DOI:**10.33099/2311-7249/2021-42-3-45-54**

УДК: 004.054

Ігор Анатолійович Костюк *(доктор філософії)* [1]
Максим Анатолійович Павленко *(доктор технічних наук, професор)* [2]
Сергій Валерійович Осієвський *(кандидат технічних наук)* [2]
Олексій Юрійович Несміян *(кандидат технічних наук)* [2]

[1] ***Національний університет оборони України імені Івана Черняховського, Київ, Україна***
[2] ***Харківський національний університет повітряних сил імені Івана Кожедуба, Харків, Україна***

МЕТОД ПІДВИЩЕННЯ НАДІЙНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗНАННЯ-ОРІЄНТОВАНИХ СИСТЕМ ЗА РАХУНОК МЕХАНІЗМІВ ПОВТОРНОГО ВИКОРИСТАННЯ КОДУ

Розглянуті питання можливості застосування механізму повторного використання коду в процесі проектування та розробки програмного забезпечення знання-орієнтованих систем. Доведена необхідність окремого дослідження питань розробки програмного забезпечення аналітичного та інформаційного ресурсів, що забезпечують підтримку всіх учасників процесу розробки програмного забезпечення знання-орієнтованих систем. Показана необхідність використання механізмів повторного використання раніше розробленого і верифікованого програмного коду, як елементу інформаційного ресурсу. В рамках розробленого методу підвищення надійності програмного забезпечення знання-орієнтованих систем за рахунок механізмів повторного використання коду запропонована нова візуальна форма подання бібліотек функцій у вигляді єдиної програмної оболонки. В якості базового інструментарію вирішення завдання розробки інформаційного ресурсу запропоновано використовувати UML (Unified Modeling Language) – уніфіковану мову моделювання, в основу якої покладено парадигму об'єктно-орієнтованого програмування. Зазначений вибір обґрунтований тим, що UML є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення та по суті являється відкритим стандартом, що використовує графічні позначення для створення абстрактної моделі системи. Отримані теоретичні положення відображені в наскрізному прикладі, що відображає однин з можливих варіантів організації бібліотек функцій як елементу інформаційного ресурсу. Розроблено та обґрунтовано діаграми варіантів використання, діаграми взаємодії, діаграми послідовності, діаграми класів. На основі отриманих практичних результатів запропоновано структурну схему методу, яка, на відміну від існуючих рішень включає процедуру вироблення коректур для основних UML-діаграм за вимогами середовищ програмування.

Ключові слова: *UML-діаграма, програмне забезпечення, програмний код, інформаційний ресурс, клас, функція, модель.*

Вступ

Постановка проблеми. Програмне забезпечення знання-орієнтованих інформаційних систем (ЗОІС), характеризується орієнтацією на предметну область для якої вона розробляється та, відповідно, реалізацією складних математичних моделей (досить часто унікальних та характерних лише для певної предметної області), які є алгоритмічним поданням реальних фізичних процесів. Нерідко, реальні завдання, вирішення яких необхідно реалізувати в ЗОІС потребують спільного застосування апарату декількох галузей знань, що суттєво ускладнює процес математичного опису і, як наслідок, програмної реалізації цих завдань. Крім цього, значна кількість логіко-математичних моделей будуються на алгебраїчних виразах, що вимагає застосування різних методів обчислювальної математики для можливості отримання по ним чисельних розрахунків, що вимагає наявності професійних знань методів обчислювальної математики та має вирішальне значення в програмній реалізації зазначених математичних моделей [1,2].

Тобто, розробка програмного забезпечення ЗОІС полягає не лише в програмній реалізації складних прикладних функцій, що дозволяють реалізувати функціональність системи, а й в розробці численних програмних механізмів які забезпечують надійність і коректність

обчислювального процесу, зручний інтерфейс користувача, раціональне використання обчислювальних ресурсів, сумісну взаємодію різнотипних програмних додатків в рамках єдиного проекту. Крім цього, програмна реалізація математичних моделей, передбачає наявність знань розробниками відповідних прикладних теорій пов'язаних зі специфічністю предметних областей.

Сформована на сьогоднішній день практика розробки програмного забезпечення передбачає розмежування сфер відповідальності в процесі розробки програмного забезпечення. Тобто розробники програмного забезпечення повинні в першу чергу відповідати за реалізацію спільних програмних механізмів, що забезпечують коректну поведінку програми в рамках відповідної операційної системи, а також коректну взаємодію програмних блоків між собою, а завдання створення програмних алгоритмів, що реалізують необхідні логіко-математичні моделі, виноситься в окрему задачу, яка вирішується інженерами зі знань, архітекторами та експертами. Висока трудомісткість і наукоємність даного процесу, а також необхідність забезпечення взаємодії фахівців в різних областях знань призводить до значних витрат на даний процес.

Саме зазначена обставина, в першу чергу, говорить про необхідність використання механізмів повторного використання раніше розробленого і верифікованого програмного коду, як елементу інформаційного ресурсу. Іншими словами, процес розробки ЗОІС, із застосуванням будь-якого з існуючих на сьогоднішній день підходів до проектування і створення програмної продукції є надзвичайно складним і трудомістким процесом. Незважаючи на те, що нові підходи до програмування дозволяють, з одного боку, суттєво підвищувати ефективність розробки програмного забезпечення, з іншого боку, постійно зростаючі вимоги до функціональної складності, можливостям міжпрограмної взаємодії, ергономіки і т.д. зазначеного класу систем вимагають все більш тривалих термінів їх розробки.

Аналіз останніх досліджень і публікацій. З метою зниження витрат часу на розробку програмної продукції, з моменту появи перших підходів до програмування, стали використовуватися і розвиватися так звані механізми повторно використовуваного коду [3-5]. Сутність цих механізмів полягає в ідеї, повторного використання в нових розробках раніше розробленого програмного коду.

В даний час до механізмів повторно використовуваного коду відносять бази або бібліотеки функцій, процедур, класів, об'єктів і агентів. Для позначення більш загальної категорії, що термінологічно об'єднує вище зазначені бази і бібліотеки, в роботі введено термін "інформаційний ресурс".

