Міністерство освіти І НАУКИ України

**Методичні вказівки до виконання лабораторних робіт з дисципліни**

**«Конструювання програмного забезпечення»**

для студентів спеціальності 121 «Інженерія програмного забезпечення»

**Мета дисципліни** - ознайомлення студентів із задачами, процесами, прийомами і засобами конструювання програмного забезпечення.

**Зміст дисципліни** Конструювання – етап життєвого циклу програмного забезпечення. Конструювання в аспекті детального проектування. Конструювання даних, програмних об’єктів та операторів. Конструювання підпрограм, модулів і класів. Захист програми та обробка помилок. Зрозумілість та документування вихідного коду. Якість конструювання. Організація результатів конструювання. Реінженерія при конструюванні програмного забезпечення. Інструменти для вирішення задач конструювання.

У даному методичному виданні висвітлюються основи конструювання програмного забезпечення. Наводяться дані щодо механізму відладки та тестування програм та якісного стилю програмування.

**Вступ**

Конструювання ПЗ – створення ПЗ з конструкцій (блоків, операторів, функцій) і його перевірка методами верифікації і тестування. До інструментів конструювання ПЗ віднесені мови конструювання, програмні методи й інструментальні системи (компілятори, СКБД, генератори звітів, системи керування версіями, конфігурацією, тестуванням й ін.). До формальних засобів опису процесу конструювання ПЗ, взаємозв'язків між людиною і комп'ютером з урахуванням середовища оточення віднесені структурні діаграми Джексона.

Область знань «Конструювання ПЗ (Software Construction)» містить у собі такі розділи:

– зниження складності (Reduction in Complexity),

– попередження відхилень від стилю (Anticipation of Diversity),

– структуризація перевірок (Structuring for Validation),

– використання стандартів (Use of External Standards).

Базову складову професійної діяльності фахівців в галузі програмної інженерії формують вміння та навички конструювання програмного забезпечення. До складу обов’язкового обсягу практичних навичок фахівця напряму «Програмна інженерія» повинні входити поняття про методи ефективного та оптимального кодування алгоритмів .

Програміст повинен генерувати не просто будь-який код, який працює, а і обов’язково володіти якісним стилем програмування, методами документування, застосовувати методи мінімізації коду, проводити ефективний пошук помилок, зокрема не явних на етапі відладки та вміти якісно тестувати власний програмний продукт.

**1. Основні теоретичні відомості конструювання програмного забезпечення.**

Термін КПЗ описує детальне створення робочої програмної системи за допомогою комбінації кодування, верифікації, модульного тестування, інтеграційного тестування і налагодження. Процес конструювання сильно пов'язаний з проектуванням і тестуванням тому конструювання відштовхується від результатів проектування, а тестування означає роботу з результатами конструювання. Досить складно визначити межу між цими складовими тому вони всі пов'язані в один комплекс життєвого циклу ПЗ.

У процесі розробки ПЗ виникають такі задачі, які пов'язані з конструюванням:перевірка виконання умов, необхідних для успішного конструювання:

* визначення способів подальшого тестування коду;
* проектування та написання класів та методів;
* створення та присвоєння імен змінних та іменованих констант;
* вибір управляючих структур та організації блоків команд;
* блочне тестування, інтеграційне тестування і відлагодження власного коду;
* взаємний огляд коду та низькорівневих програмних структур членами групи;
* "шліфування" коду шляхом його ретельного форматування та коментування;
* інтеграція програмних компонентів, створених окремо;
* оптимізація коду, яка направлена на підвищення його швидкодії і зниження міри використання ресурсів.

На рисунку 1.1 показано місце конструювання програмного забезпечення при побудові програмної системи.



Рис.1.1 Конструювання серед процесів побудови програмного забезпечення

Процеси конструювання зображені всередині сірого еліпсу. Головними компонентами конструювання є кодування та відлагодження, однак воно включає і детальне проектування, блочне тестування, інтеграційне тестування та інші процеси.

Іноді конструювання називають "кодуванням" або "програмуванням". Кодування в даному випадку видається не найкращим терміном, так як воно має на увазі механічну трансляцію розробленого плану в команди мови програмування, тоді як конструювання є зовсім не механічним процесом і часто пов’язане з творчістю та аналізом. Сенс слів "конструювання" та "програмування" досить близький.

Дана область знань пов'язана з іншими областями. Найбільш сильний зв'язок існує з проектуванням (Software Design) і тестуванням (Software Testing). Причиною цього є те, що сам по собі процес конструювання програмного забезпечення зачіпає важливі аспекти діяльності з проектування й тестування. Крім того, конструювання відштовхується від результатів проектування, а тестування (у будь-якій своїй формі) передбачає роботу з результатами конструювання. Хоча ряд операцій з проектування детального дизайну може відбуватися до стадії конструювання, великий обсяг такого роду проектних робіт відбувається паралельно з конструюванням або як його частина. Це є сутність зв'язку з областю знань "Проектування програмного забезпечення".

У свою чергу, протягом всієї діяльності з конструювання, інженери використовують модульне і інтеграційне тестування. Таким чином пов'язана дана галузь знань з "Тестуванням програмного забезпечення".

У процесі конструювання звичайно створюється більша частина активів програмного проекту - конфігураційних елементів (configuration items). Тому в реальних проектах просто неможливо розглядати діяльність по конструюванню у відриві від галузі знань "конфігураційного управління" (Software Configuration Management).

Так як конструювання неможливе без використання відповідного інструментарію і, ймовірно, дана діяльність є найбільш інструментально-насиченою, важливу роль у конструюванні грає область знань "Інструменти і методи програмної інженерії" (Software Engineering Tools and Methods).

Безумовно, питання забезпечення якості значимі для всіх галузей знань і етапів життєвого циклу. У той же час, код є основним результуючим елементом програмного проекту. Таким чином, явно напрошується і присутній зв'язок обговорюваних питань з областю знань "Якість програмного забезпечення" (Software Quality).

З пов'язаних дисциплін програмної інженерії (Related Disciplines of Software Engineering) найбільш тісний і природний зв'язок даної галузі знань існує з комп'ютерними науками (computer science). Саме в них, звичайно, розглядаються питання побудови та використання алгоритмів і практик кодування. Нарешті, конструювання стосується і управління проектами (project management), причому, в тій мірі, наскільки діяльність з управління конструюванням важлива для досягнення результатів конструювання.Рис.1.2. Область знань «Конструювання програмного забезпечення»

З іншого боку, з видів діяльності, що проходять в процесі розробки програмного забезпечення, до конструювання не відносяться: керування процесом розробки, виробки вимог, розробка високорівневої архітектури програми, проектування інтерфейсу користувача, тестування системи і її супроводження – для кожного з цих пунктів є своя наука.

