Inżynieria oprogramowania

Część 1: Tworzenie dokumentacji projektowej

WPROWADZENIE

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 UŻYTKOWNIKA

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

W tego typu dokumentacji należy przekazać klika znaczących informacji:

- Opis funkcjonalny

Należy bardzo krótko opisać usługi oferowane przez system.

- Podręcznik instalatora

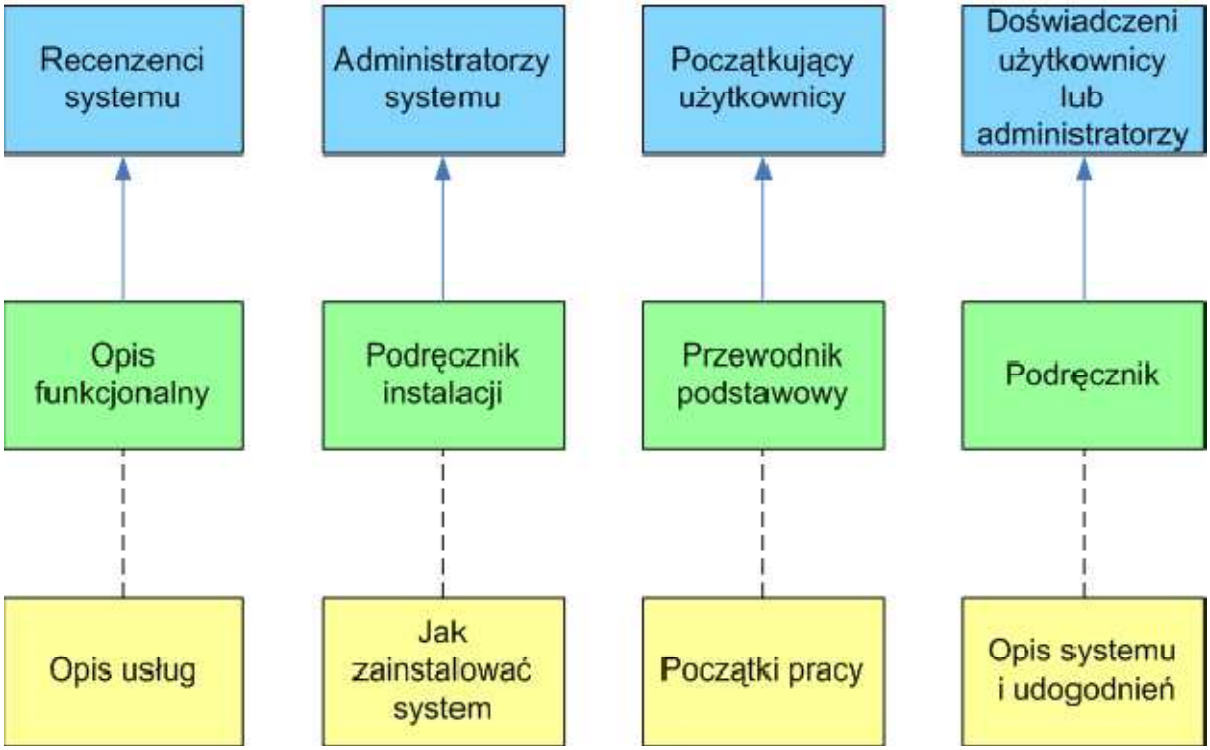
Powinien obejmować szczegółowe informacje o instalacji systemu.

- Przewodnik podstawowy

Powinien obejmować nieformalne wprowadzenie do systemu, w którym należy przedstawić jego „normalne” użycie.

- Podręcznik

Powinien zawierać opis udogodnień systemu oraz ich wykorzystywania.



Rysunek 1. Podstawowe informacje zawarte w dokumentacji użytkownika

PRZYKŁADOWY SCHEMAT PODRĘCZNIKA UŻYTKOWNIKA

1. Informacje ogólne

1.1. Autor, data opracowania podręcznika.

1.2. Krótki opis programu.

1.3. Spis treści.

2. Charakterystyka problemu

2.1. Teoretyczne wprowadzenie.

2.2. Opis notacji stosowanej w dokumentacji.

2.3. Dokładne sformułowanie problemu.

2.4. 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.

