**WOJSKOWA AKADEMIA TECHNICZNA**

**im. Jarosława Dąbrowskiego**

**WYDZIAŁ CYBERNETYKI**



PRACA DYPLOMOWA

STACJONARNE STUDIA I°

Temat: **Projekt i implementacja środowiska programowego do wspomagania oceny pracochłonności wytworzenia oprogramowania o znanej funkcjonalności**

Autor: Promotor pracy:

**Łukasz Rędziński dr hab. inż. Worwa Kazimierz**

**Warszawa 2020**

**OŚWIADCZENIE**

*„Wyrażam zgodę na udostępnianie mojej pracy przez Archiwum WAT”.*

Dnia ........................ .................................

(podpis)

*Pracę przyjąłem*

*promotor pracy  
dr hab. inż. Worwa Kazimierz*

*Tu będzie wstawione:  
Zadanie do pracy dyplomowej – str.1*

*Tu będzie wstawione:  
Zadanie do pracy dyplomowej – str.2*

# **Spis treści**

**WSTEP……………………………………………………………….........……………**

**ROZDZIAŁ I**

**POJĘCIA ROZMIARU I PRACOCHŁONNOŚCI WYTWORZENIA OPROGRAMOWANIA**

1.1 Rozmiar oraz pracochłonność wytworzenia oprogramowania……………...7

1.2 Miary rozmiaru……………………………………………………………...7

1.3 Miary pracochłonności……………………………………………………...8

1.4 Potrzeba oceny rozmiaru i pracochłonności………………………………...9

**ROZDZIAŁ II**

**WYBRANE METODY OCENY PRACOCHŁONNOŚCI WYTWORZENIA OPROGRAMOWANIA**

2.1 Dotychczasowe metody oceny pracochłonności

2.2 Metoda delficka

2.2.1 Problem metod eksperckich……………………………………….X

2.2.2 Algorytm metody…………………………………………………X

2.2.3 Wady i zalety metody……………………………………………..X

2.3 Metoda COCOMO II

2.3.1 Czynniki skali

2.3.2 Mnożniki pracochłonności

2.4 Metoda USE CASE POINTS

2.4.1 Czynniki złożoności środowiska

2.4.2 Czynniki złożoności technicznej

2.4.3 Aktorzy

2.4.4 Przypadki użycia

2.4.5 Wyznaczenie pracochłonności

**ROZDIAŁ III**

**CHARAKTERYSTYKA ISTNIEJĄCYCH ROZWIĄZAŃ**

3.1 Kalkulator COCOMO

3.2 Kalkulator USE CASE POINTS

**ROZDIAŁ IV**

**SPECYFIKACJA WYMAGAŃ I PROJEKT APLIKACJI**

4.1 Specyfikacja wymagań funkcjonalnych……………………………………X

4.2 Specyfikacja wymagań niefunkcjonalnych………………………………...X

4.3 Specyfikacja przypadków użycia…………………………………………..X

**ROZDIAŁ V**

**IMPLEMENTACJA APLIKACJI**

5.1 Wybór technologii………………………………………………………….X

5.2 Implementacja metody delfickiej…………………………………………..X

5.3 Implementacja metody COCOMO II………………………………………X

5.4 Implementacja metody USE CASE POINTS………………………………X

5.5 Implementacja interfejsu użytkownika……………………………………..X

5.4 Ocena wyników testowania………….……………………………………..X

**PODSUMOWANIE I WNIOSKI**………………………………...………………….X

**ZAKOŃCZENIE**……………………………………………………………..………X

**BIBLIOGRAFIA**………………………………………………………………..........X

**Wstęp**

**Rozdział 1. Pojęcia rozmiaru i pracochłonności wytworzenia oprogramowania.**

**1.1 Rozmiar oraz pracochłonność wytworzenia oprogramowania**

Rozmiar oprogramowania jest [1, 5] głównym parametrem wejściowym w większości modeli wytwarzania, modyfikacji i utrzymania oprogramowania. Rozmiar jest ważnym czynnikiem nakładu pracy oraz kosztów w projektach oprogramowania. Oprogramowanie jednak jest niematerialne, przez co fizyczne zmierzenie jego rozmiaru może okazać się niemożliwe. Jako że miary rozmiaru wykorzystywane są do oceny pracochłonności i wydajności procesu twórczego, rozmiar oprogramowania możemy interpretować jako rozmiar intelektualnego wkładu człowieka w bezpośrednie wytworzenie oprogramowania.