Фундаментальні основи конструювання програмного забезпечення включають:

* Мінімізація складності
* Очікування змін
* Конструювання з можливістю перевірки
* Стандарти у конструюванні

Перші три концепції застосовуються не тільки до конструювання, але й проектування, і лежать в основі сучасних методологій управління життєвим циклом програмних систем.

Основною причиною того, чому люди використовують комп'ютери в бізнес-цілях, є обмежені можливості людей в обробці і зберіганні складних структур і великих обсягів інформації, зокрема, протягом тривалого періоду часу. Це міркування є однією з основних рушійних сил у конструюванні програмного забезпечення: мінімізація складності. Потреба у зменшенні складності впливає на всі аспекти конструювання і особливо критична для процесів верифікації (перевірки) і тестування результатів конструювання, тобто самих програмних систем.

Зменшення складності у конструюванні програмного забезпечення досягається за допомогою звертання особливої уваги на створення простого коду, який легко читається, іноді на шкоду прагненню зробити його ідеальним (наприклад, з точки зору гнучкості або слідування тим чи іншим уявленням про красу, витонченість коду, вправність тих чи інших прийомів тощо). Це не означає, що повинно обмежуватися застосування тих чи інших розвинених мовних можливостей використовуваних засобів програмування. Мається на увазі "лише" надання більшої значимості читаності коду, простоті тестування, прийнятного рівня продуктивності та задоволенню заданих критеріїв, замість постійного вдосконалення коду, не оглядаючись на терміни, функціональність і інші характеристики та обмеження проекту.

Мінімізація складності досягається, зокрема, за рахунок слідування стандартам, використанням низки специфічних технік кодування і підтримкою практик, спрямованих на забезпечення якості в конструюванні.

Більшість програмних систем змінюються з плином часу. Причин цьому безліч. Очікування змін є однією з рушійних сил конструювання програмного забезпечення. Програмне забезпечення не є ізольованим від зовнішнього оточення (як системного, так і з точки зору галузі діяльності, для автоматизації задач і проблем якого воно застосовується). Більш того, програмні системи є частиною середовища, що змінюється, і повинні змінюватися разом з нею, а, іноді, і бути джерелом змін самого середовища.

Очікування змін підтримується рядом технік кодування.

"Конструювання для перевірки припускає, що побудова програмних систем повинна вестися таким чином, щоб сама програмна система допомагала вести пошук причин збоїв, будучи прозорою для застосування різних методів перевірки (і, до речі, внесення необхідних змін), як на стадії незалежного тестування (наприклад, інженерами-тестувальниками), так і в процесі операційної діяльності - експлуатації, коли особливо важлива можливість швидкого виявлення та виправлення помилок що виникають.

Серед технік, спрямованих на досягнення такого результату конструювання:

* Огляд, оцінка коду (code review)
* Модульне тестування (unit-testing)
* Структурування коду для і спільно з застосуванням автоматизованих засобів тестування (automated testing)
* Обмежене застосування складних або важких для розуміння мовних структур.

Стандарти, які безпосередньо застосовуються при конструюванні, включають:

* Комунікаційні методи (наприклад, стандарти форматів документів і оформлення вмісту)
* Мови програмування і відповідні стилі кодування (наприклад, Java Language Specification, що є частиною стандартної документації JDK - Java Development Kit і Java Style Guide, що пропонує загальний стиль кодування для мови програмування Java)
* Платформи (наприклад, стандарти програмних інтерфейсів для викликів функцій операційного середовища, такі як прикладні програмні інтерфейси платформи Windows - Win32 API, Application Programming Interface або .NET Framework SDK, Software Development Kit)
* Інструменти (не в термінах середовищ розробки, але можливих засобів конструювання - наприклад, UML як один зі стандартів для визначення нотацій для діаграм, що представляють структура коду і його елементів або деяких аспектів поведінки коду).

Конструювання залежить від зовнішніх стандартів, пов'язаних з мовами програмування, використовуваним інструментальним забезпеченням, технічними інтерфейсами і взаємним впливом конструювання програмного забезпечення та інших галузей знань програмної інженерії (в тому числі, пов'язаних дисциплін, наприклад, управління проектами). Стандарти створюються різними органами, наприклад, консорціумом OMG - Object Management Group (зокрема. Стандарти CORBA, UML, MDA,), міжнародними організаціями з стандартизації такими, як ISO/IEC, IEEE, TMF, виробниками платформ, операційних середовищ і т.д. (Наприклад, Microsoft, Sun Microsystems, CISCO, NOKIA), виробниками інструментів, систем управління базами даних і т.п. (Borland, IBM, Microsoft, Sun, Oracle, Розуміння цього факту дозволяє визначити достатній і повний набір стандартів, які застосовуються у проектній команді або організації в цілому.

Кожна програмна система протягом свого існування проходить з певною послідовністю фази або стадії від задуму до його втілення в програми, експлуатацію та вилучення. Така послідовність фаз називається життєвим циклом розробки (Software life cycle processes). На кожній фазі відбувається певна сукупність процесів, кожен з яких породжує певний продукт, використовуючи певні ресурси.

Усі продукти всіх процесів програмної інженерії являють собою певні описи — тексти вимог до розробки, погодження домовленостей, документацію, тексти програм, інструкції з експлуатації тощо.

Головними ресурсами програмної інженерії, які визначають ефективність її розробок, є час і вартість цих розробок.

Різновиди діяльності, котрі становлять процеси життєвого циклу програмної системи, зафіксовано в міжнародному стандарті ISO/IEC 12207 : 1995—0801 : Informational Technology - Software life cycle processes.

Згідно з наведеним стандартом, усі процеси поділено на три групи:

* головні процеси;
* допоміжні процеси;
* організаційні процеси.

До головних процесів віднесено такі:

* процес придбання, який ініціює життєвий цикл системи та визначає організацію-покупця автоматизованої системи, програмної системи або сервісу;
* процес розроблення, який означає дії організації — розробника програмного продукту;
* процес постачання, який означає дії під час передавання розробленого продукту покупцеві;
* процес експлуатації, який означає дії з обслуговування системи під час її використання — консультації користувачів, вивчення їхніх побажань тощо;
* процес супроводження, який означає дії з керування модифікаціями, підтримки актуального стану та функціональної придатності, інсталяцію та вилучення версій програмних систем у користувача.

