Конспект лекцій

з дисципліни «Людино-машинна взаємодія»

для студентів усіх форм навчання

спеціальності 121 – Інженерія програмного забезпечення

(освітня програма «Програмна інженерія»)

[Електронне видання]

/ Упоряд.: Р. В. Мельнікова

ЗМІСТ

ВСТУП

У сучасних умовах фахівець-професіонал в області обчислювальних систем і програмного забезпечення має бути знайомим з усіма аспектами сучасних комп'ютерних технологій, вміти використовувати їх при створенні програм.

Курс «Людино-машинний інтерфейс» має за мету надання студентові базових знань, принципів, стандартів та методів, які лежать в основі проектування високоякісного інтерфейсу користувача.

Метою дисципліни є ознайомлення студентів з поняттями, моделями, законами, методами та принципами, що застосовуються при проектуванні, створенні та тестуванні людино-машинних інтерфейсів.

Після вивчення дисципліни студенти мають **знати**: основи взаємодії людини та комп'ютера: визначення процесів навчання, сприйняття, зберігання та обробки інформації людиною з точки зору когнітивної психології; моделі, стандарти, принципи та засоби, що використовуються при проектуванні інтерфейсів; основні методики оцінки якості та зручності програмних інтерфейсів. **Уміти**: застосовувати отримані знання при проектуванні людино-орієнтованих ергономічних інтерфейсів користувача для будь-якої проблемної області; будувати модель користувача та проектувальника інтерфейсу; проводити тестування інтерфейсів різними методами.

Конспект лекцій складено згідно з освітньо-професійною програмою вищої освіти за напрямом «Програмна інженерія» та робочою програмою дисципліни.

1 Поняття інтерфейсу. Стилі інтерфейсу. Різновиди інтерфейсів. Критерії якості інтерфейсу. Основні поняття з теорії моделювання: система, моделювання, модель

Для пересічного користувача програмного продукту, інтерфейс, здебільшого, є основною частиною додатка - єдиною частиною програми, яку він бачить. Він взагалі не підозрює про код програми, що виконується у відповідь на його взаємодію з комп'ютером. Тому необхідно усвідомлювати важливість процесу проектування інтерфейсу додатка.

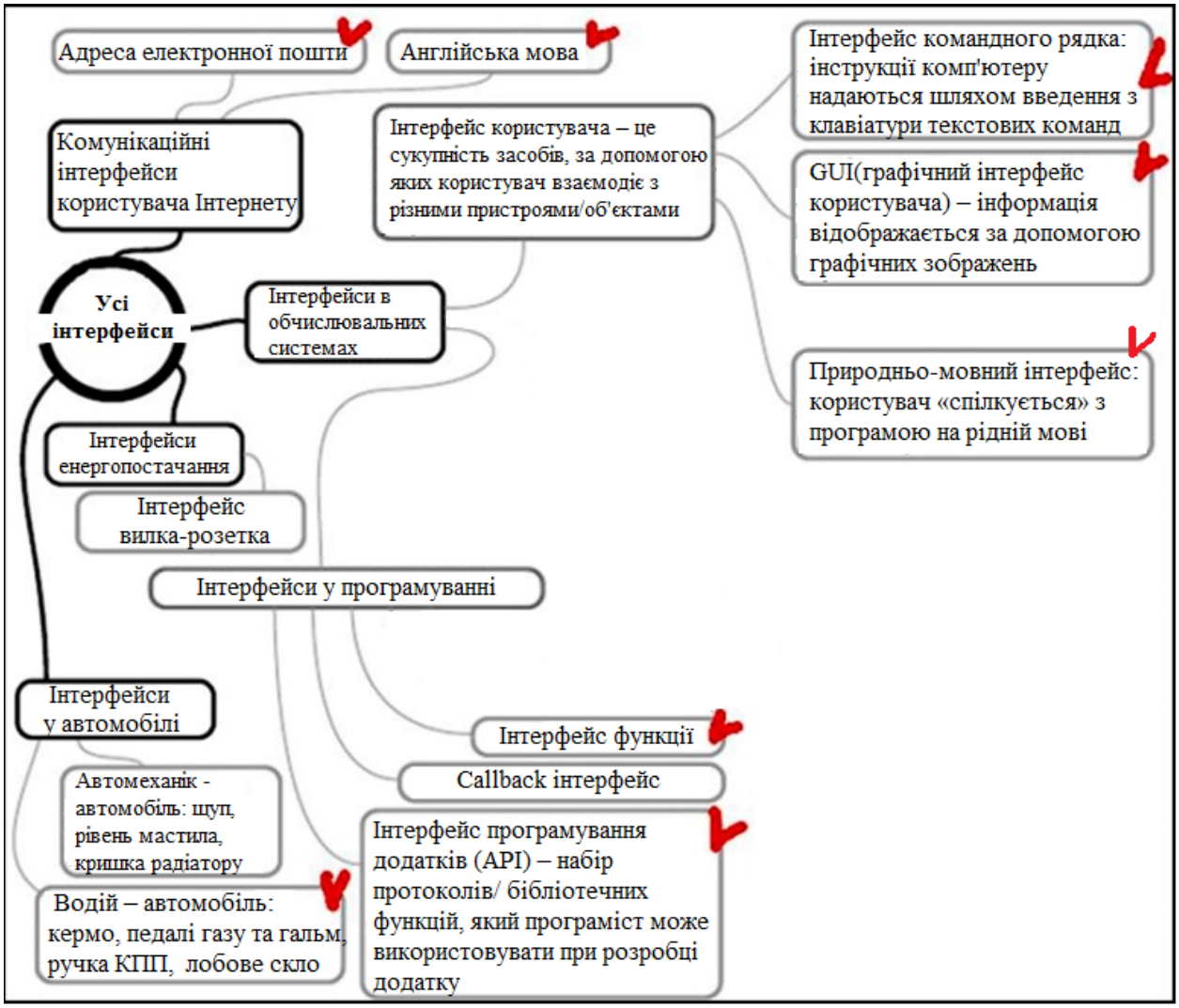

Рисунок 1.1 - Різновиди інтерфейсів

У комп'ютерній галузі слово інтерфейс означає місце, де одна незалежна система (людина) зустрічає іншу незалежну систему (програмну) і взаємодіє з нею. За іншим визначенням цього терміну – інтерфейс - це те, що користувач бачить на екрані під час виконання програми, тому його якісна розробка є не менш важливою ніж обробка даних, яка відбувається у програмі.

У більш широкому сенсі інтерфейс містить у собі пристрої вводу та виводу (апаратне забезпечення) і програмне забезпечення, що обслуговує їх, а також все те, що допомагає

користувачеві взаємодіяти з комп'ютером, у тому числі документація й довідкова інформація.

Найбільш продуктивним шляхом проектування оптимального інтерфейсу є його створення з урахуванням досвіду та у відповідності до потреб і побажань майбутнього користувача.

В процесі розвитку програмної інженерії були виявлені серед існуючих найбільш поширені стилі програмних інтерфейсів: GUI, WUI та об’єктно-орієнтований. Кожен з них має свої переваги та особливості, які впливають на вибір стилю конкретного програмного продукту (ПП).

Якість інтерфейсу оцінюється за такими критеріями:

- наскільки розробники розуміють потреби та завдання користувачів;
- чи є цей продукт результатом ретельно продуманого проектування;
- чи наочно відображені всі особливості продукту;
- наскільки продукт відповідає вимогам практичності та доцільності;
- чи задовольняє використання цього продукту естетичне почуття;
- наскільки інтерфейс може змінюватися при взаємодії з різними особами та групами;
- чи сприяє він спрощенню концептуальної моделі користувача.