ZADANIE DO ZREALIZOWANIA

1. Zgodnie z przykładowym schematem podręcznika użytkownika stworzyć dokumentację użytkownika dla programu Kalkulator (Microsoft).

DOKUMENTACJA PROJEKTOWA

Dokumentacja projektowa bardzo często nazywana jest deweloperską. Dokumentacja przeznaczona dla deweloperów systemu, którzy rozwijają lub/i poprawiają kod źródłowy systemu. Składa się z 2 części:

- dokumentacji kodu (generowanej automatycznie za pomocą narzędzi typu JavaDoc lub DoxyGen i komentarzy w kodach Źródłowych)
- dokumentów uzupełniających, omawiających sposób kompilacji i ogólną architekturę kodów źródłowych systemu.

NIEZBĘDNE ELEMENTY DOKUMENTACJI PROJEKTOWEJ

- Opis plików programu – każdy plik powinien zawierać informacje o autorze, dacie modyfikacji i nazwie
- Opis klas – każda klasa powinna zwierać opis dotyczący jej przeznaczenia
- Opis zmiennych i atrybutów klas – każda zmienna powinna zawierać komentarz na temat jej przeznaczenia oraz możliwych wartości które możne przyjmować
- Opis metod i funkcji – każda funkcja powinna zawierać opis jej działania, dokładny opis jej parametrów (co oznaczają) oraz opis wartości przekazywanej przez funkcję
- Ponadto w dokumentacji powinny zostać opisane mechanizmy tj. protokoły transmisji danych, formatu plików itp.

DOKUMENTACJA KODU Z WYKORZYSTANIEM DOXYGEN

Doxygen jest systemem do tworzenia dokumentacji kodów programów napisanych w C++, C, Java, Objective-C, Python, IDL (Corba); częściowo wspierane są PHP, C# i D. Za jego pomocą da się utworzyć dokumenty w plikach HTML, XML (strony internetowe), Latex (pdf, ps), stron podręcznika systemu Unix (man), rtf-ów. Projektem w rozumieniu Doxygen może być zarówno pojedynczy plik jak i całe drzewo plików.

Zachowanie Doxygen zależny od zawartości jego pliku konfiguracyjnego. Zwyczajowo jest to plik Doxyfile w katalogu głównym drzewa projektu. Aby ułatwić prace, możliwe jest wygenerowanie wzorca tekstowego pliku konfiguracyjnego, tak aby proces konfigurowania Doxygen ograniczyć do drobnych poprawek.

ZASADY DOKUMENTOWANIA KODU DLA DOXYGEN

Doxygen rozpoznaje bloki dokumentacji w plikach źródłowych dzięki umieszczeniu ich w specjalnych komentarzach. Bloki takie można tworzyć na kilka sposobów.

```
/**
* ...tekst...
*/
```

Przykład 1. Blok dokumentacji sposób 1

```
/*!
* ...tekst...
*/
```

Przykład 2. Blok dokumentacji sposob 2

```
///
/// ...tekst...
///
```

Przykład 3. Blok dokumentacji sposob 3

```
//!
//! ...tekst...
//!
```

Przykład 4. Blok dokumentacji sposob 4

Wewnątrz komentarzy można umieszczać specjalne polecenia (zaczynające się od znaku \ lub @ po którym następuje nazwa polecenia). Dla przykładu:

- \struct – dokumentacja struktur języka C
- \class - dokumentacja klasy
- \union - dokumentacja unii
- \enum - dokumentacja typu wyliczeniowego
- \fn - dokumentacja funkcji
- \return - dokumentowanie wartości zwracanej przez funkcję
- \param - dokumentowanie parametrów funkcji
- \var - dokumentacja zmiennej
- \def - dokumentacja #define.
- \typedef - dokumentacja definicji typ
- \file - dokumentacja pliku
- \brief – omówienie szczegółowe
- \see lub \sa - łącze do innych opisów (zobacz także)
- \namespace- dokumentacja przestrzeni nazw
- \package - dokumentacja pakietu Java
- \interface - dokumentacja interfejsu IDL

Opis pozostałych znaczników można znaleźć w dokumentacji Doxygena.

W komentarzu mogą znaleźć się listy

/*!