У свою чергу, до процесу розроблення входять такі процеси:

* інженерія вимог до системи;
* проектування;
* кодування й тестування.

До допоміжних процесів віднесено ті, що так чи інакше забезпечують якість продукту. Терміном якість продукту позначено сукупність властивостей, які зумовлюють можливість задоволення потреб замовника, котрий сформулював їх у формі вимог на розробку.

До організаційних процесів віднесено менеджмент розробки, створення структури організації, навчання персоналу, визначення відповідальності кожного з учасників процесів життєвого циклу розробки.

Стандарт ISO/IEC 12207:1995 - 0801: Informational Technology - Software life cycle processes є головним чинником визначення змісту діяльності у сфері програмної інженерії, і всі знання, яких потребують професіонали з програмної інженерії, формулюються стосовно процесів, визначених цим стандартом.

Зупинимося докладніше на процесах розробки програмного забезпечення, які в сукупності мають забезпечити шлях від усвідомлення потреб замовника до передавання йому готового продукту. На цьому шляху виділяють низку характерних робіт:

* Визначення вимог. Збір та аналіз вимог замовника виконавцем і подання їх у нотації, яка є зрозумілою як для замовника, так і для виконавця.
* Проектування. Перетворення вимог до розробки в послідовність проектних рішень щодо способів реалізації вимог: формування загальної архітектури програмної системи та принципів її прив’язки до конкретного середовища функціонування; визначення детального складу модулів кожної з архітектурних компонент.
* Реалізація. Перетворення проектних рішень на програмну систему, яка реалізує такі рішення.
* Тестування. Перевірка кожного з модулів та способів їхньої інтеграції; тестування програмного продукту в цілому (так звана верифікація); тестування відповідності функцій працюючої програмної системи вимогам (requirements), поставленим до неї замовником (так звана валідація).
* Експлуатація та супроводження готової програмної системи.

Базовими вітчизняними нормативними документами в галузі програмної інженерії є:

* ДСТУ 2873-94. Системи обробки інформації. Програмування. Терміни та визначення.
* ДСТУ 2941-94. Системи оброблення інформації. Розроблення систем. Терміни та визначення.
* ДСТУ 4302:2004. Інформаційні технології. Настанови щодо документування комп’ютерних програм.
* ДСТУ ISO/IEC 12119:2003. Інформаційні технології. Пакети програм тестування і вимоги до якості.
* ДСТУ ISO/IEC 14764:2002. Інформаційні технології. Супроводження програмного забезпечення.
* ДСТУ ISO/IEC 90003:2006. Програмна інженерія. Настанови щодо застосування ІSO 9001:2000 до програмного забезпечення (ІSO/ІЕС 90003:2004, IDT)
* ДСТУ ISO/IEC TR 12182:2004. Інформаційні технології. Класифікація програмних засобів (ISO/IEC TR 12182:1998, IDT)
* ДСТУ ISO/IEC 14598-1:2004. Інформаційні технології. Оцінювання програмного продукту. Частина 1. Загальний огляд (ISO/IEC 14598-1:1999, IDT)
* ДСТУ ISO/IEC 15288:2005. Інформаційні технології. Процеси життєвого циклу системи (ISO/IEC 15288:2002, IDT)
* ДСТУ ISO/IEC 15939:2008. Інженерія систем і програмних засобів. Процес вимірювання.
* ДСТУ 3327-96. Методика випробування процесорів мов програмування. Загальні вимоги.
* ДСТУ ISO/IEC TR 14369:2003. Інформаційні технології. Мови програмування, їхнє середовище та системний інтерфейс. Настанова щодо підготовки незалежних від мов специфікацій послуг.
* ДСТУ 4072:2001. Інформаційні технології. Мови програмування, їхнє середовище та системний інтерфейс. Настанова щодо підготовки незалежних від мов виклик процедур.
* ДСТУ ISO/IEC 2382-15:2005. Інформаційні технології. Словник термінів. Частина 15. Мови програмування (ISO/IEC 2382-15:1999, IDT)
* дсту 3008-95. "Документація. Звіти у сфері науки і техніки Структура і правила оформлення". К.: Держстандарт України,1995. – 75 с.
* ГОСТ 2.106-96. Единая система конструкторской документации. Текстовые документы. Изд. Офиц – К.: Госстандарт Украины, 1998. – 47 с.
* гост 2.109-73 ЕСКД. Основные требования к чертежам – М., 1978.
* ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам. Изд. Офиц – К.: Госстандарт Украины, 1996.
* Гост 7.32-91. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления

Певні стандарти, угоди та процедури можуть бути також створені усередині організації або навіть команди, що працює над проектом. Ці стандарти підтримують координацію між певними видами діяльності, групами операцій, мінімізують складність (у тому числі при взаємодії членів команди та за її межами), можуть бути пов'язані з питаннями очікування та обробки змін, ризиків і питаннями конструювання для перевірки та подальшого тестування. У поєднанні із зовнішніми стандартами, внутрішні стандарти покликані визначити загальні правила гри для всіх членів команди, домовившись про терміни, процедури та інші значущі угоди, незалежно від ступеня формалізації процесів конструювання зокрема і процесів життєвого циклу в загальному випадку.

**Лабораторна робота №1**

**Використання моделей проектування програмного забезпечення**

**Мета роботи**: використання навичок вибору моделі проектування для конкретного проекту.

**Завдання:**

1. На основі завдання на розробку програмного забезпечення, зробити та обґрунтувати вибір моделі життєвого циклу проекту програмного забезпечення.
2. Скласти загальний календарний графік виконання етапів проекту ПЗ.
3. Розробити та обґрунтувати вимоги до ПЗ, що розробляється.
4. Створити статут проекту згідно темою роботи.

**Теоретичні відомості:**

Моделі конструювання визначають комплекс операцій, які включають послідовність, результати та інші аспекти, пов'язані із загальним життєвим циклом розробки. У більшості випадків, моделі конструювання визначаються використовуваним стандартом життєвого циклу, застосовуваними методологіями та практиками.

Найбільш часто говорять про моделі життєвого циклу в різних джерелах з різним інтерпретацією, тому, узагальнюючи дані з різних джерел можна виділити такі 3 основні моделі: каскадна (водопадна або послідовна); ітеративна (інкрементального, еволюційна, гібридні або змішана); спіральна (модель Боема).