Система - це сукупність елементів, зв'язаних у єдине ціле для досягнення поставленої мети. Наявність мети змушує зв'язувати елементи в систему.

Наприклад: комп'ютерна система, що складається із системного блоку, монітора, зовнішніх пристроїв, проводів, що з'єднують ці елементи для забезпечення обчислень, зберігання й обробки даних і т.д.

Елемент - мінімальний неподільний об'єкт системи.

Опис системи - це сукупність відомостей про систему, що досліджується, про умови її роботи. Задається у вигляді схем, текстів, графіків, таблиць, формул тощо. Опис є основою для побудови моделі. Існують різні рівні деталізації систем.

Моделювання - подання об'єкта дослідження його моделлю й проведення експериментів з моделлю з метою одержання інформації про найважливіші властивості об'єкта, що досліджується.

Відмінна риса моделювання (від інших методів системного аналізу) полягає в можливості дослідження таких систем, прямий експеримент із якими важко виконаємо, економічно невигідний або неможливий.

У моделюванні існують 2 підходи: аналіз (процес визначення властивостей вже існуючої системи) та синтез (процес визначення структури та функцій для нової системи для одержання необхідних результатів).

Концептуальна модель відтворює опис системи в абстрактних термінах і поняттях, виділяючи істотні з погляду дослідження причинно-наслідкові зв'язки. Описує ті явища, які повинні відбуватися із системою.

Розробка концептуальної моделі вимагає глибоких знань про систему.

1.1 Запитання та завдання для самоперевірки

1 Дайте визначення поняття інтерфейс.

2 З чим пов’язаний той факт, що існує декілька видів стилів інтерфейсів ПП?

3 Наведіть приклади ПП з різними стилями інтерфейсів. Як визначити якість інтерфейсу ПП?

4 Яку мету переслідує моделювання?

2 Моделі, що використовуються при проектуванні інтерфейсу: концептуальна модель користувача, модель проектувальника.

Фахівець в області програмної інженерії, що має за завдання побудувати інтерфейс ПП, може водночас бути не тільки програмістом, але й проектувальником. Також він має отримати дані, які допоможуть йому уявити *концептуальну модель користувача*, саме для якого цей ПП призначений.

Підсвідомо користувачі завжди створюють такі моделі, коли починають користуватися будь-яким новим приладом для того, щоб:

- передбачити події;
- знайти причини помічених подій;
- виявити дії, що необхідні для здійснення потрібних змін;
- для забезпечення розуміння аналогічних пристроїв.

Для виявлення моделі користувача існують такі методи збору інформації:

- аналіз завдань користувачів;
- інтерв’ю з потенціальними користувачами;
- спостереження за користувачем у процесі виконання завдання;
- тести придатності.

Програміст, що кодує інтерфейс програми, має отримати технічне завдання від проектувальника та за допомогою відомих йому засобів розробки, методів та принципів програмування реалізує поставлене завдання.

Робота проектувальника інтерфейсу подібна роботі архітектора при будуванні оселі (він збирає ідеї майбутнього власника дому («користувача»), уточнює їх, пропонує сучасні матеріали («стандарти проектування та новітні технології»), поєднує ці дані з можливостями будівельників («програмісти») та пропонує конструкцію, яка відповідає будівельним нормам та правилам.

Таким чином, проектувальник є ланкою, що поєднує, та яка розташована в ієрархії на одну сходинку вище над користувачем і програмістом (рис. 1.2).

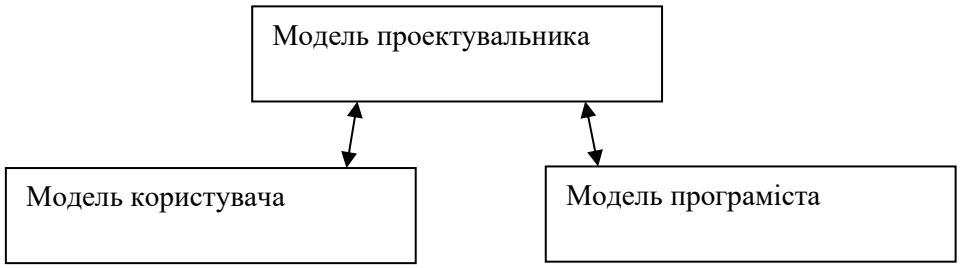

Рисунок 1.2 - Місце моделі проектувальника при розробці програмного інтерфейсу

Шаблон моделі користувача:

- мета;
- завдання (з виділенням основного);
- процеси;
- інструменти взаємодії (об'єкти проблемної галузі);
- результати.

При створенні такої моделі слід використовувати терміни проблемної області (іменники, прикметники дієслова тощо, які властиві користувачеві, наприклад: «додати новий запис до записної книжки» замість «відправити запит на додавання нового елементу у базі даних». Або «внесення особистої інформації» замість «заповнення форми реєстрації» тощо.

Розглянемо, наприклад, концептуальну модель користувача мірного кухоля.

Цілі:

- приготувати смачну вечерю; підготувати продукти до рецепту; провести експеримент з хімії;

Завдання:

- відміряти точну кількість продукту (основна);
- змішати 2 продукти;

Процеси (для основного завдання):

- помістити потрібну кількість продукту, перевірити кількість, виправити, якщо вага неправильна;

Інструменти взаємодії(об'єкти проблемної області):

- Шкала, ручка, носик, матеріал кухоля;

Результати:

- Підготовлені інгредієнти за рецептом, нагодована задоволена сім'я.

Модель проектувальника описує об'єкти, з якими працює користувач ПП, техніку маніпулювання ними та складається з 3 компонентів:

- подача інформації: візуальне оформлення, естетика (колір, анімація, звук, форма об'єктів, графіка, текст, місце розташування інформації на екрані);
- взаємодія користувача з комп'ютером: дії з клавіатурою, мишею та ін. пристрої введення, наявність клавіш швидкого доступу, структура меню, наявність зворотної зв'язку з користувачем;
- взаємозв'язок між об'єктами проблемної галузі: властивостями об'єктів та їх зв'язків (використання прийнятних метафор для узгодження задач користувача з можливостями їх реалізації на комп'ютері/пристрою).

2.1 Запитання та завдання для самоперевірки

1 Як створюється модель користувача ПП?

2 Навіщо користувачі створюють концептуальну модель при взаємодії з чимось новим, наприклад - з новою комп'ютерною грою?

3 Як проектувальник інтерфейсу може дізнатися про потреби користувача ПП, інтерфейс якого який проектується?

4 Які дві цілі проектувальника допомагають зробити редизайн успішним?

5 Чи були в Вашому житті випадки, коли Ваша концептуальна модель взаємодії з якимось об'єктом, системою або пристроєм була іншою, ніж у розробника (проектувальника), але на Ваш погляд такою, що могла покращити або спростити процес такої взаємодії? Якщо – так, наведіть приклади?

3 Прототипування інтерфейсів. Сториборди. Використання програмного забезпечення для створення прототипів інтерфейсів (Moqups,..)

Навички проектувальника досвіду користувача (UX-designer) допомагають перейти від слів (моделі користувача) до графічного відображення (сторібордів, ескізів). Вони допомагають узагальнити потреби користувача та перевести їх у візуальну форму для подальшого втілення в інтерфейсі системи,що проектується.