Слід зазначити, що при розробці прикладного програмного забезпечення загального призначення (такого, що не містить елементів інтелектуальних систем) прийнято використовувати інформаційно-аналітичний ресурс, що містить теоретичні описи завдань з використанням положень відомих методологій та їх практичні реалізації у відомих середовищах розробки програмного забезпечення. Формально, як було зазначено раніше інформаційно – аналітичний ресурс містить бази функцій, агентів, бібліотеки класів та об'єктів. При цьому розуміється, що бази об'єктів містять візуальні і невізуальні компоненти, ActiveX об'єкти, а також об'єкти крос-платформенної реалізації. Тобто бази об'єктів – це цілий ряд різнорівневих програмних механізмів (від забезпечення можливості візуалізації властивостей екземплярів класів в середовищах розробки, до забезпечення механізмів міжпрограмної взаємодії в рамках як єдиних, так і гетерогенних середовищ). Бази агентів [6], являють собою сукупність реалізованих агентів різної функціональності.

Аналіз існуючих рішень щодо розробки ЗОІС показав, що використовувати традиційний підхід , в якому контент інформаційного та аналітичного ресурсу являється нероздільним елементом підтримки процесу розробки [7-9] не завжди являється доцільним з погляду на різнорідність функціональних потреб учасників процесу розробки ПЗ ЗОІС та потребує додаткового дослідження.

Мета статті. Виходячи з концепції роздільного застосування аналітичного та інформаційного ресурсів сформуємо завдання дослідження. Питання формування аналітичного ресурсу розкрито в [3,10,11]. Окремого опрацювання потребує дослідження механізму формування інформаційного ресурсу з метою ефективної організації процесу розробки програмного забезпечення ЗОІС на рівні розробників (Developer).

В якості базового інструментарію рішення зазначеного завдання пропонується використовувати UML (Unified Modeling Language) – уніфіковану мова моделювання, в основу якої покладено парадигму об'єктно-орієнтованого програмування. Зазначений вибір обґрунтовується тим, що UML є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення та по суті являється відкритим стандартом, що використовує графічні позначення для створення абстрактної моделі системи [12, 13]. А власне саме рішення вбачається у реалізації механізму повторного використання програмного коду в процесі розробки ПЗ ЗОІС. Тобто, всі прикладні математичні вирази та відповідні алгоритми обчислень реалізуються у вигляді баз програмних функцій, класів та об'єктів і становлять ядро інформаційного ресурсу підтримки процесу розробки ПЗ ЗОІС на рівні розробника.

Це дозволить:

створити базу програмних компонент рішення типових прикладних задач та забезпечить

можливість їх повторного застосування при розробці прикладного програмного забезпечення ЗОІС;

розробникам прикладного програмного забезпечення ЗОІС основні зусилля зосередити на реалізації внутрішніх програмних рішень, а не на реалізацію складних прикладних розрахунків;

досягти підвищення надійності розроблюваного ПЗ ЗОІС, за рахунок використання в його складі вже відлагоджених та апробованих компонент, максимально зменшити можливість виникнення помилок за рахунок "блочного" характеру розробки ПЗ ЗОІС.

Передбачається, що створення і широке застосування баз програмних компонентів, на основі механізмів повторно використовуваного коду в технологічному процесі розробки прикладного програмного забезпечення ЗОІС забезпечить:

надійність, уніфікацію і стандартизацію розробляємого прикладного програмного забезпечення;

підвищення ефективності розробки і супроводу ПЗ, за рахунок зниження часових витрат на безпосереднє кодування, тестування і відлагодження ПЗ.

Структура методу підвищення надійності ПЗ ЗОІС за рахунок механізмів повторного використання коду, як сукупність відповідних моделей розробки, накопичення та використання баз програмних компонент, а також опису методики розробки баз програмних компонент (як ядра інформаційного ресурсу) наведена на рисунку 1. Зазначена структура дає загальне уявлення про сутність методу в цілому та буде деталізована за рахунок розроблених діаграм UML.

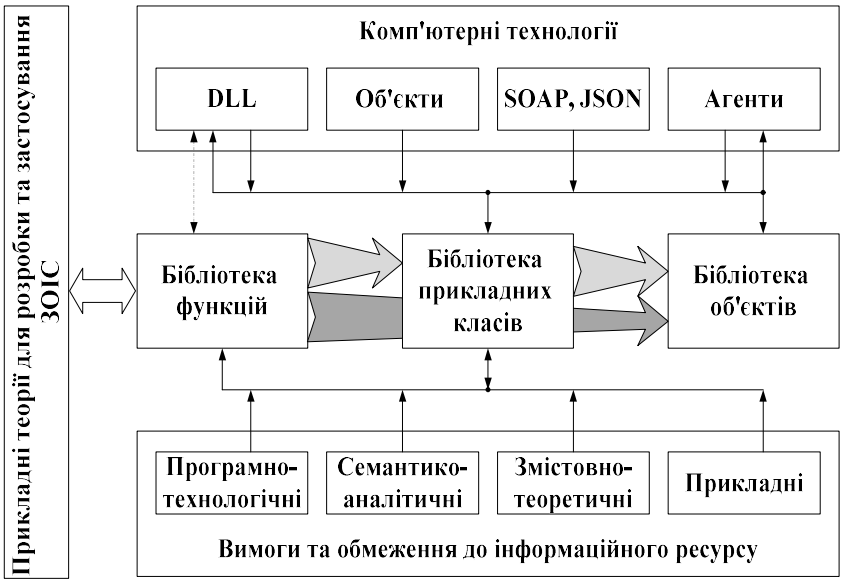

Рис. 1. Структура методу підвищення надійності ПЗ ЗОІС за рахунок механізмів повторного використання коду

Виклад основного матеріалу дослідження

Сама ідея необхідності розробки інформаційного ресурсу, що містить базу програмних компонент повторного використання коду, і зокрема, бібліотек функцій для різних областей прикладних знань в інтересах створення програмного забезпечення різного рівня в даний час не викликає сумнівів. Це пов'язано з тим, що подібні бібліотеки є найбільш доступним і в той же час загальноприйнятим способом повторного використання коду, що в свою чергу, дозволяє підвищувати надійність і ефективність розробки нового програмного забезпечення за рахунок зниження витрат часу на безпосереднє кодування, тестування та відлагодження програмних додатків [14]. Перш за все, використання бібліотек функцій та інших видів баз програмних компонент повторного використання коду призводить до підвищення надійності новостворюваного програмного забезпечення [8].

