ЗАГАЛЬНА ХАРАКТЕРИСТИКА РОБОТИ

Актуальність теми. Дефекти програмного забезпечення, в тому числі й дефекти проектування, досліджуються багатьма вченими, наприклад А. Тріфу, Д. Реч, М. Вермелінгер, М. Ланза, М. Мантула, Н. Моха, О.Кіупке, Р. Марінеску, Т. Гірба. Дефекти об'єктно-орієнтованого проектування поділяють на дві групи – функціональні та нефункціональні. Функціональні дефекти впливають на працездатність програмного забезпечення і тому потребують негайного усунення. Нефункціональні дефекти проектування негативно впливають на атрибути якості програмного забезпечення. Серед них є такі атрибути, які не пов'язані з працездатністю і тому не потребують негайного усунення, наприклад, зрозумілість, здатність до супроводу, повторного використання, тестування та перенесення. Але, з одного боку, нефункціональний дефект може прогресувати, і тому зволікання з його усуненням може призвести до значних витрат, пов'язаних із супроводженням ураженого цим дефектом програмного забезпечення. З другого боку, усунення нефункціонального дефекту проектування, що не прогресує, може виявитись марним витрачанням ресурсів. Наприклад, дефект може бути в елементах конструкцій програмного забезпечення, які не супроводжуються чи супроводжуються сторонніми організаціями, зокрема автоматично генеровані, повторно використані, чи є компонентами COTS. Таким чином, проведення робіт з усунення нефункціональних дефектів проектування має бути своєчасним і спрямованим на найбільш небезпечні дефекти. У зв'язку з цим особливої актуальності набуває завдання спостереження за розвитком нефункціональних дефектів проектування. Саме вирішенню цього завдання присвячено дисертацію.

Зв'язок теми дисертації з цільовими програмами та планами перспективних наукових досліджень. Робота виконувалась на кафедрі інженерії програмного забезпечення Національного авіаційного університету в межах науково-дослідної роботи № 24-Ф4/к44 «Методи та засоби інженерії програмного забезпечення», держбюджетної теми № 586-ДБ09 «Екологія програмного забезпечення» (номер державної реєстрації 0109U001769) та госпрозрахункової теми «Розробка програмного забезпечення та технічної документації з реєстрації фізичних осіб та оформлення документів, які посвідчують особу та підтверджують громадянство України» (договір № 579-Х08 від 1 листопада 2008 р.).

Мета та завдання дослідження. Метою дисертаційної роботи є розроблення методу моніторингу дефектів проектування об'єктно-орієнтованого програмного забезпечення. Метод реалізовано у вигляді засобу, який належить до інструментів візуалізації програмного забезпечення (Software Visualization Tools).

Для досягнення зазначеної мети в дисертаційній роботі ставляться та розв'язуються такі завдання:

– вивчення предметної галузі та опис онтології дефектів проектування, яка забезпечує повний, структурований і систематичний огляд поняття дефекту проектування та пов'язаних з ним понять;

– аналіз методів і засобів виявлення та моніторингу дефектів проектування об'єктно-орієнтованого програмного забезпечення, який забезпечує виділення невирішеної частини загальної проблеми діагностики програмного забезпечення;

– розроблення методу моніторингу дефектів проектування об'єктно-орієнтованого програмного забезпечення, який забезпечує спостереження за їх розвитком;

– розроблення методики побудови моделей дефектів проектування, які дають можливість отримувати про них інформацію;

– розроблення архітектури засобу моніторингу дефектів проектування об'єктно-орієнтованого програмного забезпечення, який забезпечує реалізацію запропонованого методу;

– апробація запропонованих методу та засобу шляхом проведення експериментального моніторингу дефектів проектування реального програмного забезпечення та аналізу його результатів.

Об'єктом дослідження є дефекти проектування об'єктно-орієнтованого програмного забезпечення.

Предметом дослідження є процеси, моделі, методи і засоби моніторингу дефектів проектування об'єктно-орієнтованого програмного забезпечення.

Методи дослідження. Для досягнення поставленої мети застосовано такі методи: аналіз та синтез – під час опису онтології дефектів проектування, аналізу та класифікації методів виявлення дефектів проектування об'єктно-орієнтованого програмного забезпечення; формалізація і моделювання – під час розроблення моделей дефектів проектування та метамоделі історії дефектів проектування; об'єктно-орієнтовані аналіз, проектування та програмування – під час розроблення засобу, який реалізує моніторинг дефектів проектування; експеримент – під час апробації запропонованих методу та засобу; вимірювання – для оцінювання інтенсивності ознак дефектів проектування.

Наукова новизна роботи полягає у розв'язанні важливого науково-практичного завдання моніторингу дефектів проектування об'єктно-орієнтованого програмного забезпечення. В дисертаційній роботі отримано такі наукові результати:

– уперше розроблено метод моніторингу дефектів проектування об'єктно-орієнтованого програмного забезпечення, який спрямовано на відстеження динаміки розвитку виявлених, у тому числі й на ранній стадії, дефектів проектування. Метод дозволяє здійснювати цілеспрямовану інспекцію та усувати найнебезпечніші дефекти й уникати зайвих витрат, що потрібні для передчасного усунення виявлених дефектів або з'ясування, які з них слід вважати небезпечними;

– уперше, шляхом розширення відомої метамоделі History Model (HISMO, Т. Гірба) побудовано метамодель історії дефектів проектування, у якій крім елементів конструкції програмного забезпечення, їх версій та історій, сутностями є версії та історії дефектів проектування, що дає змогу розробляти засоби для візуалізації та аналізу історії дефектів проектування на її основі;

– уперше запропоновано вирішення завдання побудови моделей дефектів проектування, які дозволяють розглядати дефекти проектування як сутності, що мають параметри, котрі можуть змінюватися в часі. Візуалізація цих параметрів дозволяє спостерігати дефекти проектування та їх розвиток, тому використовується для реалізації запропонованого в роботі методу.

Практичне значення отриманих результатів. Запропоновані в дисертації метод і реалізований засіб упроваджено в таких організаціях: Національному авіаційному університеті – у межах науково-дослідної роботи № 24-Ф4/к44 «Методи та засоби інженерії програмного забезпечення», і в навчальному процесі у дисциплінах «Архітектура та проектування програмного забезпечення» та «Еволюція програмного забезпечення» за напрямом «Програмна інженерія» (акт впровадження від 18.06.2010 р.); Відкритому акціонерному товаристві «КП ОТІ» – в розроблення та супроводження програмного забезпечення (акт впровадження від 10.06.2010 р.).

Особистий внесок здобувача. У праці [4] автору належать класифікація дефектів проектування та аналіз методів їх виявлення, у праці [5] – методика побудови моделей дефектів проектування об'єктно-орієнтованого програмного забезпечення.

Апробація результатів дисертації. Результати досліджень доповідались й обговорювались на VII Міжнародній науково-практичній конференції з програмування «УкрПРОГ'2010" (Кібернетичний центр ім. В.М. Глушкова Національної академії наук України, м. Київ), IX Міжнародній науково-практичній конференції молодих учених та студентів «Політ-2009» (Національний авіаційний університет, Київ 2009), Міжнародній конференції «Теоретичні та прикладні аспекти побудови програмних систем - TAAAPSD' 2008» (Національний університет «Києво-Могилянська академія» м. Київ), Міжнародній конференції аспірантів і студентів «Інженерія програмного забезпечення 2009» (Національний авіаційний університет, Київ 2009), Міжнародній конференції аспірантів і студентів «Інженерія програмного забезпечення 2008» (Національний авіаційний університет, Київ 2008), Всеукраїнській конференції аспірантів і студентів «Інженерія програмного забезпечення 2007» (Національний авіаційний університет, с. Конча-Заспа 2007), наукових семінарах Національного авіаційного університету.