*** Lista zdarzeń:**

*** - zdarzenia myszy**

*** -# zdarzenie mouse move**

*** -# zdarzenie click \n**

*** Więcej informacji na temat zdarzenia click**

*** -# podwójne kliknięcie**

*** - zdarzenia klawiatury**

*** -# zdarzenie key down**

*** -# zdarzenie key up**

*****

*** Więcej tekstu tutaj.**

***/**

Przykład 5. Kodowanie listy (przykład wzięty z dokumentacji Doxygen)

Ponadto w komentarzach można używać większości znaczników html formatujących tekst, co jest szczególnie przydatne gdy planujemy wygenerować dokumentację w tym właśnie języku.

Dla przykładu poniżej przedstawiony został sposób realizacji listy wykorzystując znaczniki html.

/*!

*** Lista zdarzeń:**

*** **

*** zdarzenia myszy**

*** **

*** mouse move event**

*** mouse click event\n**

*** Więcej informacji na temat zdarzenia click**

*** podwójne kliknięcie**

*** **

*** zdarzenia klawiatury**

*** **

*** zdarzenie key down**

*** zdarzenie key up**

*** **

*** **

*** Więcej tekstu tutaj.**

***/**

Przykład 6. Kodowanie listy za pomocą znaczników HTML

PRZYKŁADY KOMENTARZY

W tej sekcji zaprezentowane zostaną przykłady użycia bloków komentarzy wraz z poleceniami specjalnymi.

KOMENTARZ PLIKU

/**

\file pierwszy.cpp

\author Jan Kowalski

\date 10.10.2010

\brief Progam symuluje działanie prawidłowe.

***/**

KOMENTARZ ZMIENNEJ (KRÓTKI KOMENTARZ)

```
char sTab[20] [100]; ///< Tablica dwuwymiarowa przechowywujaca pewne wartości
```

KOMENTARZ ZMIENNEJ (SZCZEGÓŁOWY KOMENTARZ)

```
int nLiczba = 0; /**<
```

\brief Zmienna z wartością liczby

*** Przyjmuje:**

*** - Wartosc 0, gdy lubi**

*** - Wartosc n, gdzie n jest ilością królików**

***/**

KOMENTARZ FUNKCJI

//! Funkcja dzielaca bufor z danymi wejsciowymi na poszczegolne linie (wywolywana wielokrotnie w funkcji int main())

/

Funkcja iteracyjnie przechodzi przez zadany bufor w poszukiwaniu znaku '$', jednoczesnie wczytujac po kolei wszystkie znaki, do bufora wyjsciowego, ktore nie sa '$', az do napotkania tegoz znaku.

\param buffer Bufor zawierajacy dane wejsciowe przekazane przez symulowane urzadzenie GPS

\param dest Bufor do ktorego zapisane zostaja poszczegolne linie

\param pos Pozycja od ktorej nalezy zaczac przeszukiwanie bufora wejsciowego

\return Funkcja przekazuje wartosc:

- -1 kod bledu w przypadku konca bufora wejsciowego

- i, gdzie i oznacza pozycje w buforze na ktorej zakonczone zostalo przeszukiwanie

***/**

int get_a_line(char *buffer, char *dest, int pos)

{...}

JAVADOC

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

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 Informacja o autorze (może być wielu).

@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 używanie ich nie jest zalecane.

Dla metod:

@param parametr opis Opisuje jeden parametr metody. Może ich być 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 tę metodę.