Розробка програмного забезпечення – це складний процес, в який входить багато складових. В загальному випадку це:

* визначення проблеми;
* вироблення вимог;
* створення плану конструювання;
* розробка архітектури ПЗ, або високорівневе проектування;
* детальне проектування;
* кодування і відлагодження;
* блочне тестування;
* інтеграційне тестування;
* інтеграція;
* тестування системи;
* корегувальне супроводження

При вирішенні конкретної задачі в процесі розробки ПЗ необхідно:

1. Перевірити виконання всіх вимог

2. Визначити способи тестування

3. Акуратно підійти до створення імен змінних і констант

4. Після проведення блочного тестування й налагодження власного коду провести огляд коду всіма членами розробки низькорівневих програмних структур

5. Інтегрувати програмні компоненти та оптимізувати код для підвищення швидкодії і зниження використання ресурсів.

При конструюванні програмного забезпечення необхідно мінімізувати складність; передбачити очікувані зміни; конструювати з можливістю перевірки; використовувати стандарти при конструюванні (включають в себе: комунікаційні методи (стандарти форматів документів); мови програмування і стилі кодування; платформи; інструменти).

Для своєчасного завершення розробки ПЗ проект розбивають на кілька етапів і для цих етапів складають загальний календарний графік.

При розробці уставу проекту необхідно вказати:

* бізнес - причини виникнення проекту;
* цілі проекту;
* вимоги до проекту;
* описати обрану модель життєвого циклу;
* фінансові показники проекту;
* технічні вимоги на продукт;
* WBS-структуру проекту;

**Лабораторна робота №2**

**Планування конструювання програмного забезпечення**

**Мета роботи**: засвоєння навичок планування розробки проектів програмного забезпечення

**Завдання:**

1. Розробити перелік робіт, що потрібно виконати для здійснення плану і задовільнення вимог, що були розроблені у попередній ЛР, та визначити наявні ресурси.

2. Виконати деталізацію плану та здійснити розподіл ресурсів для виконання робіт у встановлені терміни.

**Теоретичні відомості:**

Планування – вид діяльності, пов’язаний з постановкою цілей (задач) і дій в майбутньому. Планування, в загальному вигляді, має на увазі виконання наступних етапів:

* Ідентифікація цілей і задач
* Складання програми дій
* Виявлення необхідних ресурсів і їх джерел
* Визначення безпосередніх виконавців і доведення плану до них.

Планування – оптимальне розподілення ресурсів для досягнення поставленої мети. При плануванні використовуються калібровані моделі очікувань, які засновані на зусиллях пов'язаних з обсягом завдання. Ця модель часто визначається як людино-місяці до функціональних точках або кількістю рядків коду. Можуть застосовуватися й інші моделі такі як експертна оцінка, крім того на цьому етапі необхідно ідентифікувати зв'язки і залежності між завданнями. Такі роботи проводяться часто з використанням методу аналізу критичного шляху.

Для задач на цьому етапі визначено очікувані терміни виконання (розклад) включаючи початок, тривалість і закінчення для цього використовуються часто PERT-діаграми. PERT -аналіз - це техніка оцінки очікувань щодо тривалості завдань проекту, яка проводиться на основі визначення середнього вагового значення 3-х оцінок тривалості: оптимістична, очікувана, песимістична. Ця техніка часто застосовується для оцінки зусиль, які необхідні для реалізації завдань. Часто використовують декілька різних методів оцінки. Вимоги до людських ресурсах і інструментах аналізуються з точки зору вартісних очікувань. Цей етап проводитиметься менеджером проекту і командою.

З точки зору розподілу ресурсів розглядаються завдання, для яких призначені терміни і найчастіше вони пов'язані з обладнанням і людьми. На цьому етапі планування зручно використовувати діаграми Ганта, вони визначають доступність ресурсів та їх оптимальне використання при вирішенні даної задачі.

Управління персоналом часто виділяється в окремо виділяється завдання, іноді необхідно залучати зовнішніх консультантів. При плануванні ризиків необхідно:

* ідентифікувати ризики
* оцінити ризики
* пом'якшити ризики.

Для цього застосовуються відповідні методи: побудова дерева рішень або моделювання процесів. На цьому ж етапі визначаються правила і політика припинення проекту. При плануванні ризиків слід враховувати і унікальні ризики в області програмної інженерії. При складанні плану проекту, необхідно визначити як проект буде управлятися і його план (звітність, моніторинг і контроль). На етапі планування розглядається питання управління контрактами з постачальниками.

При виконанні планування КПЗ необхідно:

1. Виправити і затвердити вимоги
2. Створити список робіт з описом модулів ПЗ
3. Визначити необхідні ресурси людські та технічні
4. Зробити деталізація плану виконання робіт. ( Для графічногопредставлення деталізації плану робіт збудувати діаграму Ганта).

**Лабораторна робота №3**

**Використання мов конструювання програмного забезпечення**

**Мета роботи**: Засвоення навичок використання відповідних мов конструювання програмного забезпечення

Завдання:

1. Виконати аналіз вимог з метою встановлення відповідних методів та

засобів їх реалізації;

2. Виконати проектування моделей реалізації вимог;

3. Провести розробку програмного забезпечення.

**Теоретичні відомості:**

Мови конструювання включають усі форми комунікацій, за допомогою яких людина може задати рішення проблеми, виконуване на комп’ютері. Як відомо, можна виділити такі типи мов конструювання:

* Конфігураційні мови, що дозволяють задавати параметри виконання програмної системи;

Наприклад: 1С:Предприятие; AutoCad: MathCad:EMACS:Latex

* Інструментальні мови – мова конструювання з елементів. Що використовуються повторно; звичайно будується як сценарна мова (скрипт), виконувана у відповідному середовищі.

Наприклад : C; FORTRAN; PHP; PERL

* Алгоритмічні

Наприклад: С++: Алгол; Лисп: Пролог; JAVA; C#: Pascal

З'являються більш сучасні методи розробки ПЗ, які дозволяють змінювати код в процесі конструювання.

Невід’ємною частиною конструювання програмного забезпечення є аналіз вимог, необхідних для встановлення методів і засобів їх реалізації.

Зробити проектування моделей реалізації вимог. Реалізацію вимог представити у вигляді діаграми зі встановленими класами, методами і зв’язками між ними. Описати атрибути сутностей та їх зв’язків. Розробити програмне забезпечення.

