Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
Факультет інформаційних технологій
Кафедра безпеки інформації та телекомунікацій

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеню магістра

студента *Гезя Максима Віталійовича*

академічної групи *125м-17-2*

спеціальності *125 Кібербезпека*

спеціалізації[1]

за освітньо-професійною програмою *Кібербезпека*

на тему *Вдосконалення організації захисту інформації на підприємстві по розробці програмних продуктів*

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	д.ф.-м.н., проф. Кагадій Т.С.			
розділів:				
спеціальний	ст. викл. Саксонов Г.М.			
економічний	к.е.н., доц. Пілова Д.П.			

Рецензент				

Нормоконтролер	ст. викл. Мєшков В.І.			

Дніпро
2018

ЗАТВЕРДЖЕНО:
завідувач кафедри
безпеки інформації та телекомунікацій
_____ д.т.н., проф. Корнієнко В.І.

«_____»_____20___року

ЗАВДАННЯ
на кваліфікаційну роботу ступеня магістра

студенту ***Гезю М.В.*** **академічної групи** ***125м-17-2***
(прізвище та ініціали) (шифр)

спеціальності ***125 Кібербезпека***

спеціалізації[1]

за освітньо-професійною програмою ***Кібербезпека***

на тему ***Вдосконалення організації захисту інформації на підприємстві по розробці програмних продуктів***

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 29.11.2018 № 2025-л_____

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень ***існуючі процедури проектування, розробки та впровадження захищеного програмного забезпечення (ПЗ) на підприємства галузі інформаційних технологій***

Предмет досліджень ***методи та моделі забезпечення безпечної процедури проектування, розробки та впровадження захищеного ПЗ***

Мета ***вдосконалити процес забезпечення безпечної процедури проектування, розробки та впровадження захищеного ПЗ та надати рекомендації щодо вдосконалення процедур***

Вихідні дані для проведення роботи ***законодавство України та міжнародні стандарти у сфері кібербезпеки та розробки ПЗ, існуючі алгоритми оцінки загроз інформаційної безпеки підприємства***

3 ОЧІКУВАНІ РЕЗУЛЬТАТИ

Наукова новизна ***результатів полягає у впровадженні нового елементу політики інформаційної безпеки підприємства, в якому розглянуті проблема***

пов'язані з процесами проектування, розробки та впровадження захищеного програмного забезпечення на підприємства галузі інформаційних технологій

Практична цінність *розроблені рекомендації щодо вдосконалення процедури проектування, розробки та впровадження захищеного програмного забезпечення*

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Закону України «Про інформацію», НД ТЗІ 1.1-002-99, НД ТЗІ 2.5-004-99, НД ТЗІ 2.5-005-99, НД ТЗІ 3.7-001-99, ДСТУ ISO/IEC 9126-1:2001, ДСТУ ISO 9000-2001, ДСТУ ISO 9001-2001, ДСТУ ISO 9004:2001, ДСТУ ISO/IEC 90003:2006

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт** (початок-кінець)
Огляд джерел за темою та напрям досліджень	03.09.18-06.10.18
Методи досліджень	07.10.18-31.10.18
Результати досліджень	01.11.18-24.11.18
Виконання економічного розділу	25.11.18-04.12.18
Оформлення пояснювальної записки	05.12.18-10.12.18

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект *від реалізації результатів роботи очікується позитивним завдяки підвищенню захищеності програмних продуктів і зниження ймовірності виникнення дефектів при розробці ПЗ*

Соціальний ефект *від реалізації результатів роботи очікується позитивним завдяки створенню умов для ознайомлення працівників підприємства з правовою базою і встановленими процедурами та правилами роботи*

7 ДОДАТКОВІ ВИМОГИ

Завдання видано _____ Кагадій Т.С.
(підпис керівника) (прізвище, ініціали)

Дата видачі: 03.09.18р.

Дата подання до екзаменаційної комісії: 14.12.18р.

Прийнято до виконання _____ Гезь М.В.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: ___ с., __ рис., __ табл., 4 додатки, 17 джерел.

Об’єкт дослідження: існуючі процедури проектування, розробки та впровадження захищеного програмного забезпечення (ПЗ) на підприємства галузі інформаційних технологій.

Мета дипломної роботи: вдосконалити процес забезпечення безпечної процедури проектування, розробки та впровадження захищеного ПЗ та надати рекомендації щодо вдосконалення процедур.

В першому розділі розглянуті основні проблеми, які пов’язані з проектуванням, розробкою, впровадженням та підтримкою ПЗ. Розглянуті переваги та недоліки методів тестування програмних продуктів. Зазначені принципи проектування безпечного ПЗ. Наведений аналіз нормативної бази в галузі оцінки якості програмних продуктів.

В другому розділі наведені загрози несанкціонованого доступу до інформації, визначені елементи моделі загроз експлуатаційної безпеки ПЗ. Розглянуті проблеми управління персоналом на підприємствах галузі інформаційних технологій. Визначені основні принципи забезпечення безпеки ПЗ. Наведена типова організаційна структура підприємства по розробці ПЗ. Зазначені особливості колективного проектування, розробки та тестування ПЗ та використання програмних рішень таких як система керування версіями.

Новизна очікуваних результатів полягає у впровадженні нового елементу політики інформаційної безпеки підприємства, в якому розглянуті проблема пов’язані з процесами проектування, розробки та впровадження захищеного ПЗ на підприємства галузі інформаційних технологій.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, УПРАВЛІННЯ ПЕРСОНАЛОМ, НЕСАНКЦІОНОВАНИЙ ДОСТУП, ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, СИСТЕМА КЕРУВАННЯ ВЕРСІЯМИ.

РЕФЕРАТ

Пояснительная записка: ___ с., __ рис., __ табл., 4 приложений, 17 источников.

Объект исследования: существующие процедуры проектирования, разработки и внедрения защищенного программного обеспечения (ПО) на предприятия области информационных технологий.

Цель дипломной работы: усовершенствовать процесс обеспечения безопасной процедуры проектирования, разработки и внедрения защищенного ПО и дать рекомендации по совершенствованию процедур.

В первом разделе рассмотрены основные проблемы, связанные с проектированием, разработкой, внедрением и поддержкой ПО. Рассмотрены преимущества и недостатки методов тестирования программных продуктов. Указанные принципы проектирования безопасного ПО. Приведенный анализ нормативной базы в области оценки качества программных продуктов.

Во втором разделе приведены угрозы несанкционированного доступа к информации, определенные элементы модели угроз эксплуатационной безопасности ПО. Рассмотрены проблемы управления персоналом на предприятиях области информационных технологий. Определены основные принципы обеспечения безопасности ПО. Приведена типовая организационная структура предприятия по разработке ПО. Указанные особенности коллективного проектирования, разработки и тестирования ПО и использования программных решений такие как система управления версиями.

Новизна ожидаемых результатов заключается во внедрении нового элемента политики информационной безопасности предприятия, в котором рассмотрены проблема связана с процессами проектирования, разработки и внедрения защищенного ПО на предприятия области информационных технологий.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, УПРАВЛЕНИЕ ПЕРСОНАЛОМ, НЕСАНКЦИОНИРОВАННЫЙ ДОСТУП, ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, СИСТЕМА УПРАВЛЕНИЯ ВЕРСИЯМИ.

ABSTRACT

Explanatory note: ___ p., __ fig., __ tab., 4 application, 17 sources.

Object of study: the existing procedures for the design, development and implementation of protected software (software) in the field of information technology.

The aim of the thesis: to improve the process of ensuring a secure procedure for designing, developing and implementing secure software and to give recommendations on how to improve procedures.

The first section discusses the main problems associated with the design, development, implementation and support of software. The advantages and disadvantages of software testing methods are considered. The specified principles for the design of secure software. The analysis of the regulatory framework in the field of assessing the quality of software products.

The second section lists the threats of unauthorized access to information, certain elements of the model of threats to the operational security of software. The problems of personnel management at the enterprises of the field of information technology. Defined the basic principles of software security. The typical organizational structure of a software development enterprise is given. These features of the collective design, development and testing of software and the use of software solutions such as version control system.

The novelty of the expected results lies in the introduction of a new element of the company's information security policy, which addresses the problem associated with the design, development and implementation of secure software in the information technology industry.

SOFTWARE, PERSONAL MANAGEMENT, UNAUTHORIZED ACCESS, SOFTWARE TESTING, VERSION MANAGEMENT SYSTEM.

СПИСОК УМОВНИХ СКОРОЧЕНЬ

CMM	—	Capability Maturity Model;
VCS	—	Version Control System;
БД	—	база даних;
ІС	—	інформаційні системи;
ІТ	—	інформаційні технології;
ЗІ	—	захист інформації;
КС	—	комп’ютерні системи;
ЛОМ	—	локальна обчислювальна мережа;
НСД	—	несанкціонований доступ;
ОІД	—	об’єкт інформаційної діяльності;
ОС	—	операційна система;
ПІБ	—	політика інформаційної безпеки;
ПЗ	—	програмне забезпечення;
ПК	—	персональний комп’ютер.

ЗМІСТ

с.

ВСТУП

В процесі розробки програмного забезпечення, що повинно відповідати вимогам до конфіденційності, цілісності і доступності, виявляються серйозні проблеми, для їх вирішення необхідний персонал з відповідним рівнем освіти, підготовкою та досвідом. При створенні високонадійних програмних комплексів важливе значення має і правильна організація управління проектами. На практиці проблеми управління набагато частіше стають причинами краху проектів, ніж невдалі технічні рішення.

Удосконалення проектування програмного забезпечення та пов’язаних з ним процесів повинно вести до створення безпечного програмного забезпечення. Йдеться про програми, які надають нам впевненості (хоч і не гарантують абсолютного захисту і повної достовірності), включають в себе функції, необхідні для підтримання безпеки на належному рівні, і мають властивості, що гарантують правильність та коректність їх використання.

Крім того, вдосконалення процесів проектування сприяє оптимізації вартості розробки і появи програм з украй малою кількістю дефектів, що вимагають низьких витрат на технічну підтримку, що тягне за собою загальне підвищення репутації програмних продуктів.

Процес проектування починається з визначення специфікацій, що впливають на поведінку програм і відображають актуальні вимоги до безпеки. Необхідно, щоб всі компоненти системи відповідали цим умовам. Специфікації повинні забезпечувати потрібну функціональність і не мати вразливих місць. Незалежно від того, чи купується система на стороні або розробляється усередині компанії, всі її компоненти зобов’язані відповідати вимогам до безпеки.

Ще один момент, що стосується забезпечення безпеки, пов’язаний з правильністю проектування та реалізації. Програмне забезпечення працює правильно тільки в тому випадку, коли всі виконувані операції точно відповідають певним специфікаціям і не допускається нічого зайвого. Сьогодні

ж часто програми виконують дії, про які не знають навіть розробники і тестувальники.

Жорсткість вимог до стійкості і безпеки призвело до того, що з’явився ряд підходів до побудови високонадійних програмних систем.

У великих компаніях таких як Apple, IBM, Microsoft виявили, що близько половини всіх проблем, пов’язаних з безпекою програм, обумовлені недоліками проектування.[4] А щоб усунути останні, необхідний висококваліфікований персонал, що володіє великим досвідом роботи в галузі проектування та забезпечення безпеки.

Перш ніж приступити до створення чогось нового, потрібно ретельно опрацювати процедури проектування і вирішити питання, пов’язані з кваліфікацією персоналу.

Більшості організацій нелегко проводити серйозні перетворення, які потрібні для проектування безпечного програмного забезпечення. Крім того, методи розробки, які позитивно позначаються на забезпеченні його захисту, поки недосконалі.

1 СТАН ПИТАННЯ. ПОСТАНОВКА ЗАДАЧІ

1.1 Актуальність проблеми

Багато вразливих місць з’являються внаслідок помилок, які ненавмисно виникають при проектуванні та розробці програмного забезпечення. Згідно з аналізом, виконаним фахівцями CERT/CC, поява більшості вразливих місць обумовлено загальними причинами (10 таких причин породжують близько 75% всіх вразливостей), причому джерелами більше 90% уразливих місць є відомі типи дефектів. Поняття «дефект» будемо трактувати ширше, відносячи до них все те, що обумовлює необхідність яких-небудь змін продукту (незалежно від того, чи пов’язано це з невідповідністю вимогам, невдалими конструкторськими рішеннями, недостатнім рівнем безпеки та зручності використання або з помилками кодування).

При використанні сьогоднішніх технологій розробки на кожну тисячу рядків нового або зміненого коду доводиться, як правило, від 1 до 7 дефектів. У типових сучасних системах, що містять мільйони рядків коду, є тисячі дефектів. Природно, таке програмне забезпечення рідко гарантує безпеку. Для створення дійсно безпечного програмного забезпечення організаціям необхідно скоротити на один-два порядки число дефектів у специфікаціях, помилок при проектуванні та реалізації.

Для вдосконалення технології проектування розроблено цілий ряд процесів і процесних моделей. Особливу увагу будемо приділяти тим з них, які дозволяють істотно зменшити кількість помилок при розробці специфікацій програмного забезпечення, його проектуванні та впровадженні, мінімізувати число вразливих місць.

1.2 Колективне проектування

Процес колективного проектування програмного забезпечення (Team Software Process, TSP) був визначений фахівцями інституту Software Engineering Institute (SEI), створеного при університеті Карнегі-Меллона. Даний

процес є використання при колективної розробки певного набору операцій. В даний час процес TSP використовується досить широко [5].

Проведене дослідження 20 проектів в 13 організаціях показало, що команди які застосовують його випускають програмне забезпечення, в якому є близько 0,06 дефектів проектування та реалізації на кожну тисячу рядків нового та зміненого коду. Терміни випуску цих продуктів у порівнянні з наміченими збільшуються в середньому на 6%. Такий результат досить непоганий, якщо врахувати, що більше половини програмних проектів завершуються невдачею, або на їх реалізацію йде мінімум удвічі більше часу, ніж планувалося [3].

Процес колективного проектування безпечного програмного забезпечення (TSP-Secure) доповнюється різними процедурами забезпечення безпеки протягом усього життєвого циклу розробки. Розробники проходять додаткову підготовку в області захисту, вивчаючи загальні причини виникнення вразливих місць, орієнтовані на безпеку методи проектування, відповідні методи реалізації (такі як контрольні відомості безпечного коду) і тестування. Процес TSP-Secure відносно новий, але провідні виробники вже активно використовують його для створення практично вільного від дефектів програмного забезпечення.

1.3 Структурна коректність

Формальні методи являють собою математично обґрунтовані підходи до створення програмного забезпечення. При цьому математичні моделі та формальна логіка використовуються для підтримки строгих програмних специфікацій, коректного проектування, кодування і контролю якістю. Один з таких способів – забезпечення структурної коректності (Correctness-by-Construction) – був запропонований компанією Praxis Critical Systems.[6] Він дозволяє за допомогою формальних методів перевіряти програмне забезпечення та своєчасно усувати в ньому дефекти на протязі його життєвого циклу.

Даний спосіб включає в себе визначення формальних нотацій для специфікацій систем і компонентів архітектури з урахуванням їх узгодженості і

коректності. Для безпечних систем виділяються категорії станів системи і операцій, які визначаються з урахуванням їх впливу на загальну безпеку. Кінцевою метою є створення архітектури, яка дозволяє звести до мінімуму число функцій, критично важливих для забезпечення захисту, та ізолювати їх. Це сприяє подальшому зниженню вартості і скорочення обсягу роботи (можливо, значно більш трудомісткою), пов’язаної з перевіркою правильності даних елементів.

1.4 «Стерильна кімната»

Проектування програмного забезпечення методом «стерильної кімнати» (він названий за аналогією з високоточним проектуванням стерильних кімнат, призначених для виробництва апаратури) являє собою теоретично обґрунтований, орієнтований на командну розробку процес створення та сертифікації коректних програмних систем зі статистичним контролем якості. [7-9] Даний процес охоплює весь життєвий цикл розробки, включаючи управління проектом в ході ітераційного проектування, визначення специфікацій функцій і архітектури, перевірку правильності функціоналу, а також статистичне тестування при сертифікації програм на придатність до використання. Ролі співробітників, які виконують відповідні проекти, розподіляються так: окремі колективи займаються визначенням специфікацій, безпосередньо розробкою і сертифікацією. Проектування методом «стерильної кімнати» передбачає статистичний контроль якості в ході розробки, який здійснюється шляхом послідовного відділення процедури проектування від процедури статистичного тестування.

Багато проектів, засновані на методі «стерильної кімнати», пов’язані з секретними відомостями і не фігурують в загальнодоступних звітах. Проте досвід показує, що загальна кількість помилок у даних випадках – приблизно 0,1 дефекту на 1 тис. рядків для програми, повністю відповідного принципам «стерильної кімнати», і 0,4 помилки про програму з частковою відповідністю. Більшість дефектів у програмному забезпеченні, побудованому з використанням методу «стерильної кімнати», відноситься до помилок

кодування. Проблеми, пов’язані зі специфікаціями і архітектурою, зустрічаються набагато рідше.

1.5 Процесні моделі

Процесні моделі містять визначення цілей і ключові атрибути конкретних процесів (наприклад, побудови системи безпеки), але не включають в себе практичних посібників з визначення та реалізації процесів. Серед найбільш відомих моделей вдосконалення процесів можна виділити модель Capability Maturity Model (CMM), що відображає залежність можливостей системи від її зрілості. Хоча про відмову від оцінки продукту або сертифікації системи не йдеться, методи CMM можна використовувати для загальної оцінки ефективності роботи організації (залежно від визначених у моделі критеріїв) та пошуку шляхів її підвищення. Практика свідчить, що застосування процесних моделей для вдосконалення програмного забезпечення дозволяє істотно скоротити кількість дефектів архітектури та реалізації в кінцевих продуктах. [10, 11]

Для підвищення рівня безпеки використовуються три різновиди CMM.

– CMM Integration (CMMI) допомагає організаціям удосконалювати процеси, пов’язані з проектуванням систем, розробкою програмного забезпечення, створенням інтегрованих продуктів і процесів, організацією взаємодії з постачальниками, управління процесами і проектами.

– Integrated CMM (iCMM) являє собою єдину модель, що узагальнює передовий досвід удосконалення процесів в масштабі всього підприємства, включаючи управління аутсорсингом і взаєминами з постачальниками. Широко застосовується Федеральним управлінням цивільної авіації США.

– Systems Security Engineering CMM (SSE-CMM) призначена для визначення вимог до реалізації компонентів захисту в програмних системах, які розробляються в інтересах IT-галузі. Модель SSE-CMM є стандартом ISO [ISO / IEC 21827:2002].

На сьогоднішній день вже більш ніж 4 тис. організацій скористалися перевагами моделі CMMI і її попередниці, програмної моделі CMM, при визначенні шляхів вдосконалення процесів та оцінки можливостей програмного забезпечення.

1.6 Технічні рішення

Деякі вразливі місця в системі безпеки виникають просто через недогляд. Це призводить до появи дефектів, які цілком можна усунути, якщо дотримуватися добре зарекомендували себе методів проектування. Поява низки інших вразливостей обумовлено конкретними особливостями моделювання, архітектури та проектування системи безпеки, у тому числі помилками при визначенні загроз, неадекватною аутентифікацією, невірної авторизацією, неправильним застосуванням шифрування, помилками при організації захисту даних і недостатньо ретельним плануванням поділу додатків. Для вирішення всіх цих проблем організаціям необхідно використовувати ефективні методи, що дозволяють вирішувати питання забезпечення безпеки безпосередньо.

