ТРЕНІНГ-КУРС З ІНФОРМАЦІЙНОГО УПРАВЛІННЯ ПІДПРИЄМСТВАМИ ТА ПРОЄКТАМИ

Самостійна робота за змістовним модулем 3

3.1 Загальна характеристика основних стадій проектування

Будь-який програмний виріб має системні ознаки, тому процес розробки програмних виробів різних класів потребує спільного аналізу характеристик джерел вхідної інформації, каналів зв'язку, технічних засобів обробки інформації та вимог її користувачів. Доцільність системного підходу до розробки програмного забезпечення та використання досвіду розробки технічних систем не потребує доказів.

Використовуючи системний підхід та керуючись вимогами ДСТУ, процес створення програмного виробу поділяють на такі стадії: розробка технічного завдання, розробка ескізного (зовнішнього) проекту, розробка технічного (внутрішнього) проекту, робоче проектування, випробування програмного виробу.

Початкові стадії визначають якість проекту й успіх розробки програмних виробів у цілому. Для початкових стадій розробки програмних виробів типовими є такі ситуації: виявлення і чітке формулювання проблеми в умовах невизначеності; вибір стратегії дослідження та розробки; точне визначення програмного виробу як системи (межі системи, входи, виходи, набір компонентів); виявлення цілей функціонування програмного виробу; виявлення функцій та складу програмного виробу, що створюється.

Технічне завдання є результатом досліджень у предметній сфері з погляду задач, що розв'язуються в ній. На цій стадії майбутній комплекс програм ретельно аналізується з урахуванням набору функцій та основних властивостей; обґрунтовується доцільність їх розробки; попередньо оцінюються трудові та вартісні витрати й строки створення; виробляються рекомендації щодо вибору інструментальних засобів і методів, що передбачаються для використання. Обов'язковим у змісті цієї стадії є формування вимог до якості програми відповідно до умов її функціонування та реалізації конкретних функцій.

Програмний виріб являє собою складний комплекс взаємопов'язаних програм, у створенні яких беруть участь багато розробників, тому ефективність розроблюваної системи управління залежить від чітких і узгоджених дій всіх її учасників, які можуть бути досягнуті тільки за допомогою ретельно розробленого технічного завдання (ТЗ).

ТЗ є той документ, яким повинен керуватись колектив розробників програмних виробів. ТЗ розроблюється замовником і узгоджується з розробником, або розроблюється спільно. Усі зміни у ТЗ оформлюються протоколами і є невід'ємною частиною ТЗ.

Технічне завдання містить вимоги до програмного виробу, який розроблятиметься. Основні вимоги до програми повинні мати такі властивості: однозначність інтерпретації, повнота вимог, несуперечливість вимог, здійсненність вимог.

ТЗ містить характеристики програмного виробу, потоків вхідних і керуючих сигналів, а також такі техніко-економічні вимоги:

1. Призначення програмного виробу та його основні функції.
2. Перелік техніко-економічних вимог: очікувана економічна ефективність; вартість розробки (грошові та матеріальні ресурси, розподілені по стадіях та строках розробки).
3. Склад і характеристики потоків вхідної інформації – перелік вхідних даних, розподіл каналів зв'язку та засобів перетворення.
4. Основні вимоги до ЕОМ (обсяг пам'яті, системне програмне забезпечення, сумісність із зовнішніми пристроями та інше.)
5. Склад і характеристики вихідної (керуючої) інформації – перелік керуючих сигналів, розподіл каналів зв'язку, припустима середня квадратична помилка, швидкість зміни сигналів, час запізнення повідомлень, формат даних.
6. Необхідний рівень функціональних характеристик – стійкість до зовнішніх впливів, надійність, точність, раціональність структури.
7. Експлуатаційні характеристики – ступінь автоматизації запуску, останову, вводу параметрів та їх змін; вимоги до засобів індикації, контролюючих процес функціонування; можливості зберігання та оперативного відновлення програм у пам'яті ЕОМ.
8. Умови внесення доопрацювань – ступінь автоматизації процесу внесення доопрацювань, перевірка їх ефективності.
9. Склад конструкторської та експлуатаційної документації, вимоги щодо її оформлення.