Існують багато різних способів концептуального проектування. Їх використовують для передачі потреб користувача та передбачаємої послідовності його дій, для приєднання до обговорення майбутнього рішення інших учасників проекту. Також на цьому етапі можна виявити та внести до проекту раніше пропущені проектувальником функції.

Таким чином, прототипування – це ефективний метод планування, тестування та перевірки функціональності та архітектури проекту, що розробляється. Зазвичай, його проводять у такій послідовності:

- покрокова розкадровка основних дій користувача (сторіборд, storyboard);
- обговорення створеного сторіборду з майбутніми користувачами та іншими учасниками проекту
- створення та тестування паперового прототипу (опціонально);
- створення ескізу (wireframe) майбутнього інтерфейсу за допомогою ПЗ (Moqups, Balsamiq,…);
- тестування ескізу;
- створення програмного коду для остаточного варіанту ескізу.

Покрокова розкадровка дій користувача (сторіборд, storyboard)

Головне для розуміння те, що розкадровка (сторіборд) - **це про завдання користувача**.

Хороший сторіборд завжди має своїм героєм конкретну людину, яка використовує цей інтерфейс. Також він допомагає побачити потік (послідовність) дій: можливо, в декілька гумористичній формі розповідає, що відбувається **в ключові моменти** часу.

Сторіборд - не просто картинки, а «взаємозв'язок ідей».

Рисунок 1.3 – Приклад взаємодії мовою сторіборду

Насправді це досить проста візуальна мову, на якій можна чітко викласти ідею проекту, а заодно і моменти взаємодії, ключові моменти.

Таким чином, в сторіборді мають міститися такі 3 речі:

- **Початкові умови** (люди, довкілля, завдання, які користувачу треба вирішити);
- **Послідовність дій** (які кроки повинні бути зроблено; що примусить (приведе) користувача до використання інтерфейсу; які завдання користувача інтерфейс допомагає виконати);
- **Задоволеність** (що буде мотивувати людину використовувати інтерфейс, що зробить можливим виконання завдань користувача; які потреби користувача виконує система; які цілі допомагає досягти).

Переваги сторіборду:

- робить наголос на завданнях, які інтерфейс допомагає виконати;
- уникає фіксації на конкретному інтерфейсі;
- допомагає зобразити всіх учасників на одній сторінці із зазначенням їх цілей.

Бажано не витрачати багато часу на створення сторіборду, що складається з 5-6 кадрів: часові обмеження сприяють створенню гарного рішення. Сторіборд можна створювати за допомогою олівця на папері або за допомогою спеціалізованих програмних засобів, наприклад, за посиланням: storyboardthat.com. Для ознайомленням з цим он-лайн сервісом рекомендується подивитися демонстраційне відео та/або прочитати розділ FAQ.

Використання програмного забезпечення для створення ескізів (wireframes) інтерфейсів

Прототипування – є ітеративним процесом, оскільки прототипи, зазвичай, створюються для виявлення недоліків та перевірки результатів проектування. Після

отримання зворотнього зв’язку проектувальник вносить зміни у прототип та проводить додаткове тестування. Після чого, успішний прототип може забезпечити перехід проектування у наступну фазу – створення ескізу (wireframe) інтерфейсу.

Вимоги для проведення тестування прототипу:

- потрібно знати, що хочуть від пристрою (інтерфейсу) користувачі;

- кожен клієнт має можливість внести свої додаткові пропозиції і побажання в прототип;

- для обговорення потрібно мати як мінімум один варіант, щоб надати як зразок;

- бачити, як користуються (пристроєм) інтерфейсом [інші] користувачі;

- зуміти оцінити і проаналізувати результат.

Для створення прототипів інтерфейсів у цифровому вигляді використовуються різні програмні додатки. Зазвичай, в них вже існують багато елементів для використання, які можна просто переміщати на робочу область за допомогою миші, є можливість імітувати послідовність дій користувача, є можливість швидкої зміни створеного ескізу, інколи існує функція конвертації створеного ескізу у файл формату .pdf.

Для створення прототипу у вигляді ескізу (wireframe) можна використовувати онлайн-додаток, який доступний за посиланням **https://moqups.com/**

Його безкоштовна версія (після реєстрації) дає можливість створити ескіз для одного проекту з використанням не більш ніж 200 елементів.

3.1 Запитання та завдання для самоперевірки

1. Що є метою прототипування?
2. Що таке сторіборд? Для чого вони використовуються? Які моменти взаємодії користувача мають бути обов’язково відображені на сторіборді?
3. Для чого використовуються ескізи (wireframes) інтерфейсів?
4. Як зімітувати перехід з одної сторінки на іншу при проектуванні ескізів інтерфейсів у Moqups?
5. Чи потребують сторіборди та ескізи обговорення з користувачами?

4 Процеси сприйняття й навчання людини з точки зору когнітивної психології. Людина vs комп'ютер: пам'ять і обробка даних. Ергономіка програмних інтерфейсів

Люди, створюючи комп'ютери для полегшення своєї праці й для розширення сфери розваг, намагаються надати комп'ютеру свої вміння й можливості (зберігання інформації, виконання розрахунків, виконання перекладів з однієї мови на іншу, граматична перевірка тексту...).

Для того щоб забезпечити наявність цих навичок у комп'ютері нам необхідно усвідомлювати, як ці дії робимо ми самі.

Розділ науки, який називається **когнітивною психологією** саме й займається вивченням того, як працює наш мозок, як ми думаємо, як запам'ятовуємо, як навчаємось, як сприймаємо навколишній світ, як проходять процеси *навчання* й *сприйняття*, що впливають на одержання й обробку інформації людиною..

Вважається, що чим краще ми будемо знати побудову нашого мозку, тим більш успішними будуть проекти по створенню систем людина-комп'ютер.

Відповідно до когнітивної психології **навчання** – це процес, що розвивається та який об'єднує попередні навички, знання, притаманні людині, та досвід, який вона здобуває.

Відповідно до когнітивної психології **навчання** – це процес, що розвивається та який об'єднує попередні навички, знання, притаманні людині, та досвід, який вона здобуває.

При взаємодії людина-комп'ютер передача інформації відбувається за допомогою інтерфейсу програм.

Тому при створенні програмних інтерфейсів (ПІ) необхідно керуватися сукупністю усіх відомостей про процес сприйнятті, тобто керуватися ергономікою процесу сприйняття: звичайна людина не має можливості перемножити за 5 сек. багаторазрядні числа – то ж безглуздо розробляти інтерфейс, який вимагав би від неї такі здібності.

Когнітивна психологія будує модель пізнання людини для наступного її застосування в комп'ютерній галузі.

Оскільки досвід, звички та інформація, яку ми одержуємо від органів почуттів, зберігаються в нашій пам'яті, то необхідно розглянути загальні принципи запам'ятовування й зберігання інформації людиною.

Існує 3 види пам'яті:

- довгострокова;
- короткострокова;
- буферна.

Об'єкт	**Переваги**	**Недоліки**
Людина	Розпізнавання образів	Короткострокова пам'ять малої ємності
	Паралельна обробка даних	Швидка втрата даних з короткострокової пам'яті
	Нескінченна ємність довгострокової пам'яті (завжди можна додати ще)	Повільна обробка даних у короткостроковій пам'яті
	Многокодовая пам'ять	Проблеми доступу до даних з довгострокової пам'яті
	Здатність до навчання	Помилки при обчисленнях

Рисунок 1.4 - Пам'ять і обробка даних: людина vs комп'ютер. Порівняльний аналіз.

Для роботи з інформацією із довгострокової пам'яті існують 2 методи: розпізнавання й відновлення.