Велика частина технічних методів вимагає досвіду роботи з системами забезпечення безпеки, особливо при визначенні специфікацій та проектуванні. У багатьох із застосовуваних для розробки безпечного програмного забезпечення технічних методів дуже мало практичних підтверджень ефективності.

1.7 Аналіз нормативної бази

Якість програмного забезпечення визначається в стандарті ISO/IEC 9126 Software engineering – Product quality is an international standard for the evaluation of software quality.

Розрізняються поняття внутрішньої якості, пов’язаного з характеристиками ПЗ самого по собі, без урахування його поведінки; зовнішньої якості, що характеризує ПЗ із точки зору його поведінки; і якості ПЗ при використанні у різних контекстах – тої якості, що відчувається користувачами при конкретних сценаріях роботи ПЗ. Для всіх цих аспектів

якості уведені метрики, що дозволяють оцінити їх. Крім того, для створення добротного ПЗ істотно якість технологічних процесів його розробки. Взаємини між цими аспектами якості за схемою, прийнятою ISO 9126. [15]

Загальні принципи забезпечення якості процесів виробництва у всіх галузях економіки регулюються набором стандартів ISO 9000. Найбільш важливі для розробки ПЗ стандарти в його складі наступні:

ДСТУ ISO 9000-2015 Системи управління якістю. Основні положення та словник [16]

ДСТУ ISO 9001-2015 Системи керування якістю – Вимоги. [16]

Визначає загальні правила забезпечення якості результатів у всіх процесах життєвого циклу.

Цей стандарт виділяє наступні процеси:

- Управління якістю;
- Управління ресурсами;
- Розвиток системи управління;
- Дослідження ринку;
- Проектування продуктів;
- Придбання;
- Виробництво;
- Надання послуг;
- Захист продуктів;
- Оцінка потреб замовників;
- Підтримка комунікацій із замовниками;
- Підтримка внутрішніх комунікацій;
- Управління документацією;
- Ведення записів про діяльність;
- Планування;
- Навчання персоналу;
- Внутрішні аудити;

– Оцінки управління;

– Моніторинг і виміри;

– Управління невідповідностями;

– Постійне вдосконалювання;

– Управління та розвиток системи в цілому.

Для кожного процесу потрібно мати плани розвитку процесу, що складаються як мінімум з наступних розділів:

– Проектування процесу;

– Документування процесу;

– Реалізація процесу;

– Підтримка процесу;

– Моніторинг процесу;

– Управління процесом;

– Удосконалення процесу.

Крім підтримки й розвитку системи процесів, націлених на задоволення потреб замовників і користувачів, ISO 9001 вимагає:

– Визначити, документувати та розвивати власну систему якості на основі вимірних показників;

– Використовувати цю систему якості як засіб управління процесами, націлюючи їх на більше задоволення потреб замовників, плануючи й постійно відслідковуючи якість результатів всіх видів діяльності, у тому числі й самого управління;

– Забезпечити використання якісних ресурсів, якісного (компетентного, професійного) персоналу, якісної інфраструктури і якісного оточення;

– Постійно контролювати дотримання вимог до якості на практиці, у всіх процесах проектування, виробництва, надання послуг і при придбаннях;

– Передбачити процес усунення дефектів, визначити й контролювати якість результатів цього процесу. [15]

Стандарти, що використовувалися раніше, ДСТУ ISO 9002-95 Системи якості. Модель забезпечення якості в процесі виробництва, монтажу та обслуговування і ISO 9003:1994 Quality systems - Model for quality assurance in final inspection and test в 2000 році були замінені відповідними ним частинами ISO 9001.

ДСТУ ISO 9004:2012 Системи управління якістю настанови щодо поліпшення діяльності [16].

ДСТУ ISO/IEC 90003:2006 Програмна інженерія. Настанови щодо застосування ISO 9001:2015 до програмного забезпечення. [16]

Цей стандарт конкретизує положення ISO 9001 для розробки програмних систем, з упором на забезпечення якості при процесі проектування. Він також визначає деякий набір технік і процедур, які рекомендується застосовувати для контролю й забезпечення якості розроблювальних програм. [15]

Стандарт ДСТУ ISO/IEC 9126-1:2013 Програмна інженерія. Якість продукту. Ч.1. Модель якості. пропонує використовувати для опису внутрішнього та зовнішнього якості ПЗ багаторівневу модель. На верхньому рівні виділено 6 основних характеристик якості ПО. Кожна характеристика описується за допомогою кількох вхідних у неї атрибутів. Для кожного атрибута визначається набір метрик, що дозволяють його оцінити. Множина характеристик і атрибутів якості згідно ISO 9126 показана на рис. 1.1.

Функціональність (functionality). [15,16] Здатність ПЗ в певних умовах вирішувати задачі, потрібні користувачам. Визначає, що саме робить ПЗ, які задачі воно вирішує:

– Функціональна придатність (suitability). Здатність вирішувати потрібний набір задач.

– Точність (accuracy). Здатність видавати потрібні результати.

– Здатність до взаємодії (interoperability). Здатність взаємодіяти з потрібним набором інших систем.

– Відповідність стандартам і правилам (compliance). Відповідність ПЗ наявних індустріальних стандартах, нормативним і законодавчим актам, іншим регулюючим нормам.

Рисунок 1.1 – Атрибути якості ПЗ за ISO 9126

– Захищеність (security). Здатність запобігати неавторизированому, тобто без вказівки особи, що намагається його здійснити, і недозволеному доступу до даних і програм.

Надійність (reliability). [15,16] Здатність ПЗ підтримувати визначену працездатність у заданих умовах:

– Зрілість, завершеність (maturity). Величина, зворотна частоті відмов ПЗ. Звичайно виміряється середнім часом роботи без збоїв і величиною, зворотною імовірності виникнення відмови за даний період часу.

– Стійкість до відмов (fault tolerance). Здатність підтримувати заданий рівень працездатності при відмовах і порушеннях правил взаємодії з оточенням.

– Здатність до відновлення (recoverability). Здатність відновлювати визначений рівень працездатності й цілісність даних після відмови, необхідні для цього час і ресурси

– Відповідність стандартам надійності (reliability compliance). Цей атрибут доданий в 2001 році.

Зручність використання (usability) або практичність. [15,16] Здатність ПЗ бути зручним у навчанні та використанні, а також привабливим для користувачів:

– Зрозумілість (understandability). Показник, зворотний до зусиль, які затрачаються користувачами на сприйняття основних понять ПЗ та усвідомлення їх застосовності для розв’язання своїх задач.

– Зручність навчання (learnability). Показник, зворотний зусиллям, затрачуваним користувачами на навчання роботі з ПЗ.

– Зручність роботи (operability). Показник, зворотний зусиллям, що вживається користувачами для розв’язання своїх задач за допомогою ПЗ.

– Привабливість (attractiveness). Здатність ПЗ бути привабливим для користувачів.

– Відповідність стандартам зручності використання (usability compliance).

Продуктивність (efficiency) або ефективність. [15,16] Здатність ПЗ при заданих умовах забезпечувати необхідну працездатність стосовно виділюваного для цього ресурсам. Можна визначити її і як відношення одержуваних за допомогою ПЗ результатів до затрачуваних на це ресурсів усіх типів:

– Часова ефективність (time behaviour). Здатність ПЗ видавати очікувані результати, а також забезпечувати передачу необхідного об’єму даних за відведений час.

– Ефективність використання ресурсів (resource utilisation). Здатність вирішувати потрібні задачі з використанням визначених об’ємів ресурсів визначених видів. Маються на увазі такі ресурси, як оперативна й довгострокова пам’ять, мережні з’єднання, пристрої вводу та виводу та ін.

– Відповідність стандартам продуктивності (efficiency compliance).

Зручність супроводу (maintainability). [15,16] Зручність проведення всіх видів діяльності, пов’язаних із супроводом програм:

– Аналізованість (analyzability) або зручність проведення аналізу. Зручність проведення аналізу помилок, дефектів і недоліків, а також зручність аналізу необхідності змін і їх можливих наслідків.

– Зручність внесення змін (changeability). Показник, зворотний трудозатратам на виконання необхідних змін.

– Стабільність (stability). Показник, зворотний ризику виникнення несподіваних ефектів при внесенні необхідних змін.

– Зручність перевірки (testability). Показник, зворотний трудозатратам на проведення тестування і інших видів перевірки того, що внесені зміни привели до потрібних результатів.

– Відповідність стандартам зручності супроводу (maintainability compliance).

Переносимість (portability). [15,16] Здатність ПЗ зберігати працездатність при перенесенні з одного оточення в інше, включаючи організаційні, апаратні й програмні аспекти оточення.

Іноді ця характеристика називається у нашій літературі мобільністю. Однак термін «мобільність» варто зарезервувати для перекладу «mobility» – здатності ПЗ й комп’ютерної системи в цілому зберігати працездатність при її фізичному переміщенні в просторі.

– Адаптованість (adaptability). Здатність ПЗ пристосовуватися різним оточенням без проведення для цього дій, крім заздалегідь передбачених.

– Зручність установки (installability). Здатність ПЗ бути встановленим або розгорнутим у визначеному оточенні.

– Здатність до співіснування (coexistence). Здатність ПЗ співіснувати з іншими програмами у загальному оточенні, ділячи з ними ресурси.

– Зручність заміни (replaceability) іншого ПЗ даним. Можливість застосування даного ПЗ замість інших програмних систем для вирішення тих же задач у певному оточенні.

– Відповідність стандартам переносимості (portability compliance).

Перераховані атрибути належать до внутрішньої та зовнішньої якості ПЗ згідно ISO 9126. Для опису якості ПЗ при використанні стандарт ISO 9126-4 пропонує інший, більш вузький набір характеристик. [15]

Ефективність (effectiveness). Здатність ПЗ надавати користувачам можливість вирішувати їхньої задачі з необхідною точністю при використанні в заданому контексті.

Продуктивність (productivity). Здатність ПЗ надавати користувачам визначені результати в рамках очікуваних витрат ресурсів.

Безпека (safety). Здатність ПЗ забезпечувати необхідно низький рівень ризику завдання втрат життю й здоров'ю людей, бізнесу, власності або навколишньому середовищу.

Задоволення користувачів (satisfaction). Здатність ПЗ приносити задоволення користувачам при використанні в заданому контексті.

Крім перерахованих характеристик і атрибутів якості, стандарт ISO 9126:2001 визначає набори метрик для оцінки кожного атрибута. Наведемо наступні приклади таких метрик.

Повнота реалізації функцій – відсоток реалізованих функцій по відношенню до перерахованого у вимогах. Використовується для виміру функціональної придатності.

Коректність реалізації функцій – правильність їх реалізації по відношенню до вимог. Використовується для виміру функціональної придатності.

Відношення числа виявлених дефектів до прогнозованого. Використовується для визначення зрілості.

Відношення числа проведених тестів до загального їх числа. Використовується для визначення зрілості.

Відношення числа доступних проектних документів до зазначеного в їх списку. Використовується для виміру зручності проведення аналізу.

Наочність і повнота документації. Використовується для оцінки зрозумілості.

Перераховані характеристики та атрибути якості ПЗ дозволяють систематично описувати вимоги до нього, визначаючи, які властивості ПЗ за даною характеристикою хочуть бачити зацікавлені сторони. [15]

Таким чином, вимоги повинні визначати наступне:

- Що ПЗ має робити, наприклад:
    - дозволяти клієнтові оформити замовлення й забезпечити їхню доставку;
    - забезпечувати контроль якості будівництва й відслідковувати проблемні місця;
    - підтримувати потрібні характеристики автоматизованого процесу виробництва, запобігаючи аварії й оптимальним способом використовуючи наявні ресурси.
- Наскільки воно має бути надійним, наприклад:
    - працювати 7 днів у тиждень і 24 години на добу;
    - допускається непрацездатність протягом не більше 3 годин у рік;
    - ніякі уведені користувачами дані при відмові не повинні губитися.
- Наскільки ним має бути зручно користуватися, наприклад:
    - покупець повинен, знаючи назву товару й маючи середні навички роботи в Інтернет, знаходити потрібний йому товар за не більш ніж 2 хв.;
    - інженер за фахом повинен протягом одного дня вміти розібратися в 80% функцій системи.
- Наскільки воно повинне бути ефективним, наприклад:
    - підтримувати обслуговування до 1000 запитів у секунду;

- час відгуку на запит при максимальному завантаженні не повинен перевищувати 3 с.;
- час реакції на зміну параметрів процесу виробництва не повинен перевищувати 0.1 с.;
- на обробку одного запиту не повинне витрачатися більше 1 Mb оперативної пам’яті.

– Наскільки зручним повинен бути його супровід, наприклад:
  - додавання в систему нового виду запитів не повинне вимагати більше 3 людино-днів;
  - додавання підтримки нового етапу процесу виробництва не повинне коштувати більше 100000 грн.

– Наскільки воно повинне бути стерпне, наприклад:
  - ПЗ повинне працювати на операційних системах Linux, Windows XP і MacOS X (та більш пізніших версіях);
  - ПЗ повинне працювати з документами у форматах MS Word і HTML;
  - ПЗ повинне зберігати файли звітів у форматах MS Word 20**, MS Excel 20**, HTML, RTF та у вигляді звичайного тексту;
  - ПЗ повинне сполучатися з існуючою системою запису даних про замовлення.

Наведені атрибути якості закріплені в стандартах, але це не означає, що вони цілком вичерпують поняття якості ПЗ. Так, у стандарті ISO 9126 відсутні характеристики, пов’язані з мобільністю ПЗ (mobility), тобто здатністю програми працювати при фізичних переміщеннях машини, на якій вона працює. Замість надійності багато дослідників воліють розглядати більш загальне поняття добротності (dependability), що описує здатність ПЗ підтримувати визначені показники якості за основними характеристиками (функціональності, продуктивності, зручності використання) із заданими ймовірностями виходу за їх рамки та визначеним максимальним збитком від можливих порушень. Крім того, активно досліджуються поняття зручності використання, безпеці й

захищеності ПЗ, – вони здаються більшості фахівців набагато більш складними, ніж це описується даним стандартом.

1.8 Принципи проектування безпечного програмного забезпечення

Звичайно, для створення безпечного програмного забезпечення недостатньо лише дотримання принципів, але вони здатні допомогти визначитися з методами розробки. Вісім принципів розробки, сформульовані ще в 1974 році [12], сьогодні аж ніяк не менш актуальні:

– економія механізмів – архітектура повинна бути якомога простішою та компактною;

– дії за замовчуванням, що забезпечують захист від збоїв – рішення про надання доступу необхідно приймати на основі призначення прав, а не їх виключення;

– повний контроль – надання доступу до будь-якого об’єкта має здійснюватися тільки після перевірки наявності відповідних прав;

– відкрита архітектура – її не слід тримати в таємниці;

– поділ привілеїв – механізм захисту, що вимагає двох ключів, більш гнучкий і стійкий в порівнянні з застосовують для отримання доступу один ключ. Тому там, де можливо, краще використовувати механізм з двома ключами;

– мінімізація привілеїв – будь-яка програма і будь-який користувач системи повинні мати мінімально необхідний для виконання поставлених перед ними завдань набір привілеїв;

– максимальне скорочення кількості загальних механізмів – механізмів, доступних або залежних більш ніж від одного користувача;

– прийнятність з психологічної точки зору – необхідно продумати особливості людино-машинного інтерфейсу, які забезпечать простоту використання програмного забезпечення, щоб споживачі, не замислюючись, коректно застосовували механізми захисту при виконанні своїх повсякденних завдань.

1.9 Керівництва розробників і контрольні відомості

Вибудовуючи свою діяльність на основі вказаних основних принципів, різні організації (від Microsoft до SAP) взяли на озброєння декілька універсальних посібників, у підготовці яких брала участь і група OWASP. В процесі створення безпечного програмного забезпечення розробники повинні:

- забезпечити захист від збоїв системи безпеки;
- перевірити правильність вхідної і вихідної інформації;
- організувати глибоко ешелоновану захист;
- застосовувати і повторно використовувати надійні компоненти;
- уникати ускладнення системи;
- використовувати відокремлення для розподілу прав доступу;
- уникати так званої «безпеки за рахунок заплутаності»;
- надавати тільки абсолютно необхідні привілеї;
- використовувати шифрування при організації всіх видів зв’язку;
- не застосовувати алгоритми шифрування власної розробки;
- ніколи не пересилати паролі у відкритому вигляді;
- забезпечити безпеку конфігурації, що налаштовується за замовчуванням;
- гарантувати безпеку доставки і установки програмного забезпечення;
- не створювати «потайних ходів».

Всі ці правила обов’язково повинні дотримуватися при розробці специфікацій програмного забезпечення, в процесах проектування, впровадження, тестування, доставки і установки.

1.10 Програмні продукти сторонніх розробників

Необхідно розглянути і питання використання готових комерційних продуктів або програм з відкритим кодом. Жодне з серйозних досліджень ще не довело, що в якомусь із цих видів програмного забезпечення присутній менше або більше вразливих місць. Однак дуже часто покупці і адміністратори не обізнані про те, що в наявних у них продуктах використано програмне

забезпечення незалежних розробників. Іншими словами, вони не знають про пролом у системі безпеки, хоча незалежний виробник вже опублікував відповідне попередження або випустив оновлення. Крім того, виробники можуть покладатися на якість продуктів незалежних розробників, але останні не гарантують, що всі процедури в них реалізовані належним чином.

Розробникам програмного забезпечення потрібно вимагати від незалежних виробників дотримання процесів і методів. В іншому випадку розробникам доведеться самостійно перевіряти якість вбудованих в їх продукти компонентів незалежних виробників. І вже, принаймні, незалежні виробники повинні повідомляти, які готові комерційні продукти і програми з відкритим кодом використовуються в їх розробках.

Разом зі своїми продуктами розробники повинні поставляти керівництва по використанню щодо забезпечення безпеки. У них потрібно вказувати всі допущення, рівень безпеки використовуваних засобів, а також давати рекомендації з налаштування конфігурації програмного забезпечення.

1.11 Тестування програмного забезпечення

Тестування програмного забезпечення – техніка контролю якості, що перевіряє відповідність між реальною та очікуваною поведінкою програми завдяки кінцевому набору тестів, що обираються певним чином.

Якість не є абсолютною, це суб’єктивне поняття. Тому тестування як процес, своєчасного виявлення помилок та дефектів, не може повністю забезпечити коректність програмного забезпечення. Воно тільки порівнює стан і поведінку продукту зі специфікацією. При цьому треба розрізняти тестування програмного забезпечення і забезпечення якості програмного забезпечення, до якого належать усі складові ділового процесу, а не тільки тестування [17].

1.11.1 Основні поняття та визначення

Тестування – це одна з технік контролю якості, що включає в себе:

- Планування робіт (Test Management);
- Проектування тестів (Test Design);

– Виконання тестування (Test Execution);
– Аналіз отриманих результатів (Test Analysis).

