Inżynieria oprogramowania

Inżynieria oprogramowania (*software engineering*), wiedza i umiejętność przydatna do budowy programów, szczególnie dużych systemów oprogramowania. Obejmuje tworzenie specyfikacji, metody programowania, aspekty uruchamiania i testowania oraz opracowywanie dokumentacji. Odrębnymi działami inżynierii oprogramowania są dowodzenie poprawności programów oraz zagadnienia przenośności. Znajomość inżynierii oprogramowania pomaga w wydajnym konstruowaniu niezawodnego oprogramowania, także w sposób automatyczny.

Działania składające się na inżynierię oprogramowania:
•specyfikowanie,
•tworzenie,
•zarządzanie,
•rozwijanie
systemów komputerowych.

System komputerowy składa się z programów, plików konfiguracyjnych, dokumentacji systemowej, dokumentacji użytkownika.

1. Tworzenie dokumentacji projektowej

Jednym z najbardziej niedocenianych elementów planowania i realizacji projektu informatycznego jest przygotowywanie jego dokumentacji. Z reguły spycha się ją na sam koniec, gdy wszystko jest gotowe, wprowadza zmiany z opóźnieniem (lub w ogóle nie wprowadza), a całość jest nie nazbyt czytelna dla większości zainteresowanych.

Dokumentacja jest pomyślana jako łącznik pomiędzy oprogramowaniem komputerowym a ludźmi, którzy będą go używać, zarządzać nim, rozwijać czy administrować.

O wielu aspektach dokumentacji nie pamięta się – co zdecydowanie obniża jakość całego projektu, zwłaszcza wtedy, gdy projekt trwa długo, podlega wielu zmianom, a zespół go przygotowujący zmienia swój skład.

Wiele programów jest uważane przez ich użytkowników za źle zrobione lub trudne w użyciu z tego tylko powodu, że nie rozumieją oni ich działania. Dobra dokumentacja może to przełamać. Zła – utrudnia pracę programem, przedłuża wykonywanie często podejmowanych czynności, co powoduje spadek efektywności; słowem – powoduje straty, niezbyt może wymierne często, ale dotkliwe.

DOKUMENTACJA PROGRAMU

Dokumentacja programu składa się z dwóch części:
- podręcznika użytkownika
- dokumentacji technicznej.

Podręcznik użytkownika jest przeznaczony dla przyszłych użytkowników programu. Powinien on zawierać podstawowe informacje o programie, w szczególności pełny opis sposobu korzystania z programu.

Dokumentacja techniczna stanowi najszerszy opis programu. Powinna ona zawierać pełny zestaw szczegółowych informacji dotyczących budowy i działania programu.

PODRĘCZNIK UŻYTKOWNIKA – przykładowy schemat

1. Informacje ogólne
   1.1. Autor, data opracowania podręcznika.
   1.2. Opis notacji stosowanej w dokumentacji.
   1.3. Spis treści

2. Charakterystyka programu
   2.1. Opis programu.
   2.2. Funkcjonalności.
   2.3. Przegląd zastosowań programu.

3. Środowisko programu
   3.1. Dane wejściowe (znaczenie, format, sposób wprowadzania, warunki poprawności, reakcja na błędy w danych).
   3.2. Komunikacja z użytkownikiem (jw.)
   3.3. Wyniki (jw.)
   3.4. Przykłady danych wejściowych i wyników programu.

4. Sytuacje niepoprawne
   4.1. Wykaz komunikatów diagnostycznych.
   4.2. Błędy wykonania (opis błędu, warunki jego powstania).

5. Literatura

Dokumentacja użytkownika powinna być przygotowana prostym językiem, krótkimi zdaniami, unikając technicznego slangu. Najlepiej, by jak najczęściej umieszczone w niej były listy czynności do wykonania i przykłady. Układ tekstu, struktura oraz oprawa ilustracyjna mają maksymalnie ułatwić znalezienie informacji i zrozumienie problemu.

DOKUMENTACJA TECHNICZNA – przykładowy schemat

1. Informacje ogólne
   1.1. Autor, data opracowania dokumentacji.
   1.2. Krótki opis programu.
   1.3. Spis treści.

2. Charakterystyka programu
   2.1. Przeznaczenie programu
   2.2. Przegląd zastosowań programu

3. Struktura programu
   3.1. Opis plików zewnętrznych (organizacja, użycie).
   3.2. Globalne struktury danych (opis, przeznaczenie, użytkownicy).
   3.3. Podział na moduły, schemat komunikacji między modułami.
   3.4. Wykaz używanych modułów systemowych.