Публікації. Результати досліджень викладено у восьми друкованих працях, з них чотири – у збірниках наукових праць, перелік яких затверджено ВАК України, і чотири – у фахових виданнях та збірниках тез доповідей наукових конференцій. Крім того, результати роботи викладено у звітах з науково-дослідних робіт.

Структура та обсяг роботи. Дисертація складається зі вступу, чотирьох розділів, що містять 51 рисунок і 9 таблиць, списку літературних джерел з 97 найменувань та 3 додатків. Загальний обсяг роботи становить 136 сторінок, із них 118 – основного тексту.

ОСНОВНИЙ ЗМІСТ РОБОТИ

У **вступі** обґрунтовано актуальність теми дисертаційної роботи, сформульовано мету та основні завдання досліджень, показано наукову новизну і практичну цінність отриманих результатів, наведено дані про зв’язок роботи з науковими темами Національного авіаційного університету.

У **першому** розділі розглянуто онтологію дефектів проектування в межах якої визначено: поняття діагностики програмного забезпечення, завдання діагностики програмного забезпечення та їх зв'язок із завданнями, що розв’язуються у дисертації; поняття дефекту проектування. З’ясовано причини виникнення дефектів та наведено їх класифікацію. Проаналізовано існуючі методи і засоби виявлення та моніторингу дефектів проектування об’єктно-орієнтованого програмного забезпечення, розроблено їх класифікацію та виділено невирішені завдання загальної проблеми діагностики об’єктно-орієнтованого програмного забезпечення. Онтологію застосовано для повного, структурованого та систематичного огляду поняття дефекту проектування та понять, що з ним пов’язані.

Основним поняттям в онтології (рис. 1) є дефект проектування (Дефект), який усувається із застосуванням одного або декількох технічних прийомів усунення (Усунення), обраних за допомогою технічного прийому індикації (Індикація).

Дефект проектування зумовлює появу ознак (Ознака) і діагностується на основі ознак, застосуванням методів та засобів діагностики (Діагностика). Ознаки визначаються шляхом аналізу (Аналіз) характеристик (Характеристика) елементів конструкції програмного забезпечення (Елемент ПЗ).

Характеристики добуваються з елемента конструкції програмного забезпечення шляхом застосування спеціальних технічних прийомів та методів (Зняття характеристик), наприклад, вимірювання чи статичний аналіз коду. Дефект проектування впливає негативно на один і більше внутрішніх атрибутів якості програмного забезпечення (Атрибут якості) (подібно шаблонам проектування (Шаблон), які переважно мають позитивний ефект на атрибути якості програмного забезпечення). Шаблони проектування можуть вноситися у програмне забезпечення за допомогою технічних прийомів усунення дефектів (Усунення). Дефект проектування спричиняється причиною (Причина) в результаті порушення правила проектування (Правило) і може бути відвернений механізмами запобігання дефектам (Запобігання).

Дефект проектування – це невідповідність структурних характеристик елемента або фрагмента конструкції програми правилам об'єктно-орієнтованого проектування. Це визначення обмежує об’єкт дисертаційного дослідження лише нефункціональними дефектами, що впливають на структуру об'єктно-орієнтованого програмного забезпечення. Ступінь розвитку дефекту проектування визначимо як кількісну характеристику відхилення структурних характеристик елемента або фрагмента конструкції програми від норми, що визначається правилами проектування.

Дефекти проектування класифікують за природою, за типом ураженого елемента і за можливістю вимірювання.

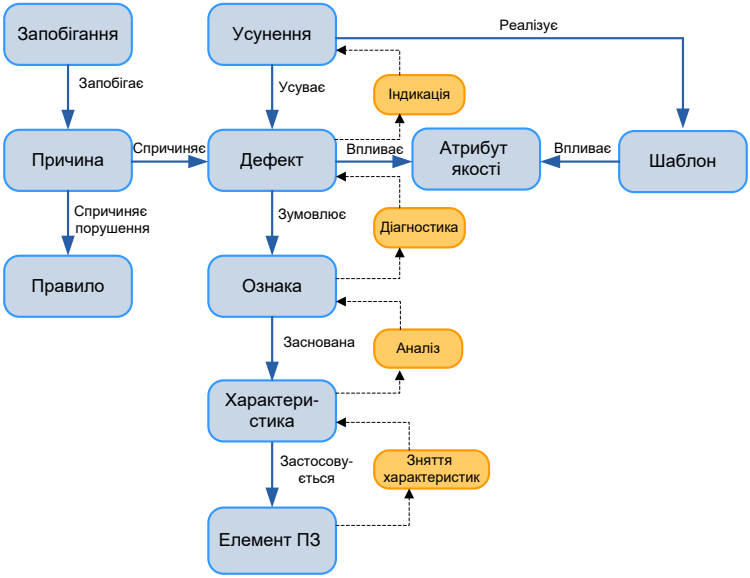


Рис. 1. Схема онтології дефектів проектування: ПЗ – програмне забезпечення

За своєю природою дефекти поділяють на такі: ідентифікаційні – дефекти, пов'язані з неправильним визначенням елементів конструкції програмної системи; кооперативні – дефекти, пов'язані з неправильним визначенням відношень між елементами конструкції програмної системи; класифікаційні – дефекти, пов'язані з некоректним використанням ієрархій класів.

За типом ураженого елемента конструкції програмного забезпечення дефекти поділено на такі: методу, класу, підсистеми.

За можливістю вимірювання дефекти поділяють на такі: вимірювані – дефекти, які в процесі супроводження програмного забезпечення, крім появи і зникнення, можуть змінювати ступінь свого розвитку; невимірювані – дефекти, які в процесі супроводження програмного забезпечення можуть тільки появлятися і зникати в результаті реструктуризації.

Методи виявлення дефектів проектування в дослідженні поділено на методи ручного виявлення та методи інструментального виявлення. Методи ручного виявлення не можуть бути масштабовані на програмні системи великих розмірів. Методи інструментального виявлення, в свою чергу, поділено на методи на основі зіставлення зі структурним зразком та на основі метрик. За відомими методами і засобами лише виявляють дефекти проектування, однак потрібно проводити моніторинг дефектів проектування, що дозволило б спостерігати за їх розвитком і своєчасно вживати заходів щодо їх усунення.

У **другому розділі** дисертації розглянуто метод, запропонований для вирішення завдання моніторингу дефектів проектування об’єктно-орієнтованого програмного забезпечення.