**Лабораторна робота №4**

**Забезпечення якості конструювання програмного забезпечення**

**Мета роботи**: завсвоєння навичок забезпечення якості конструювання програмного забезпечення, таких як його тестування та відлагодження.

Завдання:

1. Обрати та обгрунтувати методику тестування розробленого ПЗ.
2. Розробити пакети тестів для тестування розробленого ПЗ.
3. Виконати тестування розробленого ПЗ.
4. За результатами тестування зробити висновки щодо якості розробленого ПЗ та в разі необхідності виправити знайдені недоліки ПЗ.

**Теоретичні відомості:**

Якість ПЗ визначається при відсутності в ньому дефектів. Дефектами можна вважати властивості і особливості ПЗ, які роблять майже неможливим отримання від нього заявленої функціональності. Це можуть бути неточності у визначенні вимог або помилки проектування.

Тестування програмного забезпечення – процес дослідження програмного забезпечення з метою отримання інформації про якість продукту. Всі існуючі на нинішній день методики тестування ПЗ не дозволяють цілком і повністю виявити усі дефекти, а також встановити коректність функціонування компонентів ПЗ. Через це усі існуючі методики тестування ПЗ являються засобом, який дозволяє формально встановити показник рівня якості розробленого ПЗ.

З точки зору стандарту ІSО 9126 якості програмних засобів можна визначити як сукупну характеристику досліджуваного ПЗ з урахуванням таких сладових:

* надійність
* сопровождаемість
* практичність
* ефективність
* мобільність
* функціональність

Тестувальники використовують тестові скрипти на різних рівнях, їх пишуть для перевірки компонентів, в яких можливі помилки.

На тестування рекомендується виділяти від 8% до 25% часу, який йде на розробку системи. Часто тестоване програмне забезпечення збирається зі спеціальними настройками або бібліотеками і запускається в спеціальному оточенні - процес «покриття коду», практично це тестування білого ящику . Останнім часом існують різні методики тестування, які включають в себе кілька методологій, найчастіше такими методами є ТDD . Воно базується на двох методологіях: JUNIT-тестуванні і рефакторингу. Це ітеративний процес, кроки повторюються до тих пір поки не досягнуто потрібної якості коду. ТDD дає переваги:спрощує розробку, дає можливість постійно використовувати регресіонний тестування покращує комунікацію, пояснює вимоги до програм і покращує дизайн, зменшує складність за рахунок невнесення зайвого коду. JUNIT-тестування є тестовим для JAVA і використовує відповідні анотації. Попереднє написання тестів (до коду) дозволяє звести мінімум до часу між виявленням і виправленням дефектів, це одна з методик ефективної розробки ПЗ.

Існує декілька ознак по яким прийнято проводити класифікацію видів тестування, звичайно виділяють такі:

**По об'єкту тестування:**

* функціональні тестування
* тестування продуктивності
* стрес-тестування
* тестування стабільності
* тестування зручності використання
* тестування інтерфейсу користувача
* тестування безпеки
* тестування локалізації
* тестування сумісності.

**По знанню системи:**

* Тестування чорного ящику
* Тестування білого ящику
* Тестування сірого ящику

**За ступенем автоматизації:**

* ручне тестування
* автоматизоване
* напівавтоматизоване.

**За ступенем ізольованості компонент:**

* модульне тестування
* інтеграційне тестування
* системне тестування

**За часом проведення тестування**

**За ознакою позитивності сценаріїв**

**За ступенем підговори до тестування**

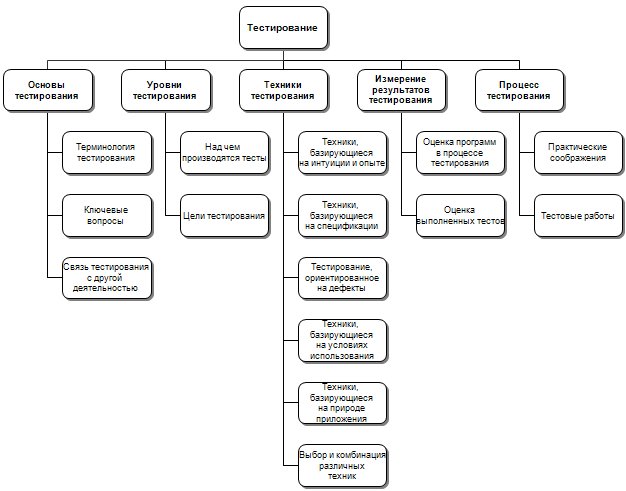
Тестування — це одна з технік контролю якості, що включає в себе:

* Планування робіт (Test Management)
* Проектування тестів (Test Design)
* Виконання тестування (Test Execution)
* Аналіз отриманих результатів (Test Analysis).

Існують такі рівні тестування:

* Приймальне тестування
* Установче тестування
* Модульне тестування
* Інтеграційне тестування
* Системне тестування
* Альфа-тестування
* Бета-тестування
* Тестування продуктивності
* Тестування «білого ящика» і «чорного ящика»
* Статичне і динамічне тестування
* Регресійне тестування

На рисунку 3. Зображено схему тестування програмного забезпечення.

Рисунок 3 – Тестування програмного забезпечення

ПЗ для полегшення роботи з мінливою інформацією називають системами управління версіями. Системи керування версіями дозволяють зберігати декілька версій одного і того ж документа, при необхідності дозволяють звертатися до більш ранніх версій і визначати хто і що зробив, в який час і багато іншого. Такі системи широко застосовуються при розробці ПЗ, зберігання початкового коду. Існують кілька пакетів, які дозволяють керувати версіями (CVS, SVN,Subversion, Git, Mercurial, MS SS, MS TFS, MAKE, GMAKE, CMAKE, SCons, QMAKE).

**Лабораторна робота №5**

**Застосування шаблонів проектування**

**Мета роботи**: засвоєння навичок використання шаблонів проектування для прискорення і підвищення надійності конструювання програмного забезпечення

Завдання:

1. Запропонувати та обгрунтувати 2-3 шаблони проектування для реалізації компонентів ПЗ, що розробляється.
2. Реалізувати зазначені компоненти ПЗ з використанням обраних шаблонів проектування.
3. Прокоментувати переваги і, якщо такі є, недоліки використаних шаблонів проектування для зазначених проектних рішень.

**Теоретичні відомості:**

В розробці ПЗ, шаблон проектування або паттерн – повторювана архітектурна конструкція, що представляє собою рішення проблеми проектування в рамках деякого контексту.