Спостерігаємий останнім часом бурхливий розвиток обчислювальної техніки та не менш стрімкий розвиток методологій програмування, анітрохи не зменшили потреби розробників ПЗ ЗОІС в бібліотеках функцій, як в одному з основних механізмів повторного використання коду.

Цю потребу в найзагальніших випадках компенсують за рахунок розробки програмного забезпечення, що містить в своєму складі найбільш часто використовувані бібліотеки, наприклад, математичних, статистичних, строкових та інших функцій. Але потреба в безпосередньо прикладних функціях, тобто функціях, що вирішують завдання прикладних галузей знань, необхідних при реалізації ПЗ ЗОІС (наприклад, таких як: радіолокації, аеродинаміки і т.д.), до цих пір залишається незадоволеною. Складність ситуації полягає в тому, що існуючі стандарти на розробку програмних додатків, не прийнято використовувати при розробці бібліотек

функцій, тому що бібліотеки функцій не виступають в ролі кінцевого продукту. Вони використовуються в межах засобів розробки, і їх окрема стандартизація досі також не була принциповою. З іншого боку, використання, загальноприйнятого підходу до розробки бібліотек функцій, що реалізують складні математичні залежності прикладних галузей знань не завжди може бути виправданим. Це пов'язано з достатньою складністю та унікальністю галузей знань. При цьому текстовий опис подібних функцій, що використовується в документації до бібліотек, далеко не завжди може бути повним та пов'язаний з персональним баченням розробника.

Тобто, вихідні лістинги і документація не можуть бути інформативно-ємним і повним описом для бібліотек прикладних функцій. Виникає потреба в новій формі подання бібліотек функцій, з урахуванням можливостей сучасних мов програмування, вимог теперішнього часу та потреб розробників. Особливо гостро це питання постає при розподілі інформаційного та аналітичного ресурсу підтримки процесу розробки ЗОІС.

Крім цього, самим прикладним функціям, як готовим компонентам програмного коду, властиве протиріччя, що полягає в недостатньо чіткому логіко-математичному опису області визначення функції з одного боку, а з іншого боку явному описі області визначення функції в програмній реалізації. Зазначене протиріччя призводить до значної кількості помилок при синтезі програмного коду для ПЗ ЗОІС на основі баз програмних компонент. Ця ситуація, як правило, посилюється тим, що неможливо вимагати від розробників програмних додатків глибоких знань з прикладних теорій, в інтересах яких створюється програмне забезпечення. Крім вищесказаного для існуючих бібліотек функцій притаманні і внутрішні недоліки: відсутність автоматизованих механізмів тестування функцій, одноваріантного подання функцій, відсутність зручних механізмів пошуку.

Виходячи з вищесказаного, в рамках розробленого методу підвищення надійності ПЗ ЗОІС за рахунок механізмів повторного використання коду пропонується нова візуальна форма подання бібліотек функцій у вигляді єдиної програмної оболонки, що дозволить:

відображати список всіх реалізованих в бібліотеці функцій;

відображати для кожної функції її математичний опис та перелік аргументів;

безпосередньо в програмній оболонці виконувати розрахунок будь-якої з реалізованих в бібліотеці функцій, як зі значеннями за замовчуванням, так і з будь-якими введеними користувачем значеннями;

формувати текстовий опис функцій в форматі Windows Help;

формувати лістинг функції з коментарями на двох та більше мов програмування за вибором розробника.

Подібне візуальне представлення бібліотек прикладних функцій здатне досить повно задовольнити потреби розробників проектів при створенні і супроводі програмного забезпечення для ЗОІС.

Як було зазначено вище, в рамках запропонованого методу підвищення надійності ПЗ ЗОІС за рахунок механізмів повторного використання коду передбачається, що об'єктно орієнтований аналіз і проектування баз програмних компонент, таких як візуальні бібліотеки прикладних функцій, здійснюється з використанням візуальних засобів уніфікованої мови моделювання UML. Відмінною особливістю даного підходу є створення в процесі аналізу і проектування візуальних моделей розробляємого програмного додатку на мові UML, яким притаманна висока семантична насиченість. До моменту початку створення будь-якого програмного забезпечення розробники повинні найбільш повно ознайомитися з вимогами, що висуваються замовниками до створюваної системи. Протягом тривалого часу в процесі як об'єктно-орієнтованої, так і традиційної розробки програмного забезпечення застосовувалися типові сценарії, що дозволяють розробникам краще зрозуміти вимоги до системи. Однак ці сценарії зазвичай трактувалися досить неформально – їх майже завжди використовували, але рідко документували. Ця ситуація була змінена в програмно-технологічному підході Objectory [12-13]. Сценаріям (варіантам використання), була надана така значимість, що, вони перетворилися в один з основних елементів аналізу та планування реалізації проекту розробки програмного забезпечення. Тобто, варіант використання – це опис типової взаємодії користувача з комп'ютерною системою. Нотація, що використовується в UML, дозволяє зручно і з достатнім ступенем деталізації застосовувати варіанти використання для опису вимог до системи. На рисунку 2 у відповідності до наведених раніше функціональних представлено діаграму варіантів використання бібліотеки функцій.

При створенні діаграм варіантів використання слід зосередити увагу на реалізацію та рішення специфічних для розробника завдань. Обмеженням являється лише передбачення застосування зовнішніх інформаційних ресурсів. В такому випадку слід на діаграмах варіантів використання відображати не завдання користувача, а системні взаємодії, що відображають процеси (яким чином система повинна виконувати призначені для користувача завдання). Кожен варіант використання доповнюється текстовим описом – потоком подій. В потоці подій в стандартному вигляді описані події, їх взаємозв'язок, що викликаються реалізацією завдань користувача. Формально, наведений опис являється першим етапом методу підвищення надійності ПЗ ЗОІС за рахунок механізмів повторного використання коду.