Сутність методу полягає в спостереженні за зміною параметрів дефектів проектування. Метод реалізується шляхом використання пропонованої метамоделі історії дефектів проектування (Design Flaws History Model, DDHM) об'єктно-орієнтованого програмного забезпечення і багатоаспектної візуалізації дефектів проектування уражених елементів конструкції різного рівня абстракції. У DDHM дефект уперше моделюється як окрема сутність, здатна змінювати свої параметри з плином часу. Модель дефекту проектування та методика її побудови дозволяє отримати значення таких параметрів дефекту: ступінь розвитку дефекту, інтенсивності ознак дефекту і середню інтенсивність ознак дефекту. Для проведення моніторингу дефектів за допомогою запропонованого методу необхідно виконати такі дії:

1) побудувати моделі дефектів проектування, моніторинг яких необхідно провести;

2) добути за допомогою засобів зворотної інженерії екземпляр DDHM з декількох версій вихідного коду аналізованого програмного забезпечення;

3) побудувати на основі екземпляра DDHM графічні зображення, які потрібні для моніторингу дефектів проектування;

4) виконати візуальний аналіз графічних зображень через взаємодію з ними.

Модель дефекту проектування – це опис елемента програмного забезпечення, ураженого дефектом проектування. Нехай $E = \{e_1, e_2, e_3, ..., e_k\}$ – множина елементів конструкції програмного забезпечення, а $D = \{d_1, d_2, d_3, ..., d_p\}$ – множина дефектів проектування, які можуть виникнути в елементі конструкції $e \in E$. Тоді визначаємо модель дефекту проектування як пару функцій $< \varphi_d, \mu_d >$:

– $\varphi_d : E \to [0,1000]$ – функція для визначення ступеня розвитку дефекту $d \in D$;

– $\mu_d : E \to [0,1000]$ – функція для визначення середньої інтенсивності простих ознак дефекту $d \in D$.

Функції моделі дефекту будуємо комбінуванням функцій агрегування на основі правила проектування.

Пропонуємо методику побудови моделі дефекту проектування, за якою необхідно виконати такі кроки:

1. Сформувати правило проектування елемента конструкції програмного забезпечення. На основі аналізу літератури та власного досвіду з урахуванням особливостей проекту розроблення програмного забезпечення визначити множину правил проектування $R = \{r_1, r_2, r_3, ..., r_p\}$. Визначимо $violate: R \to D$ як взаємно однозначну відповідність. Якщо $d \in D$ – дефект проектування, а $r \in R$ – правило проектування, тоді $d = violate(r)$ означає, що порушення правила r спричиняє дефект d.

2. Виконати аналіз правила для визначення ознак його порушення. Шляхом покрокової деталізації правила $r \in R$ виділити для кожного $d = violate(r)$

множину ознак $S_d = \{s_{d,1}, s_{d,2}, s_{d,3}, ..., s_{d,m}\}$ того, що елемент $e \in E$ уражений дефектом $d \in D$.

3. Визначити метрику для обчислення інтенсивності кожної простої ознаки. Інтенсивність ознаки – це кількісна характеристика її прояву. Ознака може бути простою та складеною з інших ознак. Простою ознакою вважатимемо ознаку, інтенсивність якої можна оцінити за допомогою однієї прямої метрики. Для кожної простої ознаки $s \in S$ обрати метрику $M_s : E \to Q$, де Q – множина всіх раціональних чисел, за допомогою якої можна оцінити інтенсивність ознаки s. Якщо оцінюється ознака, для якої неможливо обрати існуючу метрику, то необхідно визначити нову метрику.

4. Установити значення порога для кожної з метрик. Значення порога $Z_s \in Q$ розділяє область значень метрики M_s на дві підобласті. Залежно від того, в якій підобласті міститься результат вимірювань, приймається рішення про наявність ознаки $s \in S$. Є такі основні підходи визначення порогів для метрик: використання статистики – пороги визначаються за вибіркою результатів вимірювань метрики; використання загальноприйнятої семантики – пороги визначаються на основі загальних знань.

5. Побудувати функції φ_d та μ_d моделі дефекту, комбінуючи функції для визначення інтенсивностей його ознак за допомогою функцій агрегування. Для оцінювання інтенсивності простої ознаки $s \in S$ залежно від того, чи є поріг верхньою або нижньою межею значень оцінної метрики, пропонуємо такі функції:

– $I_s(e) = HigherThan\,(M_s(e), Z_s) = \dfrac{M_s(e)}{Z_s} \cdot 100$ – якщо поріг є верхньою межею значень метрики;

– $I_s(e) = LowerThan\,(M_s(e), Z_s) = \dfrac{Z_s}{M_s(e)} \cdot 100$ – якщо поріг є нижньою межею значень метрики.

Для побудови функцій знаходження інтенсивностей складених ознак (в тому числі і φ_d), комбінуючи функції знаходження інтенсивностей складових, пропонуємо застосувати агрегувальні функції:

– $I_s(e) = min(I_{s_1}(e), I_{s_2}(e))$ – у випадку складеної ознаки $s \in S$, що визначається одночасним проявом складових ознак $s_1, s_2 \in S$;

– $I_s(e) = max(I_{s_1}(e), I_{s_2}(e))$ – у випадку складеної ознаки $s \in S$, що визначається проявом хоча б однієї із складових ознак $s_1, s_2 \in S$.

Для побудови функції μ_d застосуємо функцію $mean$, тобто функцію знаходження середнього арифметичного аргументів.

Метамодель об'єктно-орієнтованого програмного забезпечення – це опис сутностей програмного забезпечення, їх властивостей і відношень. Подамо DDHM у вигляді вертикальних та горизонтальних шарів (рис. 2).

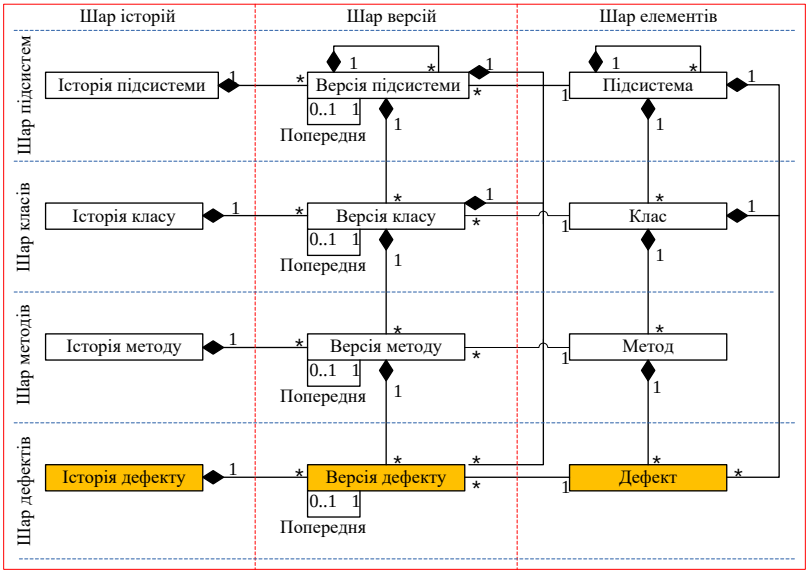


Рис.2. Метамодель історії дефектів проектування

Вертикальні шари містять такі узагальнені сутності: елементи, версії, історії. Горизонтальні шари містять сутності, що репрезентують елементи модельованого програмного забезпечення, їх версії та історії. Особливість DDHM полягає в тому, що сутностями метамоделі, крім версій і історій елементів конструкції програмного забезпечення (методу, класу, підсистеми), є також версії та історії дефектів. Формально DDHM визначено як типізований, навантажений орієнтований мультиграф $MModel =< N, E >$, де $N = \{n_1, n_2, n_3, ..., n_m\}$ – вузли, що описують сутності; $E = \{e_1, e_2, e_3, ..., e_k\}$ – ребра, що описують відношення між сутностями. Засоби зворотної інженерії використовуються для добування екземпляра DDHM з вихідного коду програмного забезпечення.