Pracochłonność rozumiana jest jako [1, 5] ilość czasu zużytego przez człowieka na wytworzenie oprogramowania. W procesie oceny kosztów projektu, nacisk kładziony jest zazwyczaj na ocenę pracochłonności członków zespołu. Powodem tego jest fakt, że godziny pracy zwykle napędzają całkowite koszty projektu. W związku z tym, pracochłonność traktowana jest jako suma pracy wszystkich osób pracujących przy projekcie.

**1.2 Miary rozmiaru oprogramowania**

W zależności od celu, rozmiar oprogramowania [1, 6] możemy interpretować w różny sposób. Rozmiar jest zazwyczaj używany jako kluczowy czynnik niezbędny do określenia nakładu, stosuje się go również w kontekście funkcjonalności, złożoności, redundancji i powtórnego użycia. W zależności od kontekstu użycia inna miara może okazać się bardziej odpowiednia. Stało się to przyczyną powstania wielu miar rozmiaru istotnych z punktu widzenia różnych atrybutów.

**LOC** – [6] linie kodu źródłowego (lines of code) – najprostsza miara rozmiaru oprogramowania. Wielkość ta daje ogólne pojęcie o skali programu jednak nie jest wystarczająca do bardziej szczegółowych analiz – wartość ta jest silnie zależna od użytego języka programowania. Jednostki:

**KLOC** – tysiące linii kodu źródłowego

**eLOC** – liczba linii kodu z pominięciem komentarzy, pustych linii czy nawiasów klamrowych

**ILOC** – liczba logicznych linii kodu (miara zależna od użytego języka programowania)

**Punkty funkcyjne ­­**– [2] znaczące grupy mierzalnego kodu z perspektywy użytkownika, a nie programisty. Uwzględniają one potrzeby użytkownika oraz zebraną w trakcie analizy wymagań funkcjonalność. Wartość punktów funkcyjnych to ważona suma pięciu składników, które cechują aplikację.

- 4 x liczba wejść zewnętrznych do systemu (lub typów transakcji),

- 5 x liczba typów lub raportów zewnętrznych,

- 10 x liczba wewnętrznych plików logicznych (zbiorów danych),

- 7 x liczba zewnętrznych plików interfejsu (zbiorów danych),

- 4 x liczba obsługiwanych zewnętrznych zapytań do systemu.

**1.3 Miary pracochłonności**

Najczęściej stosowaną jednostką nakładu jest [1, 11] **osobomiesiąc** rozumiany jako miesięczny rozmiar pracy jednej osoby. Pewien problem stanowi różna liczba dni roboczych w zależności od miesiąca, a także urlopy czy inne dni wolne od pracy. Z tego powodu osobomiesiąc traktuje się jako jedną dwunastą średniej liczby przepracowanych dni w roku.

Innymi znanymi jednostkami pracochłonności są [1, 8] **osobodni**, **osobogodziny** lub **FTE** (full time equivalent). Osobodzień oznacza rozmiar pracy, jaką jedna osoba może wykonać w ciągu jednego dnia, a osobogodzina – w ciągu godziny. FTE to etap przeliczeniowy, czyli zaangażowanie jednego lub więcej ludzi na poziomie jednego etapu. Na przykład 2 FTE to 2 osoby na pełen etat lub 4 osoby na poł etatu.

**1.4 Potrzeba oceny rozmiaru i pracochłonności**

Wiedza dotycząca rozmiaru oprogramowania oraz nakładu pracy ludzkiej potrzebnej do ukończenia projektu jest [1] podstawowym problemem stojącym przed każdym projektem wytwórczym. Wiedza ta jest wymagana do oszacowania kosztów czy czasu potrzebnego na wykonanie projektu. Zależność pomiędzy rozmiarem wytworzonego oprogramowania, a pracochłonnością służy do planowania procesu wytwórczego oraz konstruowania harmonogramów. Stanowi to podstawę skutecznego zarządzania projektem informatycznym. Z punktu widzenia zarządzania projektem informatycznym istotne są dwa aspekty określania rozmiaru i pracochłonności:

• Bezpośredni lub pośredni pomiar

• Oszacowanie

Pomiar dotyczy istniejącego oprogramowania. Oszacowanie zwykle dotyczy oprogramowania, które dopiero ma być wytworzone. Trafna ocena nakładu pracy ludzkiej przed rozpoczęciem projektu jest kluczowa dla powodzenia projektu informatycznego.