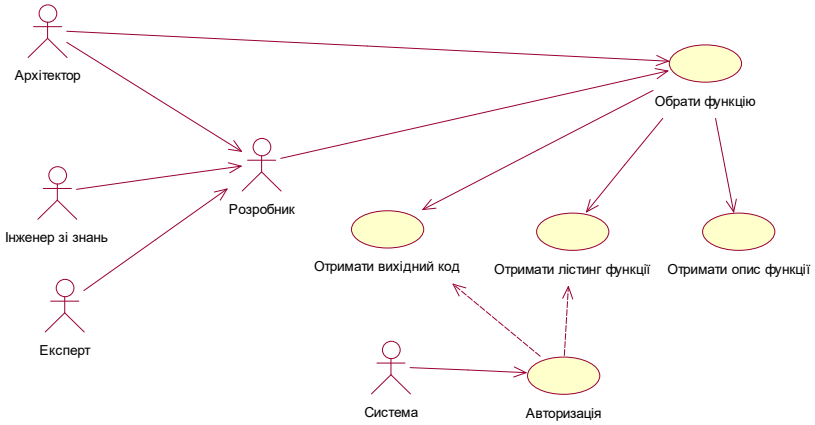

Рис. 2. UML діаграма варіантів використання бібліотеки функцій

На другому етапі проводиться опис в UML-нотації моделі потоку подій для кожного з варіантів використання. Нижче наведено, як приклад, стандартний формат опису моделі потоку подій для одного з варіантів використання, а саме модель потоку подій варіанту використання “Отримати лістинг функції” (рис. 3).

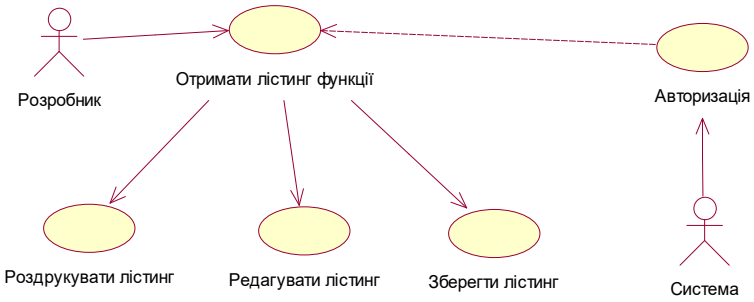

Рис. 3. Діаграма реалізації варіанту використання “Отримати лістинг функції”

Коротко опис цієї моделі полягає в наступній констатації – дозволяє користувачеві вибрати необхідну йому функцію, по її назві, із списку доступних опцій. Для неї основний потік подій наступний: варіант використання починається, коли користувач обирає з пропонованого списку назву конкретної функції. При виборі користувачем функції – відображається її лістинг та список аргументів. При цьому альтернативним потоком являється блокування завдання отримання лістингу функції за умови не проходження авторизації.

Подання вимог до бази програмних компонент в подібному вигляді дозволяє вже на етапах аналізу і проектування не лише врахувати вимоги, що пред'являються до ПЗ ЗОІС, але і розробити конкретні варіанти їх реалізації.

Діаграма варіантів використання наведена на рис. 2, відображає найбільш високий рівень абстракції, який містить проект, для відображення завдань користувача. Для більш низьких рівнів завдань також використовуються діаграми варіантів використання та розкривають сутність варіантів використання верхнього рівня. Таким чином, створюється ієрархічна структура варіантів необхідного ступеня деталізації. Зокрема, в розглянутому прикладі діаграма на рис. 3 розкривають сутність варіанту використання “Отримати лістинг функції”.

На основі даних з діаграм варіантів використання здійснюється конструювання основних функціональних блоків бази програмних компонент для синтезу програмного забезпечення ЗОІС. Тобто діаграми варіантів використання дають необхідну інформацію для подальшого аналізу і проектування. Подібне конструювання дозволяє легко простежити не лише призначення, але і взаємодію програмних блоків.

З метою визначення порядку взаємодії як між програмними модулями так і взаємодії з системо передбачається розробка діаграм послідовності для кожного з основних та альтернативних варіантів використання. По суті це моделі взаємодії елементів програмної реалізації проекту.

Наприклад, для потоку подій “Отримати лістинг функції” розроблена діаграма послідовностей, яка зображена на рисунку 4 та діаграма взаємодії (рис. 6).

Перевага в застосуванні діаграм послідовностей полягає в можливості вибору будь-якого прийнятного рівня абстракції. Це проілюстровано на рисунку 5, де представлено модель тієї ж діаграми послідовностей для потоку подій “Отримати лістинг функції”, але на рівні абстракції етапу проектування, який виконується безпосередньо перед початком програмної реалізації.

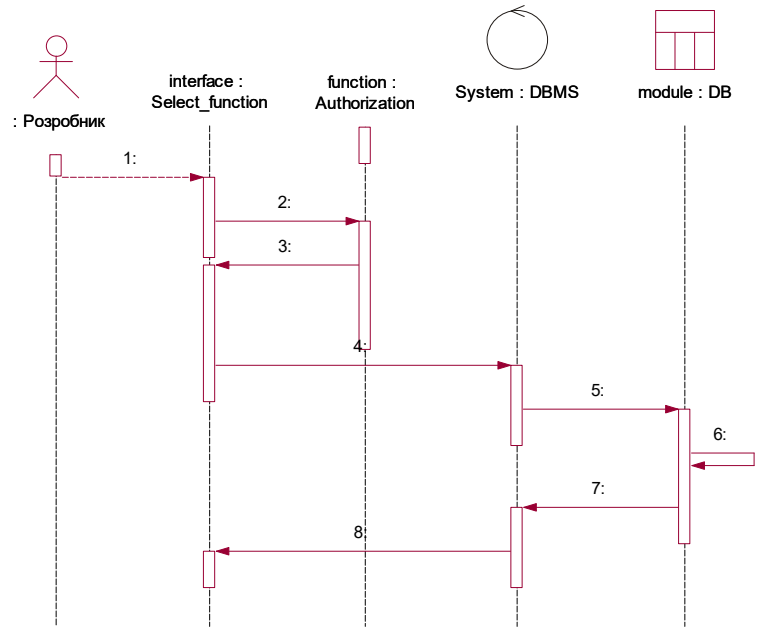

Рис. 4. UML-діаграма послідовностей потоку подій “Отримати лістинг функції”