Ключову роль у побудові зображень для моніторингу дефектів відіграє ступінь розвитку дефекту проектування. У контексті запропонованого методу зображення показують дефекти проектування елементів конструкції програмного забезпечення таких рівнів абстракції: рівня методу, рівня класу, рівня підсистеми, та в таких аспектах: історії розпаду програмного забезпечення, історії розвитку дефекту проектування, історії розвитку ознак дефекту проектування.

Для відображення дефектів проектування в аспекті історії розпаду програмного забезпечення пропонуємо зображення «Рентгенограма», призначене для вирішення таких завдань: моніторингу розподілу дефектів за елементами конструкції; формування загального уявлення про стан розпаду програмного забезпечення.

«Рентгенограму» будуємо застосовуючи алгоритми візуалізації до матриці $A = [A_{ij}]$, де $A_{ij} \in N$ – вузол типу $methodV$ або $classV$ (версія методу або версія класу) графу-екземпляра DDHM; i – номер складового елемента конструкції

програмного забезпечення (методу або класу); j – номер версії програмного забезпечення. Група рядків репрезентує складений елемент конструкції, який складається зі складових елементів. Матрицю показано у вигляді таблиці (рис.3, *а*). Колір комірки таблиці відображає ступінь розвитку найбільш розвиненого дефекту проектування відповідної версії елемента конструкції. Використовуються відтінки сірого кольору – чим темніший колір, тим більший ступінь розвитку дефекту.

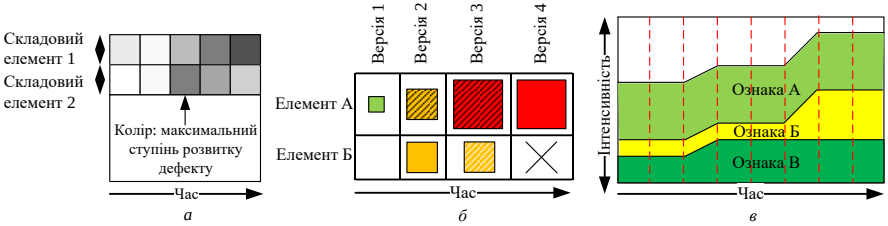


Рис.3. Правила представлення: *а* – «Рентгенограма»; *б* – «Історія дефекту»; *в* – «Історія ознак дефекту»

Для відображення дефектів проектування в аспекті історії розвитку дефекту проектування пропонуємо зображення «Історія дефекту», призначене для відстеження дефектів проектування певного типу. Тип елементів конструкції (підсистеми, класу, методу) і відповідний тип дефектів проектування вибирає користувач.

Зображення будуємо застосовуючи алгоритми візуалізації до матриці $A=[A_{ij}]$, де $A_{ij} \in N$ – вузол типу *defectV* (версія дефекту) графу-екземпляра DDHM; i – номер дефекту проектування; j – номер версії програмного забезпечення. Матрицю показано у вигляді таблиці (рис.3, *б*). Прямокутники в комірках таблиці відображають версії дефектів проектування. Три можливі кольори прямокутників вказують на ступінь розвитку дефекту проектування відповідно до підобласті, в яку потрапляє його значення. На області значень ступеня розвитку дефекту проектування виділимо такі підобласті: зелену – інтервал (0,75], в якому значення ступеня розвитку дефекту проектування далеке від перевищення порога; жовту – інтервал (75,100], в якому значення ступеня розвитку дефекту проектування близьке до перевищення порога; червону – інтервал (101,1000], в якому значення ступеня розвитку дефекту проектування перевищило поріг. Довжина діагоналі прямокутника відповідає ступеню розвитку версії дефекту. Якщо прямокутника немає в комірці таблиці, то дефекту в цій версії елемента конструкції теж немає. Візерунок чорного кольору всередині прямокутника означає, що ступінь розвитку версії дефекту збільшився порівняно з попереднім, або залишався незмінним, але збільшилася середня інтенсивність ознак дефекту. Візерунок білого кольору всередині прямокутника означає, що ступінь розвитку версії дефекту зменшився порівняно з попереднім, або залишався незмінним, але зменшилася середня інтенсивність ознак дефекту. Таким чином, невеликі зміни ступеня розвитку дефекту проектування, які візуально не помітні, все одно будуть зафіксовані.

Для відображення дефектів проектування в аспекті історії розвитку ознак дефекту пропонуємо зображення «Історія ознак дефекту», яке призначене для відстеження ознак дефекту проектування. Елемент конструкції (підсистема, клас, метод) визначається обраним користувачем дефектом проектування.

Зображення будуємо застосовуючи алгоритми візуалізації до матриці-рядка $A=[A_j]$, де $A_j \in N$ – вузол типу *defectV* графу-екземпляра DDHM; j – номер версії програмного забезпечення. Матрицю-рядок показано у вигляді набору червоних пунктирних ліній (рис. 3, *в*). Кожну ознаку дефекту проектування показано горизонтальним прошарком, товщина якого в точках перетину з пунктирними лініями відповідає інтенсивності прояву ознаки відповідної версії дефекту. Товщина всіх шарів відповідає сумі інтенсивностей всіх ознак і пропорційна середній інтенсивності ознак дефекту.

Для проведення моніторингу дефектів проектування пропонуємо методику, розроблену на основі досвіду застосування запропонованого методу. В основу методики покладено інтерпретацію графічних зображень. Основним зображенням для моніторингу дефектів проектування є «Історія дефекту», оскільки воно зосереджує основну увагу на власне дефекті проектування, а зображення «Рентгенограма» та «Історія ознак дефекту» є допоміжними. Тому для інтерпретації зображення «Історії дефекту» пропонуємо каталог категорій дефектів проектування. Категорію дефектів проектування визначимо як групу дефектів проектування, об'єднаних спільністю їх історії. Кожній категорії відповідає: ім'я, за яким категорію ідентифікують в каталозі; зразок, за яким можна встановити належність дефекту категорії при аналізі зображення «Історія дефекту». У цьому випадку зразок візуальний, хоча може бути заданий в іншій формі, наприклад, для автоматичного встановлення належності дефекту категорії; зразок інтерпретації історії дефекту, відображеного на зображенні «Історія дефекту». Оскільки дефекти однієї категорії мають однакові історії, то інтерпретувати їх можна однаково. Тому процедура інтерпретації зводиться до віднесення дефекту на основі його зображення до тієї чи іншої категорії, для якої задано зразок інтерпретації.

Основні категорії каталогу: «Збільшення», «Зменшення», «Пульсування», «Стабільність».

До категорії «Збільшення» належать дефекти проектування, ступінь розвитку яких збільшується протягом історії (рис. 4, *а*). Зростання дефекту вказує на те, що під час підготовки нових версій роботи із супроводу ураженого дефектом елемента конструкції програмного забезпечення проводять ігноруючи правила проектування.

До категорії «Зменшення» належать дефекти проектування, ступінь розвитку яких зменшується протягом історії (рис. 4, *б*). Зменшення дефекту вказує передусім на проведену реструктуризацію, спрямовану на усунення даного дефекту.