Зазвичай, шаблон не є кінцевим зразком, який може бути відразу перетвореним в код; це лише приклад вирішення задачі, який можна використати в різних ситуаціях. Об’єктно-орієнтовані шаблони показують відношення і взаємодію між класами або об’єктами, без визначення того, які кінцеві класи або об’єкти додатку будуть використовуватися.

«Низькорівневі» шаблони, що враховують специфіку конкретної мови програмування називаються ідіомами. Це хороші рішення проектування, характерні для певної мови або програмної платформи, і тому не універсальні.

На найвищому рівні беруть участь архітектурні шаблони, вони включають архітектуру усієї програмної системи.

Алгоритми по своїй суті також являються шаблонами, але не проектування, а розрахунку, саме через те, що вирішують різноманітні задачі. Головна користь кожного окремого шаблона полягає в тому, що він описує рішення цілого класу абстрактних проблем. Також той факт, що кожен шаблон має своє ім’я, полегшує дискусію про абстрактні структури даних (ADT) між розробниками. Так як вони можуть посилатися на відомі шаблони. Таким чином, за рахунок шаблонів виконується уніфікація термінології, назв модулів та елементів проекту. Правильно сформульований шаблон проектування дозволяє, відшукавши вдале рішення, користуватися ним знову і знову. Але вони і можуть пропагувати і погані стилі розробки додатків. Для подолання цих недоліків використовується рефакторинг.

Типи шаблонів проектування:

1. Твірні шаблони – шаблони проектування, які абстрагують процес інстанціювання. Вони дозволяють зробити систему незалежною від способу створення, композиції і представлення об’єктів. Шаблон, породжуючий класи, використовує наслідування щоб змінювати інстанційований клас, а шаблон, що породжує об’єкти, делегує інстанціювання іншому об’єкту.
2. Структурні шаблони – визначають різноманітні складні структури, які міняють інтерфейс вже існуючих об’єктів або його реалізацію, дозволяючи полегшити розробку і оптимізувати програму.
3. Поведінкові шаблони – визначають взаємодію між об’єктами, підвищуючи таким чином його гнучкість.
4. Шаблони паралельного програмування – використовується для більш ефективного написання багато поточних програм, і надає готові рішення проблем синхронізації.

Також, станом на нинішній день, існує ряд інших шаблонів:

* Carrier Rider Mapper описують надання доступу до інформації що зберігається
* Аналітичні шаблони описують основний підхід для створення вимог для програмного забезпечення до початку самого процесу програмної розробки
* Комунікаційні шаблони описують процес спілкування між окремими учасниками/співробітниками організації
* Організаційні шаблони описують організаційну ієрархію підприємства/фірми
* Анті-паттерни описують, як не слід робити при розробці програм, показуючи характерні помилки в дизайні і в реалізації.

Антіпаттерни - класи, найбільш часто впроваджуваних поганих рішень проблеми, вони використовуються для розпізнання при вивченні непрацюючої системи. Вони діляться на групи: управління розробкою ПЗ; розробки ПЗ; в ООП; в програмуванні; методологічні; управління конфігурацією; організаційні; соціальні; «жартівливі».

**Лабораторна робота №6**

**Документування результатів розробки**

**проекту програмного забезпечення**

**Мета роботи**:

Завдання:

1. Навести в роботі склад усіх модулів, що були розроблені у межах проекту програмного забезпечення.
2. Щодо кожного модулю, навести склад програмних компонент, що увійшли до модулю.
3. Навести програмний код, що був розроблений для реалізації функціональних можливостей кожного модулю.

**Теоретичні відомості:**

Документація вимагає значних ресурсів для її застосування. Управління документацією повинне безупинно підтримувати її повноту, коректність та узгодженість з програмним продуктом. Помилки і дефекти документів не менш небезпечні для застосування програмних продуктів чим помилки в структурах, інтерфейсах, файлах, текстів програм.

Методи і засоби документування кожної процедури зазвичай не розкриваються і адресуються до спеціальними нормативними документами різного рівня. Для цих етапів створюються нормативні документи - шаблони. Сукупні витрати на документування програмного продукту можуть досягати 20-30% від загальної трудомісткості. Якщо продукт не дуже складний, то витрати на документування знижуються до 10%.

За своїм призначенням та орієнтації на певні групи користувачів документацію можна розділити на:

* - технологічну
* - експлуатаційну

При складанні документації, обов'язково складається план проекту ПЗ. Дуже часто використовуються для управління документами програмного продукту спеціальна автоматизована БД, яка включає в себе весь комплекс обміну інформацією і документами між учасниками проекту розробки ПЗ.

Необхідно складати специфікацію вимог до документації таким чином, щоб первинний перелік набору документів повністю відповідав концепції вимог і призначенням ПО. Ці вимоги до документів полягають в контракті і технічному завданні.

Після проведення моніторингу масштабу проекту, формуються вимоги до комплекту документів, специфікацій вимог, це може бути обов'язкове робоче проектування.

Для управління документацією складного програмного продукту при детальному проектуванні, треба враховувати ступінь корисності різнорідних документів, з метою узагальнення їх характеристик.

Керівництво процесом документування можна розділити на рівні:

* Адаптація складу і змісту документів до даної ділової проблемно-орієнтованої області
* Адаптація номенклатури, структури для кожного проекту.

Номенклатура документів для крупного програмного продукту може доходити до 50 видів, на обсяг впливають тексти програм з коментарями, тестові сценарії та результати тестування модулів, специфікації програми даних. Як правило, на кожен рядок тексту програми може вимагатися 10% від повної сторінки документації. Більша частина документів залишається у вигляді файлів, але кожен такий документ повинен бути оформлений відповідно до стандартів і шаблонами та скріплений підписами замовників і розробників.

При складанні графіка виконання робіт в життєвому циклі ПЗ, потрібно враховувати графіки створення окремих документів, які мають бути зв'язані і узгоджені з цим базовим графіком. Дуже часто в реальних проектах доводитися відхилятися від лінійної схеми документообігу: Переривається цей процес після проміжного етапу проектування - йде повернення на попередню системи документообігу.

До составу базової версії ПЗ входить підсистема документообігу та комплект документів для користувачів.