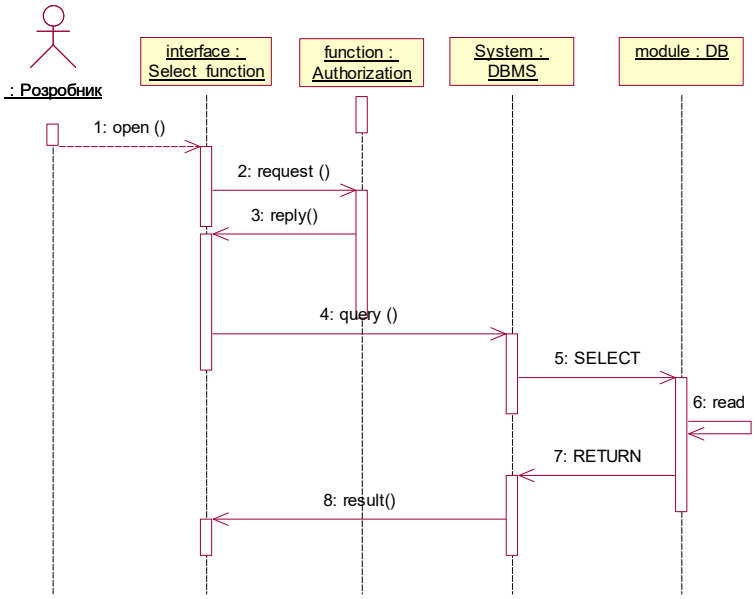

Рис. 5. UML-діаграма послідовностей потоку подій “Отримати лістинг функції” з визначенням функцій та класів

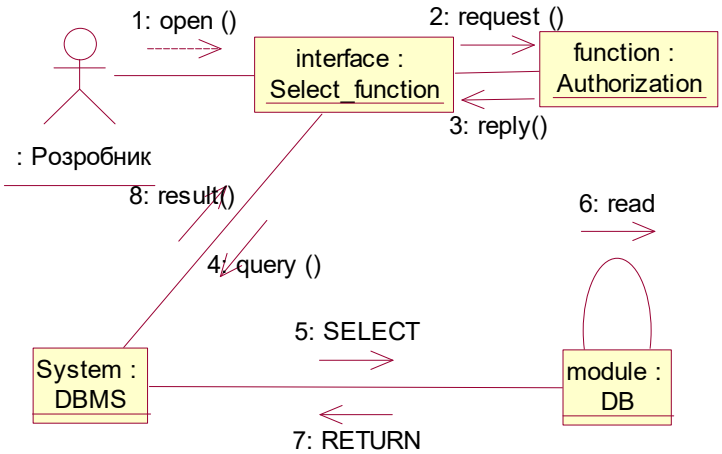

Рис. 6. UML-діаграма взаємодії для варіанту використання “Отримати лістинг функції”

Слід зазначити, що, як правило, на діаграмах UML не відображаються всі реально використовувані або присутні блоки, класи, об'єкти, зв'язки, сутності і т.д. Це надзвичайно перевантажує діаграми і відповідно погіршує якість їх сприйняття. Тому на діаграмах прийнято відображати лише найбільш суттєві елементи, що забезпечують розкриття сутності відповідних програмних процесів.

Центральною ланкою по суті всіх об'єктно-орієнтованих методів є діаграми класів, що показано в [13,16,17-19]. Діаграма класів використовуються для відображення типів об'єктів програмної реалізації та різного роду статичних зв'язків, які існують між цими об'єктами. Крім цього, діаграми класів здатні відображати атрибути класів, операції і обмеження, які накладаються на зв'язки між об'єктами.

Безпосереднє проектування класів починається, в рамках запропонованого методу, з найвищого рівня абстракції, на якому зручно за допомогою діаграм класів представляти словник предметної області, при цьому визначаються імена, атрибути, операції та зв'язки. Виходячи з вищесказаного, діаграма класів, що розкриває схему функціонального призначення основних класів бібліотеки функцій, як типового варіанту бази програмних компонент повторно використовуваного коду наведено на рисунку 7.

Перехід до нижчих рівнів абстракції дозволяє розробити ієрархію класів системи, сформувавши повний перелік класів, що входять безпосередньо в кожну з екранних форм з визначеними іменами, атрибутами, операціями та зв'язками. Таким чином, діаграма класів з додаванням класів нижчої ієрархії матиме вигляд наведений на рисунку 8.

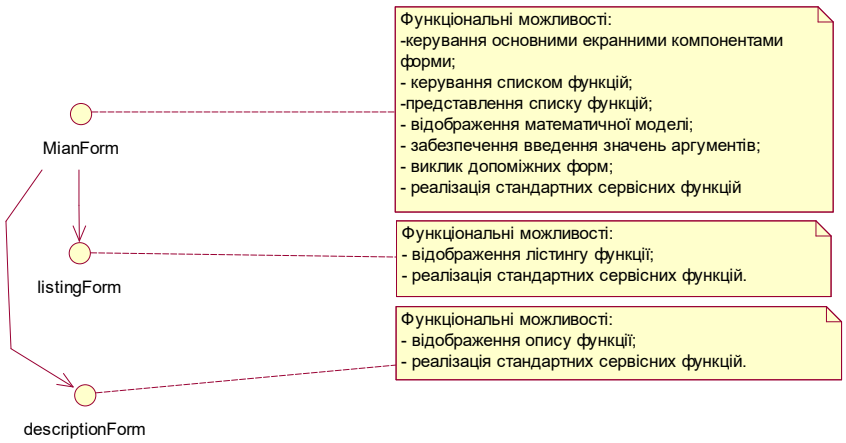

Рис. 7. Діаграма класів основної форми візуального представлення бібліотек функцій для найвищого рівня абстракції