До категорії «Пульсування» належать дефекти проектування, ступінь розвитку яких поперемінно то збільшується, то зменшується протягом історії (рис. 4, *в*). Пульсування ступеня розвитку дефекту проектування вказує на те, що уражений елемент конструкції піддається змінам, що порушують правила

проектування програмного забезпечення. Далі дефект виявляють, установлюють його негативний вплив на здатність до супроводу й усувають. Проте з часом інженери із супроводу знову змушені порушувати правила проектування. Такі дефекти найбільш небезпечні для подальшої еволюції програмного забезпечення, оскільки вказують на серйозні проблеми щодо супроводу ураженого елемента конструкції і нездатність персоналу усунути їх.

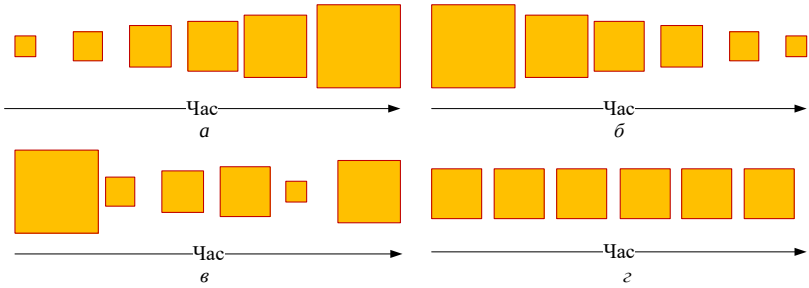


Рис.4. Візуальні зразки: *а* – «Збільшення»; *б* – «Зменшення»; *в* – «Пульсування»; *г* – «Стабільність»

До категорії «Стабільність» належать дефекти проектування, ступінь розвитку яких залишається стабільним протягом історії (рис. 4, *г*). Стабільність дефекту проектування означає, що він не створює проблем із супроводу ураженого елемента конструкції, тому такий дефект безпечний. Кожну з категорій, крім «Стабільність», поділимо на підкатегорії. Першій підкатегорії належать дефекти, ступінь розвитку яких протягом усієї історії вищий за 100% (верхня), другій – ступінь розвитку яких протягом історії переходить значення 100% у будь-якому напрямку (перехідна), третій – ступінь розвитку яких протягом усієї історії нижчий за 100% (нижня).

Таким чином, за методикою проведення моніторингу дефектів проектування необхідно виконати такі загальні кроки:

1. Оцінити стан розпаду програмного забезпечення, оцінюючи наскільки затемненою є «Рентгенограма» для прийняття рішення про необхідність подальшого моніторингу дефектів. Особливо несприятливим є затемнення в правій частині по всій висоті зображення, що свідчить про інтенсивний розпад програмного забезпечення.

2. Визначити на «Рентгенограмі» складні елементи конструкції (такі як підсистема), де найбільш імовірно можна виявити дефекти проектування, що належать до найнебезпечніших категорій, як то «Верхнє пульсування» або «Верхнє збільшення». Такі елементи, як правило, зображуються особливо затемненими зонами.

3. Перейти до моніторингу дефектів проектування, в першу чергу визначених на попередньому кроці елементів конструкції, за допомогою зображення «Історія дефекту».

4. Знайти фрагменти зображення «Історія дефекту», що відповідають візуальним зразкам категорій дефектів проектування.

5. Віднести дефекти, історії яких репрезентовані знайденими фрагментами «Історії дефекту», до відповідних категорій.

6. Інтерпретувати історії віднесених до категорій дефектів на основі зразків інтерпретації відповідних категорій.

7. Розглянути представлення «Історія ознак дефекту» для дефектів, причину розвитку яких необхідно з'ясувати (спостереження за розвитком інтенсивностей ознак – за товщиною відповідних прошарків) та історію їх середньої інтенсивності (спостереження за розвитком сумарної інтенсивності всіх ознак – за товщиною усіх прошарків).

У **третьому** розділі розглянуто архітектуру засобу моніторингу дефектів проектування об'єктно-орієнтованого програмного забезпечення (рис.5).

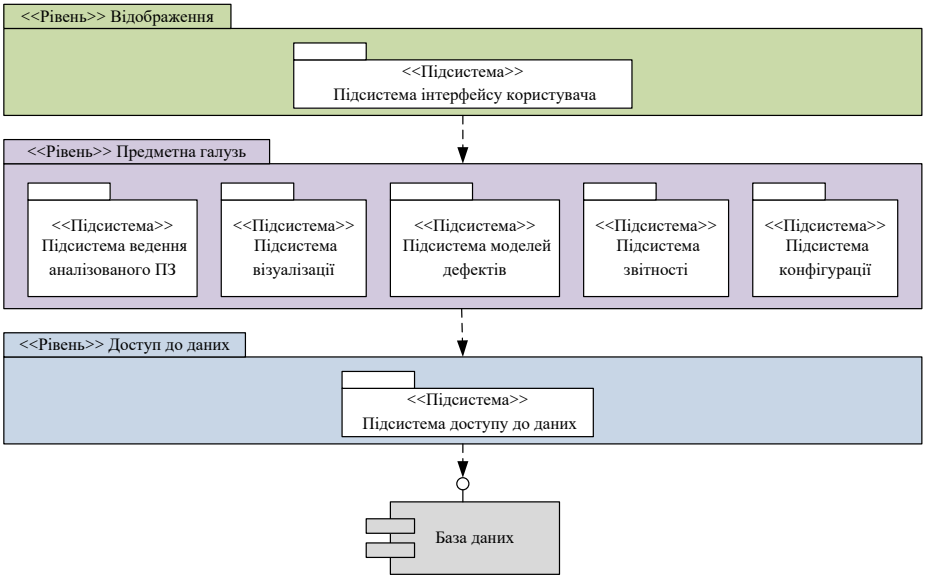


Рис.5. Архітектура засобу моніторингу дефектів проектування: ПЗ – програмне забезпечення

Розроблений засіб належить до інструментів візуалізації програмного забезпечення (Software Visualization Tools) і виконує такі функції: ведення моделей аналізованого програмного забезпечення, ведення моделей дефектів проектування, добування екземпляра DDHM з вихідних кодів аналізованого програмного забезпечення, збереження екземпляра DDHM у базі даних, побудова графічних зображень для моніторингу дефектів проектування в різних аспектах, взаємодія користувача із зображеннями, складання звітів про проведений моніторинг.

Для забезпечення здатності до перенесення, розширюваності – якостей, необхідних для дослідницьких інструментів, засіб розроблено в архітектурному стилі «багаторівнева архітектура» та включає такі рівні:

– відображення – верхній рівень, який реалізує взаємодію користувача із засобами в діалоговому режимі, а також уведення і виведення даних з підсистем рівня предметної галузі;

– предметної галузі – середній рівень, що реалізує основну функціональність засобів;

– доступу до даних – нижній рівень, реалізує доступ підсистем рівня предметної галузі до даних в базі даних.

Рівні відображення та доступу до даних містять по одній підсистемі (підсистему інтерфейсу користувача і підсистему доступу до даних відповідно).

Для того щоб зменшити кількість залежностей підсистеми інтерфейсу користувача від компонентів підсистем рівня предметної галузі до однієї, застосуємо шаблон проектування «Фасад». Роль «фасаду» у кожній підсистемі відіграє компонент керування, який надає спрощений інтерфейс підсистеми. Компонент керування обробляє запити, що надходять від підсистеми інтерфейсу користувача, і забезпечує спільну роботу компонентів відповідної підсистеми. Підсистема інтерфейсу користувача реалізує взаємодію користувача із засобом моніторингу дефектів у діалоговому режимі.