Слід відзначити, що при використанні модульного принципу програмування, для складних модулів розробляють окремі ТЗ.

Ескізне проектування – процес опису очікуваної поведінки системи з погляду зовнішнього щодо неї спостерігача. Мета цього опису – отримання вичерпаного, детального опису зовнішніх взаємодій користувача з майбутнім продуктом, не торкаючись його внутрішньої будови. Стадія зовнішнього проектування значною мірою залежить від коректності та повноти вимог, що були сформульовані в технічному завданні.

Ескізний проект містить принципові конструктивні рішення – загальну оптимальну структуру програмного виробу, призначення його модулів, організацію взаємозв'язку між ними, обмін даними та динамічний розподіл ресурсів ЕОМ, попередню оцінку необхідного обсягу пам'яті та припустимий діапазон характеристик вхідних і вихідних величин для кожного модуля. Він також містить моделювання роботи основних алгоритмів для апробації основних принципів обробки даних і управління, уточнення основних положень методики випробувань і оцінювання якості функціонування

алгоритмів і програм, рішення технологічних і організаційних питань розробки логічного та інформаційного стикування модулів.

Ескізний проект виражається у формі зовнішніх специфікацій. Специфікація – це документ, в якому перераховуються умови, яким повинен відповідати програмний продукт, що розроблятиметься. До складу зовнішніх специфікацій належать: схема зовнішніх функцій; функціональні специфікації; структурне подання даних.

На стадії ескізного проектування мають бути розроблені всі принципові методичні та технологічні питання розробки програмного виробу. Ескізний проект затверджується замовником і є керівництвом для розробки технічного проекту.

Технічне проектування – це розробка сукупності проектних рішень щодо алгоритмічної структури програмного виробу та організації інформаційного забезпечення. До документації цієї стадії належать: пакет HIPO – схем або алгоритмічних схем; наочна таблиця змісту програмного виробу; зовнішні специфікації модулів.

Робоче проектування – це реалізація проектних рішень, що вироблені у відповідності із раніше сформульованими вимогами. Ця стадія включає безпосереднє кодування алгоритму (програмування), налагодження окремих компонентів і всього виробу в цілому, оформлення експлуатаційних документів.

На цій стадії використовується праця багатьох розробників, які працюють паралельно і координовано через тісний взаємозв’язок між модулями. Програмування кожного модуля закінчується автономним налагоджуванням, мета якого – локалізація та усунення помилок. Після цього розпочинається комплексне налагоджування програмного виробу.

До складу експлуатаційних документів належать: тексти програми; опис програми; опис застосування; посібник системного програміста; посібник програміста; опис вхідної мови і т. ін.

Випробування програм – це перевірка відповідності програмного виробу його специфікаціям на реальних даних або даних контрольного прикладу, наведеного в технічному завданні.

Досвід розробки пакетів загального призначення показує, котрі алгоритми та програми, значення їх параметрів і констант, взаємозв’язки, їх операторна частина під час створення піддаються неперервним змінам.

Крім того, можуть вводитись нові оператори, блоки і модулі, що реалізують нові додаткові функції, нові структурні зміни, що оптимізують процес управління і є об’єктивно необхідними, оскільки зумовлені набуттям досвіду і більш глибоким пізнанням взаємозв’язків між елементами системи та закономірностей процесу управління.

Певна кількість доопрацювань виконується для усунення неминучих помилок і прорахунків.

Стадії розробки та експлуатації програмного виробу є етапами його життєвого циклу, які пов’язані між собою прямими і зворотними зв’язками, причому останні означають повернення до одного з етапів, який уже

виконано, коли виявляються помилки, яких там припустилися. Вартість доопрацювання помилок зростає від останніх стадій до перших. Максимальні доопрацювання збігаються з роботами на стадії програмування і стикування програмного виробу з джерелами інформації.