**Rozdział 2. Wybrane metody oceny pracochłonności wytworzenia oprogramowania**

**2.1 Dotychczasowe metody oceny pracochłonności**

Najpopularniejsze metody szacowania pracochłonności w projektach IT dzielą się na dwie kategorie – [6] metody statystyczne i metody eksperckie. Metody eksperckie opierają się na wiedzy i intuicji ekspertów. Do metod eksperckich zalicza się m.in. negocjacje, estimating poker i metodę delficką.

Metody statyczne korzystają z modeli matematycznych opierających się na analizy tysięcy projektów. Są to m.in. metoda punktów funkcyjnych czy COCOMO.

**2.2 Metoda delficka**

**2.2.1 Problem metod eksperckich**

Opieranie się na wiedzy jest praktyką wykorzystywaną dość powszechnie [16]. Jednak proces negocjacji w szerszym gronie ekspertów niesie za sobą pewne problemy związane z ludzką psychiką.

- **Problem zależności** – może wystąpić w przypadku, gdy pomiędzy ekspertami istnieje hierarchiczna zależność np. jak pomiędzy szefem i pracownikiem. Sprzeciwienie się pracodawcy na forum może nie być zadaniem łatwym.

**- Problem autorytetu** – opinia osób obdarzonych autorytetem może wpłynąć na opinię pozostałych ekspertów.

**- Problem urażonej dumy** – osoba świadoma swojego błędu, nie chcąc przyznać się do błędu może podtrzymywać swoje zdanie, unosząc się dumą.

Metoda delficka jest jedną z metod, próbujących niwelować problemy negocjacji.

**2.2.2 Algorytm metody**

Metoda delficka polega na [1, 16] **indywidualnym**, **niezależnym** szacowaniu przez kilku ekspertów. Stosując metodę delficką staramy się osiągnąć konsensus, czyli wypracować wspólne zdanie. Algorytm metody delfickiej, w wersji *Wideband Delphi*, składa się z siedmiu kroków.

1. Zaopatrzenie ekspertów w odpowiednie materiały przybliżające zakres projektu (np. dokument specyfikacji wymagań) oraz specjalny formularz estymacji. Każdy z ekspertów otrzymuje również czas, aby przeanalizować dokumentację.
2. Wymiana poglądów na temat projektu w gronie ekspertów.
3. Anonimowe głosowanie - każdy z ekspertów precyzuje własne oszacowanie pracochłonności.
4. Opracowanie wyników przez moderatora oraz przeniesienie ich na formularze estymacji poszczególnych ekspertów.

Sporządzony przez moderatora ***raport estymacji***, jest unikalny dla każdego eksperta. Przedstawia on ocenę własną eksperta, oceny pozostałych ekspertów oraz średnie oszacowanie. Umożliwia on weryfikację własnego stanowiska i jego krytyczną ocenę. Ekspert ma możliwość porównania wystawionych ocen ze średnią wartością oraz ocenami pozostałych osób. Nie jest natomiast podawana informacja umożliwiająca powiązanie ocen z konkretnymi osobami.

Analiza danych przedstawionych w raporcie ma przede wszystkich na celu skłonić autora oszacowań do refleksji i powtórnego przyjrzeniu się problemowi.

1. Analiza wyników w gronie ekspertów.

Aby zachować anonimowość negocjacji należy wystrzegać się podawania dokładnych wartości oszacowań. Bardziej istotne jest nakreślanie problemów, oraz obrazowanie przesłanek, w oparciu o które eksperci podejmowali swoje decyzje.

1. Powtarzanie procedury od kroku trzeciego, aż szacunki ekspertów nie będą wystarczająco zbliżone.
2. Oszacowanie końcowe.

Jeśli nie udało się wypracować wspólnej oceny dla wszystkich ekspertów, należy posłużyć się średnią ważoną, dla określenia oszacowania końcowego. Średnia ta uwzględnia ocenę pesymistyczną  oraz optymistyczną z wagą 1 oraz  ocenę średnią z wagą 4.

**Oszacowanie = (P + 4A + O) / 6**

***P*** - ocena pesymistyczna,

***A*** - ocena średnia,

***O*** - ocena optymistyczna.

Można powiedzieć, że przychylamy się do średniej oceny, mając na uwadze także skrajne oszacowania.