Верифікація (Verification) – це процес оцінки системи або її компонентів з метою визначити чи задовольняють результати поточного етапу розробки умовам, сформованим на початку цього етапу. Тобто чи виконуються цілі, терміни, завдання з розробки проекту, визначені на початку поточної фази.

Валідація (Validation) – це визначення відповідності розроблюваного програмного забезпечення між очікуваннями і потребами користувача, вимогам до системи.

План тестування (Test Plan) – це документ, що описує весь обсяг робіт з тестування, починаючи з опису об’єкта, стратегії, розкладу, критеріїв початку і закінчення тестування, до необхідного в процесі роботи обладнання, спеціальних знань, а також оцінки ризиків з варіантами їх вирішення.

Тест дизайн (Test Design) – це етап процесу тестування програмного забезпечення, на якому проектуються і створюються тестові випадки (тест кейси), відповідно до визначених раніше критеріями якості та цілями тестування.

Тестовий випадок (Test Case) – це документ, що описує сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації тестованої функції або її частини.

Баг/Дефект Репорт (Bug Report) – це документ, що описує ситуацію або послідовність дій (Steps), що призвела до некоректної роботи об’єкта тестування (Misbehavior), із зазначенням причин та очікуваного результату (Expected Result).

Тестове покриття (Test Coverage) – це одна з метрик оцінки якості тестування, що представляє із себе щільність покриття тестами вимог або коду, що виконується.

Деталізація Тест Кейсів (Test Case Specification) – це рівень деталізації опису тестових кроків і необхідного результату, при якому забезпечується розумне співвідношення часу проходження до тестового покриття.

Час Проходження Тест Кейса (Test Case Pass Time) – це час від початку проходження кроків тест кейса до отримання результату тесту [17].

1.11.2 Види тестування програмного забезпечення

Існує кілька ознак, за якими прийнято робити класифікацію видів тестування [17]. Зазвичай виділяють такі:

За ступенем автоматизації:

- Ручне тестування (manual testing);
- Автоматизоване тестування (automated testing);
- Напівавтоматизованих тестування (semiautomated testing).

За ступенем підготовленості до тестування:

- Тестування по документації (formal testing);
- Тестування ad hoc або інтуїтивне тестування (ad hoc testing) – тестування без тест плану та документації, що базується на методиці передбачення помилки та власному досвіді тестера.

За знанням системи:

- Тестування чорного ящика (black box);
- Тестування білого ящика (white box);
- Тестування сірого ящика (grey box).

За ступенем ізольованості компонентів:

- Компонентне (модульне) тестування (component/unit testing);
- Інтеграційне тестування (integration testing);
- Системне тестування (system/end-to-end testing).

За часом проведення тестування:

- Альфа-тестування (alpha testing):
  - Тестування при прийманні або Димове тестування(smoke testing);
  - Тестування нової функціональності (new feature testing);
  - Регресивне тестування (regression testing);
  - Тестування при здачі (acceptance testing);
- Бета-тестування (beta testing).

За об’єктом тестування:

- Функціональне тестування (functional testing);
- Тестування продуктивності (performance testing):
  - Навантажувальне тестування (load testing);
  - Стрес-тестування (stress testing);
  - Тестування стабільності (stability/endurance/soak testing);
- Тестування зручності використання або Юзабіліті-тестування (usability testing);
- Тестування інтерфейсу користувача (UI testing);
- Тестування безпеки (security testing);
- Тестування локалізації (localization testing);
- Тестування сумісності (compatibility testing).

За ознакою позитивності сценаріїв:

- Позитивне тестування (positive testing);
- Негативне тестування (negative testing).

1.11.3 Рівні тестування

- Модульне тестування тестує мінімальний компонент програми, або модуля. Кожний модуль тестується для перевірки правильності його реалізації.
- Інтеграційне тестування виявляє дефекти в інтерфейсах та у взаємодії між компонентами (модулями).
- Системне тестування тестує інтегровану систему для перевірки відповідності всім вимогам.
- Системне інтеграційне тестування перевіряє, чи система інтегрується в будь-яку зовнішню систему (або системи) відповідно до системних вимог.
- Приймальне тестування може проводитись кінцевим користувачем, замовником, або клієнтом для перевірки, чи може продукт бути прийнятий до використання.

- альфа-тестування – це симульоване або реальне операційне тестування потенційними користувачами/замовником або командою тестувальників на боці розробника.
- бета-тестування йде після альфа-тестування. Версії програмного забезпечення, відомі як бета-версії, надаються у користування обмеженій кількості людей поза компанією для того, щоб упевнитись, що програма не містить великої кількості помилок [17].

1.11.4 Методи випробувань (тестування)

Існує три методи тестування програмного забезпечення: тестування «білий ящик», «сірий ящик» та «чорний ящик».

Тестування «білий ящик» – тестування, при якому тестувальник (фахівець з тестування програмного забезпечення) має доступ до коду. Його ще називають тестуванням скляного ящика або тестуванням прозорого ящика.

Крім того, що тестувальник може переглядати код, він ще й сам може писати код, який використовує бібліотеки існуючого програмного продукту. З його допомогою часто програмісти самі перевіряють свій продукт.

Тестування «чорний ящик» – тестування, при якому тестувальник має доступ до програми тільки через інтерфейс.

Тестувальник очима користувача дивиться на програму, не маючи при цьому доступу до коду. Але так як він все-таки тестировщик, то перевіряє програму не як користувач, а за допомогою своїх стратегій і методів: або вручну, або за допомогою інструментів тестування.

Перевага такого виду тестування в тому, що воно не вимагає знання мов програмування.

Мета цього способу тестування в тому, щоб перевірити розбіжність поведінки програми з документацією.

Тестування «сірий ящик» – воно знаходиться в прикордонному стані між білим ящиком і чорним, тому його і називають сірим. Поєднання відбувається таким чином: зовні на продукт дивимося як на чорний ящик, але вибір тестів засновуємо на знанні внутрішнього устрою програми, знанні її коду.

Цей метод часто використовується для тестування веб-додатків.

1.12 Висновок

Завдання, поставлені в даному дипломній роботі, на сьогоднішній день є актуальними в силу різноманіття вразливостей та дефектів у програмному забезпеченні яке розроблюється. Розглянуті в першому розділі дипломній роботі основні підходи до проектування, розробки та впровадження програмного забезпечення мають свої переваги та недоліки. Для вирішення проблем, пов’язаних з розробкою безпечного (захищеного) програмного забезпечення, необхідно вирішити ряд основних проблем зв’язаних з організаційною структурою підприємств галузі інформаційних технологій, виробничих процесів, програмних рішень які використовуються при проектуванні, розробці та впровадженні програмного забезпечення.

РОЗДІЛ 2. СПЕЦІАЛЬНИЙ РОЗДІЛ

2.1 Моделювання загроз

Моделювання загроз являє собою методологію аналізу захисту, яку можна використовувати при визначенні ризиків та прийнятті рішень при побудові архітектури, кодуванні і тестуванні. Дана методологія застосовується переважно на початкових стадіях реалізації проекту (до яких відносяться розробка специфікацій, архітектурних уявлень, діаграм потоків даних, функціональних діаграм тощо), але її можна використовувати і при написанні детально опрацьованого технічного завдання, а також коду. Кінцева мета полягає в усуненні потенційних загроз додатком, здатних завдати найбільшої шкоди.

В цілому моделювання загроз передбачає декомпозицію програми та визначення його основних властивостей з подальшим виявленням і класифікацією загроз, спрямованих проти кожної властивості або компонента. Загрози класифікуються в залежності від ступеня ризику. В результаті формуються стратегії ліквідації загроз на етапах проектування, кодування і тестування.

Щоб отримати реальні шанси на успіх, на всіх рівнях управління на чільне місце слід поставити завдання створення безпечного програмного забезпечення. Це означає, що розробникам повинні бути доступні всі необхідні ресурси, а в разі потреби вони можуть вдатися до послуг зовнішніх фахівців. Необхідно, щоб корпоративна культура забезпечувала навчання, вмотивованим, наполегливим, дисциплінованим, що володіють необхідним рівнем кваліфікації і заслуговують довіри персоналу виконання узгодженого плану і оцінку результатів.

При організації технічної підтримки, виправленні помилок і випуск оновлень використання оптимальних методів і процесів відіграє не менш важливу роль, ніж при розробці нового програмного забезпечення. Особливу увагу слід приділяти процесам управління конфігурацією: зміни у вже існуючі

програми слід вносити тільки після визначення чіткої процедури таких змін і настройки конфігурації.

2.2 Елементи моделі загроз експлуатаційної безпеки ПЗ

Аналіз загроз експлуатаційної безпеки ПЗ дозволяє, розділити їх на два типи: випадкові і навмисні, причому останні поділяються на активні і пасивні.

Активні загрози спрямовані на зміну технологічно зумовлених алгоритмів, програм функціональних перетворень або інформації, над якою ці перетворення здійснюються.

Пасивні загрози орієнтовані на порушення безпеки інформаційних технологій без реалізації таких модифікацій.

Загрози, що носять випадковий характер і пов’язані з відмовами, збоями апаратури, помилками операторів і т.п. припускають порушення заданого власником інформації алгоритму, програми її обробки або спотворення змісту цієї інформації. Суб’єктивний фактор появи таких загроз обумовлений обмеженою надійністю роботи людини і проявляється у вигляді помилок (дефектів) у виконанні операцій формалізації алгоритму функціональних перетворень чи опису алгоритму на деякій мові, зрозумілій обчислювальної системі.

Загрози, що носять зловмисний характер викликані, як правило, навмисним бажанням суб’єкта здійснити несанкціоновані зміни з метою порушення коректного виконання перетворень, достовірності та цілісності даних, яке проявляється в спотвореннях їх змісту або структури, а також з метою порушення функціонування технічних засобів в процесі реалізації функціональних перетворень і зміни конструктиву обчислювальних систем.[13]

2.3 Загрози несанкціонованого доступу до інформації

Загрози знищення, розкрадання апаратних засобів автоматизованої системи носіїв інформації шляхом фізичного доступу до її елементів.

1 Крадіжка ПЕОМ;

2 Крадіжка носіїв інформації;

3 Крадіжка ключів та атрибутів доступу;

4 Крадіжки, модифікації, знищення інформації;

5 Вихід з ладу вузлів ПЕОМ, каналів зв’язку;

6 Несанкціоноване відключення засобів захисту.

Загрози розкрадання, несанкціонованої модифікації або блокування інформації за рахунок несанкціонованого доступу (НСД) із застосуванням програмно-апаратних і програмних засобів.

1 Дії шкідливих програм (вірусів);

2 Недекларовані можливості системного ПЗ та ПЗ для обробки персональних даних;

3 Установка ПЗ не пов’язаного з виконанням службових обов’язків.

Загрози не навмисних дій користувачів і порушень безпеки функціонування АС і систем захисту в її складі через збої в програмному забезпеченні, а також від загроз техногенного (збоїв апаратури через ненадійність елементів, збоїв електроживлення) і стихійного (ударів блискавок, пожеж, повеней і т.п.) характеру.

1 Втрата ключів та атрибутів доступу;

2 Ненавмисна модифікація (знищення) інформації співробітниками;

3 Ненавмисне відключення засобів захисту;

4 Вихід з ладу апаратно-програмних засобів;

5 Збій системи електропостачання;

6 Стихійне лихо.

Загрози навмисних дій внутрішніх порушників.

1 Доступ до інформації, модифікація, знищення осіб не допущених до її обробці;

2 Розголошення інформації, модифікація, знищення співробітниками допущеними до її обробці.

Загрози несанкціонованого доступу по каналах зв’язку.

1 Загроза «Аналіз мережевого трафіку» з перехопленням передається з АС прийнятої із зовнішніх мереж інформації:

1.1 Перехоплення за межами контрольованої зони;

1.2 Перехоплення в межах контрольованої зони зовнішніми порушниками;

1.3 Перехоплення в межах контрольованої зони внутрішніми порушниками.

2 Загрози сканування, спрямовані на виявлення типу або типів використовуваних операційних систем, мережевих адрес робочих станцій АС, топології мережі, відкритих портів і служб, відкритих з'єднань.

3 Загрози виявлення паролів по мережі.

4 Загрози нав'язування хибного маршруту мережі.

5 Загрози підміни довіреного об'єкта в мережі.

6 Загрози впровадження помилкового об'єкта як в АС, так і в зовнішніх мережах.

7 Загрози типу «Відмова в обслуговуванні».

8 Загрози віддаленого запуску додатків.

9 Загрози впровадження по мережі шкідливих програм.

2.4 Управління персоналом

Вплив загрози антропогенного характеру необхідно зменшувати за рахунок проведення комплексу заходів з менеджменту, а точніше розробки нових підходів по управлінню персоналом підприємства: проведення підготовки з питань захисту інформації, розробки посадових інструкцій, переосмислення бізнес-логіки процесів проектування, розробки і впровадження програмних продуктів.

В умовах переходу до ринкової економіки управління персоналом має бути системним і завершеним на основі комплексного вирішення кадрових проблем, використання нових і вдосконалення існуючих форм і методів кадрової роботи.

Комплексний підхід до управління персоналом передбачає врахування організаційних, економічних, соціальних, психологічних, правових, технічних, педагогічних та інших аспектів управління у їх сукупності і взаємозв'язку при визначальній ролі соціально-економічних чинників.

Системний підхід відбиває взаємозв'язки між окремими аспектами управління персоналом і виражається в розробці кінцевої мети, визначенні шляхів її досягнення, створенні відповідного механізму управління, який забезпечує комплексне планування, організацію та стимулювання системи роботи з персоналом на підприємстві.

Варто виділяти три аспекти поняття «управління персоналом»: функціональний (змістовний), організаційний та освітній (як навчальна та наукова дисципліна). [1]

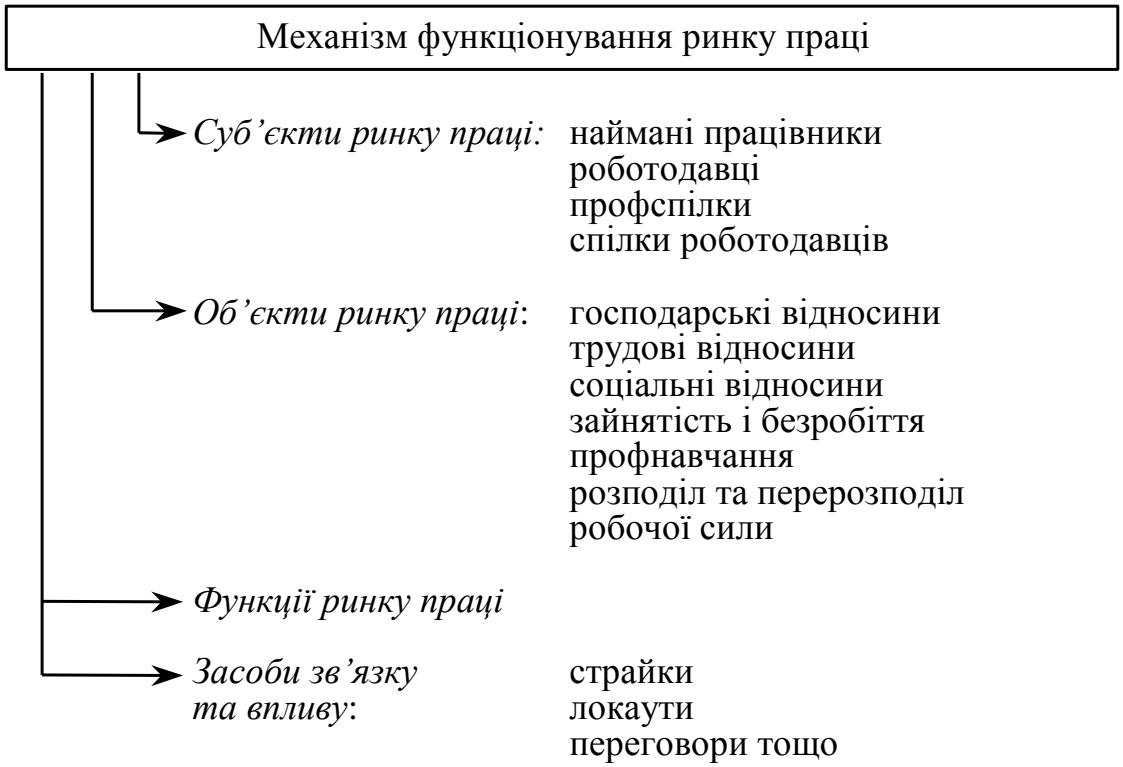

Рисунок 2.1 – Механізм функціонування ринку праці

Наймані працівники – це працівники, які виконують роботу за певну заробітну плату, обумовлену трудовим договором (контрактом). Це поняття асоціюється з поняттями кадри та персонал, які співвідносяться з іншими поняттями таким чином:

Трудові ресурси і робоча сила виступають, зазвичай, об'єктами державного і виробничо-територіального управління, а кадри і персонал – галузевого і фірмового управління людськими ресурсами.

Під кадрами розуміють штатних (постійних) кваліфікованих працівників, які пройшли попередню професійну підготовку, володіють трудовими

навичками, досвідом роботи, спеціальними знаннями в обраній сфері діяльності і перебувають у трудових відносинах з керівництвом фірм.

Під персоналом розуміють весь особовий склад підприємства, всіх постійних і тимчасових працівників, представників кваліфікованої і некваліфікованої праці.[1]

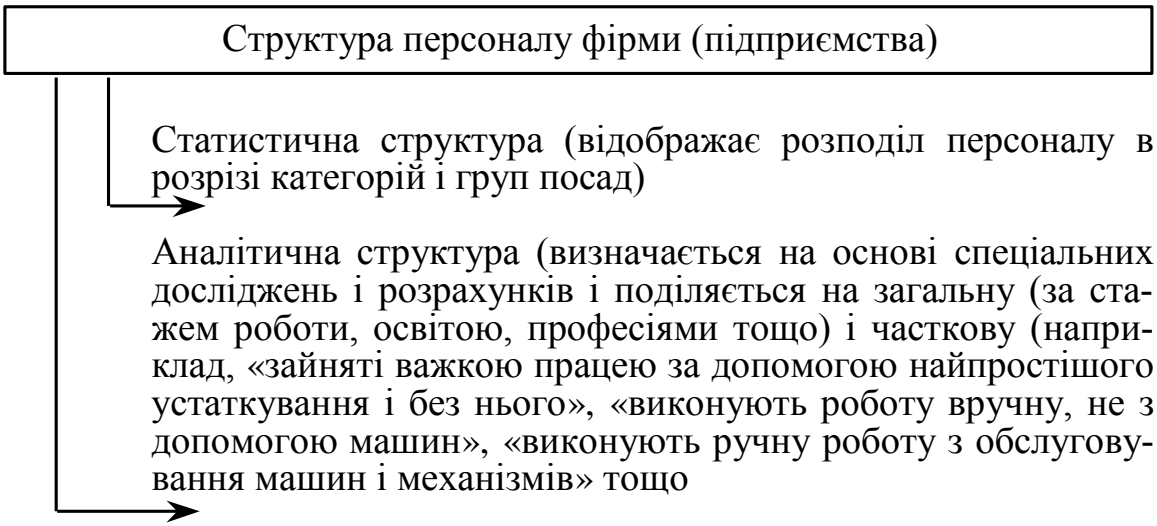

Рисунок 2.2 – Структура персоналу підприємства

Критерієм оптимальності структури персоналу є відповідність між:

– чисельністю працівників різних посадових груп;

– обсягами робіт (у нормогодинах), які виконуються кожною посадовою групою.