Підсистема ведення аналізованого програмного забезпечення виконує такі функції: додавання, змінення, видалення екземплярів DDHM; додавання, змінення, видалення версій систем аналізованого програмного забезпечення; перегляд і навігація за елементами конструкції кожної версії систем аналізованого програмного забезпечення; пошук за елементами конструкції кожної версії систем аналізованого програмного забезпечення.

Підсистема візуалізації виконує такі функції: підготовку даних для побудови графічних зображень з урахуванням обмежень на тип і кількість зображених дефектів проектування або конструкцій програмного забезпечення; побудова графічних зображень; взаємодія користувача із зображеннями.

Підсистема моделей дефектів вирішує такі завдання: перегляд списку моделей дефектів проектування, що зберігаються в базі даних, та їх описів; завдання граничних значень метрик як опорних точок для визначення інтенсивностей ознак дефектів; додавання, змінення, видалення, пошук і тестування моделей дефектів; перегляд списку метрик, які можна отримати за допомогою засобу; вибір метрик, за допомогою яких можна оцінити інтенсивності ознак дефектів; додавання, змінення, видалення ознак дефектів проектування.

Підсистема звітності вирішує такі завдання: додавання, змінення, видалення шаблонів звітів; додавання, змінення, видалення звітів; додавання, змінення, видалення запитів даних про результати моніторингу; побудова запитів; генерація звітів; перегляд звітів; пошук звітів; перегляд результатів вимірювань аналізованого програмного забезпечення.

Підсистема конфігурації вирішує такі завдання: перегляд і пошук налаштувань засобу; змінення параметрів засобу; збереження і відновлення налаштувань засобу.

Базу даних призначено для зберігання екземплярів DDHM, моделей дефектів проектування, шаблонів та звітів про моніторинг, налаштувань і службової інформації.

У **четвертому розділі** описано результати практичного застосування запропонованого методу і засобу моніторингу дефектів проектування об'єктно-орієнтованого програмного забезпечення. Згідно з розробленою методикою проведено моніторинг дефектів проектування засобу UML моделювання ArgoUML (http://argouml.tigris.org/) з відкритим вихідним кодом, написаним мовою програмування Java. Виконано моніторинг дефектів «God Class», «Brain Method» і «Shotgun surgery». Ці дефекти обрано тому, що вони часто трапляються в об'єктно-орієнтованому програмному забезпеченні та негативно впливають на його здатність до супроводу. Для кожного з цих дефектів побудовано відповідну модель. Для прикладу наводимо результати тільки для дефекту проектування «God Class». Модель цього дефекту складається з таких функцій:

$$\varphi_d(e) = min(HigherThan(WMC(e),36), LowerThan(TCC(e),0.33), HigherThan(ATFD(e),3));$$

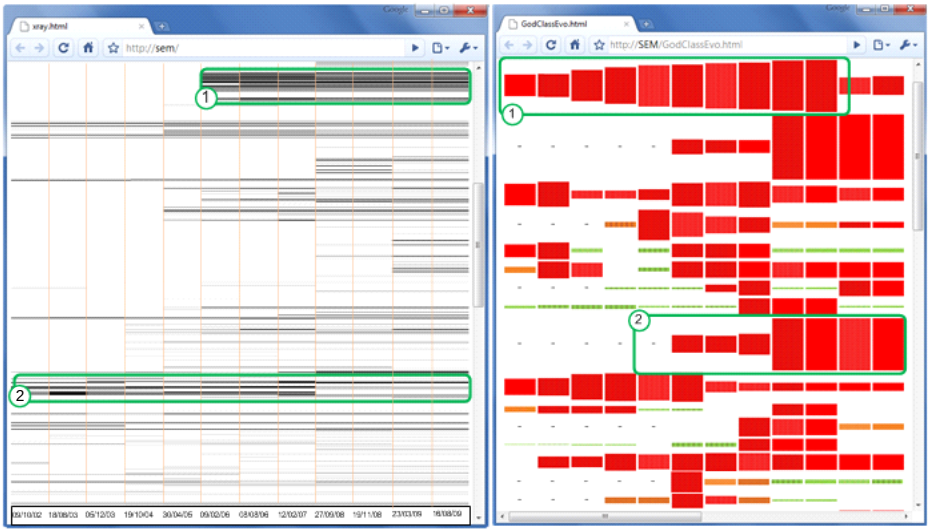
$$\mu_d(e) = mean(HigherThan(WMC(e),36), LowerThan(TCC(e),0.33), HigherThan(ATFD(e),3)),$$

де WMC (Weighted Method Count), TCC (Tight Class Cohesion), ATFD (Access To Foreign Data) – метрики. Розподіл дефектів проектування «God Class» за запропонованими категоріями показує, що 34% дефектів протягом усієї історії не змінювалися і становлять найчисельнішу категорію. Ці результати пояснюють нездатність існуючих засобів точно розпізнавати дефекти проектування – значна частина дефектів не змінюються, а отже, на супровід уражених ними елементів конструкції не витрачаються зусилля. Розподіл дефектів за підкатегоріями категорій «Пульсування» і «Збільшення» показує, що найчисельніша підкатегорія – нижня, в якій містяться дефекти, що зароджуються, тобто ті, ступінь розвитку яких не перевищував 100%. Найменша категорія – верхня, що містить найбільш небезпечні дефекти: «Верхнє збільшення» – 4%, «Верхнє пульсування» – 5%.

Для формування відправної точки діагностики та загальної картини розпаду системи ArgoUML побудовано графічне зображення «Рентгенограма» (рис. 6, *а*). Як складений елемент конструкції обрано пакет, а як складовий – клас. Кольором прямокутника показано ступінь найбільш розвиненого дефекту проектування відповідної версії елемента конструкції, а оскільки в розділі розглядається лише один дефект проектування класу, то саме його ступінь розвитку показано кольором прямокутника. На зображенні видно велику кількість затемнених ділянок, дві з яких виділено для подальшого розгляду.

На виділеній підобласті *1* зображено пакет org.argouml.model.mdr. Видно, що пакет, імовірно, був створений у шостій версії, оскільки всі дефекти проектування виникають різко й одночасно. Після створення велика частина класів цього пакета уражена дефектами проектування з високими ступенями розвитку, що вказує на необхідність моніторингу дефектів у цьому пакеті. У дев'ятій версії, очевидно, була проведена реструктуризація частини класів пакета, про що свідчить освітлені деякі лінії, після чого пакет залишається досить стабільним. Виділена підобласть *2* зображує пакет org.argouml.uml.diagram.deployment.ui, вивчення якого показує, що з початку історії пакета більша частина його класів були уражені дефектами проектування, крім того, протягом п'яти років (2002 – 2007рр.)

дефекти були нестабільними (інтенсивна зміна рівня сірого, що вказує на нерівномірність). Починаючи з дев'ятої версії кількість дефектів зменшено до трьох, причому їх ступінь розвитку не високий і стабільний до останньої версії. Це свідчить про перероблення розглянутого пакета, тому подальша його інспекція не потрібна.