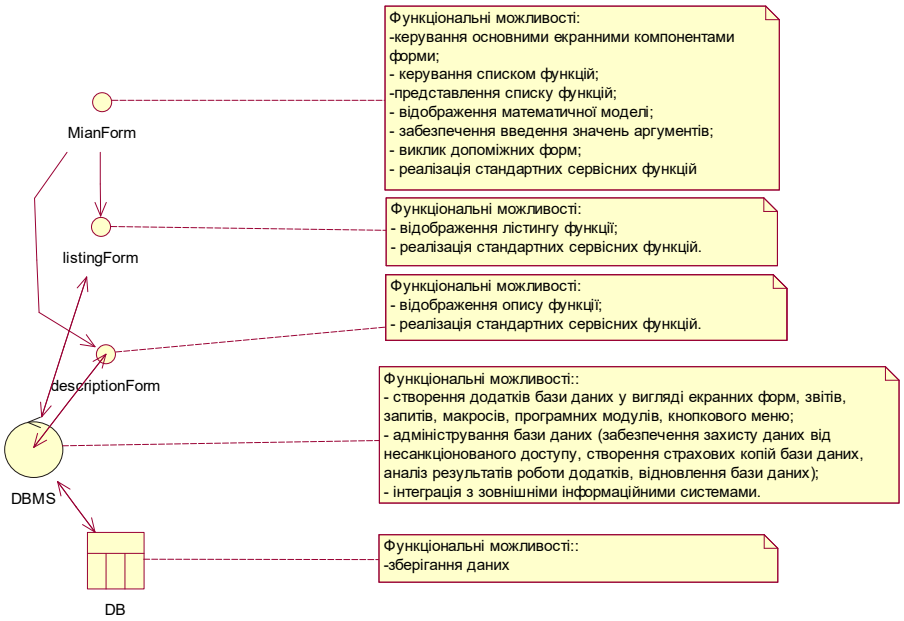

Рис. 8. Діаграма класів основної форми візуального представлення бібліотек функцій з включенням системних класів Control та Table

В даному випадку назви класів вже прив'язані до конкретної програмної реалізації, що з одного боку дещо ускладнює її читання, але з іншого боку більш зручно при переході до безпосередньої програмної реалізації. Діаграма класів, є видом статичних діаграм, і її доцільно застосовувати при

поясненні статичних зв'язків проектуємої бази програмних компонент для синтезу, розвитку і супроводу ПЗ ЗОІС.

Найбільш важливу роль в пропонуємому рішенні несе на собі клас T Functions, що реалізований в модулі Descript Functions, діаграма класів якого наведена на рисунку 9.

Даний модуль призначений для опису функцій, що безпосередньо реалізуються в компонентній бази інформаційного ресурсу. Крім цього зазначений модуль вирішує завдання зберігання необхідної інформації для подання реалізованих прикладних функцій

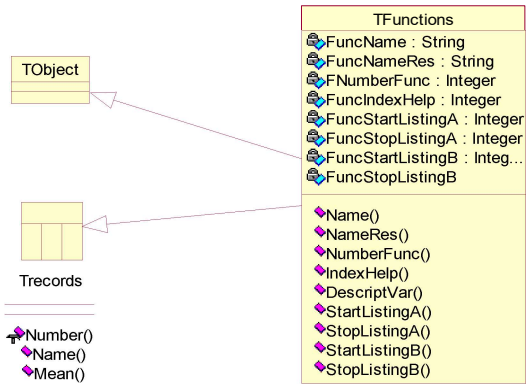

Рис. 9. UML-діаграма класів модуля Descript Functions

Зазначений клас знаходиться у відношенні асоціації, яке безпосередньо показує залежності між структурами проекту. На діаграмі класу T Functions представлені поля і властивості даного класу, для кожного з яких зображується тип видимості, найменування і тип подання. Як видно з діаграми, відповідно до об'єктно-орієнтованої парадигми, поля класу мають тип видимості public, А властивості – тип видимості private. Крім класів, на даній діаграмі представлені також типи даних використовуються в проекті. Завершальним кроком проектування візуальної бібліотеки функцій для синтезу, розвитку і супроводу ПЗ ЗОІС є остаточна розбивка передбачуваного програмного коду на модулі. Для цього в рамках запропонованого методу використовуються діаграми компонентів.

Таким чином, запропонований метод підвищення надійності ПЗ ЗОІС за рахунок механізмів повторного використання коду дозволяє створювати глибоко верифіковані бази програмних компонент повторюваного коду в рамках загальної візуальної оболонки. Завдяки визначеним функціональним можливостям дозволяє ефективно вирішити задачу підвищення надійності зазначеного програмного забезпечення.

Послідовний виклад сутності методу дещо нівелює складну структуру (логічну послідовність) його реалізації. Для виключення цього факту наведемо узагальнену структуру методу підвищення надійності ПЗ ЗОІС за рахунок механізмів повторного використання коду (рис. 10).

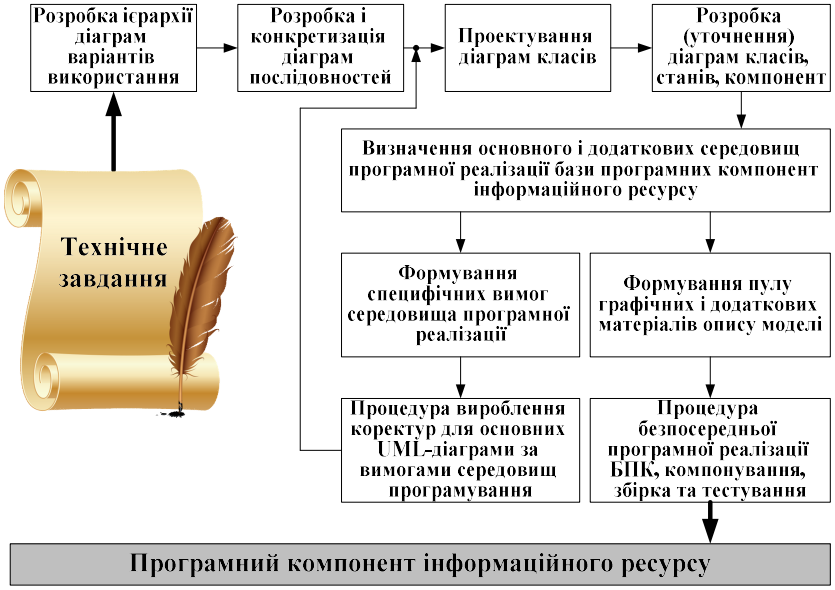

Рис. 10. Структурна схема методу підвищення надійності ПЗ ЗОІС за рахунок механізмів повторного використання коду