**Завдання до лабораторних робіт**

|  |  |
| --- | --- |
| **Варіант** | **Тема** |
| 1 | Банкомат-видача готівки |
| 2 | Касовий апарат-оплата товарів відповідно до їх бар-кодами |
| 3 | Апарат з продажу напоїв: кава або інші напої |
| 4 | Склад-прийом та відвантаження товарів будь-якого виду |
| 5 | Авіакаса-продаж квитків |
| 6 | ЖД каса-продаж квитків |
| 7 | Автобусна каса-продаж квитків |
| 8 | Прохідна підприємства-облік робочого часу |
| 9 | Облік обладнання в сервіс-центрі |
| 10 | Відділ кадрів невеликого підприємства-облік кадрів складу |
| 11 | Невеликий електронний магазин-продаж товарів |
| 12 | Склад будівельних матеріалів |
| 13 | Магазин аудіо-відеозаписів |
| 14 | Термінал оплати послуг |
| 15 | Морські вантажоперевезення |
| 16 | Електронна бібліотека |
| 17 | Термінал продажу білетів |
| 18 | Обік у готелі |
| 19 | Каса кінотеатру |
| 20 | Обслуговування АЗС |
| 21 | Музикальний автомат |
| 22 | Система обліку пацієнтів у лікарні |

**Контрольні питання**

1) Що таке життевий цикл програмного забезпечення.

2) Які різновиди моделей життевого циклу програмного забезпечення Вам відомі.

3) Основні характеристики і фази водоспадної моделі.

4) Основні характеристики і фази ітеративної моделі.

5) Основні характеристики і фази спіральної моделі.

6) Основні аспекти планування проекту програмного забезпечення.

7) Основні характеристики планування процесу розробки.

8) Основні характеристики визначення результатів проекту.

9) Основні характеристики оцінки зусиль, розкладу та вартісних очикувань проекту.

10)Основні характеристики розподілу ресурсів проекту.

11) Яким чином визначається критичний шлях проекту.

12)Основні характеристики керування ризиками проекту.

13)Основні типи мов конструювання програмних проектів.

14)Основні можливості конфігураційних мов конструювання.

15)Основні можливості інструментальних мов конструювання.

16)Основні можливості мов програмування.

17)Назвіть методи інтеграції.

18)Назвіть основні характеристики методів інтеграції.

19)Назвіть основні засоби безперервної інтеграції.

20)Що таке побудова, перевірка, розгортання?

21)Що таке зворотній зв'язок?

22)Цілі тестування програмного забезпечення.

23)Цілі тестування програмного забезпечення.

24)Охарактеризуйте техніки що базуються на інтуїції та досвіді інженера.

25)Охарактеризуйтетехніки що базуються на специфікаціях.

26)Охарактеризуйте техніки що орієнтовані на код.

27)Охарактеризуйте техніки що орієнтовані на дефекті.

28)Охарактеризуйте техніки що базуються на умовах використання.

29)Охарактеризуйте техніки що базуються на природі застосування.

30)Методи оцінки результатів тестування.

31)Перелічить основні техніки відлагодження програмного забезпечення.

32)Які інструменти відлагодження Вам відомі та які в них переваги та недоліки.

33)Основні типи шаблонів проектування та їх призначення.

34)Фундаментальні шаблони, їх склад та призначення.

35)Опишіть можливості шаблону Незмінний об'єкт.

36)Опишіть можливості шаблону Інтерфейс.

37)Породжуючі шаблони, їх склад та призначення.

38)Опишіть можливості шаблону Абстрактна фабрика

39)Опишіть можливості шаблону Будівник.

40)Опишіть можливості шаблону Фабричний метод.

41)Опишіть можливості шаблону Клас-одинак.

42)Опишіть можливості шаблону Відстрочена іниціалізація.

43)Опишіть можливості шаблону Прототип.

44)Структурні шаблони, їх склад та призначення.

45)Опишіть можливості шаблону Адаптер.

46)Опишіть можливості шаблону Міст.

47)Опишіть можливості шаблону Компоновник.

48)Опишіть можливості шаблону Декоратор.

49)Опишіть можливості шаблону Фасад.

50)Опишіть можливості шаблону Пристосуванець.

51)Опишіть можливості шаблону Заступник.

52)Поведінкові шаблони, їх склад та призначення.

53)Опишіть можливості шаблону Команда.

54)Опишіть можливості шаблону Інтерпретатор.

55)Опишіть можливості шаблону Ітератор.

56)Опишіть можливості шаблону Посередник.

57)Опишіть можливості шаблону Наглядач.

58)Опишіть можливості шаблону Стан.

**Список використанних джерел**

1. ДСТУ 2873-94. Системи обробки інформації. Програмування. Терміни та визначення. - К.: Держстандарт України, 1994.
2. ДСТУ 2941-94. Системи оброблення інформації. Розроблення систем. Терміни та визначення. - К.: Держстандарт України, 1994.
3. ДСТУ 4302:2004. Інформаційні технології. Настанови щодо документування комп’ютерних програм. - К.: Держстандарт України, 2004.
4. ДСТУ ISO/IEC 12119:2003. Інформаційні технології. Пакети програм тестування і вимоги до якості. - К.: Держстандарт України, 2003.
5. ДСТУ ISO/IEC 14764:2002. Інформаційні технології. Супроводження програмного забезпечення. - К.: Держстандарт України, 2002.
6. ДСТУ ISO/IEC 90003:2006. Програмна інженерія. Настанови щодо застосування ІSO 9001:2000 до програмного забезпечення (ІSO/ІЕС 90003:2004, IDT) - К.: Держстандарт України, 2006.
7. ДСТУ ISO/IEC TR 12182:2004. Інформаційні технології. Класифікація програмних засобів (ISO/IEC TR 12182:1998, IDT) - К.: Держстандарт України, 2004.
8. ДСТУ ISO/IEC 14598-1:2004. Інформаційні технології. Оцінювання програмного продукту. Частина 1. Загальний огляд (ISO/IEC 14598-1:1999, IDT) - К.: Держстандарт України, 2004.
9. ДСТУ ISO/IEC 15288:2005. Інформаційні технології. Процеси життєвого циклу системи (ISO/IEC 15288:2002, IDT) - К.: Держстандарт України, 2005.
10. ДСТУ ISO/IEC 15939:2008. Інженерія систем і програмних засобів. Процес вимірювання. - К.: Держстандарт України, 2008.
11. ДСТУ 3327-96. Методика випробування процесорів мов програмування. Загальні вимоги. - К.: Держстандарт України, 1996.
12. ДСТУ ISO/IEC TR 14369:2003. Інформаційні технології. Мови програмування, їхнє середовище та системний інтерфейс. Настанова щодо підготовки незалежних від мов специфікацій послуг. - К.: Держстандарт України, 2003.
13. ДСТУ 4072:2001. Інформаційні технології. Мови програмування, їхнє середовище та системний інтерфейс. Настанова щодо підготовки незалежних від мов виклик процедур. - К.: Держстандарт України, 2001.
14. ДСТУ ISO/IEC 2382-15:2005. Інформаційні технології. Словник термінів. Частина 15. Мови програмування (ISO/IEC 2382-15:1999, IDT) - К.: Держстандарт України, 2005.
15. дсту 3008-95. "Документація. Звіти у сфері науки і техніки Структура і правила оформлення". К.: Держстандарт України,1995. – 75 с.
16. ГОСТ 2.106-96. Единая система конструкторской документации. Текстовые документы. Изд. Офиц – К.: Госстандарт Украины, 1998. – 47 с.
17. гост 2.109-73 ЕСКД. Основные требования к чертежам – М., 1978.
18. ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам. Изд. Офиц – К.: Госстандарт Украины, 1996.
19. ДСТУ Гост 7.1:2006. Система стандартів з інформації, бібліотечної та видавничої справі. Загальні вимоги та правіла складання. - К.: Держстандарт України, 2007. – 47 с.
20. ДСТУ Гост 2.104:2006. ЕСКД. Основні написи. - К.: Держстандарт України, 2006.
21. Інформатика: Комп’ютерна техніка. Комп’ютерні технології. Посіб. / За ред. О.І.Пушкаря – К: Видавничий центр “Академія”, 2001. – 696с. (Альма-матер)
22. Макконнелл С. Совершенный код. Мастер-класс / Пер. с англ.- М.: Издательско-торговый дом "Русская Редакция"; СПб.: Питер, 2005.- 896 стр.: ил.
23. Основы Программной Инженерии (по SWEBOK). 3. Конструирование программного обеспечения.