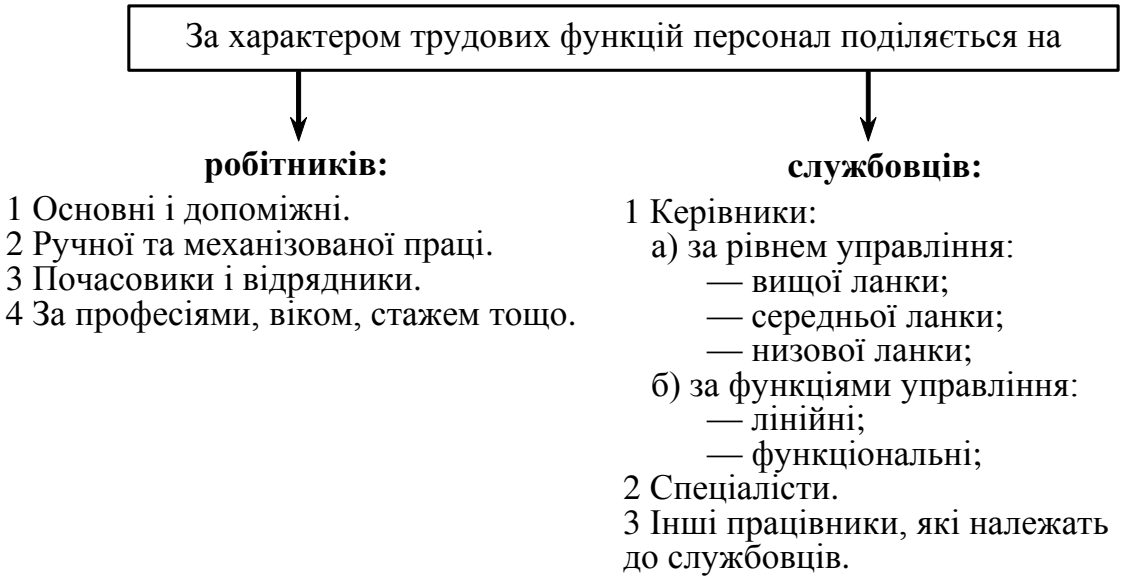

Рисунок 2.3 – Трудові функції персоналу

Чисельність персоналу визначається характером, масштабами, складністю, трудомісткістю виробничих процесів, ступенем їх механізації, автоматизації, комп’ютеризації.[1]

Показники чисельності персоналу:

– нормативна чисельність;

– облікова (фактична) чисельність:

а) постійні;

б) тимчасові;

в) сезонні;

– середньооблікова чисельність;

– середньооблікова чисельність всього персоналу в еквіваленті повної зайнятості;

– явочна чисельність;

– чисельність працівників, які не перебувають в обліковому складі.

Важливу роль в процесі розробки програмних продуктів є правильно спланований необхідний час на виконання спланованого обсягу робіт по проектуванню інтерфейсів, написанню програмного коду, проведення тестування додатків.

Якщо в процесі визначення термінів робіт було допущено помилку, це може призвести до збільшення навантаження на працівників, що призведе до виникнення у програмному забезпеченні яке розробляється дефектів та помилок.

2.4.1 Три сфери управління персоналом

Управління працею (вдосконалення виробничих процесів, розподіл та кооперація праці, проектування методів праці та організації робочих місць, контроль за трудовою дисципліною, поліпшення умов праці та її охорони, організація оплати праці і стимулювання).

Управління кадрами (комплектування і підготовка кадрів, оцінка і розстановка, організація професійної адаптації, мотивація і виховання працівників, зниження плинності кадрів тощо).[1]

Управління соціально-демографічними процесами (допомога сім’ям, будівництво й утримання об’єктів соціально-культурного призначення, поліпшення методичного обслуговування, організація дозвілля, розвиток підсобних господарств, підвищення загальноосвітнього і культурного рівня працівників).

2.4.2 Система лінійного та функціонального управління персоналом

1 Підсистема планування та маркетингу персоналу (розробка кадрової політики, розробка стратегії управління персоналом, аналіз кадрового потенціалу, аналіз ринку праці, організація кадрового планування, планування і прогнозування потреби в персоналі, організація реклами, підтримання взаємозв’язку із зовнішніми джерелами, які забезпечують організацію кадрами).

2 Підсистема найму та обліку персоналу (організація найму персоналу, організація співбесід, оцінки, добору і прийому персоналу, облік прийому, переміщень, заохочень і звільнень персоналу, управління зайнятістю персоналу, діловодство).

3 Підсистема трудових відносин (аналіз та регулювання групових і особистих відносин, аналіз і регулювання відносин керівництва, управління виробничими конфліктами та стресами, соціально-психологічна діагностика, дотримання етичних норм і взаємовідносин, управління взаємодії з профспілками, розгляд скарг і претензій, управління стабільністю трудового колективу, аналіз плинності кадрів, планування виходу на пенсію).

4 Підсистема використання персоналу (розстановка персоналу, розробка посадових інструкцій, регламентація роботи, контроль за трудовою дисципліною).

5 Підсистема управління умовами праці (дотримання вимог психофізіології праці, дотримання вимог ергономіки праці, дотримання вимог технічної естетики, охорона праці і техніки безпеки, охорона навколишнього середовища, фізична охорона організації та окремих посадових осіб).

6 Підсистема розвитку персоналу (професійна орієнтація персоналу, професійна підготовка персоналу, перепідготовка і підвищення кваліфікації персоналу, введення в посаду, адаптація нових працівників, оцінка кандидатів на вакантну посаду, поточна періодична оцінка кадрів, організація раціоналізації і винахідництва, реалізація ділової кар'єри і службово-професійного просування, організація роботи з кадровим резервом).

7 Підсистема мотивації поведінки персоналу (управління мотивацією поведінки персоналу, тарифікація трудового процесу, розробка систем оплати праці, розробка форм участі персоналу в розподілі прибутку і капіталу, розробка форм морального заохочення, організація нормативно-методичного забезпечення системи управління персоналом).

8 Підсистема соціального розвитку (організація харчування, управління житлово-побутовим обслуговуванням, розвиток культури і фізичного виховання, забезпечення охорони здоров'я і відпочинку, забезпечення дитячими установами, управління соціальними конфліктами і стресами, організація соціального страхування).

9 Підсистема розвитку оргструктур управління (аналіз діючої оргструктури управління, проектування нової оргструктури управління, розроблення штатного розкладу, формування нової оргструктури управління, розробка і реалізація рекомендацій щодо розвитку стилю і методів управління).

10 Підсистема правового забезпечення (дотримання трудового законодавства в трудових відносинах, узгодження розпорядчих документів з управління персоналом, проведення консультацій з правових питань).

11 Підсистема інформаційного забезпечення (статистичний облік персоналу, інформаційне і технічне забезпечення управління персоналом, забезпечення персоналу науково-технічною інформацією).

12 Підсистема організації праці (нормування праці, організація та обслуговування робочих місць).[1]

2.4.3 Аналіз роботи працівника підприємства

Аналіз роботи (Job analysis) може бути визначений як одержання інформації про неї. Аналіз відповідає на такі запитання: як називається робота, де потрібно виконувати її, якою є мета роботи, хто відповідає за працівника, за що відповідає працівник, з ким працівник має справу під час виконання роботи, в чому полягають основні завдання роботи, як ці завдання виконуються, які робочі стандарти (норми) потрібні, які вміння, знання, досвід потрібні для того, щоб відповідати цим стандартам, які зусилля потрібні для виконання цієї роботи, чи потребує ця робота фізичних і/або розумових зусиль, які завдання є простими, які – складними?[1]

До методів аналізу робіт відносять:

– фотографування робочого часу, в ході якого визначаються та реєструються в часі виконувані працівником завдання та дії, і за результатами якого можуть бути досить точно оцінені ступінь доцільності та ранг значущості окремих трудових дій;

– співбесіда з працівниками або їхніми безпосередніми керівниками;

– можливе також анкетне опитування, коли працівники заповнюють стандартний опитувальний лист або дають у вільній формі письмовий опис змісту виконуваної ними роботи.

Ці методи, хоча й дають реальну можливість враховувати думку безпосереднього виконавця роботи, менш точні, ніж безпосереднє спостереження, оскільки на оцінку змісту роботи може вплинути суб’єктивна оцінка працівника, стереотипність його уявлень про трудовий процес.

Описуючи виконання обов’язків, необхідно вказати основні обов’язки та витрати часу у відсотках на виконання кожного обов’язку протягом типового робочого дня.[1]

Інформація, отримана в результаті аналізу робіт, повинна бути використана для опису роботи (Job description), який включає:

– відомості про посаду працівника;

– посаду начальника працівника;

– посади підлеглих працівників;

– мету роботи;

– основні завдання роботи.

Опис роботи служить для: розробки опису потрібної людини; написання об’яви про наймання на роботу; розробки форми заяви; попереднього розгляду претендентів на роботу; оцінки кандидатів під час співбесіди; підготовки нових працівників до нової посади; оцінки працівників; порівняння різних робіт (за умовами та зарплатою).

2.4.4 Набір персоналу

У специфікації особи (Person specification) детально описується тип потрібної людини, зокрема вміння, знання та досвід. У практиці відомий «план семи точок», [розроблений Алеком Роджером] як основа для специфікації особи: фізичний стан, досягнення, загальний інтелект, здібності, інтереси, характер, інші умови.

Існує два джерела набору персоналу: зовнішній та внутрішній.

Внутрішні джерела: вивішування оголошень та подання заяв, меморандуми для керівників, публікації у внутрішніх виданнях.

Зовнішні джерела: реклама, агентства з працевлаштування, залучення персоналу співробітниками, самостійні візити та листи, набір випускників вузів.[1]

Переваги внутрішніх джерел: фірма краще знає сильні та слабкі якості кандидатів, кандидат краще знає компанію, можлива мотивація працівників за рахунок підвищення по службі, збільшується віддача інвестицій у людські ресурси.

Недоліки внутрішніх джерел: працівники можуть підвищуватись до рівня, на якому вони вже не відповідають вимогам; боротьба за кар’єру може створити поганий психологічний клімат; можуть виникнути проблеми з новими ідеями та впровадженнями.

Переваги зовнішніх джерел: більша вірогідність появи нових ідей та перспектив, буває дешевше набирати кваліфікованих працівників «ззовні».

Недоліки зовнішніх джерел: складніше залучати, контактувати та оцінювати кандидатів; великий період адаптації нових працівників; можливі моральні проблеми, оскільки часто «свої» працівники відчувають себе достатньо кваліфікованими для виконання такої роботи.

2.5 Типова організаційна структура підприємства з розробки програмного забезпечення

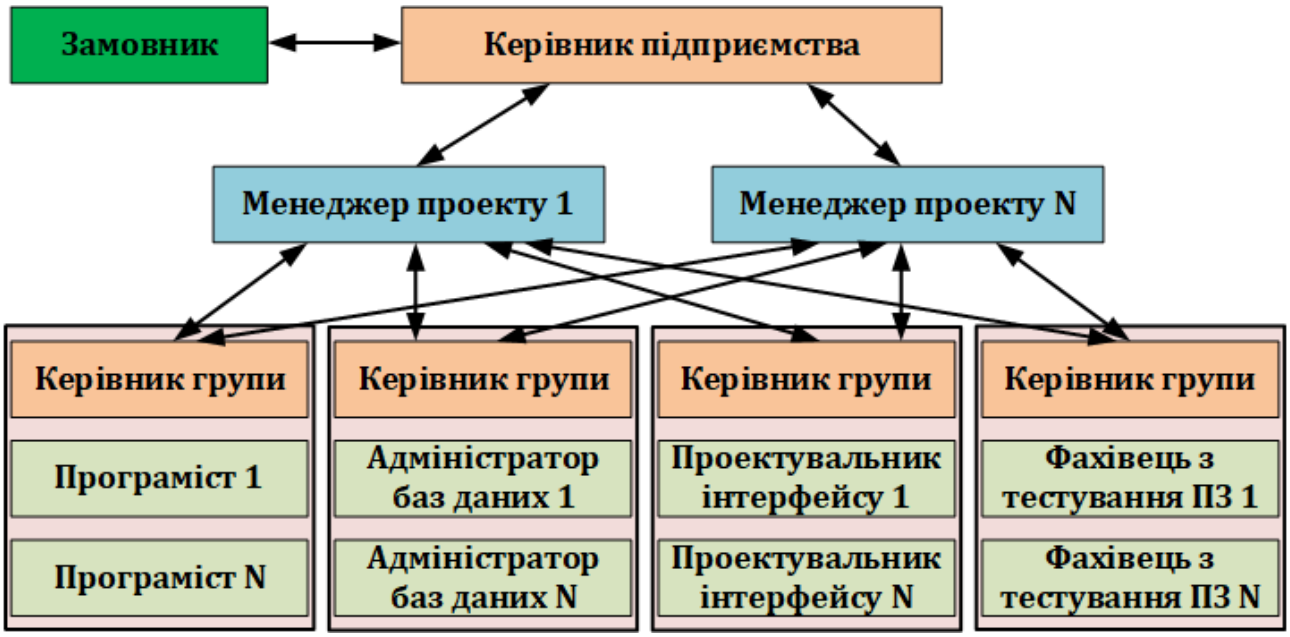

Рисунок 2.4 – Структура підприємства

Типова організаційна структура підприємства з розробки програмного забезпечення має наступний вид (рис 2.4):

1 Керівник підприємства – відповідає за діяльність всіх відділів підприємства, контролює та корегує бізнес процеси.

2 Менеджер проекту – відповідає за проектування, розробку специфікацій, розподіл роботи по проекту яким керує.

3 Керівник групи – відповідає за виконання поставлених задач у встановлені строки.

4 Програміст – виконує поставлені задачі по написанню програмного коду.

5 Адміністратор баз даних (програміст) – виконує нормування даних, розробляє логічні і фізичні моделі баз даних, виконує написання запитів до баз даних.

6 Проектувальник інтерфейсу (програміст) – розробляє інтерфейс програмного забезпечення відповідно до специфікації продукту.

7 Фахівець з тестування ПЗ (фахівець з якості, програміст-тестувальник) – виконує функції з тестування програмного забезпечення, виявляє дефекти та помилки в ПЗ, надає інформацію програмістам, адміністраторам БД, проектувальникам інтерфейсу для виправлення дефектів і помилок виявлених під час тестування.

2.6 Основні принципи забезпечення безпеки програмного забезпечення

В якості об’єкта забезпечення технологічної та експлуатаційної безпеки ПЗ розглядається вся сукупність його компонентів в рамках конкретної системи.

Сукупність заходів щодо забезпечення технологічної та експлуатаційної безпеки компонентів ПЗ повинна носити конфіденційний характер. Необхідно забезпечити постійний, комплексний та дієвий контроль за діяльністю розробників і користувачів компонентів. Крім загальних принципів, звичайно необхідно конкретизувати принципи забезпечення безпеки ПЗ на кожному етапі його життєвого циклу.

Принципи забезпечення технологічної безпеки при обґрунтуванні, плануванні робіт і проектному аналізі ПЗ

Принципи забезпечення безпеки ПЗ на даному етапі включають принципи:

– Комплексності забезпечення безпеки ПЗ, яка передбачає розгляд проблеми безпеки інформаційно-обчислювальних процесів з урахуванням всіх структур системи, можливих каналів витоку інформації і несанкціонованого доступу до неї, часу та умов їх виникнення, комплексного застосування організаційних і технічних заходів.

– Плановане застосування засобів безпеки програм, що передбачає перенесення акценту на спільне системне проектування ПЗ і засобів її безпеки, планування їх використання у передбачуваних умовах експлуатації.

– Обґрунтованості засобів забезпечення безпеки ПЗ, що полягає в глибокому науково-обґрунтованому підході до прийняття проектних рішень по оцінці ступеня безпеки, прогнозування загроз безпеки, всебічної апріорної оцінки показників засобів захисту.

– Достатності безпеки програм, що відбиває необхідність пошуку найбільш ефективних і надійних заходів безпеки при одночасній мінімізації їх вартості.

– Гнучкості управління захистом програм, що вимагає від системи контролю та управління забезпеченням інформаційної безпеки ПЗ здатності до діагностування, випереджальної нейтралізації, оперативному та ефективному усуненню виникаючих загроз в умовах різких змін обстановки інформаційної боротьби.

– Завчасності розробки засобів забезпечення безпеки і контролю виробництва ПЗ, що полягає в попереджувальному характері заходів забезпечення технологічної безпеки робіт в інтересах недопущення зниження ефективності системи безпеки процесу створення ПЗ.

– Документованої технології створення програм, що припускає розробку пакета нормативно-технічних документів з контролю програмних засобів на наявність умисних дефектів.

Принципи досягнення технологічної безпеки ПЗ в процесі його розробки

Принципи забезпечення безпеки ПО на даному етапі включають принципи:

– Регламентації технологічних етапів розробки ПЗ, що включає впорядковані фази проміжного контролю, специфікацію програмних модулів і стандартизацію функцій і формату представлення даних.

– Автоматизації засобів контролю керуючих і обчислювальних програм на наявність дефектів, створення типової загальної інформаційної бази

алгоритмів, вихідних текстів і програмних засобів, що дозволяють виявляти навмисні програмні дефекти.

– Послідовної багаторівневої фільтрації програмних модулів в процесі їх створення із застосуванням функціонального дублювання розробок та поетапного контролю.

– Типізації алгоритмів, програм і засобів інформаційної безпеки, що забезпечує інформаційну, технологічну і програмну сумісність, на основі максимальної їх уніфікації за всіма компонентами і інтерфейсів.

– Централізованого управління базами даних проектів ПЗ і адміністрування технології їх розробки з жорстким розмежуванням функцій, обмеженням доступу відповідно до засобами діагностики, контролю та захисту.

– Блокування несанкціонованого доступу співвиконавців та абонентів державних мереж зв'язку, підключених до стендів для розробки програм.

– Статистичного обліку і ведення системних журналів про всіх процесах розробки ПЗ з метою контролю технологічної безпеки.

– Використання тільки сертифікованих і вибраних як єдиних інструментальних засобів розробки програм для нових технологій обробки інформації та перспективних архітектур обчислювальних систем.

Принципи забезпечення технологічної безпеки на етапах стендових і приймально-здавальних випробувань

Принципи забезпечення безпеки ПЗ на даному етапі включають принципи:

– Тестування ПЗ на основі розробки комплексів тестів, параметрів для конкретних класів програм з можливістю функціонального та статистичного контролю в широкому діапазоні зміни вхідних і вихідних даних.

– Проведення натурних випробувань програм при екстремальних навантаженнях з імітацією впливу активних дефектів.

– Здійснення «фільтрації» програмних комплексів з метою виявлення можливих навмисних дефектів певного призначення на базі створення моделей загроз та відповідних скануючих програмних засобів.

– Розробки та експериментальної відпрацювання засобів верифікації програмних виробів.