Отже, процес розробки програмного виробу є багатоетапний і нерозривний; внесення змін в алгоритми і програми на всіх стадіях є об'єктивна закономірність, і це необхідно враховувати під час планування робіт і організації взаємодії між колективами розробників.

3.2. Проектування взаємодії користувача з програмним виробом

Під час проектування зовнішніх сполучень розробник повинен передбачити реалізацію таких властивостей програмного виробу, як зручність використання, надійність і безпека експлуатації, а також технологічність програмного виробу.

Реалізація цих властивостей може бути досягнена при виконанні таких груп правил: правила мінімізації помилок користувача; правила виявлення помилок користувача; правила мінімізації складності програмного виробу.

Правила мінімізації помилок користувача

1. Поведінка системи щодо користувача має бути гнучкою, тобто такою, щоб користувач не був змушений діяти виключно передбаченим способом. Це означає, що система повинна забезпечувати в будь-якому стані якомога більше функцій. У будь-якій ситуації мають бути припустимі різновиди формальних типів діалогу, які користувач може вільно вибирати.
2. Повідомлення команди та директиви, що вводяться користувачем, повинні бути якомога стислими, але не настільки, щоб втратився їх зміст.
3. Стандартизація і уніфікація типів повідомлень, що вводяться та виводяться. Повідомлення повинні мати однакові формати, стиль побудови, скорочення.
4. Узгодженість способу взаємодії з рівнем підготовки (рівнем кваліфікації) користувача. Цей рівень визначається в технічному завданні і має бути врахований для запобігання технологічним помилкам з боку користувачів.
5. Поведінка системи та результати її функціонування мають бути зрозумілими користувачу. Завжди на будь-яке вхідне повідомлення чи дію необхідно проектувати видачу якогось повідомлення. Порушення цього правила призводить до того, що користувач буде вагатися, чи вірно зроблено звернення, і може спробувати повторити ввід, внаслідок чого може виникнути небажана або хибна ситуація.
6. Система повинна бути завжди готовою допомогти користувачу, ніколи не слід ставити користувача у скрутне становище. Реалізація цього правила виражається в наявності засобів допомоги (інструкцій, підказок), звернення до яких має виконуватись за бажанням користувача. Програмний

виріб має бути побудований таким чином, щоб його використання було можливим без спеціального навчання діалогу.

7. Проект системи повинен брати до уваги фізичні та психологічні особливості користувача під час його роботи на машині. Це стосується, наприклад, швидкості введення даних і директив і реакції системи на ввід, яка не повинна перевищувати межу безстресової роботи, але й не повинна бути дуже повільною. Крім того, система має давати користувачеві пояснення будь-якої затримки відповіді.

Правила виявлення помилок користувача

1. Система має приймати будь-які дані. Якщо введені дані є неприпустимими, то система повинна обов’язково повідомити про це користувача, щоб він не прийняв рішення на основі недостовірної інформації.

2. Користувачеві має надаватись можливість перевірки введених даних ще до початку їх обробки.

3. Помилки користувача повинні виявлятися негайно, а не після того, як програма завершила роботу. Наприклад, якщо певний модуль має виконуватися тільки після успішного завершення попереднього, необхідно передбачити реалізацію цієї умови при розробці.

4. Система повинна передбачати реакцію на випадкові дії користувача, і, якщо ці дії загрожують процесу обробки інформації або призводять до пошкодження даних, блокувати їх і вимагати від користувача підтвердження або відміни виконання попередньої команди.

5. Там, де особливо важлива підвищена вірогідність результатів обробки, слід використовувати надлишок даних для викриття помилки.

Правила мінімізації складності програмного виробу

Реалізація цього правила найбільш відчутна щодо зовнішнього проекту, коли закладається остов майбутньої програми. І шлях мінімізації складності – уніфікація способу взаємодії користувача з програмним виробом. Відповідно до цього краще мати порівняно невеликій набір добре погоджених функцій з мінімальною кількістю специфічних особливостей, ніж якомога більший набір незалежних і нескоординованих функцій.