**2.2.3 Wady i zalety metody**

Jedną z największych zalet [1, 16] metody delfickiej jest możliwość wycofania się z podjętej wcześniej decyzji na każdym etapie negocjacji. Nawet eksperci z wieloletnim doświadczeniem mogą się mylić i nie chcieć do tego przyznać. Problem ten zostaje rozwiązany poprzez anonimowość głosowania. Negocjacje w metodzie delfickiej mają charakter burzy mózgów. Stymuluje to cały proces oraz umożliwia spojrzenie na problem z punktu widzenia wielu osób.

Niestety, metoda delficka jest dość wolna – wymaga dużo czasu i przygotowań. Wymaga zaangażowania dodatkowych osób do przygotowania formularza estymacji i opracowania wyników. Eksperci biorący udział w negocjacjach muszą posiadać odpowiednią wiedzę i doświadczenie, a metoda zakłada, że zgodność opinii ekspertów jest równoznaczna z ich prawdziwością i trafnością, co nie zawsze jest prawdą.

**2.3 Metoda COCOMO II**

Metoda COCOMO II [11, 16] bazuje na rozmiarze oprogramowania oraz opiera się na modelach matematycznych stworzonych na podstawie ponad 100 projektów informatycznych. W metodzie tej wyróżnia się 3 modele:

**Application Composition Model** - odpowiedni dla projektów wykorzystujących nowoczesne narzędzia do tworzenia graficznych interfejsów użytkownika.

**Early Design Model** – wykorzystywany we wczesnej fazie projektu, gdy kompletna architektura nie została jeszcze zdefiniowana.

**Post-Architecture Model** – stosowany, gdy została już opracowana architektura systemu. Jest najlepiej udokumentowanym oraz najbardziej kompletnym modelem do oceny nakładów pracy potrzebnych do stworzenia lub rozwoju oprogramowania komputerowego.

Podstawowy wzór do określenia pracochłonności w metodzie COCOMO II, modelu **Post-Architecture** to:

gdzie:

Size – wielkość oprogramowania w tysiącach linii kodu źródłowego (KLOC)

SF – czynniki skali (scale factors)

EM – mnożniki pracochłonności (effort multipliers)

Przy użyciu metody COCOMO II otrzymujemy wartość pracochłonności wyrażoną w **osobomiesiącach**. Dla projektów **prostych** wartości mnożników pracochłonności wynoszą 1.

**2.3.1 Czynniki skali**

W ramach metody COCOMO II wyszczególniono 5 czynników skali projektu [8, 9, 11, 16] są to:

**Typowość**

**Elastyczność**

**Zarządzanie Ryzykiem**

**Spójność zespołu**

**Dojrzałość procesu**

W zależności od oceny poszczególnych czynników wyznacza się ich wpływ w skali od bardzo niskiego do extra wysokiego. Wartości poszczególnych czynników skali przedstawiono w Tab. 1.

Tab. 1. Wartości czynników skali

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Bardzo niski** | **Niski** | **Normalny** | **Wysoki** | **Bardzo wysoki** | **Extra wysoki** |
| Typowość | 6.20 | 4.96 | 3.72 | 2.48 | 1.24 | 0.00 |
| Elastyczność | 5.07 | 4.05 | 3.04 | 2.03 | 1.01 | 0.00 |
| Zarządzanie ryzykiem | 7.07 | 5.65 | 4.24 | 2.83 | 1.41 | 0.00 |
| Spójność zespołu | 5.48 | 4.38 | 3.29 | 2.19 | 1.10 | 0.00 |
| Dojrzałość procesu | 7.80 | 6.24 | 4.68 | 3.12 | 1.56 | 0.00 |

Źródło: [10]

**Typowość** - określa, czy realizowany projekt, jest typowy dla danej organizacji oraz czy zawiera elementy innowacyjności

**Elastyczność** - określa czy istnieje potrzeba zgodności z dokumentem wymagań oraz z zewnętrznymi interfejsami

**Zarządzanie Ryzykiem** - bierze pod uwagę istnienie planu zarządzania ryzykiem, uszeregowania zadań, ustalenie budżetu

**Spójność zespołu** - ma na uwadze trudności w synchronizacji między uczestnikami projektu

**Dojrzałość procesu** - ocena dojrzałości procesu na podstawie modelu CMM

**2.3.2 Mnożniki pracochłonności**

Mnożniki pracochłonności [8, 9, 11, 16] wyszczególnione w COCOMO II to:

**Wymagana niezawodność systemu**

**Rozmiar użytej bazy danych**

**Re-używalność**

**Zakres wymaganej dokumentacji**

**Złożoność**

**Wymagania czasu wykonania**

**Ograniczenia pamięciowe**

**Płynność platformy tworzenia**

**Możliwości analityków**

**Możliwości programistów**

**Doświadczenie z aplikacją**

**Doświadczenie z platformą**

**Doświadczenie z językiem i narzędziami**

**Ciągłość zatrudnienia personelu**

**Użycie narzędzi programowych**

**Jakość komunikacji zespołu**

**Napięty harmonogram**

W zależności od oceny poszczególnych czynników wyznacza się ich wpływ w skali od bardzo niskiego do extra wysokiego. Wartości poszczególnych mnożników pracochłonności przedstawiono w Tab. 2.

Tab. 2. Wartości mnożników pracochłonności

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Bardzo niski** | **Niski** | **Normalny** | **Wysoki** | **Bardzo wysoki** | **Extra wysoki** |
| Wymagana niezawodność systemu | 0.82 | 0.92 | 1.00 | 1.10 | 1.26 |  |
| Rozmiar użytej bazy danych |  | 0.90 | 1.00 | 1.14 | 1.28 |  |
| Re-używalność | 0.73 | 0.87 | 1.00 | 1.17 | 1.34 | 1.74 |
| Zakres wymaganej dokumentacji |  | 0.95 | 1.00 | 1.07 | 1.15 | 1.24 |
| Złożoność | 0.81 | 0.91 | 1.00 | 1.11 | 1.23 |  |
| Wymagania czasu wykonania |  |  | 1.00 | 1.11 | 1.29 | 1.63 |
| Ograniczenia pamięciowe |  |  | 1.00 | 1.05 | 1.17 | 1.46 |
| Płynność platformy tworzenia |  | 0.87 | 1.00 | 1.15 | 1.30 |  |
| Możliwości analityków | 1.42 | 1.19 | 1.00 | 0.85 | 0.71 |  |
| Możliwości programistów | 1.34 | 1.15 | 1.00 | 0.88 | 0.76 |  |
| Doświadczenie z aplikacją | 1.29 | 1.12 | 1.00 | 0.90 | 0.81 |  |
| Doświadczenie z platformą | 1.22 | 1.10 | 1.00 | 0.88 | 0.81 |  |
| Doświadczenie z językiem i narzędziami | 1.19 | 1.09 | 1.00 | 0.91 | 0.85 |  |
| Ciągłość zatrudnienia personelu | 1.20 | 1.09 | 1.00 | 0.91 | 0.84 |  |
| Użycie narzędzi programowych | 1.17 | 1.09 | 1.00 | 0.90 | 0.78 |  |
| Jakość komunikacji zespołu | 1.22 | 1.09 | 1.00 | 0.93 | 0.86 | 0.80 |
| Napięty harmonogram | 1.43 | 1.14 | 1.00 | 1.00 | 1.00 |  |

Źródło: [10]

**2.4 Metoda USE CASE POINTS**

Metoda Use Case Points [12, 16] bazuje na wymaganiach systemu. Pozwala na oszacowanie pracochłonności na etapie, gdy nie posiadamy jeszcze zdefiniowanej architektury systemu.

W metodzie Use Case Points podstawowymi artefaktami branymi pod uwagę są:

**Czynniki złożoności środowiska**

**Czynniki złożoności technicznej**

**Aktorzy**

**Przypadki użycia**

**2.4.1 Czynniki złożoności środowiska**

Czynniki złożoności środowiska [3, 4, 12, 16] (ECF – Environmental Complexity Factors) charakteryzują zespół/organizację, która wytwarza oprogramowanie. Wyszczególniono 8 czynników środowiska. Każdemu przypisano wagę liczbową, mówiącą, jak bardzo dany czynnik wpływa na ostateczny wynik estymacji. Poszczególne czynniki złożoności środowiska wraz z odpowiadającymi im wagami przedstawiono w Tab. 3.