Розпізнавання - це спроба згадати інформацію, використовуючи будь-який зв'язок.

Відновлення - це спроба згадати інформацію без будь-якої допомоги.

Ергономічний інтерфейс – інтерфейс, що спроектований з урахуванням когнітивних особливостей людини.

Одне з найбільш важливих завдань проектування ПІ:

- **використовувати переваги комп'ютера** та **зменшити навантаження на** розумові (ментальні) процеси **користувача** (тобто враховувати його слабості).

4.1 Запитання та завдання для самоперевірки

6. Що є метою прототипування?
7. Що таке сторіборд? Для чого вони використовуються? Які моменти взаємодії користувача мають бути обов’язково відображені на сторіборді?
8. Для чого використовуються ескізи (wireframes) інтерфейсів?
9. Як зімітувати перехід з одної сторінки на іншу при проектуванні ескізів інтерфейсів у Moqups?
10. Чи потребують сторіборди та ескізи обговорення з користувачами?

5 Використання стандартних елементів інтерфейсу користувача

Необхідно вміти проектувати інтерфейси чітко уявляючи для себе, які користувачі будуть працювати з ним та за яких умов це буде відбуватися. Для різних груп користувачів (звичайні люди , літні, малюки, працівники рятувальних служб, швидкої допомоги, call-центрів тощо) необхідно розробляти інтерфейси враховуючи їх особливості або вимоги до їх умов праці (необхідність мінімізації часу відгуку).

Для кожної задачі, яку необхідно вирішити, завжди існують декілька варіантів рішення. Створення інтерфейсу для заданої проблемної області можливо як за допомогою комбінацій стандартних елементів, так і за допомогою поєднання графічних примітивів, малюнків з технологіями click-to-choose, click-to-activate, drag-and-drop, можливостями Web-інтерфейсів для дескопів або смартфонів.

Майже кожна програма, що призначена для роботи з користувачем має вікно, оскільки саме через нього вона буде приймати уведення з миші та клавіатури, виконувати виведення інформації на екран. Всі інші стандартні елементи інтерфейсу користувача - меню, лінійки прокручування, курсор миші й т.п. - грають лише допоміжну роль по відношенню до вікна.

Вікно - це, як правило, прямокутна область на екрані. Вікно програми за звичай містить заголовок (title bar), меню (menu), рамку (sizing border), смуги прокручування (scroll bar), кнопки закриття, згортання та робочу область.

Активне вікно - це вікно, що у цей момент перебуває на передньому плані.

Вікно з фокусом введення - активне вікно або його дочірнє вікно. Саме в це вікно в цей момент спрямоване введення із клавіатури.

Необхідно пам’ятати, що недоречне використання стандартних елементів може призвести до дезорієнтації користувача. Надмірна кількість елементів або невдале їх групування також знижує ефективність інтерфейсу.

5.1 Запитання та завдання для самоперевірки

1. Навіщо між елементами інтерфейсу залишають проміжок? Яким він має бути за розміром?
2. Який пристрій введення швидше дозволяє маніпулювати: миша або тачпад? Чому?
3. Що таке фокус введення? Як користувач може їм керувати? Як їм може керувати програміст?

4. Для яких даних використовується стандартний елемент інтерфейсу TreeView?

5. Що допомагає позбутися використання стандартного елементу інтерфейсу ProgressBar?

6. У яких випадках при проектуванні інтерфейсу краще використовувати ListBox замість Edit?

7. У яких випадках при проектуванні інтерфейсу краще використовувати TabControl?

6 Модель GOMS. Законі Хіка та Фітса. Наслідки цих законів.

Кількісні методи допомагають звести спірні питання евристичних методів (наприклад, метод тестування інтерфейсу за участю користувача) до простих обчислень.

Евристика (heuristic) - знання, придбане людиною в міру нагромадження досвіду в будь-якій діяльності, тобто в рішенні практичних завдань певного класу (без застосування математичних розрахунків).

Крім цього, кількісні методи допомагають нам зрозуміти найважливіші аспекти взаємодії людини з комп'ютером.

Існує безліч методів кількісної оцінки інтерфейсів, розглянемо декілька з них.

Модель GOMS дозволяє передбачити час виконання завдання складного редагування при використанні заданого варіанта інтерфейсу.

Обмеження:

- розрахунки виробляються тільки для досвідчених користувачів;
- без обліку можливих помилок користувачів;

До такого формального аналізу прибігають у випадках, коли необхідно вибрати один з 2 варіантів розробки, якщо навіть невеликі розходження у швидкості можуть давати психологічний або економічний ефект (часто повторювані дії, швидка допомога, ...).

Розробники моделі GOMS помітили, що загальний час виконання роботи системою користувач-комп'ютер є сумою всіх інтервалів, які будуть потрібні системі на виконання елементарних жестів, що становлять дане завдання.

Хоча цей час змінюється для різних користувачів, ретельні лабораторні дослідження показали, що можна застосовувати набір стандартних інтервалів.

У даній моделі використовуються *середні* значення для тимчасових інтервалів. Тому, така модель дозволяє провести ПОРІВНЯЛЬНИЙ АНАЛІЗ 2 і більше варіантів інтерфейсів, а не розрахувати абсолютний час використання кожного з них.

$$T_{EXECUTE} = T_K + T_P + T_H + T_D + T_M + T_R$$

Рисунок 1.5 - Загальний час виконання роботи системою користувач-комп'ютер згідно моделі GOMS

Де:

K - час, необхідний для натискання клавіші на клавіатурі /клік миші (0,2 с);

P - час, необхідний для переміщення миші з метою вказати на об'єкт (1,1 с);

H - час, необхідний для переміщення руки із клавіатури на мишу або з миші на клавіатуру (0,4 с);

D - час перемальовування (для інтерфейсів зі складною графікою. Обчислюється експериментально);

M - ментальна підготовка (1,35 с);

R - час відповіді системи (має бути < 0,25 с; для того, щоб уникнути появи занепокоєння користувача).

Алгоритм GOMS:

- перелічити список необхідних жестів (проставити оператори K, P, H);
- визначити ментальні операції (оператор M);
- прибрати деякі оператори M;
- підрахувати час.

Для розрахунків взаємодії зі смартфонами існує Touch-level model (TLM).

Закон Фитса дозволяє кількісно визначити той факт, що чим далі об'єкт від поточної позиції курсору або чим менше його розміри, тим більше часу буде потрібно користувачеві для переміщення до нього курсору.

Закон Хика дозволяє кількісно виразити, що: “Чим більше варіантів вибору надається, тим більше часу потрібно на вибір ”.