4. Opis modułu
   4.1. Informacje ogólne (zwięzła charakterystyka modułu).
   4.2. Opis funkcjonalny modułu
      4.2.1. Przeznaczenie modułu.
      4.2.2. Sposób wykorzystania modułu
         (procedury - przeznaczenie, parametry, sekwencja wołania;
         moduły korzystające z opisywanego modułu).
      4.2.3. Korzystanie z innych modułów i nielokalnych struktur danych.
   4.3. Charakterystyka działania modułu
      4.3.1. Szczegółowy opis algorytmu.
      4.3.2. Lokalne struktury danych.
      4.3.3. Budowa modułu (procedury pomocnicze - przeznaczenie, parametry,
         korzystanie z nielokalnych struktur danych).
      4.3.4. Wymagania czasowe i pamięciowe modułu.
   4.4. Sytuacje niepoprawne
      4.4.1. Kontrola poprawności danych wejściowych.
      4.4.2. Wykaz komunikatów diagnostycznych.
      4.4.3. Błędy wykonania (opis błędu, warunki jego powstania).

5. Literatura

6. Załączniki
   6.1. Informacja o przebiegu testowania modułów.
   6.2. Przykładowe testy programu (dane testowe + omówienie testu).
   6.3. Wydruk programu.

ZADANIE 1

Na podstawie przykładowego schematu podręcznika użytkownika utworzyć dokumentację użytkownika dla jednego z programów: Kalkulator (Microsoft), Paint (Microsoft), Adobe Reader, (ew. inne).

JAVADOC

Javadoc to generator dokumentacji stworzony przez firmę Sun Microsystems. Narzędzie to generuje dokumentację kodu źródłowego Javy na podstawie zamieszczonych w kodzie komentarzy Javadoc.

Wiele nowoczesnych IDE wspomaga tworzenie dokumentacji i pozwala na automatyczne jej generowanie. Umożliwiają one również wykorzystanie takiej dokumentacji jako pomoc dla programisty podczas pisania kodu.

Komentarz Javadoc oddzielony jest znacznikami /** i */, które sprawiają, że ich zawartość (czyli to, co znajduje się między nimi), jest ignorowana przez kompilator. Pierwsza jego linia to opis metody lub klasy, która zadeklarowana jest poniżej. Dalej znajduje się pewna liczba opcjonalnych tagów, które z kolei opisują parametry metody (@param), wartość zwracaną (@return) itp.

```
/**
 * Validates a chess move. Use {@link #doMove(int, int, int, int)} to move a piece.
 *
 * @param theFromFile file from which a piece is being moved
 * @param theFromRank rank from which a piece is being moved
 * @param theToFile   file to which a piece is being moved
 * @param theToRank   rank to which a piece is being moved
 * @return            true if the chess move is valid, otherwise false
 */
boolean isValidMove(int theFromFile, int theFromRank, int theToFile, int theToRank)
{
    ...
}

/**
 * Move a chess piece.
 *
 * @see java.math.RoundingMode
 */
boolean doMove(int theFromFile, int theFromRank, int theToFile, int theToRank)
{
    ...
}
```

Opis znaczników dokumentacyjnych:

@author autor	Informacje o autorze (autorach).
@version wersja	Informacja o wersji.
@see klasa lub klasa#metoda lub napis	Generuje odwołanie do dokumentacji innej klasy, metody lub informację o odwołaniu.
@deprecated	Oznacza składowe, które zostały zastąpione przez nowsze wersje i uzywanie ich nie jest zalecane.

Dla metod:

@param parametr opis	Opisuje jeden parametr metody. Może być ich dowolnie dużo, jednak na ogół jest ich tyle ile parametrów metody, w takim samym porządku.
@return opis	Opisuje wartość zwracaną przez metodę.
@throws wyjątek opis (albo **@exception** wyjątek opis)	Opisuje wyjątek, który może być rzucony przez tą metodę.

ZADANIE 2

- W środowisku NetBeans przygotować dowolny mały projekt (np. HelloWorld)
- Utworzyć dokumentację Javadoc (Run -> Generate Javadoc)
- Dodać komentarz klasy (np. autor, opis)
- Dodać komentarz zmiennej (opis)
- Dodać komentarz metody (parametry, co zwraca, wyjątek)
- Wygenerować dokumentację HTML