Запропонований метод забезпечує не лише підвищення надійності, але і зниження витрат часу та зростання результативності розробки програмного забезпечення ЗОІС на основі подібних баз програмних компонент (в т.ч. візуальних бібліотек функцій), як типових представників методології повторного використання розробленого і верифікованого коду.

Висновки і перспективи подальших досліджень

Запропонований метод підвищення надійності ПЗ ЗОІС за рахунок механізмів повторного використання коду має ряд особливостей як методичних, так і інструментальних, що відрізняють його від відомих та альтернативних рішень.

Зокрема, структурно метод розроблений для забезпечення функціонування інформаційної складової системи підтримки процесу розробки знання-орієнтованих систем. Тобто, отримані рішення не розповсюджуються на не стійку аналітичну складову. Це надасть можливість розробникам програмного забезпечення:

адекватно оцінювати поточну якість програмного забезпечення ЗОІС, з метою вироблення коригуючого впливу на хід проектування та розробки;

використовувати запропонований апарат на всіх етапах життєвого циклу програмного забезпечення ЗОІС.

знизити рівень ітеративності технологічного процесу розробки програмного забезпечення ЗОІС;

підвищити рівень складності реалізованих завдань, а як наслідок знизити трудомісткість;

підвищити безпомилковість роботи ЗОІС, в цілому.

Повторне використання програмного коду, за рахунок системного створення і використання різних баз програмних компонент (бібліотек функцій, класів, об'єктів і сервісів), забезпечує суттєве зниження трудомісткості процесу створення нового програмного забезпечення, а також значне зростання верифікованості і надійності програмного забезпечення ЗОІС, як програмних виробів.

Запропонований метод підвищення надійності програмних забезпечення ЗОІС за рахунок механізмів повторного використання коду дозволяє створювати глибоко верифіковані бази програмних компонент повторюваного коду в рамках загальної візуальної оболонки, що в свою чергу дозволяє ефективно вирішити задачу підвищення надійності програмного забезпечення.

Література

1. Павленко М. А. Методологічні основи підвищення якості програмного забезпечення інтелектуальних систем прийняття рішення / М. А. Павленко, С. В. Осієвський, Ю. В. Данюк. // Системи обробки інформації. – 2021. – №1. – С. 55–64. https://doi.org/10.30748/soi.2021.164.06. **2.** Domínguez, Oscar & Torres, L.M.. (2010). Technology intelligence: Methods and capabilities for generation of knowledge and decision making. 1-9. **3. Turinskyi, O.**, Pievtsov, H., Pavlenko, M., Osievskiy, S., Herasimov, S., Djus, V. (2020). The problem of structuring indicators of quality of decision software support system. International Journal of Advanced Trends in Computer Science and Engineering, 9(5), 7916-7923. https://journal-hnups.com.ua/index.php/zhups/article/view/528. **4.** Serifi, Veis & Dašić, Predrag & Ječmenica, R. & D.Labović,. (2013). Functional and information modeling of production using IDEF methods. Strojniski Vestnik. 55. 131-140. **5. Taylor**, Richard & van der Hoek, Andre. (2007). Software Design and Architecture The once and future focus of software engineering. FoSE 2007: Future of Software Engineering. 226-243. https://ieeexplore.ieee.org/document/4221623.

6. Ванн Тассел, Д. Стиль, разработка, эффективность, отладка и испытание программ [Текст]: пер. с англ. / Ванн Тассел. – М.: Мир, 1995. -248 с. **7. Брагина Т. И.** Сравнительный анализ итеративных моделей разработки программного обеспечения / Т. И. Брагина, Г. В. Табунщик // Радіоелектроніка, інформатика, управління. – 2010. – Вып. № 2 (23). – С.130–139. **8. Левыкин В. М.** Модель архитектурного фреймворка ускоренной разработки информационной системы / В. М. Левыкин, М. В. Евланов // Нові технології. – 2013. – № 1-2 (39-40). – С.51–57. **9. Осипова Т. Ф.** Моделирование процесса проектирования автоматизированной информационной системы структурным методом / Т. Ф. Осипова // Актуальные проблемы экономики и управления. – 2015. – № 2(6). – С. 89–96. **10. Павленко М.А.,** Осієвський С.В., Золотухіна О.А., Модель підтримки процесів розробки інтелектуальних систем підтримки прийняття рішень Телекомунікаційні та інформаційні технології. 2020. № 4 (69) 130 – 139 с. http://tit.dut.edu.ua/index.php/telecommunication/article/view/2363. **11. Осієвський С.В.,** Третяк В.Ф., Модель інформаційно–аналітичної підтримки процесів розробки знання–орієнтованих інформаційних систем //Колективна монографія Сучасний стан проведення наукових досліджень у ІТ-технологіях, галузях електроніки, інженерії, нанотехнологіях та транспортній сфері / за редакцією Голденблата М.А. та Валеренко Г.І. – Вінниця:Європейська наукова платформа https://ojs.ukrlogos.in.ua/index.php/monographs/article/view/14109. **12. D. S. Maylawati**, W. Darmalaksana, and M. A. Ramdhani, "Systematic Design of Expert System Using Unified Modelling Language," IOP Conf. Ser. Mater. Sci. Eng., vol. 288, no. 1, p. 012047, 2018. **13. Mouheb D.** et al. (2015) Unified Modeling Language. In: Aspect-Oriented Security Hardening of UML Design Models. Springer, Cham. https://doi.org/10.1007/978-3-319-16106-8_2.