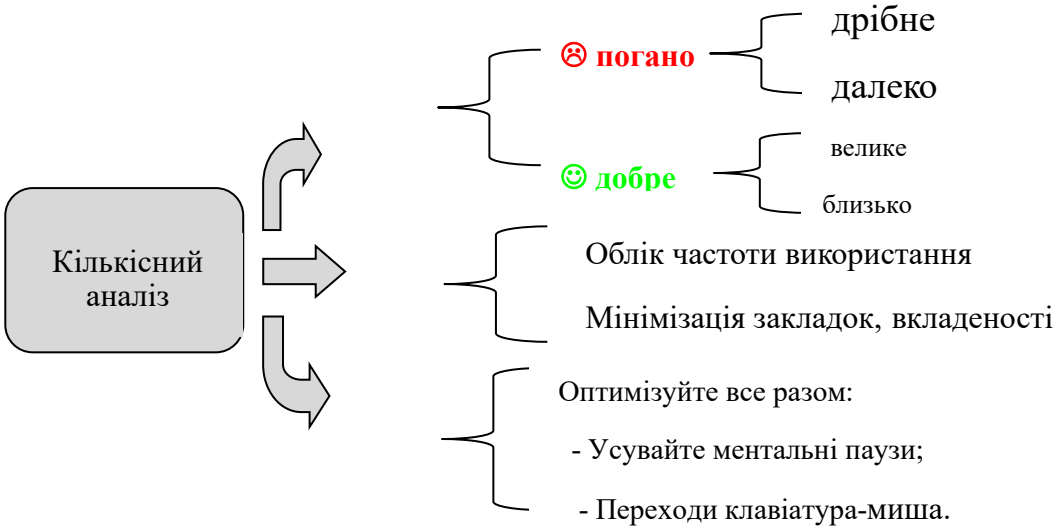

Рисунок 1.6 - Квантифікаційний аналіз. Итоги

6.1 Запитання та завдання для самоперевірки

1. Що таке квантифікаційний аналіз інтерфейсу? Які методи такого аналізу Ви знаєте?
2. Чому для різних груп користувачів чи для різних контекстів є сенс використовуватися різні варіанти інтерфейсу?
3. Які є переваги у інтерфейсу, що розроблений за допомогою стандартних елементів (button, checkbox,…)? За рахунок чого є можливість прискорити час роботи користувача з такими інтерфейсами?
4. На яких параметрах базуються викладки моделі GOMS? Перелічить основні жести користувача згідно цієї моделі? Які відмінності має TLM модель?
5. Перелічите послідовність жестів, що необхідні згідно моделі GOMS, для запуска додатку, іконка якого розташована на Робочому столі.
6. Чи дають обрахунки за моделлю GOMS для двох інтерфейсів знайти мінімально можливе значення часу роботи?

7 Тестування (оцінювання) інтерфейсу без участі реальних користувачів. 10 основних принципів (евристик) створення інтерфейсу. Перевірка інтерфейсів за контрольними списками.

Usability - зручність роботи, простота використання.

Евристика (heuristic) - знання, придбане людиною в міру нагромадження досвіду в будь-якій діяльності, тобто в рішенні практичних завдань певного класу.

Евристичні методи, на відміну від алгоритмічних, не вимагають математичних розрахунків, але досвід розробників свідчить про їх ефективність при вдосконаленні інтерфейсів.

Ці 10 евристик не є твердими правилами, яким обов'язково необхідно слідувати, але досвід розроблювачів показує, що їх застосування неодмінно сприяє підвищенню зручності, збільшує ефективність і простоту використання інтерфейсів (не тільки програмних , але і будь-яких інших): *Feedback - Metaphor - Navigation - Consistency - Prevention - Recognition - Efficiency - Design – Recovery - Help*

- *Feedback* - інформування про поточний стан системи;

Система завжди повинна вчасно повідомляти користувачеві про свій поточний стан за допомогою підходящого зворотного зв'язка

- *Metaphor* - використання метафор реального світу;

Система має говорити мовою користувачів, використовуючи слова, фрази і поняття, знайомі користувачеві, а не слова, що орієнтовані на систему або розробника. Слідуйте за прийнятими в реальному світі умовними позначками і правилами, відображаючи інформацію в природному й логічному порядку.

- *Navigation* - надання контролю та свободи користувачеві у навігації;

Надайте користувачеві можливість вибору: працювати з мишкою, клавіатурою або й тим і іншим. Створіть умови для негайних оборотних дій. Користувачі часто вибирають функції системи помилково й мають потребу в ясно визначеному "запасному виході", щоб вийти з небажаного стану, минаючи тривали діалоги. Забезпечте "Undo" і "Redo".

- *Consistency* - послідовність і слідування стандартам;

Використовуйте загальновідомі терміни, назви об'єктів і дій так, як звичайно прийнято. Користувачі повинні бути впевненими в послідовному поводженні об'єктів інтерфейсу. Додержуйтеся загальноприйнятих угод.

- *Prevention* - попередження помилок;

Ретельно продумане попередження помилок є кращим засобом боротьби з ними, у порівнянні з добре продуманим повідомленням про зроблену помилку.

- *Recognition* - розпізнавання, а не відновлення інформації;

Мінімізуйте навантаження на пам'ять користувача, роблячи видимими можливі варіанти. Користувач не повинен запам'ятовувати інформацію, обрану їм на попередньому кроці. Правила користування системою мають бути видимими і доступними по запиту в будь-який момент

- *Efficiency* - гнучкість і ефективність у використанні;

Акселератори не видимі для новачків, але досвідченим користувачам дають можливість збільшити швидкість роботи. Тому система повинна задовольняти потреби як тих, так і інших. Давайте можливість користувачеві настроїти під себе найбільше часто використовувані дії.

- *Design* - естетична привабливість і мінімалізм;

У текстовій інформації не має бути даних, що не стосуються проблеми або рідко використовуються. Кожний зайвий елемент інтерфейсу відволікає увагу від потрібних елементів і, таким чином, зменшує його "візуальну читабельність". Гарний дизайн заснований на принципах: *контрасту*, *повторюваності*, *вирівнювання* й *подібності*

- *Recovery* - допомога користувачеві у визначенні помилкових дій, в їх діагностиці та виправленні;

- *Help* - надання довідкової інформації й документації.

Повідомлення про помилки повинні бути написані простою мовою, не містити кодованої інформації, точно вказувати на проблему й пропонувати її конструктивне рішення.

7.1 Запитання та завдання для самоперевірки

1 Що таке юзабіліті?

2 Що таке евристика? Що дає використання евристик при розробці інтерфейсів?

3 Приведіть декілька прикладів використання вказаної евристики юзабіліті (Інформування про поточний стан системи) у повсякденному житті або у ПП.

4 Приведіть декілька прикладів використання вказаної евристики юзабіліті («Використання метафор реального світу») у повсякденному житті або у ПП

5 Які проблеми дозволяє знайти евристичний аналіз?

6 Приведіть декілька прикладів використання вказаної евристики юзабіліті («Розпізнавання, а не відновлення інформації») у повсякденному житті або у ПП.

8 Тестування інтерфейсу без участі реальних користувачів. Когнітивний наскрізний перегляд.

Тестування без участі реальних користувачів

- таке тестування розробники проводять самостійно на різних стадіях розробки інтерфейсу, починаючи з паперового прототипу;

- зазвичай рівень "комп'ютерної грамотності" проектувальника вище рівня потенційної аудиторії.

Результат:

- проектувальник вибирає рішення, які забезпечують ефективність роботи, але споживачам такі рішення не зрозумілі;
- у підсумку вони виявляються нездатними скористатися системою.

Тестування із залученням реальних користувачів

- рівень фінансування проекту визначає етап, з якого починається проведення тестування з користувачами: чим раніше, тим краще (але дорожче);
- зазвичай проектувальник інтерфейсу знає про предметну область менше, ніж майбутні користувачі.

Результат:

- відсутність тестування з користувачами може привести до того, що система виявиться нездатної вирішувати завдання, про необхідність яких проектувальник не знає;
- тестування при участі користувачів призначено для визначення слабких місць інтерфейсу, які не були знайдені під час тестування без їхньої участі.