– Проведення стендових випробувань ПЗ для визначення ненавмисних програмних помилок проектування і помилок розробника, що призводять до невиконання цільових функцій програм, а також виявлення потенційно «вузьких» місць у програмних засобах для руйнівного впливу.

– Відпрацювання засобів захисту від несанкціонованого впливу порушників на ПЗ.

Принципи забезпечення безпеки при експлуатації програмного забезпечення

Принципи забезпечення безпеки ПЗ на даному етапі включають принципи:

– Збереження та обмеження доступу до еталонів програмних засобів, недопущення внесення змін до них.

– Профілактичного вибіркового тестування і повного сканування програмних засобів на наявність умисних дефектів.

– Ідентифікації ПЗ на момент введення його в експлуатацію відповідно до передбачуваними загрозами безпеки ПЗ і його контроль.

– Забезпечення модифікації програмних виробів під час їх експлуатації шляхом заміни окремих модулів без зміни загальної структури та зв’язків з іншими модулями.

– Суворого обліку та каталогізації всіх супроводжуваних програмних засобів, а також інформації яка збирається, оброблюється і зберігається.

– Статистичного аналізу інформації про всі процеси, робочі операції, відступи від режимів штатного функціонування ПЗ.

– Гнучкого застосування додаткових засобів захисту ПЗ в разі виявлення нових, непрогнозованих загроз інформаційної безпеки.

2.7 Створення алгоритмічно безпечного програмного забезпечення

Основною відмінною рисою підходу, пов’язаного зі створенням алгоритмічно безпечного програмного забезпечення (ПЗ) є те, що початок процесу забезпечення безпеки програм при їх розробці можна перенести на більш ранні етапи життєвого циклу програмного забезпечення, наприклад, на етапи, які передують етапу випробування програм, тим самим збільшивши загальний час на внесення до програми захисних функцій. [2]

Основна складність у розробці безпечних програм зазначеного типу полягає в труднощі знаходження ефективних алгоритмів їх функціонування, які були до того ж доказовою коректними. І, ймовірно, проблема нерозв’язності доказовості безпеки для складних програм при використанні методів їх тестування на етапі випробувань залишається, в разі застосування методів створення алгоритмічно безпечного програмного забезпечення, як і раніше нерозв’язною.

Одним з головних методичних питань створення безпечного програмного забезпечення є постановка задачі (формулювання проблеми) її розробки. Припустимо, деякого розробнику (колективу розробників) пропонується розробити програму «Р» для деякого об’єкту автоматизації. При цьому наслідки некоректного функціонування програми такі, що можуть привести до якихось «небажаних» або навіть катастрофічних наслідків для об’єкта автоматизації. Виходячи з гіпотези, що в загальному випадку проблема доказовості безпеки для складних комплексів програм є нерозв’язною, неформальна якісна постановка задачі розробки зазначених програм може бути тоді сформульована таким чином.[2]

Постановка 1. «При деяких умовах і обмеженнях необхідно розробити програму «Р», яка коректно обчислює результат майже для всіх своїх вхідних значень».

Таким чином, розробник констатує факт, що лише для незначної (можливо досить малої) частки вхідних значень програма може видати некоректне вихідне значення в результаті не виявленого програмного дефекту або активізації невиявленою програмної закладки.

Зрозуміло, що в деяких випадках виникає певна необхідність отримання яких-небудь більш точних кількісних оцінок ступеня захисту створеної програми від внутрішніх дефектів, з метою отримання певних гарантій надійного функціонування об’єкта автоматизації.

При використанні контрольно-випробувальних методів аналізу безпеки програм, коли аналізу піддається тільки виконуваний код програми можна отримати (як правило, експертним шляхом), або наближені кількісні характеристики виявлення дефектів в контрольованих програмах, або стратифікаційних характеристики не порушення програмою деяких умов безпеки. В цьому випадку постановка 1 завдання розробки програми «Р» принципово не міняється.

Постановка ж завдання розробки безпечного програмного забезпечення за рахунок алгоритмічно безпечних процедур змінюється з точки зору отримання точних кількісних (в даному випадку імовірнісних) характеристик кардинальним чином.

Постановка 2. «При деяких умовах і обмеженнях необхідно розробити програму «Р*», яка коректно обчислює результат для всіх своїх вхідних значень з малою ймовірністю помилки».

На прикладі постановок 3 4 5 і 6, можна також простежити істотну різницю в парадигмі розробки безпечних програм. Постановки 3 і 4 на відміну від постановок 1 і 2 враховують не «зміст» наборів вхідних значень програми «Р», а їх розподілу ймовірностей, а постановки 5 і 6 є в деякому сенсі узагальненими і враховують заданий структурний критерій тестування.

Постановка 3. «При деяких умовах і обмеженнях необхідно розробити програму «Р», яка некоректно обчислює результат лише для деякого приватного розподілу ймовірностей своїх вхідних величин».

Постановка 4. «При деяких умовах і обмеженнях необхідно розробити програму «Р*», яка коректно обчислює результат для будь-якого розподілу ймовірностей своїх вхідних величин з малою ймовірністю помилки».

Постановка 5. «При деяких умовах і обмеженнях необхідно розробити програму «Р», яка коректно обчислює результат майже на всіх тестах повної системи тестів програми щодо заданого структурного критерію тестування».

Постановка 6. «При деяких умовах і обмеженнях необхідно розробити програму «Р*», яка коректно з малою ймовірністю помилки обчислює результат на всіх тестах повної системи тестів щодо заданого структурного критерію тестування».

Одним із принципових умов вирішення завдання в постановці 2, 4 і 6 є наявність якогось властивості алгоритмічної трансформації, що дозволяє переходити від традиційного шляху створення програм, які будуть потім перевірятися на наявність дефектів, до апріорно безпечним програмам. До числа таких прикладів можна віднести самостійно корегуючи і самостійно тестувальні програми які володіють властивістю випадкової самозводження і програми, створені на базі методів конфіденційних обчислень.

Розглянуті до цього методи аналізу безпеки ПЗ пов’язані зі спробами убезпечити фактично вже розроблене програмне забезпечення від дій зловмисника. Це означає, що розробка безпечного ПЗ можлива за рахунок створення засобів протидії програмним закладок для продуктів, створених на основі існуючих інформаційних технологій створення програмного забезпечення. Тобто, тільки після факту розробки програм починається верифікаційний аналіз, тестування або контроль їх на технологічну безпеку. У цьому сенсі проблема забезпечення технологічної безпеки програмного забезпечення більш близька до фундаментальної проблеми його надійності.

У той же час, дані методи створення безпечного ПЗ характеризуються рядом істотних недоліків, до числа яких можна віднести: неможливість перекриття для більшості програмних комплексів всього спектру тестових наборів вихідних даних, необхідність створення високонадійних механізмів тестування програм, наприклад, механізмів екстраполяції результатів випробувань; істотні ресурсовитрати на проведення випробувань і т.п.[2]

Тому останнім часом з’явилася нагальна необхідність у створенні нових інформаційних технологій розробки ПЗ, початково орієнтованих на створення безпечних програмних продуктів відносного заданого класу. У цьому випадку проблема досліджень зводиться до розробки таких математичних моделей, які представляються адекватної формальної основою для створення методів захисту програмного забезпечення на етапі його проектування і розробки. При цьому спочатку передбачається, що:

– один або декілька учасників проекту є (або, принаймні, можуть бути) зловмисниками;

– в процесі експлуатації зловмисник може вносити в програми зміни (необов’язково пов’язані з впровадженням програмних закладок);

– засоби обчислювальної техніки, на яких виконуються програми, не вільні від апаратних закладок.

Тоді, виходячи з цих припущень, формулюється наступна постановка задачі: «Потрібно розробити програмне забезпечення таким чином, що незважаючи на зазначені вище перешкоди, воно функціонувало б правильно». Одне з основних переваг полягає в тому, що одні й ті ж методи дозволяють захищатися від зловмисника, що діє як на етапі розробки, так і на етапі експлуатації програмного забезпечення. Однак, це досягається за рахунок деякого уповільнення обчислень, а також підвищення витрат на розробку програмного забезпечення.[2]

2.8 Система керування версіями

Система керування версіями (Version Control System, VCS) дозволяє зберігати декілька версій одного і того ж документа, при необхідності повертатися до більш ранніх версій, визначати, хто і коли зробив зміни в коді, і багато іншого. Такі системи найбільш широко використовуються при розробці програмного забезпечення для зберігання вихідних кодів програм які розробляються. Управління версіями використовується в інструментах конфігураційного управління (Software Configuration Management Tools). [14]

2.8.1 Загальні відомості

Ситуація, в якій електронний документ за час свого існування зазнає ряд змін, досить типова. При цьому часто буває важливо мати не тільки останню версію, а й кілька попередніх. У простому випадку можна просто зберігати декілька варіантів документа, нумеруючи їх відповідним чином. Такий спосіб неефективний (доводиться зберігати кілька практично ідентичних копій), вимагає підвищеної уваги і дисципліни і часто веде до помилок, тому були розроблені засоби для автоматизації цієї роботи.

Традиційні системи управління версіями використовують централізовану модель, коли є єдине сховище документів, кероване спеціальним сервером, який і виконує велику частину функцій з управління версіями. Користувач, який працює з документами, повинен спочатку отримати потрібну йому версію документа зі сховища; зазвичай створюється локальна копія документа, так звана «Робоча копія».

Може бути отримана остання версія або будь-яка з попередніх, яка може бути обрана за номером версії або датою створення, іноді й за іншими ознаками. Після того, як в документ внесено потрібні зміни, нова версія поміщається в сховище. На відміну від простого збереження файлу, попередня версія не стирається, а теж залишається в сховище і може бути звідти отримана в будь-який час. Сервер може використовувати дельта-компресію – такий спосіб зберігання документів, при якому зберігаються тільки зміни між послідовними версіями, що дозволяє зменшити обсяг збережених даних. Оскільки зазвичай найбільш затребуваною є остання версія файлу, система може при збереженні нової версії зберігати її цілком, замінюючи в сховище останню раніше збережену версію на різницю між цією і останньою версією.

Деякі системи підтримують збереження версій обох видів: більшість версій зберігається у вигляді дельт, але періодично (по спеціальній команді адміністратора) виконується збереження версій всіх файлів в повному вигляді; такий підхід забезпечує максимально повне відновлення історії в разі пошкодження сховища.

Іноді створення нової версії виконується непомітно для користувача (прозоро), або прикладної програмою, що має вбудовану підтримку такої функції, або за рахунок використання спеціальної файлової системи. У цьому випадку користувач просто працює з файлом, як звичайно, і при збереженні файлу автоматично створюється нова версія.

Часто буває, що над одним проектом одночасно працюють декілька людей. Якщо дві людини змінюють один і той же файл, то один з них може випадково скасувати зміни, зроблені іншим. Системи управління версіями відстежують такі конфлікти і пропонують засоби їх вирішення. Більшість систем може автоматично об’єднати (злити) зміни, зроблені різними розробниками. Однак таке автоматичне об’єднання змін, звичайно, можливо тільки для текстових файлів і за умови, що змінювалися різні (непересічні) частини цього файлу. Таке обмеження пов’язане з тим, що більшість систем керування версіями орієнтовані на підтримку процесу розробки програмного забезпечення, а вихідні коди програм зберігаються в текстових файлах. Якщо автоматичне об’єднання виконати не вдалося, система може запропонувати вирішити проблему вручну.

Часто виконати злиття неможливо ні в автоматичному, ні в ручному режимі, наприклад, якщо формат файлу невідомий або занадто складний. Деякі системи управління версіями дають можливість заблокувати файл у сховищі. Блокування не дозволяє іншим користувачам отримати робочу копію або перешкоджає зміні робочої копії файлу (наприклад, засобами файлової системи) і забезпечує, таким чином, винятковий доступ тільки тому користувачеві, який працює з документом.

Багато систем управління версіями надають ряд інших можливостей:

– Дозволяють створювати різні варіанти одного документа, так звані гілки, із загальною історією змін до точки розгалуження і з різними - після неї.

– Дають можливість дізнатися, хто і коли додав або змінив конкретний набір рядків у файлі.

– Ведуть журнал змін, в який користувачі можуть записувати пояснення про те, що і чому вони змінили в даній версії.

– Контролюють права доступу користувачів, дозволяючи або забороняючи читання або зміну даних, залежно від того, хто запитує цю дію.[14]

2.8.2 Типовий порядок роботи з системою

Кожна система управління версіями має свої специфічні особливості в наборі команд, порядок роботи користувачів і адмініструванні. Тим не менш, загальний порядок роботи для більшості VCS абсолютно стереотипів. Тут передбачається, що проект, яким би він не був, вже існує і на сервері розміщений його репозиторій, до якого розробник отримує доступ.

2.8.2.1 Початок роботи з проектом

Першою дією, яку повинен виконати розробник, є отримання робочої копії проекту або тієї його частини, з якою доведеться працювати. Ця дія виконується за допомогою стандартної команди вилучення версії (checkout або clone). Розробник задає версію, яка повинна бути скопійована, за замовчуванням зазвичай копіюється остання (або вибрана адміністратором в якості основної) версія.[14]

За командою вилучення встановлюється з’єднання з сервером і проект (або його частина - один з каталогів з підкаталогами) у вигляді дерева каталогів і файлів копіюється на комп’ютер розробника. Звичайною практикою є дублювання робочої копії: крім основного каталогу з проектом на локальний диск (або в окремий, спеціально вибраний каталог, або в системні підкаталоги основного дерева проекту) додатково записується ще одна його копія. Працюючи з проектом, розробник змінює лише файли основної робочої копії. Друга локальна копія зберігається як еталон, дозволяючи в будь-який момент

без звернення до сервера визначити, які зміни внесені в конкретний файл або проект в цілому; як правило, будь-яка спроба ручного зміни цієї копії призводить до помилок в роботі програмного забезпечення VCS.

2.8.2.2 Щоденний цикл роботи

При деяких варіаціях, що визначаються особливостями системи та деталями прийнятого бізнес-процесу, звичайний цикл роботи розробника протягом робочого дня виглядає наступним чином.

Оновлення робочої копії

У міру внесення змін до проекту робоча копія на комп'ютері розробника старіє, розбіжність її з основною версією проекту збільшується. Це підвищує ризик виникнення конфліктних змін. Тому зручно підтримувати робочу копію в стані, максимально близькому до поточної основної версії, для чого розробник виконує операцію поновлення робочої копії (update) наскільки можливо часто.

Модифікація проекту

Розробник модифікує проект, змінюючи вхідні в нього файли в робочій копії відповідно до проектного завдання. Ця робота проводиться локально і не вимагає звернень до сервера VCS.

Фіксація змін

Завершивши черговий етап роботи над завданням, розробник фіксує (commit) свої зміни, передаючи їх на сервер (або в основну гілку, якщо робота над завданням повністю завершена, або в окрему гілку розробки даного завдання). VCS може вимагати від розробника перед фіксацією обов'язково виконати оновлення робочої копії. При наявності в системі підтримки відкладених змін (shelving) зміни можуть бути передані на сервер без фіксації. Якщо затверджена політика роботи в VCS це дозволяє, то фіксація змін може проводитися не щодня, а тільки по завершенні роботи над завданням; в цьому випадку до завершення роботи всі пов'язані із завданням зміни зберігаються тільки в локальній робочої копії розробника.

2.8.2.3 Розгалуження

Робити дрібні виправлення в проекті можна шляхом безпосередньої редагування робочої копії і наступною фіксацією змін прямо в головній галузі (стовбурі) на сервері. Однак при виконанні скільки-небудь значних за обсягом робіт такий порядок стає незручним: відсутність фіксації проміжних змін на сервері не дозволяє працювати над чимось в груповому режимі, крім того, підвищується ризик втрати змін при локальних аваріях і втрачається можливість аналізу і повернення до попередніх варіантам коду в межах даної роботи. Тому для таких змін звичайною практикою є створення гілок (branch), тобто розгалуження від стовбура в якійсь версії нового варіанту проекту або його частини, розробка в якому ведеться паралельно зі змінами в основний версії. Гілка створюється спеціальною командою. Робоча копія гілки може бути створена заново звичайним чином (командою вилучення робочої копії, із зазначенням адреси або ідентифікатора гілки), або шляхом перемикання наявної робочої копії на задану гілку.[14]

Базовий робочий цикл при використанні гілок залишається точно таким же, як і в загальному випадку: розробник періодично оновлює робочу копію (якщо з гілкою працює більше однієї людини) і фіксує в ній свою щоденну роботу. Іноді гілку розробки так і залишається самостійною (коли зміни породжують новий варіант проекту, який далі розвивається окремо від основного), але частіше за все, коли робота, для якої створена гілка, виконана, гілка реінтегрується в стовбур (основну гілку).

Це може робитися командою злиття (зазвичай merge), або шляхом створення патча (patch), що містить внесені в ході розробки гілки зміни і застосування цього патча до поточної основної версії проекту.

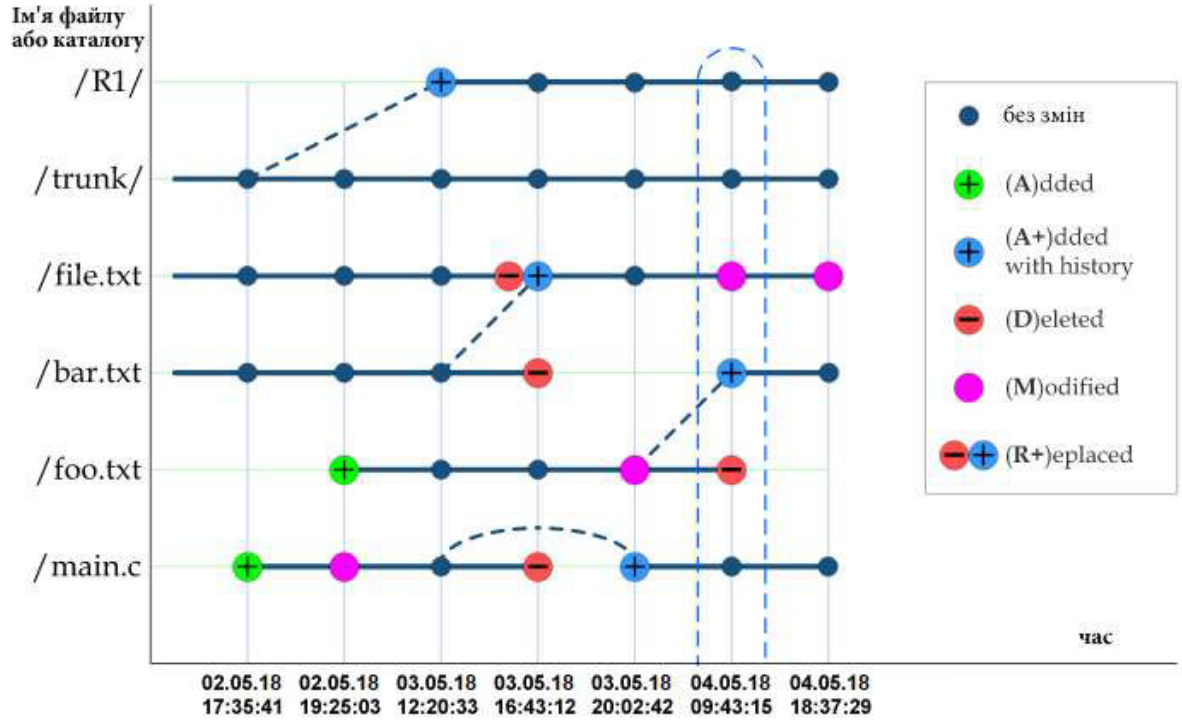

Рисунок 2.5 – Робочий цикл системи керування версіями

2.8.2.4 Злиття версій

Три види операцій, які виконуються в системі управління версіями, можуть призводити до необхідності об’єднання змін. Це:

– Оновлення робочої копії (зміни, зроблені в основної версії, зливаються з локальними).