<http://swebok.sorlik.ru/3_software_construction.html>.

1. Bohm, Corrado; and Giuseppe Jacopini (May 1966). "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules". Communications of the ACM 9 (5): 366–371. [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1145/355592.365646](http://dx.doi.org/10.1145%2F355592.365646)
2. Dijkstra, E. W. (Aug 1972). ["The Humble Programmer"](http://www.cs.utexas.edu/%7EEWD/transcriptions/EWD03xx/EWD340.html). [Communications of the ACM](http://en.wikipedia.org/wiki/Communications_of_the_ACM) 15 (10): 859–866. [doi](http://en.wikipedia.org/wiki/Digital_object_identifier" \o "Digital object identifier):[10.1145/355604.361591](http://dx.doi.org/10.1145%2F355604.361591). [http://www.cs.utexas.edu/~EWD/transcriptions/EWD03xx/EWD340.html](http://www.cs.utexas.edu/%7EEWD/transcriptions/EWD03xx/EWD340.html).  (EWD340) [PDF](http://www.cs.utexas.edu/%7EEWD/ewd03xx/EWD340.PDF), 1972 [ACM Turing Award](http://en.wikipedia.org/wiki/ACM_Turing_Award) lecture
3. Dijkstra, E.W., "Structured Programming," Software Engineering Techniques, Buxton, J.N., and Randell, B., eds. Brussels, Belgium, NATO Science Committee, 1969.
4. B. Meyer, Object-Oriented Software Construction, second ed., Prentice Hall, 1997, Chap. 6, 10, 11.
5. Guide to the Software Engineering Body of Knowledge (SWEBOK). CHAPTER 4. SOFTWARE CONSTRUCTION.

<http://www.computer.org/portal/web/swebok/html/ch4>K. Beck, Test-Driven Development: By Example, Addison-Wesley, 2002.

1. McCabe : Complexity Measure, IEEE Transacions on Software Engineering, Volume 2, No 4, pp 308-320, December 1976
2. M. Fowler and al., Refactoring: Improving the Design of Existing Code, Addison-Wesley, 2002.
3. Russell Gold, Thomas Hammell, Tom Snyder. Test Driven Development: A J2EE Example.- Apress, 2005.- 296 pages.
4. Амблер С. Гибкие технологии: Экстремальное программирование и унифицированый процесс разаботки. Библиотека программиста.―Спб.: Питер, 2005.―412 с.
5. Бек К. Экстремальное программирование: разработка через тестирование. Библиотека программиста.―СПб.: Питер, 2003.―224 с.
6. Влиссидес Джон. Применение шаблонов проектирования. Дополнительные штрихи.: Пер. с англ.― М.: Издательский дом “Вильямс”, 2003. ― 144 с
7. Е.Гамма, Р.Хелм, Р.Джонсон, Д. Влиссидес. Приемы объекетно-ориентированного проектирования. Паттерны проектирования,-ДМК Пресс, 2010,-366 с.
8. Иан Соммервилл. Инженерия программного обеспечения, 6-е издание.: Пер. с англ. ―М.:Издательский дом “Вильямс”, 2002.―624 с.
9. Лавріщева К.М. ПРОГРАМНА ІНЖЕНЕРІЯ.–К.– 2008.–319 с.
10. Мартин Р. Быстрая разработка программ: принципы, примеры, практика. Пер. с англ.―М.: Издательский дом “Вильямс”, 2004.―752 с.
11. Макконел С. Совершенный код. Мастер класс / Пер. с англ.―М.: Издательско-торговый дом «Русская редакция»; Спб.: Питер, 2005.―896 с.
12. Фаулер М. Рефакторинг: улучшение существующего кода.―Пер. С англ.―СПб: Символ-Плюс, 2003.―432 с.

ЗМІСТ

Вступ.………………………………………………………………………………3

Основні теоретичні відомості конструювання програмного забезпечення.….4

Лабораторна робота №1 Використання моделей проектування програмного забезпечення……………………………………………………………………...15

Лабораторна робота №2 Планування конструювання програмного забезпечення……………………………………………………………………...17

Лабораторна робота №3 Використання мов конструювання програмного забезпечення……………………………………………………………………...19

Лабораторна робота №4 Забезпечення якості конструювання програмного забезпечення……………………………………………………………………...20

Лабораторна робота №5 Застосування шаблонів проектування……………..23

Лабораторна робота №6 Документування результатів розробки проекту програмного забезпечення………………………………………………………27

Завдання до лабораторних робіт………………………………………………..29

Контрольні питання ……………………………………………………………..30

Список використанних джерел……………………………………...34