Тестування інтерфейсу без участі реальних користувачів (Evaluating the Interface Without Users)

- **Когнітивний наскрізний перегляд (Cognitive Walkthrough);**

  Обмірковування можливих проблем і помилкових шляхів користувача.
- **Аналіз дій (Action Analysis);**
  - ✓ *Аналіз по моделі GOMS (Key-Stroke Model);* (був розглянутий раніше)

Чому важливі види аналізу, у яких не беруть участь користувачі:

- незважаючи на важливість залучення користувачів у процес проектування інтерфейсу, у проектувальників ПІ існує необхідність оцінити проект, що розробляється, на ранній стадії (без користувачів) або при відсутності належного фінансування;
- час потенційних користувачів (майже завжди) є дорогим і обмеженим ресурсом;
- добре проведене тестування без користувачів виявляє проблеми, які можуть бути пропущені при тестуванні за участю невеликої кількості користувачів;

Обидва види аналізу: без користувачів і з їх участю можуть не виявити всіх проблем (окремо), але об'єднання результатів кожного з них дозволяє істотно поліпшити якість ПІ.

Початкові умови й алгоритм проведення когнітивного наскрізного перегляду (КСП)

Персоні, яка проводе аналіз, **надається:**

- ✓ зразок ПІ, що тестується, або детальний опис проекту інтерфейсу **(1);**
- ✓ характеристика цільової аудиторії користувачів **(2);**

Вибирається одне з типових завдань, рішення якої забезпечується інтерфейсом **(3);**

Продумується і записується реальна послідовність дій користувача, які він має зробити, щоб виконати поставлене завдання **(4);**

Для одержання чіткої картини процесу, що відбувається, необхідно обґрунтувати кожну можливу дію користувача, покладаючись на загальне знання цільової аудиторії і на можливості зворотного зв'язка, наданого інтерфейсом та відповівши на перевірочні питання **(4*);**

Знайшовши проблемні місця в інтерфейсі – запропонувати покращення **(5).**

Список обов'язкових при проведенні КНП питань:

- Чи догадається користувач і чи буде намагатися зробити необхідні для виконання завдання дії?
- Чи перебувають керуючі елементи (кнопки, меню, перемикачі) в області видимості? Наскільки візуально доступна інформація про можливі дії?
- Якщо наявність керуючого елемента була помічена, чи зможе користувач розпізнати, що саме ця кнопка, пункт меню й тому подібне потрібні йому для виконання завдання?
- Якщо користувач зумів скористатися потрібним елементом керування, як він може довідатися про його доречність і правильність у цей момент, щоб із упевненістю перейти до наступної дії?

Використання контрольних списків є ефективним і економічним засобом підвищення якості інтерфейсів.

Природно, що в кожному конкретному випадку необхідно розробляти свій власний контрольний список, оскільки він має враховувати специфіку програмного засобу, що

розроблюється, і можливості засобів розробки. Тому наведений нижче контрольний список є скоріше шаблоном, у якому представлені основні розділи стандарту ISO-9241.

Вимоги до елементів керування

Кнопки

- Всі кнопки, що запускають дії, мають текст в інфінітивній формі дієслова (приклад: шукати), а не іншу частину мови або форму дієслова (приклад: готовий). Давати кнопці текст "ОК" можна, тільки якщо яке-небудь дієслово не вміщується на кнопці;
- Між кнопками, що розміщені поруч, має бути порожній простір, клацання по якому не спрацьовується;
- Недоступні команди не зникають із екрана, а стають заблокованими;
- Частотні кнопки супроводжуються не тільки текстом, але й піктограмами; кнопки, що використовуються рідко - тільки текстовими підписами.

Поля уведення

- Полях уведення вже мають найбільш імовірні значення;
- Якщо в полі вводиться чисельне значення, границі діапазону виводяться в спливаючій підказці;
- Якщо в полі вводиться чисельне значення з обмеженого діапазону, поле супроводжується Spinner-ом;
- Довжина полів не менше, і, по можливості, не більше, довжини даних, що вводяться в них.
- Якщо поле призначене для уведення помітної кількості тексту, воно многорядкове.

Списки

- Списки вже мають найбільш імовірні значення;
- Якщо список містить більше 50 елементів, використовується фільтр або режим пошуку;
- Немає коротких списків (менш п'яти елементів); такі списки представлені як групи радіокнопок або чекбоксів;
- Ширина списків не менше ширини вхідних у них елементів;

- Елементи списку відсортовані; або структурно, тобто по загальних ознаках, або за алфавітом, або по частотності (тільки списки менше 7 елементів);
- Якщо в списку більше 50 відсортованих за алфавітом елементів, першими трьома елементами є найбільш частотні елементи. Вони також повторюються на своїх алфавітних місцях.

Чекбокси й радіокнопки

- Якщо чекбоксів у групі більше 10, вводиться додатковий, що виставляє/знімає всі чекбокси;
- Чекбокси й радіокнопки усередині своїх груп розставлені по вертикалі;
- Якщо у вікні, крім термінаційних кнопок, є тільки набір радіокнопок, подвійне клацання по радіокнопці встановлює її й закриває вікно.

Взаємодія

- Система, завершивши тривалу операцію (більше мінути роботи), пищить через вбудований динамік комп'ютера;
- Якщо в інтерфейсі не використовується безпосереднього маніпулювання, система не має своїх курсорів. Якщо безпосереднє маніпулювання є, свої курсори застосовуються тільки якщо аналогів з ОС не існує.

тощо.

9 Візуалізація даних. Афорданс. Золотий переріз. Основи теорії кольору. Класичні схеми кольорів. Сприйняття кольорів.

Призначення будь-якого інтерфейсу – забезпечити комунікацію між двома системами, у випадку з програмним інтерфейсом (ПІ), між людиною та комп’ютером.

Знання принципів візуального сприйняття інформації людиною є необхідним при проектуванні ПІ.

Важливо усвідомлювати відмінності між подачею інформації та її сприйманням у реальному світі і можливостями її відображення та її сприйманням людиною за посередництвом комп’ютера.

Людина завжди підсвідомо намагається знайти якісь закономірності, тому важливо навчитися правильно їх подавати на екрані комп’ютера за допомогою засобів візуального подання інформації.

Необхідно подавати інформацію використовуючи принцип графічної цілісності:

- максимально точно відображати дані;
- максимально візуально відображати закономірності, які несе в себе інформація.

Це дозволяє скоротити час, що затрачується на сприйняття поданої інформації, за рахунок того, що візуально виражені закономірності сприймаються набагато швидше й легше, ніж чисельні або взагалі не виражені у явний спосіб.

Термін афорданс означає ситуацію, при якій об'єкт показує суб'єктові спосіб свого використання своїми невід'ємними властивостями.

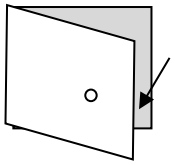

Рисунок 1.7 – Афорданс двері

Наприклад, напис "На себе" на двері не є афордансом, а вид дверей, що підказує людині, що вона відкривається на себе, несе в собі афорданс.

Користь афорданса полягає в тім, що він дозволяє користувачам обходитися без будь-якого попереднього навчання, завдяки цьому афорданс є найефективнішим і надійним засобом забезпечення зрозумілості.

Існує кілька способів передачі афорданса у комп’ютерних системах, з яких самими значними є наступні чотири:

- мапинг, або повторення конфігурації об'єктів конфігурацією елементів керування;
- видима приналежність керуючих елементів об'єкту;
- візуальний збіг афордансов екранних об'єктів з такими ж афордансами об'єктів реального миру (кнопка в реальному світі пропонує користувачеві натиснути на неї; псевдотривимірна кнопка в програмному інтерфейсі також пропонує натиснути на неї за аналогією);
- зміна властивостей об'єкта при підведенні до нього курсору (блідий аналог тактильної взаємодії).