– Фіксація змін (локальні зміни зливаються із змінами, уже зафіксованими в основний версії).

– Злиття гілок (зміни, зроблені в одній гілці розробки, зливаються із змінами, зробленими в інший).

Абсолютна більшість сучасних систем управління версіями орієнтовано, в першу чергу, на проекти розробки програмного забезпечення, в яких основним видом вмісту файлу є текст. Відповідно, механізми автоматичного злиття змін орієнтуються на обробку текстових файлів, тобто файлів, які містять текст, що складається з рядків буквено-цифрових символів, прогалин і табуляцій, розділених символами переведення рядка.

При визначенні допустимості злиття змін в межах одного і того ж текстового файлу працює типовий механізм рядкового порівняння текстів, який порівнює версії які об’єднуються з базовою і будує список змін, тобто доданих, віддалених і замінених наборів рядків . Мінімальною одиницею даних для цього алгоритму є рядок, навіть найменше відміну робить рядки різними. З урахуванням того, що символи-роздільники, в більшості випадків, не несуть смислового навантаження, механізм злиття може ігнорувати ці символи при порівнянні рядків.

Ті знайдені набори змінених рядків, які не перетинаються між собою, вважаються сумісними і їх злиття робиться автоматично.

2.8.2.5 Конфлікти та їх вирішення

Ситуація, коли при злитті декількох версій зроблені в них зміни перетинаються між собою, називають конфліктом. При конфлікті змін система управління версіями не може автоматично створити об’єднаний проект і змушена звертатися до розробника. Як вже говорилося вище, конфлікти можуть виникати на етапах фіксації змін, оновлення або злиття гілок. У всіх випадках при виявленні конфлікту відповідна операція припиняється до його дозволу. Деякі системи навіть не намагаються вирішувати конфлікти на етапі фіксації або злиття гілки зі стовбуром; при виникненні таких конфліктів операція повністю скасовується і розробнику пропонується спочатку виконати оновлення робочої копії, вирішити конфлікти і тільки після цього виконувати об’єднання своїх змін з базовою гілкою.[14]

Для вирішення конфлікту система, в загальному випадку, пропонує розробнику три варіанти конфліктуючих файлів: базовий, локальний і серверний.

2.8.2.6 Блокування

Механізм блокування дозволяє одному з розробників захопити в монопольне використання файл або групу файлів для внесення в них змін. На той час, поки файл заблокований, він залишається доступним всім іншим

розробникам тільки на читання, і будь-яка спроба внести в нього зміни відкидається сервером.

2.8.3 Базові принципи розробки ПЗ в VCS

Порядок використання системи управління версіями в кожному конкретному випадку визначається технічними регламентами і правилами, прийнятими в конкретній фірмі або організації, що розробляє проект. Тим не менш, загальні принципи правильного використання VCS нечисленні і єдині для будь-яких розробок і систем управління версіями.

– Будь-які робочі, тестові або демонстраційні версії проекту збираються тільки з репозиторію системи. «Персональні» збірки, що включають ще не зафіксовані зміни, можуть робити тільки розробники для цілей проміжного тестування. Таким чином, гарантується, що репозиторій містить все необхідне для створення робочої версії проекту.

– Поточна версія головної гілки завжди коректна. Не допускається фіксація в головній гілці неповних або таких які не пройшли хоча б попереднє тестування змін. В будь-який момент складання проекту, проведена з поточної версії, повинна бути успішною.

– Будь-яка значима зміна повинна оформлятися як окрема гілка. Проміжні результати роботи розробника фіксуються в цю гілку. Після завершення роботи над зміною гілку об’єднується зі стволом. Винятки допускаються тільки для дрібних змін, робота над якими ведеться одним розробником протягом не більш ніж одного робочого дня.[14]

– Версії проекту позначаються тегами. Виділена і позначений тегом версія більш ніколи не змінюється.

2.9 Процес тестування програмного продукту

Розглянемо процес проведення тестування програмного продукту.

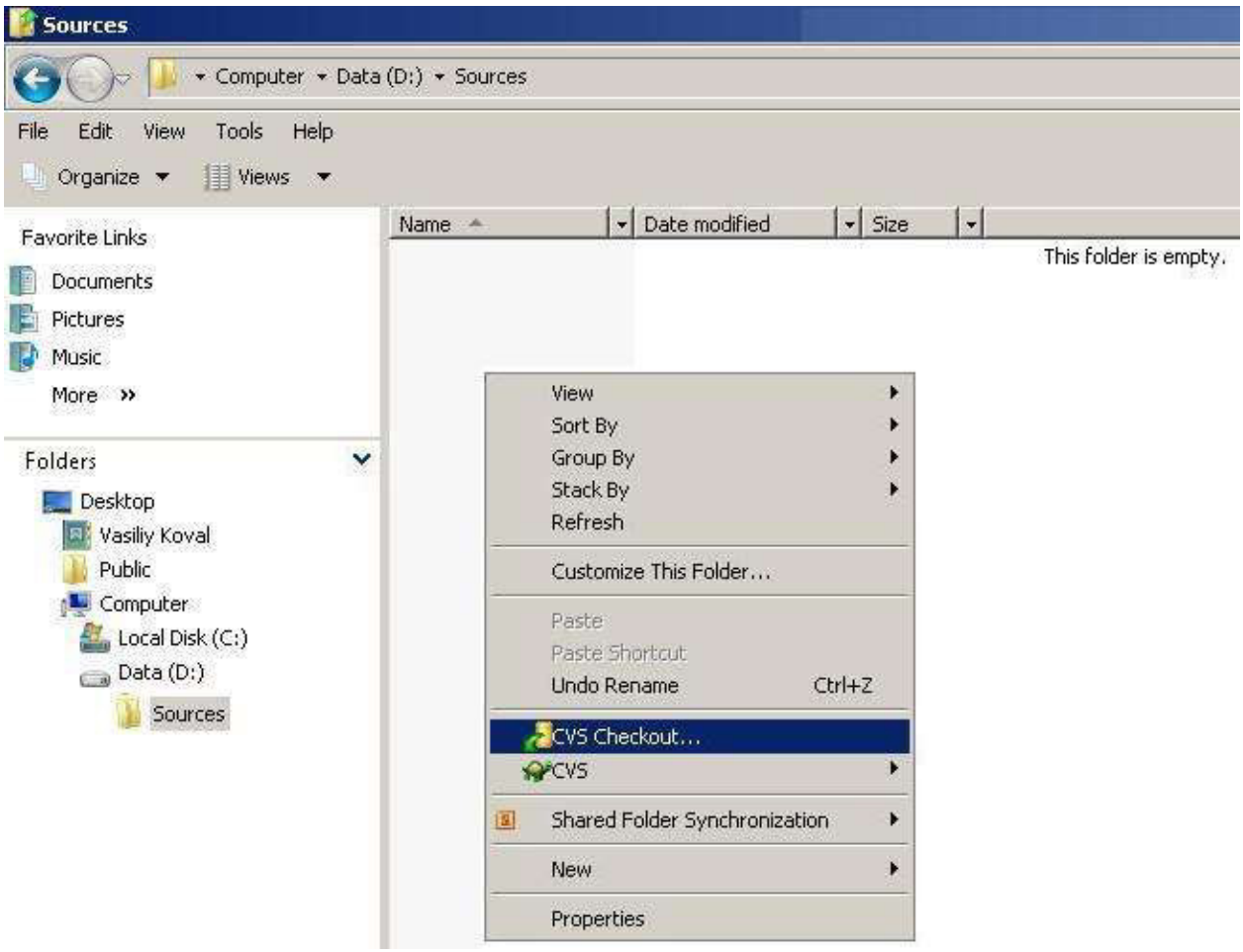

Рисунок 2.6 – Завантаження програмного коду

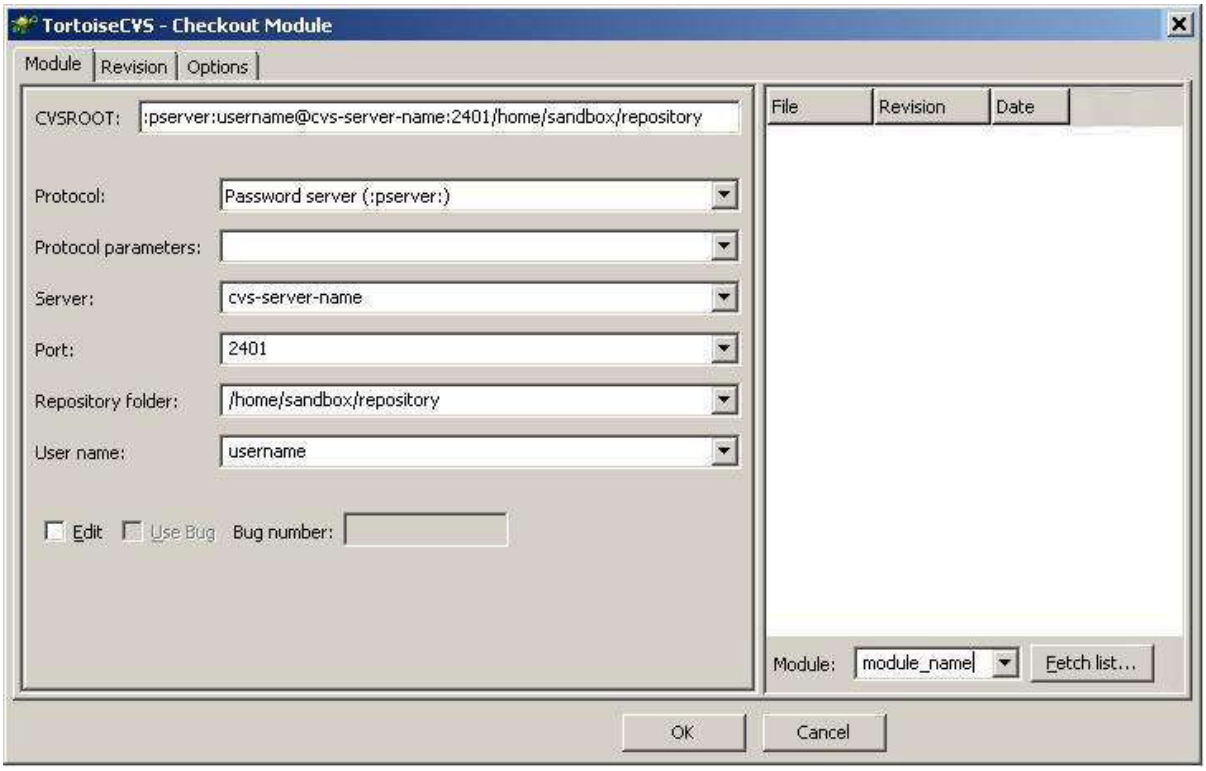

Рисунок 2.7 – Вікно налаштувань

Рисунок 2.8 – Процес завантаження модулів

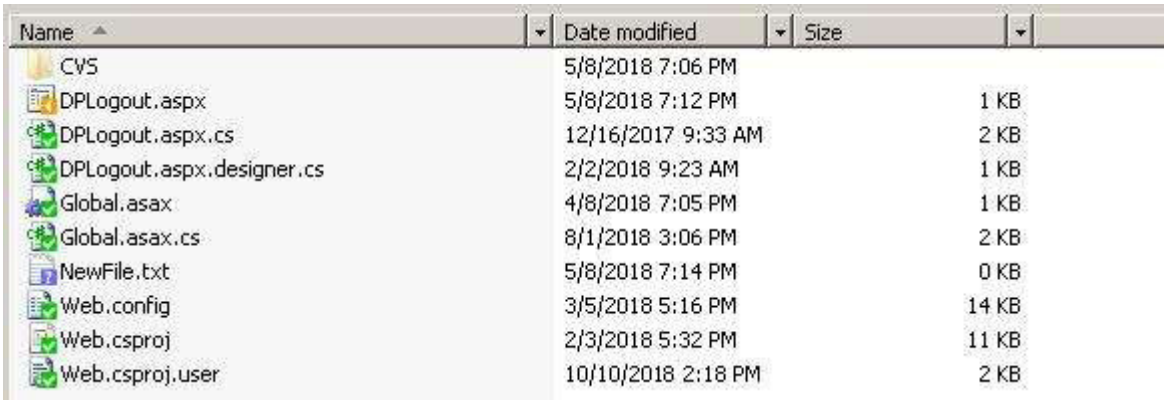

Рисунок 2.9 – Робочій простір

На рисунку 2.9 файли помічені помаранчевим прапорцем – файли змінені користувачем, зеленим прапорцем – файли не змінювали, та синім прапорцем – файли які буди створені користувачем і відсутні на сервері CVS.

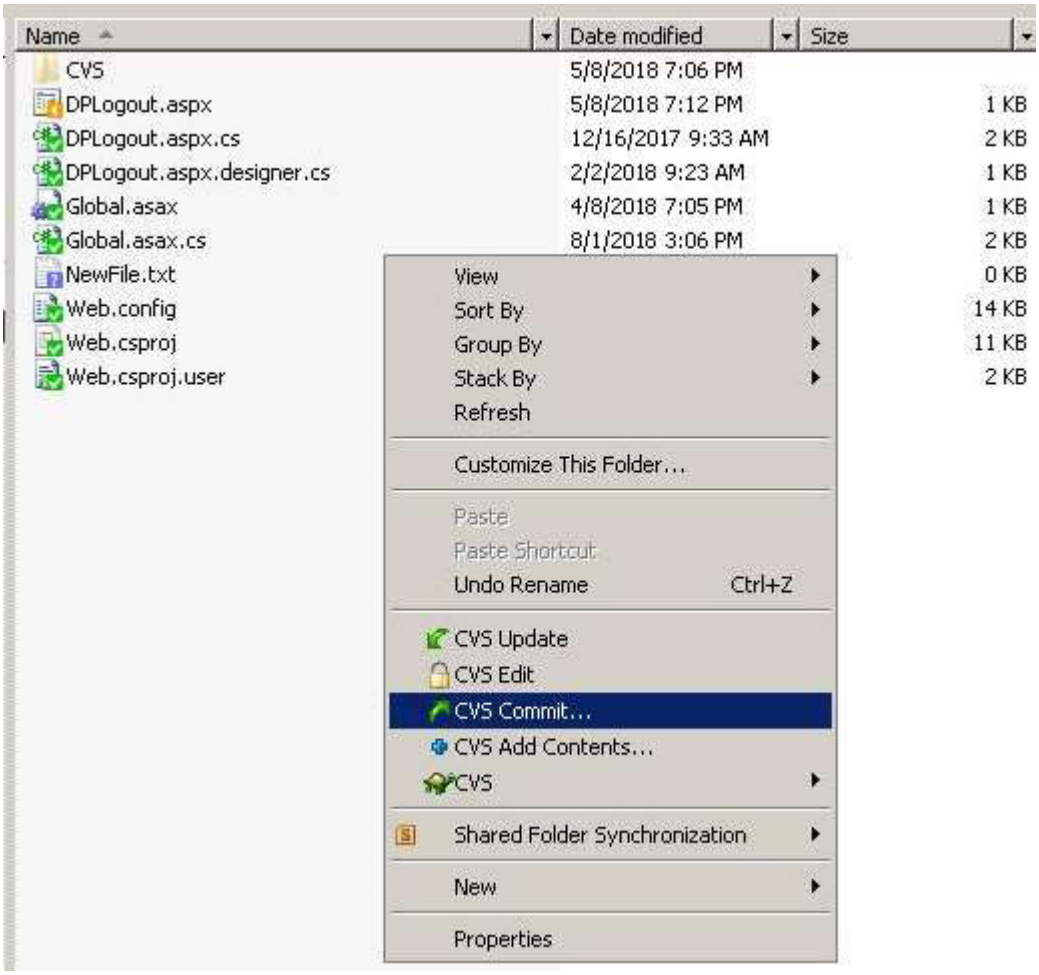

Рисунок 2.9 – Завантаження файлів на сервер після виконаної роботи

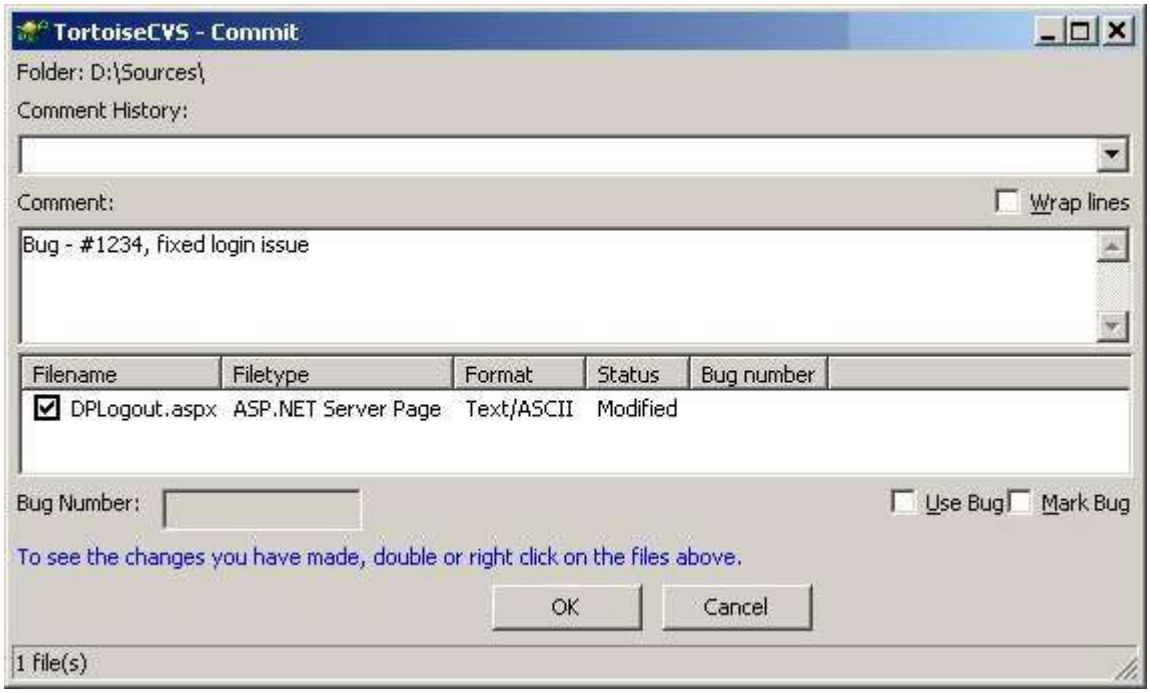

Рисунок 2.10 – Вибір файлів для завантаження на сервер та описання виправлених помилок