14. Липаев В.В. Обеспечение качества программных средств. Методы и стандарты [Текст] / В.В. Липаев. – М.: МГТУ «Станкин», 2002. -302 с. **15. Плескач В.Л.** , Рогушина Ю.В. Агентні технології: Монографія. – К.: Київ. нац. торг.–екон. ун–т, 2005. – 344 с. (Табл.10. Рис.58. Бібліограф.361). **16.** Sojer, Manuel and Henkel, Joachim, Code Reuse in Open Source Software Development: Quantitative Evidence, Drivers, and Impediments (March 9, 2010). Journal of the Association for Information Systems, Vol. 11, No. 12, pp. 868-901, 2010, Available at SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1489789.

17. Haefliger, S., Krogh, G. V. and Spaeth, S. (2008). Code Reuse in Open Source Software. Management Science, 54(1), pp. 180-193. https://pubsonline.informs.org/doi/abs/10.1287/mnsc.1070.0748

18. Osis J., Asnina E. Is modeling a treatment for the weakness of software engineering? in:Garcia Diaz V., Cueva Lovelle J., García-Bustelo B. (Eds.), Handbook of Research onInnovations in Systems and Software Engineering, IGI Global, Hershey, NY, 2015, pp. 411-427.

19. Sejans J., Nikiforova N. Practical Experiments with Code Generation from the UML ClassDiagram. Proceedings of the 3rd International Workshop on Model-Driven Architectureand Modeling-Driven Software Development, SciTePress, Beijing, China, 2011, pp. 57-67.

МЕТОД ПОВЫШЕНИЯ НАДЕЖНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ЗНАНИЯ-ОРИЕНТИРОВАННЫХ СИСТЕМ ЗА СЧЕТ МЕХАНИЗМОВ ПОВТОРНОГО ИСПОЛЬЗОВАНИЯ КОДА

***Игорь Анатольевич Костюк** (доктор философии)* [1]
***Максим Анатольевич Павленко** (доктор технических наук, профессор)* [2]
***Сергей Валерьевич Осиевский** (кандидат технических наук)* [2]
***Алексей Юрьевич Несмиян** (кандидат технических наук)* [2]

[1] *Национальный университет обороны Украины имени Ивана Черняховского, Киев, Украина*
[2] *Харьковский национальный университет Воздушных Сил имени Ивана Кожедуба, Харьков, Украина*

Рассмотрены вопросы возможности применения механизма повторного использования кода в процессе проектирования и разработки программного обеспечения знаний-ориентированных систем. Доказана необходимость отдельного исследования вопросов разработки программного обеспечения аналитического и информационного ресурсов, обеспечивающих поддержку всех участников процесса разработки программного обеспечения знаний-ориентированных систем. Показана необходимость использования механизмов повторного использования ранее разработанного и верифицируемого программного кода как элемента информационного ресурса. В рамках разработанного метода повышения надежности программного обеспечения знаний-ориентированных систем за счет механизмов повторного использования кода предложена новая визуальная форма представления библиотек функций в виде единой программной оболочки. В качестве базового инструментария для решения задачи разработки информационного ресурса предложено использовать UML (Unified Modeling Language) – унифицированный язык моделирования, в основу которого положена парадигма объектно-ориентированного программирования. Указанный выбор обоснован тем, что UML является неотъемлемой частью унифицированного процесса разработки программного обеспечения и, по сути, является открытым стандартом, использующим графические обозначения для создания абстрактной модели системы. Полученные теоретические положения отражены в сквозном примере, отражающем один из возможных вариантов организации библиотек функций как элемента информационного ресурса. Разработаны и обоснованы диаграммы вариантов использования, диаграммы взаимодействия, диаграммы последовательности, диаграммы классов. На основе полученных практических результатов предложена структурная схема метода, которая, в отличие от существующих решений включает процедуру выработки корректур для основных UML-диаграмм по требованиям сред программирования.

Ключевые слова: *UML-диаграмма, программное обеспечение, код, информационный ресурс, класс, функция, модель.*

METHOD OF INCREASING THE RELIABILITY OF KNOWLEDGE SOFTWARE-ORIENTED SYSTEMS DUE TO CODE RE-USE MECHANISMS

***Igor Kostyuk** (Doctor of Philosophy)* [1]
***Maksym Pavlenko** (Doctor of Technical Sciences, Professor)* [2]
***Sergey Osievsky** (Candidate of technical sciences)* [2]
***Oleksiy Nesmiyan** (Candidate of Technical Sciences)* [2]

[1] *National Defence University of Ukraine named after Ivan Cherniakhovskyi, Kyiv, Ukraine*
[2] *Ivan Kozhedub Kharkiv National University of the Air Force, Kharkiv, Ukraine*

The possibilities of application of the code reuse mechanism in the process of designing and developing software of knowledge-oriented systems are considered. The necessity of a separate study of the issues of software development of analytical and information resources that provide support to all participants in the process of software development of knowledge-oriented systems is proved. The necessity of using the mechanisms of reuse of previously developed and verified program code as an element of the information resource is shown. In the framework of the developed method of increasing the reliability of knowledge-oriented systems software due to the mechanisms of code reuse, a new visual form of representation of function libraries in the form of a single software shell is proposed. As a basic tool for solving the problem of information resource development, it is proposed to use UML (Unified Modeling Language) - a unified modeling language, which is based on the paradigm of object-oriented programming. This choice is justified by the fact that UML is an integral part of a unified software development process and is essentially an open standard that uses graphical notation to create an abstract model of the system. The obtained theoretical provisions are reflected in a cross-cutting example, which reflects one of the possible options for organizing libraries of functions as an element of the information resource. Diagrams of use cases, interaction diagrams, sequence diagrams, class diagrams are developed and substantiated. Based on the obtained practical results, a block diagram of the method is proposed, which, in contrast to existing solutions, includes a procedure for making corrections for basic UML diagrams according to the requirements of programming environments.

Keywords: *UML-diagram, software, software code, information resource, class, function, model.*