Інтерфейс, за звичай, призначений для тривалого використання, що зближає його із книжковим дизайном і дизайном середовища перебування.

Але інтерфейс зароджується у функціональності, "інтерфейс ні до чого" просто не може існувати.

Тому необхідно шукати компроміс між естетикою й функцією.

Принципи багатьох напрямків дизайну цілком можуть бути застосовні при дизайні інтерфейсу. Головний з них:

- намагайтеся зробити інтерфейс максимально насиченим візуальними закономірностями;
- чим більше закономірностей, тим більше гармонії.

При людино-машинній взаємодії 90 відсотків інформації передається візуально, через дисплей, і сприймається людиною очами. Оскільки колір є сильним засобом залучення уваги, то при проектуванні ПІ необхідно правильно користуватися основами теорії кольору.

У сприйнятті кольору були виділені наступні феномени:

- зв'язок кольору з різними перцептивними модальностями: смаком, кольором, тактильними відчуттями, сприйняттям простору й рухи;
- індукція кольором характерного психічного стану;
- можливість використання кольору як засобу вираження відносини.

Теорія кольору - це набір принципів, який використовується для створення гармонічних сполучень кольорів.

Колірні співвідношення можуть бути візуально представлені спектральним колом - замкнутим колірним спектром.

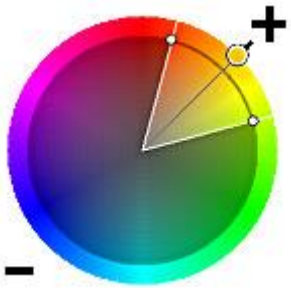

Рисунок 1.8 – Спектральне коло (Колірне колесо, Color Wheel), RGB

Через сторіччя після Ньютона, Йоханн Гєте почав вивчати психологічний вплив кольорів. Він помітив, що синій дає почуття прохолоди, а жовтий – зігріває.

Й. Гєте поділ кольори на дві групи:

- «плюс» (від червоного, жовтогарячого до жовтого);
- «мінус» (від зеленого, фіолетового до синього).

Кольори зі знаком «+» підвищують емоційне схвилювання й життєрадісність.

Кольори зі знаком «–» асоціюються зі слабкістю й поганим настроєм.

Яскравість - визначає наскільки колір світлий або темний, у тому розумінні, наскільки він близький до білого. Залежить від кількості світла, що випромінюється кольором. Найпростіший спосіб запам'ятати це поняття - уявити собі шкалу сірого кольору, зі зміною чорного на білий, у якій містяться всі можливі варіанти монохроматичного сірого кольору.

Гармонічні колірні комбінації називаються **колірними схемами.**

Відповідно до колірної теорії гармонійні колірні комбінації створюють:

- будь-які два кольори розташовані один навпроти одного на спектральному колі;
- будь-які три кольори, що формують рівносторонній трикутник;
- будь-які чотири кольори, що формують прямокутник (дві пари кольорів один навпроти одного).

Класичні колірні схеми мають такі назви:

- монохроматична;
- суміжна;
- комплементарна;
- суміжно-комплементарна;
- тріадна;
- тетріадна (двокомплементарна).

Згідно принципу «Безпосереднього маніпулювання даними»:

- введені вхідні дані мають бути відразу відображені (бути добре помітними);

- дизайн має «сигналити» про те, якою має бути «правильна когнітивна модель користувача» для даного інтерфейсу, тобто спрямовувати його найкоротшим шляхом для отримання позитивного результату;

Шість основних питань, які проектувальник має пам’ятати про його пристрій/інтерфейс і його користувача - *як легко* кожен користувач зможе:

- визначити функції пристрою (інтерфейсу)?
- визначити, які дії можливі?
- визначити, як перейти від наміру до дії?
- як виконати дію?
- визначити стан системи на даний момент?
- інтерпретувати стан як (не) відповідний після скоєної дії?

Правильне використання таких засобів візуального дизайну як - групування даних, лінійки, сітки, вирівнювання, шрифти, колір – значно сприяють досягнення користувачем

позитивного результату. Методи зниження когнітивної навантаження користувачів також допомагають цьому.

Виходячи з вищесказаного, необхідно проектувати інтерфейси так, щоб вони дозволяли користувачу: швидко вловлювати асоціації та закономірності для розуміння дій, що необхідні для виконання завдання, зменшувати завантаження короткочасної пам'яті в процесі взаємодії з інтерфейсом.

Інформація, представлена візуально, завжди більш зрозуміла, ніж текстова або цифрова.

Для зменшення проблем порозуміння необхідно забезпечувати:

- візуалізацію (аффорданси, символи і аналогії);

- зворотний зв'язок між системою і користувачем (щоб завжди було зрозуміло, що відбувається);

- використання існуючих стандартів - «бути послідовним»;

- відсутність «деструктивних» операцій, звідси необхідність undo (знання про те, що ніякі дії користувача не приведуть до краху, дає можливість користувачеві вивчати систему більш широко);

- зрозумілість доступних дій: всі наявні операції мають бути зрозумілі після детального вивчення меню (люди знайомляться з новим сайтом - вивчають опції його меню);

- надійність - не повинно бути випадкових непередбачуваних подій.

Інтерфейси допомагають користувачеві, якщо вони:

- заохочують експерименти;

- підтримують процес вивчення і знижують кількість помилок;

- показують (вказують) на зміни;

- замінюють повільні розрахунки в короткостроковій пам'яті швидким сприйняттям;

- підтримують асоціативне мислення, особливо для фахівців (що добре розбираються в проблемної області даного інтерфейсу);

- збільшують ефективність роботи;

- сприяють спільній роботі групи людей.

Розглянемо приклад інтерфейсу, у якому порушується принцип «безпосереднього маніпулювання даними». Після аналізу проблеми буде запропонований паперовий прототип редизайну інтерфейсу, який усуває помилки.

Опис проблеми в телефоні: яке з напрямків руху джойстика приведе до зниження гучності? Відповідь неоднозначний. Горизонтальний слайдер суперечить підказці(стрілки вверх та вниз).

Рисунок 1.9 – Приклад інтерфейсу, у якому порушується принцип «безпосереднього маніпулювання даними» у програмному інтерфейсі

Рішення (редизайн) інтерфейсів наведений на рис. 1.10:

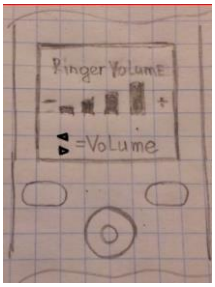

Рисунок 1.10 – Паперовий прототип редизайну інтерфейсу

9.1 Запитання та завдання для самоперевірки

1 Які недоліки та переваги монохромної колірної схеми? Коли її краще використовувати?

2 Який колір з прислів'я про мисливця («Каждый охотник желает знать, где сидит фазан») має найбільше значення яскравості (brightness)? Як це можна визначити чи обрахувати?

3 Якщо всі можливі кольори можна уявити за допомогою «сфери кольорів» , верхня точка якої має значення RGB(255,255,255), нижня RGB(0,0,0) , визначте точку, де буде знаходитися на цієї сфері точка зі значенням RGB(0,0,255)?

4 Що таке афорданс? Наведіть приклади. Перелічить способи його передачі на екрані.

5 Що таке гармонічне поєднання кольорів? За якими правилами можна знайти такі поєднання?

6 Задля чого краще використовувати вирівнювання та групування елементів інтерфейсу?

7 Які чотири основні цілі візуального дизайну при проектуванні взаємодії в інтерфейсах?