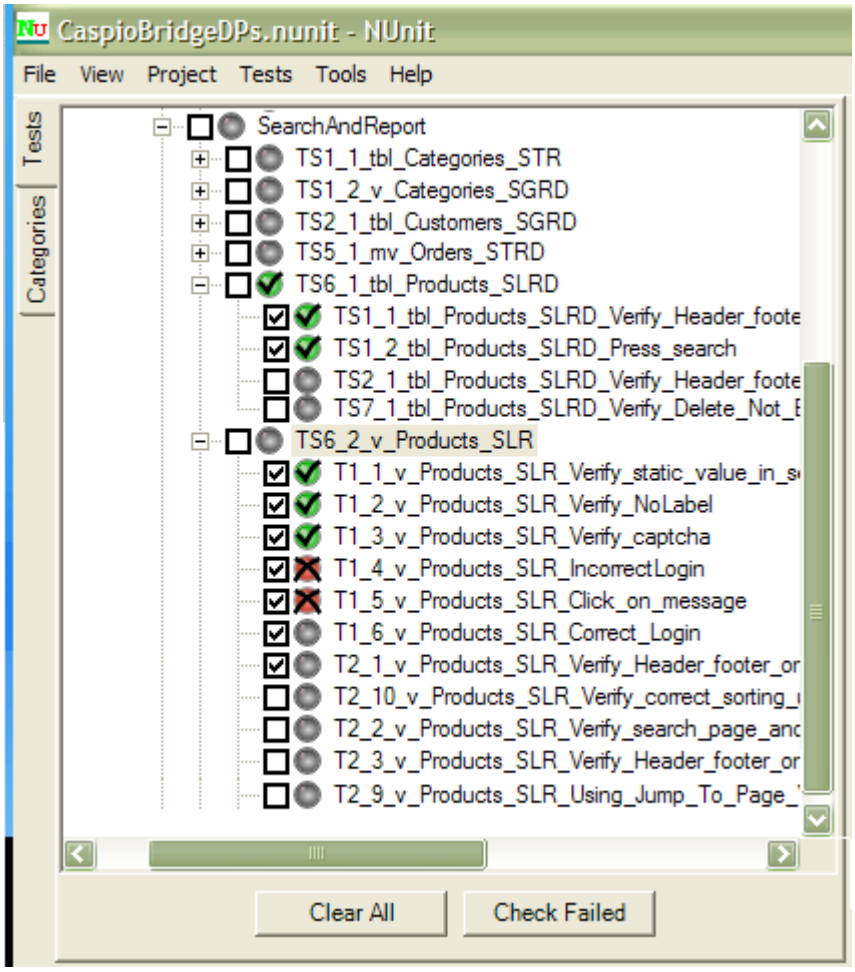

Рисунок 2.11 – Структура проекту

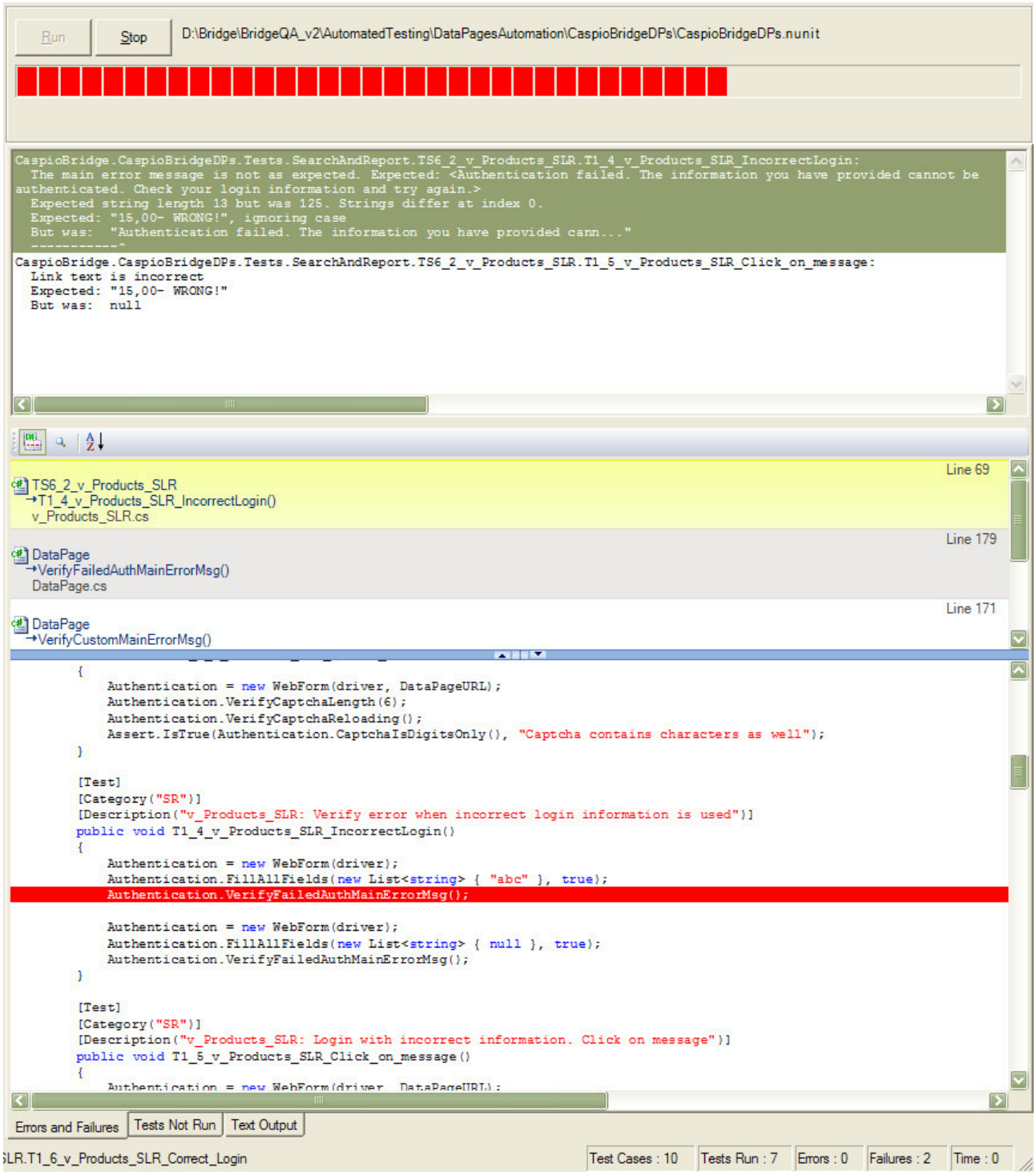

Рисунок 2.12 – Запуск процесу тестування

Рисунок 2.13 – Результат тестування

Після проведення тестування програмного забезпечення отримуємо звіт, в якому вказано кількість помилок допущених при розробці програмного продукту.

2.10 Рекомендації з вдосконалення системи захисту інформації

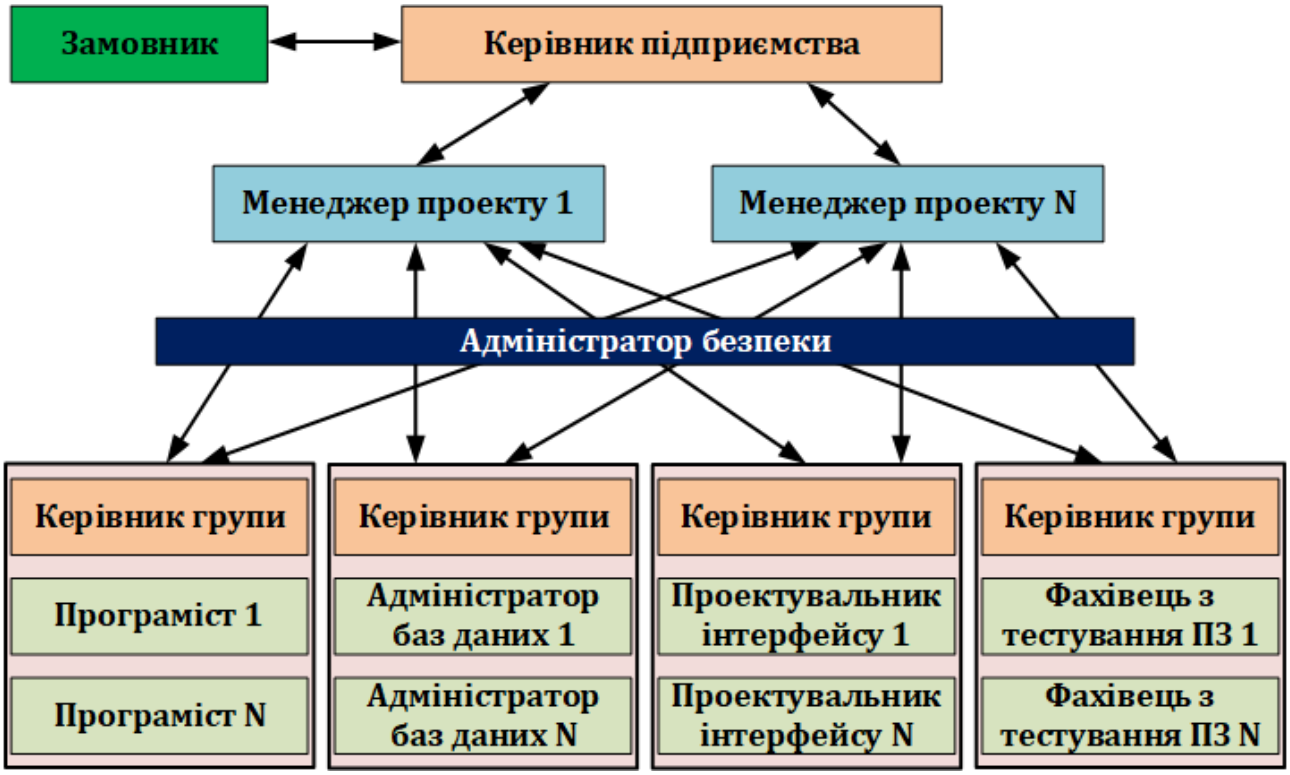

Рисунок 2.14 – Структура підприємства

На малих підприємствах з розробки програмного забезпечення захист інформації стоїть на останньому місті. Більшість підприємств обходяться лише організаційними заходами, такими як підписання документів з нерозголошення комерційної інформації та ознайомленням персоналу з посадовими інструкціями.

В цілому ці заходи забезпечать захист підприємства лише на законодавчому рівні, а канали витоку залишаться.

Для вдосконалення процедур розробки захищених програмних продуктів, на підприємстві необхідно виділити нову штатну одиницю «Адміністратор безпеки».

В діючих посадових інструкціях необхідно внести зміни, відповідно до яких менеджери проектів і керівники груп будуть надавати інформацію о процесах розробки ПЗ адміністратору безпеки, який в свою чергу надає

рекомендації по вдосконаленню процедур розробки захищеного ПЗ та надає відповідну звітність керівнику підприємства.

2.11 Рекомендації на середньострокову перспективу

Розробники ПЗ повинні постійно підвищувати рівень безпеки своїх продуктів і процесів, роблячи упор на специфікаціях та архітектури. Велика частина існуючого програмного забезпечення не відрізняється добрими характеристиками захисту. Виробникам програмного забезпечення слід використовувати для перебудови і реструктуризації систем з неприйнятною архітектурою і конструктивними рішеннями високоякісні процеси розробки, що дозволяють вивести параметри безпеки на належний рівень.

Державі слід розробити програму оцінки і вимірювання основних параметрів, яка дозволить визначати ефективність таких процесів і сертифікувати ті з них, які дійсно підходять для випуску безпечного програмного забезпечення. Крім того, необхідно звернутися в центр CERT з проханням зайнятися разом з постачальниками програмного забезпечення та іншими зацікавленими організаціями оцінкою процесів і створенням методики тестування, що дозволяє проаналізувати поліпшення які з’являються. Нарешті, необхідно розглянути стан національної інформаційної інфраструктури, спільно з представниками програмної галузі сформулювати порівняльну оцінку щорічні мети вдосконалення систем безпеки і стежити за темпами досягнення поставлених цілей.

2.12 Рекомендації на довгостроковий період

2.12.1 Сертифікація

Певною мірою програми сертифікації (зокрема, Common Criteria і ITSEC) орієнтовані на перевірку вже існуючого програмного забезпечення. П’ятий, шостий і сьомий рівні програми Common Criteria спрямовані на те, щоб показати, що конструктивні рішення відповідають вимогам до захисту. У них навіть передбачена перевірка використання при розробці методів, що забезпечують створення програмного забезпечення з мінімальною кількістю

дефектів. На практиці, однак, ці рівні використовуються досить рідко, а число інцидентів, пов’язаних з безпекою, як і раніше має тенденцію до зростання. Державі необхідно стежити за розробкою програм контролю та аналізу, оцінювати їх якість, а також ініціювати сертифікацію процесів і методів, які довели свою високу ефективність при створенні безпечного програмного забезпечення.

2.12.2 Освіта та професійна підготовка

Більшість університетів пропонують навчальні курси, орієнтовані на вивчення предметів напрямку криптографії і націлені, головним чином, на теоретичні аспекти конфіденційності та цілісності. Це добре для майбутніх дослідників, але навчання і підготовки практиків достатньої уваги не приділяється. Цілісні підходи до розробки безпечного програмного забезпечення, які можна використовувати в подальшій практиці, як і раніше зустрічаються досить рідко.

Міністерству внутрішніх справ та Служби безпеки України слід стимулювати і фінансувати університети, які пропонують відповідне навчання і проводять дослідження процесів розробки безпечного програмного забезпечення. Особливу увагу необхідно приділити розвитку вже наявних в даній області програм. Університетам слід запрошувати експертів в сфері безпеки і фахівців-практиків з організацій, в яких використовуються сертифіковані процеси розробки безпечного програмного забезпечення, для надання допомоги в підготовці майбутніх інженерів-програмістів та IT-спеціалістів.

Має сенс заохочувати і фінансувати дослідження, спрямовані на їх визначення, документування та на забезпечення доступу до інших процесів розробки безпечного програмного забезпечення, засобів тестування, конструкторських рішень і способів реалізації систем безпеки.

2.13 Висновок

У спеціальній частині дипломної роботи булі розглянуті загрози інформації що циркулює на підприємстві, наведені підходи з управління

персоналом на підприємстві, наведено етапи проведення тестування програмного забезпечення з використання системи керування версіями, розглянута типова організаційна структура підприємства з розробки програмного забезпечення та дані рекомендації по вдосконаленню організаційної структури підприємства.

РОЗДІЛ 3. ЕКОНОМІЧНИЙ РОЗДІЛ

Метою цього розділу є обґрунтування економічної доцільності вдосконалення організації захисту інформації на підприємстві по розробці програмних продуктів. Для досягнення поставленої мети необхідно здійснити наступні розрахунки:

– капітальні витрати на придбання і налагодження складових системи інформаційної безпеки або витрат, що пов’язані з виготовленням апаратури, приладів, програмного забезпечення;

– річні експлуатаційні витрати на утримання і обслуговування об'єкта проектування;

– річний економічний ефект;

– показники економічної ефективності організації захисту інформації на підприємстві по розробці програмних продуктів.

3.1 Розрахунок (фіксованих) капітальних витрат

Капітальні інвестиції – це кошти, призначені для створення і придбання основних фондів і нематеріальних активів, що підлягають амортизації.

3.1.1. Визначення витрат на створення програмних засобів захисту інформації

3.1.1.1 Визначення трудомісткості проведення тестування програмного забезпечення з використання системи керування версіями

Трудомісткість тестування програмного забезпечення з використання системи керування версіями визначається тривалістю кожної робочої операції:

$$t = t_{mз} + t_{в} + t_{а} + t_{пр} + t_{опр} + t_{д}, \text{ годин},$$

де $t_{тз}$ – тривалість складання технічного завдання на розробку ПЗ, $t_{тз}$=10;

$t_в$ – тривалість вивчення ТЗ, літературних джерел за темою тощо;

$t_а$ – тривалість розробки блок-схеми алгоритму;

$t_{пр}$ – тривалість програмування за готовою блок-схемою;

$t_{опр}$ – тривалість опрацювання програми на ПК;

$t_д$ – тривалість підготовки технічної документації на ПЗ.

Умовна кількість операторів у програмі:

$$Q = q \cdot c\ (1 + p) = 10*1,5*(1+0,1) = 16,5$$

де q – очікувана кількість операторів;

c – коефіцієнт складності програми;

p – коефіцієнт корекції програми в процесі її опрацювання.

Тривалість вивчення технічного завдання, опрацювання довідкової літератури з урахуванням уточнення ТЗ і кваліфікації програміста:

$$t_в = \frac{Q \cdot B}{(75...85) \cdot k} = \frac{16,5 \cdot 1,4}{85 \cdot 1} = 0,27 \text{ годин},$$

де В – коефіцієнт збільшення тривалості етапу внаслідок недостатнього опису завдання, В = 1,2...1,5;

k – коефіцієнт, що враховує кваліфікацію програміста і визначається стажем роботи за фахом:

- до 2 років – 0,8;
- від 2 до 3 років – 1,0;
- від 3 до 5 років – 1,1...1,2;
- від 5 до 7 років – 1,3...1,4;

- понад 7 років – 1,5...1,6.

Тривалість розробки блок-схеми алгоритму:

$$ta = \frac{Q}{(20...25)\cdot k} = \frac{16,5}{20\cdot 1} = 0,825 \text{ годин.}$$

Тривалість складання програми за готовою блок-схемою:

$$tпр = \frac{Q}{(20...25)\cdot k} = \frac{16,5}{20\cdot 1} = 0,825, \text{ годин.}$$

Тривалість опрацювання програми на ПК:

$$tопр = \frac{1,5Q}{(4...5)\cdot k} = \frac{1,5\cdot 16,5}{5\cdot 1} = 4,95, \text{ годин.}$$

Тривалість підготовки технічної документації на ПЗ:

$$t_д = \frac{Q}{(15...20)\cdot k} + \frac{Q}{(15...20)}\cdot 0,75 = \frac{16,5}{20\cdot 1} + \frac{16,5}{20\cdot 1}\cdot 0,75 = 1,44$$

Отже,

t = 10+ 0,27 + 0,825 + 0,825 + 4,95+ 1,44 = 18,31 годин,

3.1.1.2. Розрахунок витрат на створення програмного продукту

Витрати на створення програмного продукту ***Кпз*** складаються з витрат на заробітну плату виконавця програмного забезпечення ***Ззп*** і вартості витрат машинного часу, що необхідний для опрацювання програми на ПК ***Змч***:

$$К_{пз} = З_{зп} + З_{мч} = 2197,2 + 168,44 = 2365,64 \text{ грн.}$$

$$З_{зп} = t\ З_{пр} = 18,31*120 = 2197,2 \text{ грн.}$$

де t – загальна тривалість створення ПЗ, годин;

Зпр – средньогодинна заробітна плата програміста з нарахуваннями, грн/годину.

Вартість машинного часу для налагодження програми на ПК визначається за формулою:

$$З_{мч} = (t_{опр} + t_{д}) \cdot С_{мч} = (4,95 + 1,44) \cdot 26,36 = 168,44 \text{ грн.}$$

де $t_{опр}$ – трудомісткість налагодження програми на ПК, годин;

$t_{д}$ – трудомісткість підготовки документації на ПК, годин;

$С_{мч}$ – вартість 1 години машинного часу ПК, грн./година.