а *б*

Рис.6. Зображення, побудовані для дефекту «God Class»: *а* – «Рентгенограма»; *б* – «Історія дефекту»

Перейдемо до аналізу зображення «Історія дефекту» (6, *б*), в ході якого також звернемося до аналізу зображення «Історія ознак дефекту» для детального розгляду історії ознак дефекту «God Class». На зображенні виділені підобласті, ідентифіковані цифрами.

Виділена підобласть *1* зображує дефект проектування, що уразив клас org.argouml.uml.diagram.ui.FigNodeModelElement. З рисунка видно, що дефект в цьому класі існував з його першої версії і зростав протягом майже всієї історії класу. Білий візерунок у п’ятій та сьомій версіях свідчить про те, що, не зважаючи на збільшення ступеня розвитку дефекту, середня інтенсивність його ознак в цих версіях зменшувалась, що могло бути викликано зменшенням інтенсивності ознаки, яка не визначає ступеня розвитку дефекту з огляду на свою немінімальність. Незважаючи на значне зменшення дефекту в передостанній версії, в останній версії знову намітилось його зростання і ступінь розвитку становив 300%. У зв’язку з цим можна зробити висновок, що дефект належить категорії «Верхнє пульсування» і його необхідно усунути. Оскільки цей дефект дуже виражений, розглянемо збудоване для нього графічне зображення «Історія ознак дефекту» (рис 7).

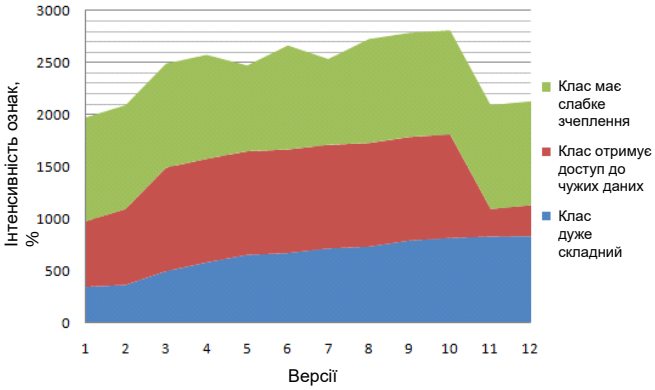


Рис.7. «Історія ознак дефекту» для дефекту «God Class» класу org.argouml.uml.diagram.ui.FigNodeModelElement

Три шари зображують три ознаки дефекту «God Class». Видно, що протягом історії з десяти версій визначальною залишалася ознака дефекту «Клас дуже складний» (її зображено шаром найменшої товщини). Причому ця ознака весь час продовжувала зростати. Різке зменшення ступеня розвитку спричинено різким зменшенням в одинадцятій версії ознаки «Клас отримує доступ до чужих даних», після чого визначальною стає ця ознака. Ознака «Клас має слабке зчеплення» протягом усієї історії має максимальну інтенсивність за винятком п'ятої та сьомої версії, де спостерігаються невеликі спади, що пояснюють появу у відповідних версіях білого візерунка на зображенні «Історія дефекту».

Виділена підобласть *2* зображує дефект проектування, що уразив клас org.argouml.model.mdr.CoreFactoryMDRImpl. У дев'ятій версії дефект досяг ступеня розвитку 800%, який надалі не змінювався, тому дефект належить до категорії «Верхнє збільшення». Незважаючи на стабільність ступеня розвитку дефекту в останніх чотирьох версіях, його збільшення майже в три рази протягом історії свідчить на користь необхідності усунення цього дефекту.

ВИСНОВКИ

Дисертаційна робота є теоретично обґрунтованим дослідженням, містить узагальнення й практичне вирішення важливого науково-практичного завдання, сутність якого полягає в розробленні методу та засобу моніторингу дефектів проектування об'єктно-орієнтованого програмного забезпечення. Метод призначено для застосування під час супроводження та еволюційного розроблення програмного забезпечення і дозволяє відстежувати розвиток дефектів проектування. Застосування методу забезпечується засобом, який розроблено і реалізовано в дисертації.

Основні наукові та практичні результати роботи полягають у такому:

1. Уперше, шляхом аналізу процесів і практик виявлення та усунення дефектів проектування, сформульовано і вирішено завдання моніторингу дефектів проектування об'єктно-орієнтованого програмного забезпечення, що сприяє вирішенню завдань діагностики програмного забезпечення, наприклад контроль та прогнозування технічного стану програмного забезпечення.

2. Уперше запропоновано метод моніторингу дефектів проектування об'єктно-орієнтовного програмного забезпечення, сутність якого полягає у відстеженні змін параметрів дефектів проектування у часі. Метод реалізується використанням пропонованої метамоделі історії дефектів проектування (DDHM) об'єктно-орієнтованого програмного забезпечення і багатоаспектної візуалізації дефектів проектування елементів конструкції різного рівня абстракції.

3. Розроблено методику побудови моделей дефектів проектування.

4. Розроблено метамодель історії дефектів проектування, яка дозволяє реалізовувати засоби для візуалізації та аналізу історій дефектів проектування та уражених ними елементів конструкції програмного забезпечення.

5. Розроблено набір графічних зображень, які спрощують відстеження розвитку дефектів проектування та алгоритми підготовки даних для побудови цих графічних зображень на основі екземплярів DDHM.

6. Розроблено методику проведення моніторингу дефектів проектування.

7. Розроблено та реалізовано, на основі архітектурного стилю «багаторівнева архітектура», архітектуру засобу моніторингу дефектів, який створює екземпляри DDHM на основі вихідних кодів програмного забезпечення та виконує візуалізацію дефектів проектування відповідно до запропонованого методу.

8. За допомогою розроблених у дисертації методу та засобу проведено моніторинг дефектів проектування реального програмного забезпечення з відкритим вихідним кодом ArgoUML. Вибір зумовлено зрозумілістю предметної галузі і тим, що це програмне забезпечення супроводжують понад сім років. Таким чином, показано дієвість і практичну значущість методу та засобу для діагностики програмного забезпечення. Результати дисертаційної роботи впроваджено в навчальний процес Національного авіаційного університету та в розроблення програмного забезпечення у відкритому акціонерному товаристві «КП ОТІ».

СПИСОК ОПУБЛІКОВАНИХ АВТОРОМ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1. Нечай О.С. Реверсивная инженерия программного обеспечения // Інженерія програмного забезпечення 2007 : Всеукр. конф. асп. і студ., Київ, 4–5 грудня 2007р. : тези доп. – К.: НАУ, 2007. – С.18.

2. Нечай О.С. Дефекты конструкции объектно-ориентированного программного обеспечения // Теоретичні та прикладні аспекти побудови програмних систем : п'ята міжнар. конф. TAAPSD'08, Київ, 22–24 вересня 2008р. : тези доп. – К.: ПУЛЬСАРИ, 2008. – С.47–51.

3. Нечай О.С. Метод побудови нечітких моделей дефектів проектування програмного забезпечення // ПОЛІТ-2009 : IX міжнар. наук. конф. студ. та молодих учених, Київ, 8–10 квітня 2009р. : зб. тез доп. – К. : Вид-во Нац. авіац.

ун-ту «НАУ-друк», 2009. – С.223.

4. Нечай О.С. Методи та засоби виявлення дефектів проектування об'єктно-орієнтованого програмного забезпечення / О.С. Нечай, М.О. Сидоров // Вісник НАУ. – 2009. – №3. – С. 200–205.