8 При розміщенні інформації на web-сторінці, які її частини є більш пріоритетними? Чому?

9 Які техніки проектування інтерфейсів сприяють більш зручному процесу сортування або пошуку даних користувачем?

10 Електронна підтримка. Засоби спрощення навчання користувача. Піктограми.

Одне з визначень інтерфейсу: "У більше широкому сенсі **інтерфейс** містить у собі пристрої введення і виведення (апаратне забезпечення), програмне забезпечення, що обслуговує їх, а також все те, що **допомагає користувачеві взаємодіяти з комп'ютером (програмою)**, **у тому числі** документація й **довідкова інформація**"

Основна мета розробників програмного інтерфейсу - зробити все можливе, щоб полегшити користувачеві створення його концептуальної моделі ПІ, тобто спростити процес роботи із програмою.

Розробники інтерфейсів у своїй роботі все більше **орієнтуються на користувачів, які вчаться в процесі взаємодії із програмою**. І дійсно, більшість людей, користуючись ПП, одержують нову інформацію, аналізують її і згодом перетворюють її у свій досвід - тобто вчаться.

Працюючи із програмою, користувачі навчаються. Тому необхідно пам'ятати мінімальний список засобів спрощення процесу навчання:

- загальна "зрозумілість" системи (метафори, афорданс, піктограми, візуальні засоби, колір, анімація, місце розташування , розміри об'єктів,..тощо)
- навчальні матеріали (електронна підтримка - довідкові системи, контекстні, підказки, Майстри (Wizards), демороліки, навігаційні карти, індикатори режиму).

Люди роблять що-небудь тільки при наявності стимулу. *Якщо користувачеві навчатися роботі із програмним продуктом легко* (тобто йому не складно побудувати концептуальну модель для роботи з ним), *то йому буде досить слабкого стимулу для подальшого навчання й роботи із програмою;* якщо важко, то стимул прийдеться збільшувати.

Щоб користувач почав процес навчання, йому важливо розповісти про функціональність системи, причому не просто повідомляти про наявність тієї або іншої функції, але і розповісти, яку перевагу він одержить, навчившись цієї функції, тобто

пояснити **"що зробити => що це дає"**. А далі користувач сам буде вчити, якщо, звичайно, стимул достатній.

Відповідно до одного з принципів юзабилити (*usability - зручність роботи, простота у використанні*), ПП мають пропонувати такі навчальні допоміжні елементи ***щоб спростити процес навчання користувача***:

- візуальний зворотний зв'язок;
- систему електронної допомоги (довідка по F1, контекстна підказка);
- демонстраційні ознайомлювальні ролики;
- Майстрів (Wizards).

Електронна підтримка (ЕП) - це "інтуїтивно зрозумілі способи одержання рекомендацій, порад, потрібної інформації".

ЕП може складатися з контекстної, пояснювальної, довідкової або навчальної інформації. У навчальній інформації програм вищого рівня пропонуються зразки рішення типових завдань. Це допомагає користувачам вивчити, як працює продукт, використовуючи аналогічні дані і завдання (приклади в MSDN, варіанти завдання аргументів функцій при роботі з Excel, …).

Вдало розроблена й зручна у використанні електронна підтримка може значно зменшити витрати на тренінг і навчання в будь-якій компанії (статистика затверджує, що левову частину у вартості ПП становлять саме ці два види послуг, які надаються користувачам компанією-виробником).

Зазвичай у користувачів при роботі із ПП виникають такі питання:

- що я можу зробити за допомогою цієї програми?(нова комп'ютерна іграшка)
- що це таке? (в MS Paint)
- що воно робить? (довідкова інформація про яку-небудь функцію)
- де я зараз, куди можу перейти? (карта сайту)
- що трапилося, що це означає? (обраний уже існуючий логін)
- що я можу зробити зараз? (контекстна довідка в Word)

ЕП має відповідати саме на такі питання. Інформація надається по запиту користувача, саме тоді, коли він її потребує.

Піктограми – це знайомі всім маленькі картинки, які слугують для позначення кнопок та інших об'єктів, що є необхідною ознакою сучасних інтерфейсів.

Переваги піктограм:

- Піктограми можуть значним чином вплинути на ясність та посилити привабливість додатка;

- використання піктограм дозволяє набагато спростити процес інтернаціоналізації ПП.

Піктограми доречні при нестачі площі, де можна виводити інформацію (пояснюючі слова замість них забирають багато місця) або коли тимчасово треба скрити/показати якусь інформацію.

Недоліки піктограм: їх треба створювати та тестувати з урахуванням фокусної групи з цільової аудиторії користувачів (різні культури, звичаї та особливості сприйняття).

Числа є поганими засобами для швидкої передачі їх реальних значень для **взаємозв'язку між ними**. **Бажаніше представляти числові дані**, показуючи їх не як послідовність, але **як візуальні об'єкти**, **властивості яких тим або іншим способом пов'язані із самим числом**.

10.1 Запитання та завдання для самоперевірки

1 Чому необхідно спрощувати процес навчання користувачів ПП? Спрогнозуйте дії користувача, якщо йому інтерфейс ПП йому не зрозумілий.

2 Які переваги та недоліки використання піктограм Ви знаєте?

3 Що таке електронна підтримка? Який з її елементів, може підказати користувачу його місце знаходження при роботі з ПП?

4 Які дії користувача вказують на те, що він не може знайти потрібну інформацію на сайті?

5 Яким даним надає перевагу більшість користувачів: текстовим або візуальним? Чому?

Перелік посилань

1 Раскин Д. Интерфейс: новые направления при проектировании компьютерных систем, [Текст]: пер. с англ., -СПб: Символ-Плюс, 2015, -272 с.

2 Алан Купер, Роберт Рейман, Дэвид Кронин, Кристофер Носсел / Интерфейс. Основы проектирования взаимодействия, [Текст]: -СПб: Питер, 2016, -720 с.

3 Элам, К. Графический дизайн. Принцип сетки = Grid Systems: Principles of Organazing Type (Desihn briefs) [Текст]:: пер. с англ./ К. Элам. –Санкт-Перербург: Питер, 2014, -120 с.

4 Круг С. Веб-дизайн: книга Стива Круга или «не заставляйте меня думать!» [Текст]: 2-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2008. – 224 с.: цв. ил.

5 Лидвелл, У. Универсальные принципы дизайна [Текст]: пер. с англ./ У Лидвелл, К. Холден, Д. Батлер. -СПб: Питер, 2012, -272 с.

6 Мандел Т.: Разработка пользовательского интерфейса [Текст]: Пер. с англ. – М. : ДМК Пресс, 2001, - 416 с

7 Унгер, Р. UX-дизайн. Практическое руководство по проектированию опыта взаимодействия [Текст]:пер. с англ./Р. Унгер, К. Чендлер. –СПб -М: Символ-Плюс, 2011, - 336 с.

8 Дональд Норман "Дизайн обыденных вещей“ (Donald Norman, Design of everyday things).

9 Нильсен Я. Веб-дизайн: анализ удобства использования веб-сайтов по движению глаз = Eyetracing Web Usability [Текст]: пер. с англ./ Я. Нильсен, К. Перниче.-М. –СПб, -К: Вильямс, 2010, -480 с.

10 nngroup.com/articles/ten-usability-heuristics ; nngroup.com/articles

11 youtube.com/watch?v=Edqjao4mmxM

12 https://www.coursera.org/lecture/datavisualization/2-3-1-tuftes-design-rules-4yiGl

13 https://material.io/resources/color

14 https://color.adobe.com/create/color-wheel