Вартість 1 години машинного часу ПК визначається за формулою:

$$С_{мч} = 1,5 \cdot 10 \cdot 1,64 + \frac{4800 \cdot 0,6}{1920} + \frac{2500 \cdot 0,2}{1920} = 26,36 \text{ грн.}$$

Вдосконалення організації захисту інформації на підприємстві по розробці програмних продуктів передбачає використання системи керування версіями (Version Control System, VCS), а також інструменті конфігураційного управління (Software Configuration Management Tools). Ці продукти мають безкоштовні версії з відкритим кодом для 25 вільних вузлів, що відповідає потребам підприємства.

Таким чином, капітальні (фіксовані) витрати на проектування та впровадження проектного варіанта системи інформаційної безпеки складають:

$$К = К_{пр} + К_{зпз} + К_{пз} + К_{аз} + К_{навч} + К_{н} = 2365,64 + 6000 = 8365,64 \text{ грн.}$$

де $К_{пр}$ – вартість розробки проекту інформаційної безпеки та залучення для цього зовнішніх консультантів, тис. грн;

$К_{зпз}$ – вартість закупівель ліцензійного основного й додаткового програмного забезпечення (ПЗ), тис. грн;

$К_{пз}$ – вартість створення основного й додаткового програмного забезпечення, тис. грн;

$К_{аз}$ – вартість закупівлі апаратного забезпечення та допоміжних матеріалів, тис. грн;

$К_{навч}$ – витрати на навчання технічних фахівців і обслуговуючого персоналу, $К_{навч} = 6000$ грн;

$К_{н}$ – витрати на встановлення обладнання та налагодження системи інформаційної безпеки, тис. грн.

3.1.1 Розрахунок поточних витрат

Річні поточні витрати на функціонування системи інформаційної безпеки складають:

$$С = С_{в} + С_{к} + С_{ак}, \quad грн.$$

де $С_{в}$ - вартість відновлення й модернізації системи ($С_{в} = 0$);

$С_{к}$ - витрати на керування системою в цілому;

$С_{ак}$ - витрати, викликані активністю користувачів системи інформаційної безпеки ($С_{ак} = 0$ грн.).

Витрати на керування системою інформаційної безпеки ($С_{к}$) складають:

$$С_{к} = С_{н} + С_{а} + С_{з} + С_{ел} + С_{о} + С_{тос}, \text{ грн.}$$

Витрати на навчання адміністративного персоналу й кінцевих користувачів визначаються ($С_{н} = 0$ грн.).

Річний фонд заробітної плати інженерно-технічного персоналу, що обслуговує систему інформаційної безпеки ($С_з$), складає:

$$С_з = З_{осн} + З_{дод}, \text{ грн.}$$

Основна заробітна плата визначається, виходячи з місячного посадового окладу, а додаткова заробітна плата – в розмірі 8-10% від основної заробітної плати.

Основна заробітна плата одного системного адміністратора на місяць складає 8000 грн. Додаткова заробітна плата – 8% від основної заробітної плати. Отже,

$$С_з = 8000*12 + 8000*12*0,08= 112320 \text{ грн.}$$

З 01.01.2016 року ставка ЄСВ для всіх категорій платників складає 22%.

$$С_{єв} = 112320 *0,22 = 24710,4 \text{ грн.}$$

Вартість електроенергії, що споживається апаратурою системою інформаційної безпеки протягом року ($С_{ел}$), визначається за формулою:

$$С_{ел} = Р \cdot F_р \cdot Ц_е, \text{ грн.,}$$

де Р – встановлена потужність апаратури інформаційної безпеки, (Р = 1,5 кВт);

$F_р$ – річний фонд робочого часу системи інформаційної безпеки ($F_р$ = 1920 год.);

$Ц_е$ – тариф на електроенергію, ($Ц_е$ = 1,64 грн./кВт за годину).

Вартість електроенергії, що споживається апаратурою системою інформаційної безпеки протягом року:

$$С_{ел} = 1,5*1920*1,64 = 4723,2 \text{ грн.}$$

Витрати на технічне й організаційне адміністрування та сервіс системи інформаційної безпеки визначаються у відсотках від вартості капітальних витрат - 2% ($С_{тос}$ = 8365,64 *0,02=167,31 грн).

Витрати на керування системою інформаційної безпеки ($С_к$) визначаються:

$$С_к = 112320 + 24710,4 + 4723,2 + 167,31 = 151920,91 \text{ грн.}$$

Річні поточні витрати на функціонування системи інформаційної безпеки:

$$С = 151920,91 \text{ грн.}$$

3.2 Оцінка можливого збитку від атаки на вузол, або сегмент мережі

3.2.1 Оцінка величини збитку

Для розрахунку вартості збитку застосовуємо спрощену модель оцінки.

Необхідні вхідні дані для розрахунку:

де $t_п$ – час простою вузла внаслідок атаки ($t_п$ = 12 годин);

$t_в$ – час відновлення після атаки персоналом ($t_в$ = 8 годин);

$t_{ві}$ – час повторного введення загубленої інформації співробітниками атакованого сегменту мережі ($t_{ві}$ = 7 годин);

$З_о$ – заробітна плата обслуговуючого персоналу ($З_о$ = 9000 грн. на місяць);

$З_с$ – заробітна плата співробітників атакованого вузла ($З_с$ = 12000 грн. на місяць);

$Ч_о$ – чисельність обслуговуючого персоналу ($Ч_о$ = 3 особи);

$Ч_с$ – чисельність співробітників атакованого сегменту мережі ($Ч_с$ = 3 особи);

О – обсяг продажів атакованого сегменту мережі, (О = 320000 грн. на рік);

$П_{зч}$ – вартість заміни устаткування або запасних частин, ($П_{зч}$ = 0 грн.);

I – число атакованих вузлів (I = 3);

N – середнє число атак на рік (N = 30).

Упущена вигода від простою атакованого вузла становить:

$$U = П_п + П_в + V,$$

де $П_п$ – оплачувані втрати робочого часу та простої співробітників атакованого вузла, грн.;

$П_в$ – вартість відновлення працездатності сегмента мережі, грн.;

V – втрати від зниження обсягу продажів за час простою атакованого вузла, грн.

Упущена вигода від простою атакованого вузла:

$$U = 818,18 + 886,36 + 4153,85 = 5858,39 \text{ грн.},$$

Втрати від зниження продуктивності співробітників атакованого сегмента мережі являють собою втрати їхньої заробітної плати за час простою внаслідок атаки:

$$П_п = \frac{\sum Зс}{F} \cdot t_п,$$

де F – місячний фонд робочого часу (при 40-а годинному робочому тижні становить F = 176 годин).

Втрати від зниження продуктивності співробітників атакованого сегмента мережі:

$$П_п=(12000/176)*12=818,18 \text{ грн},$$

Витрати на відновлення працездатності сегмента мережі включають кілька складових:

$$П_в = П_{ві} + П_{пв} + П_{зч},$$

де $П_{ві}$ – витрати на повторне уведення інформації, грн.;

$П_{пв}$ – витрати на відновлення сегмента мережі, грн.;

$П_{зч}$ – вартість заміни устаткування або запасних частин, грн..

Витрати на відновлення працездатності сегмента мережі:

$$П_в = 477,27+409,09= 886,36 \text{ грн.},$$

Витрати на повторне введення інформації $П_{ві}$ розраховуються, виходячи з розміру заробітної плати співробітників атакованого сегмента мережі $З_с$, які зайняті повторним введенням втраченої інформації, з урахуванням необхідного для цього часу $t_{ві}$:

$$П_{ві} = \frac{\sum Зс}{F} \cdot t_{ві} \ .$$

Витрати на повторне введення інформації:

$$П_{ві}=(12000/176)*7=477,27 \text{ грн.}$$

Витрати на відновлення сегмента мережі $П_{пв}$ визначаються часом відновлення після атаки $t_в$ і розміром середньогодинної заробітної плати обслуговуючого персоналу:

$$П_{пв} = \frac{\sum Зо}{F} \cdot t_в \ .$$

Витрати на відновлення сегмента мережі:

$$П_{пв}=(9000/176)*8=409{,}09 \text{ грн.}$$

Втрати від зниження очікуваного обсягу продажів за час простою атакованого сегмента мережі визначаються виходячи із середньогодинного обсягу продажів типового підприємства і сумарного часу простою атаковано сегмента мережі:

$$V = \frac{O}{Fp} \cdot (t_п + t_в + t_{вi}) \ , \text{ грн.}$$

де $F_р$ – річний фонд часу роботи організації (52 робочих тижні, 5-ти денний робочий тиждень, 8-ми годинний робочий день) становить близько ($F_р$ = 2080 годин).

Втрати від зниження очікуваного обсягу продажів типового підприємства:

$$V=(320000/2080)*(12+8+7)=4153{,}85 \text{ грн.}$$

Таким чином, загальний збиток від атаки на вузол або сегмент корпоративної мережі організації складе:

$$B = \sum_i \sum_n U \ = 3*30*5858{,}39 = 527255{,}1 \text{ грн.}$$

3.2.2 Загальний ефект від впровадження системи інформаційної безпеки

Загальний ефект від впровадження системи інформаційної безпеки визначається з урахуванням ризиків порушення інформаційної безпеки і становить:

$$E = B \cdot R - C \text{ грн.},$$

де B – загальний збиток від атаки у разі перехоплення інформації, тис. грн.;

R – вірогідність успішної реалізації атаки на сегмент мережі, частки одиниці (40%);

С – щорічні витрати на експлуатацію системи інформаційної безпеки, тис. грн.

Загальний ефект від впровадження системи інформаційної безпеки визначається з урахуванням ризиків порушення інформаційної безпеки:

$$E = 527255,1*0,4 - 151920,91 = 58981,13 \text{ грн.}$$

3.3 Визначення та аналіз показників економічної ефективності системи інформаційної безпеки

Коефіцієнт повернення інвестицій ROSI показує, скільки гривень додаткового прибутку приносить одна гривня капітальних інвестицій на впровадження системи інформаційної безпеки:

$$ROSI = \frac{E}{K}, \quad \text{частки одиниці,}$$

де E – загальний ефект від впровадження системи інформаційної безпеки грн.;

К – капітальні інвестиції за варіантами, що забезпечили цей ефект, грн.

Коефіцієнт повернення інвестицій ROSI:

$$ROSI = \frac{58981,13}{8365,64} = 7,05, \quad \text{частки одиниці,}$$

Проект визнається економічно доцільним, якщо розрахункове значення коефіцієнта повернення інвестицій перевищує величину річної депозитної ставки з урахуванням інфляції:

$$ROSI > (N_{деп} - N_{інф})/100),$$

де $N_{деп}$ – річна депозитна ставка, (18 %);

$N_{інф}$ – річний рівень інфляції, (12%).

Розрахункове значення коефіцієнта повернення інвестицій:

$$7,05 > (18 - 12)/100 = 7,05 > 0,06.$$

Термін окупності капітальних інвестицій T_o показує, за скільки років капітальні інвестиції окупляться за рахунок загального ефекту від впровадження системи інформаційної безпеки:

$$T_O = \frac{K}{E} = \frac{1}{ROSI} = \frac{1}{7,05} = 0,14, \quad \text{років.}$$

3.4 Висновок

Результатом проведеної роботи в даному розділі є обґрунтування економічної доцільності вдосконалення організації захисту інформації на підприємстві по розробці програмних продуктів.

Розраховані поточні витрати на експлуатацію системи інформаційної безпеки становлять 151920,91 грн. Визначена величина економічного ефекту складає 58981,13 грн. Коефіцієнт повернення інвестицій складає 7,05 та швидкість повернення - 0,14 року.

Аналіз проведених розрахунків дозоляє зробити висновок про економічну доцільність вдосконалення організації захисту інформації на підприємстві по розробці програмних продуктів.

ВИСНОВКИ

Дипломні роботі розглянуті основні проблеми які пов’язані з проектуванням, розробкою, впровадженням та підтримкою програмного забезпечення. Наведені способи проектування захищеного (безпечного) програмного забезпечення. Зазначені основні методи тестування програмних продуктів з використання методів «білої», «сірої» та «чорної» скриньки. Зроблений аналіз нормативної бази в галузі оцінки якості програмних продуктів.

Наведені загрози несанкціонованого доступу до інформації визначені елементи моделі загроз експлуатаційної безпеки програмного забезпечення. Розглянуті проблеми пов’язані з управлінням персоналом на підприємствах галузі інформаційних технологій. Визначені особливості відбору кадрів для виконання поставлених задач при проектування, розробці, тестуванні та підтримки програмних продуктів. Наведена типова організаційна структура підприємства по розробці програмного забезпечення. Зазначені особливості колективного проектування, розробки та тестування програмного забезпечення та використання програмних рішень таки як система керування версіями.

ПЕРЛІК ПОСИЛАНЬ

1 Петюх В. М. Управління персоналом: Навч.-метод. посібник для самост. вивч. дисц. — К.: КНЕУ, 2000. — 124 с.

2 Open Security Group. (Електрон. ресурс) / Спосіб доступу: URL: http://www.open-security.org/bezopasnostj_po_ks/metody_sozdaniya_algoritmicheski_bezopasnyh_procedur. – Загол. з екрана.

3 Открытые системы (Електрон. ресурс) / Спосіб доступу: URL: http://www.osp.ru/os/2004/08/185088/. – Загол. з екрана.

4 G.E. McGraw. DIMACS Workshop on Software Security, IEEE Security & Privacy, vol. 1, no. 2, 2003.

5 N. Davis, J. Mullaney. The Team Software Process in Practice: A Summary of Recent Results, tech. report CMU/SEI-2003-TR-014, Software Eng. Inst., Carnegie Mellon Univ., September 2003.

6 A. Hall, R. Chapman. Correctness by Construction: Developing a Commercial Secure System. IEEE Software, vol. 19, no. 1, 2002.

7 R. Linger, S. Powell. Developing Secure Software with Cleanroom Software Engineering. Cybersecurity Summit Task Force Subgroup on Software Process, February 2004.

8 H. Mills, R. Linger. Cleanroom Software Engineering. Encyclopedia of Software Engineering, 2nd ed., J.Marciniak, ed., John Wiley & Sons, 2002.

9 S. Prowell et al. Cleanroom Software Engineering: Technology and Process, Addison Wesley, 1999.

10 J. Herbsleb et al. Benefits of CMM-Based Software Process Improvement: Initial Results, tech. report CMU/SEI-94-TR-013, Software Engineering Institute, Carnegie Mellon University, 1994.

11 D.R. Goldenson, D.L. Gibson. Demonstrating the Impact and Benefits of CMMI, special report CMU/SEI-2003-SR-009, Software Eng. Inst., Carnegie Mellon Univ., 2003.

12 J. Saltzer, M. Schroeder. The Protection of Information in Computer Systems. Proc. IEEE, vol. 63, no. 9, 1975.

13 Open Security Group. (Електрон. ресурс) / Спосіб доступу: URL: http://www.open-security.org/bezopasnostj_po_ks/yelementy_modeli_ugroz_yeksplu atacionnoj_bezopasnosti_po. – Загол. з екрана.

14 Свободная библиотека (Електрон. ресурс) / Спосіб доступу: URL: http://en.wikipedia.org/wiki/Revision_control. – Загол. з екрана.

15 Якість програмного забезпечення (Електрон. ресурс) / Спосіб доступу: URL: http://www.unicyb.kiev.ua/~boiko/it/5kp_.htm. – Загол. з екрана.

16 International Standards for Business, Government and Society (Електрон. ресурс) / Спосіб доступу: URL: http://www.iso.org/iso/home.html. – Загол. з екрана.

17 Тестування програмного забезпечення (Електрон. ресурс) / Спосіб доступу: URL: http://en.wikipedia.org/wiki/Software_testing. – Загол. з екрана.

ДОДАТОК А. Відомість матеріалів дипломного проекту

№	Формат	Найменування	Кількість листів	Примітка
1	А4	Реферат	3	
2	А4	Список умовних скорочень	1	
3	А4	Зміст	3	
4	А4	Вступ	2	
5	А4	1 Розділ	22	
6	А4	2 Розділ	36	
7	А4	3 Розділ	13	
8	А4	Висновки	1	
9	А4	Перелік посилань	2	
10	А4	Додаток А	1	
11	А4	Додаток Б	1	
12	А4	Додаток В	1	
13	А4	Додаток Г	1	

ДОДАТОК Б. Перелік документів на оптичному носії

1 Титульна сторінка.doc

2 Завдання.doc

3 Реферат.doc

4 Список умовних скорочень.doc

5 Зміст.doc

6 Вступ.doc

7 Розділ 1.doc

8 Розділ 2.doc

9 Розділ 3.doc

10 Висновки.doc

11 Перелік посилань.doc

12 Додаток А.doc

13 Додаток Б.doc

14 Додаток В.doc

15 Додаток Г.doc

Презентація.pptx

ДОДАТОК В. Відгуки керівників розділів

Відгук керівника економічного розділу:

Керівник розділу _____ _____

(підпис) (ініціали, прізвище)

ДОДАТОК Г. ВІДГУК

на дипломну роботу магістра на тему:
Вдосконалення організації захисту інформації на підприємстві по розробці програмних продуктів
студента групи 125м-17-2
Гезя Максима Віталійовича

Пояснювальна записка складається з титульного аркуша, завдання, реферату, списку умовних скорочень, змісту, вступу, трьох розділів, висновків, переліку посилань та додатків, розташованих на __ сторінках та містить __ рисунків, __ таблиць, 17 джерела та 4 додатка.

Актуальність теми полягає в необхідності вдосконалення організації захисту інформації на підприємстві по розробці програмних продуктів.

Зміст та структура дипломної роботи дозволяють розкрити поставлену тему повністю.

Студент показав добрий рівень володіння теоретичними положеннями з обраної теми, показав здатність формувати власну точку зору (теоретичну позицію).

Практична значущість полягає у розробці рекомендації щодо вдосконалення процедури проектування, розробки та впровадження захищеного програмного забезпечення і зниження ймовірності виникнення дефектів при розробці ПЗ.

Робота виконана самостійно. У роботі виконано аналіз загроз несанкціонованого доступу до інформації, визначені елементи моделі загроз експлуатаційної безпеки ПЗ. Розглянуті проблеми управління персоналом на підприємствах галузі інформаційних технологій. Визначені основні принципи забезпечення безпеки ПЗ. Наведена типова організаційна структура підприємства по розробці ПЗ. Зазначені особливості колективного проектування, розробки та тестування ПЗ та використання програмних рішень таких як система керування версіями.

Це підтверджує самостійність обробки даних, практичні рекомендації та висновки.

Робота оформлена та написана грамотною мовою. Містить необхідний ілюстрований матеріал. Автор добре знає проблему, уміє формулювати наукові та практичні завдання і знаходить адекватні засоби для їх вирішення.

В цілому дипломна робота задовольняє усім вимогам і може бути допущена до захисту, а його автор Гезь Максим Віталійович заслуговує на оцінку «_____».

Керівник дипломної роботи,
д.ф.-м.н., проф. Т.С. Кагадій

Керівник спец. част.
ст. викл. кафедри БІТ Г.М. Саксонов