціночні шкали мають однакову розмірність (1 - погано, 2 - незадовільно, 3 - задовільно, 4 - добре, 5 - відмінно). Це необхідно для

отримання інтегральної оцінки життєздатності системи на основі оцінок підхарактеристик життєздатності. Дані шкали розроблені для систем класу CRM на базі експертного аналізу загальної моделі класу CRM-систем [11]. Для інших класів систем побудова шкал потребує додаткового дослідження.

Підхарактеристики життездатності	Ваговий коефіцієнт	Бали				
		1	2	3	4	5
Аналізованість помилок	K_{EA}	0,3	0,45	0,55	0,7	1
Аналізованість коду	K_{CA}	0,01	0,03	0,1	0,3	1
Адаптованість (модернізованість)	K_{AM}	0,2	0,4	0,6	0,8	1
Стабільність	K_S	50	30	15	5	0
Тестованість	K_T	0,3	0,5	0,7	0,9	1

Таблиця 1. Відповідність верхніх границь інтервалів абсолютних значеннь метрик до бальної оцінки підхарактеристик якості.

Бальну оцінку характеристики будемо позначати літерою В з індексом з назви характеристики. Наприклад бальну оцінку стабільності будемо позначати як B_S, аналізовності помилок - B_{EA} і т.і.

Кожен розробник може сам визначати які підхарактеристики життєздатності є для нього більш важливі. Тому вагові коефіцієнти підхарактеристик життєздатності ми будемо визначати експертним методом, з розрахунком, що їх сума дорівнює одиниці.

Тоді бальна оцінка життєздатності всієї системи буде визначатися за формулою:

$$B_{sys} = B_{EA} * K_{EA} + B_{CA} * K_{CA} + B_{AM} * K_{AM} + B_S * K_S + B_T * K_T \quad (8)$$

За цією методикою ми можемо як порівнювати два або більше варіантів моделі системи, як за окремими підхарактеристиками якості. Так і давати взагалі оцінку життєздатності одного варіанта системи, з урахуванням важливості підхарактеристик якості для замовників. Тобто, це дозволяє розробнику прийняти рішення щодо створення системи на основі того чи іншого варіанту моделі цієї системи. Крім того, при наявності лише моделі для оцінки, розробник має можливість порівняти отримані оцінки з середніми або (при наявності таких) еталонними оцінками систем такого класу.

Перелік посиланнь

1. Ігнатенко П.П. Проблеми забезпечення життєздатності програмних систем та підходи до їх вирішення // Пробл. программирования. – 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. Основы инженерии качества программных систем / Ф.И.Андон, Г.И.Коваль, Т.М.Коротун, В.Ю.Суслов – К.: Академпериодика, 2002. – 504с.

7. Бабенко Л.П., Лаврищева Е.М. Основи програмної інженерії. – Київ: "Знання", - 2001. - 269с.

8. Ларман К. Применение UML и шаблонов проектирования. – М.: Вильямс, 2001. – 496 с.

9. S. Chidamber, C. Kemerer. Towards a metrics suite for object-oriented design // Conf. Object-Oriented Programming System, Languages and Applications (OOPSLA'91). – 1991. - P. 197-211.

10. Extreme programming applied: playing to win / K. Auer, R. Miller. - Pearson Education, 2002. – 326 P.

11. Ігнатенко П.П., Ткаченко В.М., Стрєлов І.А., Дуднік Р.О. Підхід до моделювання та проектування CRM-систем. - УСІМ, № 2, 2005 с.56-67.