ZADANIE DO ZREALIZOWANIA

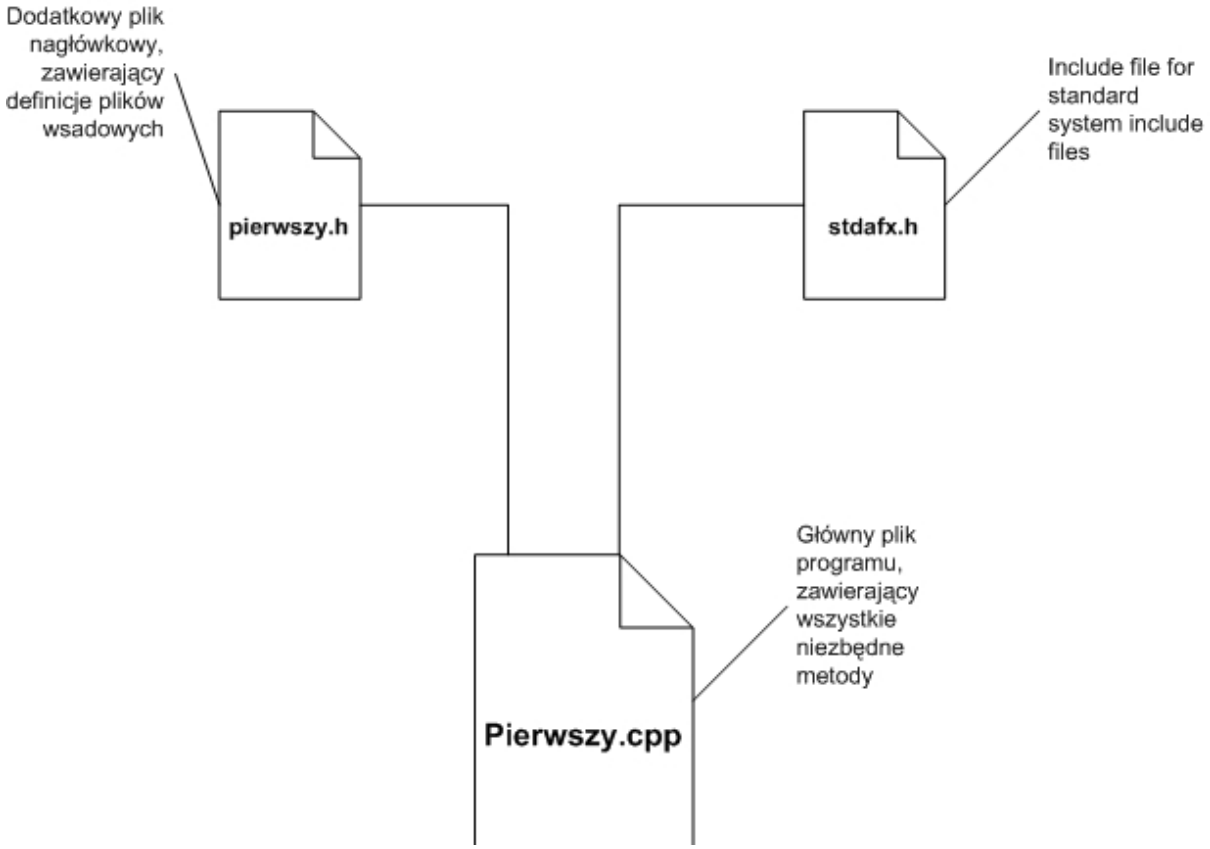
- W środowisku NetBeans przygotować dowolny projekt wypisujący HelloWorld.
- Utworzyć dokumentację Javadoc (Run -> Generate Javadoc)
- Dodać komentarz klasy (autor, opis, odwołanie do innej metody)

- Dodać komentarz pliku (autor, opis)
- Dodać komentarz zmiennej (opis)
- Dodać komentarz metody (parametry, co zwraca, wyjątek)
- Wygenerować dokumentację HTML

UZUPEŁNIENIE DOKUMENTACJI PROJEKTOWEJ

Jeżeli sposób uruchamiania, a w tym kompilacji nie jest oczywisty, należy go opisać. W tym celu dołączamy do dokumentacji kodu załącznik z instrukcją dotyczącą kompilacji i sposobu uruchomienia aplikacji. Dodatkowo należy również podać wszystkie pliki nagłówkowe niezbędne do uruchomienia aplikacji.

Ponadto należy również dołączyć ogólną architekturę kodów źródłowych systemu. Metoda takiego właśnie opisu przedstawiona została na rysunku 2. Metoda ta opera się na sporządzeniu ilustracji pokazującej powiązania między plikami składającej się na projektowany system. Przy sporządzaniu takiego rysunku należy pamiętać, iż nazwy wszystkich plików muszą zawierać rozszerzenia oraz wszystkie obiekty muszą posiadać powiązanie. Dodatkowo dobrze jest sporządzić krótki opis każdego z plików. Dobrą praktyką jest również manipulacja wielkością sygnatur plików wskazując tym samy na ważność konkretnych plików czy też zaznaczając gdzie znajduje się rdzeń systemu.



Rysunek 2. Przykład opisu ogólnej charakterystyki architektury systemu