5. Нечай О.С. Метод побудови моделей дефектів проектування обєктно-орієнтованого програмного забезпечення / О.С. Нечай, М.О. Сидоров // Наукоємні технології. – 2009. – № 2. – С.58–64.

6. Нечай О.С. Метод діагностики об'єктно-орієнтованого програмного забезпечення / О.С.Нечай // Вісник НАУ. – 2010. – №1. – С. 172–180.

7. Нечай О.С. Засоби діагностики об'єктно-орієнтованого програмного забезпечення / О.С. Нечай // Наукові записки НаУКМА. Комп'ютерні науки : зб. наук. пр. – К. : Видавничо-поліграфічний центр НаУКМА, 2010. – Т. 112. – С. 61–70.

8. Нечай А.С. Мониторинг дефектов проектирования объектно-ориентированного программного обеспечения / А.С. Нечай // Проблеми програмування; НАН України. – 2010.- №2–3. – С. 242–251.

АНОТАЦІЯ

Нечай О.С. Метод та засіб моніторингу дефектів проектування об'єктно-орієнтованого програмного забезпечення. – Рукопис.

Дисертація на здобуття вченого ступеня кандидата технічних наук зі спеціальності 01.05.03 – математичне та програмне забезпечення обчислювальних машин і систем. – Національний авіаційний університет, Київ, 2010.

Дисертаційну роботу присвячено розробленню методу та засобу моніторингу дефектів проектування об'єктно-орієнтованого програмного забезпечення.

Моніторинг дефектів проектування проводять при діагностиці програмного забезпечення. Запропоновано метод моніторингу, сутність якого полягає у спостереженні за зміною параметрів дефектів проектування у часі. Метод реалізується шляхом використання пропонованої метамоделі історії дефектів проектування (DDHM – Design Flaws History Meta-Model) об'єктно-орієнтованого програмного забезпечення і багатоаспектної візуалізації дефектів проектування елементів конструкції різного рівня абстракції. Розроблено архітектуру засобу, який забезпечує реалізацію запропонованого методу, та перевірено його працездатність.

Ключові слова: дефекти проектування, об'єктно-орієнтоване проектування, правила проектування, реструктуризація, еволюція програмного забезпечення

АННОТАЦИЯ

Нечай А.С. Метод и средство мониторинга дефектов проектирования объектно-ориентированного программного обеспечения. – Рукопись.

Диссертация на соискание ученой степени кандидата технических наук по специальности 01.05.03 – математическое и программное обеспечение вычислительных машин и систем. – Национальный авиационный университет, Киев, 2010.

Диссертационная работа посвящена разработке метода и средства мониторинга дефектов проектирования объектно-ориентированного программного обеспечения.

Мониторинг дефектов проектирования проводится при диагностике программного обеспечения. Предложен метод мониторинга, сущность которого состоит в наблюдении за изменением параметров дефектов проектирования во времени, в первую очередь степени развития. Метод реализуется путем использования предлагаемой метамодели истории дефектов проектирования (DDHM – Design Flaws History Meta-Model) объектно-ориентированного программного обеспечения и многоаспектной визуализации дефектов проектирования элементов конструкции разного уровня абстракции. Разработана архитектура средства, обеспечивающего реализацию предложенного метода, и проверена его работоспособность.

Ключевые слова: дефекты проектирования, объектно-ориентированное проектирование, правила проектирования, реструктуризация, эволюция программного обеспечения

ABSTRACT

Alexander S. Nechay. The Method and Tool for Object-Oriented Software Design Defects Monitoring. – Manuscript.

Thesis for a candidate's degree on technical science by specialty 01.05.03 – Mathematical support and software of the computing machines and systems. – National Aviation University, Kyiv, Ukraine, 2010.

The thesis is devoted to developing method and tool for monitoring design defects in object-oriented software.

The problem of design defects monitoring is formulated and solved by means of devoted method development. We describe design defect ontology in order to make systematic, comprehensive and complete overview of design defect notion and related notions.

Design defect causes the emergence of symptoms (signs) and is diagnosed based on symptoms, using the diagnostic methods and tools. Symptoms are determined by software design items' characteristics analysis. Characteristics are extracted from the software design items by using special techniques and methods such as measurement or static code analysis. Design defect has a negative impact on one or more internal attributes of software quality (similar to design patterns, which mainly have a positive effect on software quality attributes). Patterns can be implemented in software by using techniques for defects removal. Design defect is caused by cause as result of the design rules violation and can be prevented by defect preventing techniques.

Design defect is nonconformance of the structural characteristics of an software item or software fragment to the rules of object-oriented design. This definition restricts the object of our research only to non-functional defects that affect the structure of object-oriented software. Design defects are classified by their nature, by the type of affected software item and by the ability to be measured.

By their nature, defects are divided in such groups: identification – defects related to incorrect definition of structural elements of a software system, cooperation – defects related to incorrect definition of the relationship between the elements of software

design, classification – defects related to incorrect application classes' hierarchies. By type of affected software design element defects are divided into the following groups: method defects, class defects, subsystem defects. By possibility to be measured defects are divided in following groups: measurable defects – in the process of software maintenance they can not only appear and disappear, but change the degree of their progress; immeasurable defects – defects, which in the process of software maintenance can only appear and disappear as a result of restructuring.

We propose method for monitoring of object-oriented design defects, the essence of which is control changes of design flaws' parameters (especially flaw's progress degree) in time. The method is implemented by using object-oriented software design defects history meta-model (DDHM) and multi-dimensional visualization of design defects in software elements at different levels of abstraction. In DDHM, unlike in other meta-models, defect for the first time is modeled as a separate entity which can change its parameters in time. This enables one to implement tools for defects' and flawed software elements' history visualization and analysis. To apply proposed method one must accomplish the following: build models of defects in interest, using reverse engineering tools in order to extract an instance of DDHM from several versions of analyzed software source code, build on basis of obtained instance set of visualizations for design defects monitoring, perform visual analysis of resulting views via interaction with them.

To make possible visualization of design defects they have to be modeled. We propose a design defect model, which consists of two functions. First function is used to find design defect's progress degree and second – to find mean value of intensities of all defect's simple signs. Such a model allows considering of defect to be an entity, which has changeable in time parameters. Visualization of these parameters allows tracking of design defects and their progress therefore model is used in proposed method implementation. To build a design defect model one should accomplish the following: formulate rule for software item design, analyze resulting rule to find a signs of nonconformance with this rule, choose a metric for estimation of intensity of every sign, for every resulting metric set a threshold, build defect model's functions combining function for signs intensity calculation by mean of aggregating functions.

The set of views for facilitation of design defect progress observation is developed. They are: "X-Ray", "Defect history", "Defect's signs history". First view is used for defects visualization in software decay aspect and allows estimate defect distribution among software elements. Second view is used for defects visualization in design defect history's aspect and allows tracking design defects of certain type. Third view is used for defects visualization in design defect signs history's aspect and allows tracking design defect's signs. Also defect monitoring technique based on these views is developed. The technique is build upon searching in views of visual patterns with known interpretation. The architecture of a tool to support implementation of the proposed method is developed. Its efficiency verified by monitoring of design defects during the history of open source project ArgoUML. Results of this thesis are implemented in educational process of National Aviation University and industry.

Keywords: design flaws, object-oriented design, design rules, restructuring, evolution of software.