Tab. 3. Czynniki złożoności środowiska

|  |  |
| --- | --- |
| **Nazwa** | **Waga** |
| Zaznajomienie z projektem | 1.5 |
| Doświadczenie zespołu | 0.5 |
| Znajomość technik obiektowych | 1.0 |
| Umiejętności głównego analityka | 0.5 |
| Motywacja zespołu | 1.0 |
| Stabilność wymagań | 2.0 |
| Udział niepełnoetatowców | -1.0 |
| Trudność języka programowania | -1.0 |

Źródło: [14]

**Zaznajomienie z projektem** – określa, czy zespół jest zaznajomiony z dziedziną problemu i technicznymi aspektami realizowanego projektu

**Doświadczenie zespołu** – ogólnie rozumiane doświadczenie zespołu w wytwarzaniu oprogramowania

**Znajomość technik obiektowych** - umiejętności projektowania aplikacji obiektowych i wykorzystywania narzędzi do projektowania

**Umiejętności głównego analityka** – zdolności analityka do pozyskania wymagań od klienta oraz posiadanie wiedzy z dziedziny rozwiązywanego problemu

**Motywacja zespołu** – zdolność zespołu do zaangażowania się w powierzone im zadania

**Stabilność wymagań** – określa, czy wymagania nie są nastawione na częste zmiany

**Udział niepełnoetatowców** – określa, czy wśród zespołu znajduje się duża liczba pracowników pracujących w niepełnym wymiarze czasu

**Trudność języka programowania** – określa, jak trudny do opanowania jest język programowania, w którym implementowany będzie system.

Aby wyznaczyć czynnik złożoności środowiska należy określić wpływ każdego z czynników w skali od 0 do 5, a następnie wyliczyć wartość **ECF** z poniższego wzoru:

gdzie:

wi – waga i-tego czynnika

impacti - ocena wpływu i-tego czynnika w skali od 0 do 5

**2.4.2 Czynniki złożoności technicznej**

Czynniki złożoności technicznej [3, 4, 12, 16] (TCF – Technical Complexity Factor) charakteryzują przyszły system informatyczny. Wyszczególniono 13 czynników technicznych. Każdemu przypisano wagę liczbową, mówiącą, jak bardzo dany czynnik wpływa na ostateczny wynik estymacji. Poszczególne czynniki złożoności technicznej wraz z odpowiadającymi im wagami przedstawiono w Tab. 4.

Tab. 4. Czynniki złożoności technicznej

|  |  |
| --- | --- |
| **Nazwa** | **Waga** |
| Rozproszenie systemu | 2.0 |
| Wydajność systemu | 1.0 |
| Wydajność dla użytkownika końcowego | 1.0 |
| Złożone przetwarzanie wewnętrzne | 1.0 |
| Re-używalność | 1.0 |
| Łatwość instalacji | 0.5 |
| Łatwość użycia | 0.5 |
| Przenośność | 2.0 |
| Łatwość wprowadzania zmian | 1.0 |
| Współbieżność | 1.0 |
| Specjalne zabezpieczenia | 1.0 |
| Udostępnianie użytkownikom zewnętrznym | 1.0 |
| Dodatkowe szkolenia użytkowników | 1.0 |

Źródło: [14]

**Rozproszenie systemu** – definiuje, czy w systemie wymagane jest rozproszone przetwarzanie danych

**Wydajność systemu** – czas reakcji systemu na zdarzenia, szybkość przetwarzania

**Wydajność dla użytkownika końcowego** – definiuje wydajność dla użytkownika końcowego w kontekście jego percepcji

**Złożone przetwarzanie wewnętrzne** – określa, czy wymagane są skomplikowane operacje przetwarzania danych, użycie zaawansowanych algorytmów.

**Re-używalność** – określa, czy kod tworzonej aplikacji może zostać użyty w przyszłych projektach

**Łatwość instalacji** – określa sposób instalacji, czy wymagana jest obecność specjalistów podczas instalacji

**Łatwość użycia** – określa dostosowanie interfejsu użytkownika do jego potrzeb, wygodę w korzystaniu i łatwość nauki obsługi

**Przenośność** – określa, czy aplikacja powinna działać w różnych środowiskach

**Łatwość wprowadzenia zmian** – określa, czy system musi być konstruowany w sposób pozwalający na możliwe zmiany

**Współbieżność** – określa, czy w aplikacji będzie miało miejsce przetwarzanie współbieżne

**Specjalne zabezpieczenia** – definiuje, czy system będzie wymagał użycia dodatkowych systemów zabezpieczeń.

**Udostępnianie użytkownikom zewnętrznym ­**– określa w jakim stopniu z systemu będą korzystać inne zewnętrzne systemy i aktorzy

**Dodatkowe szkolenia użytkowników** – określa potrzebę organizacji szkoleń dla użytkowników

Aby wyznaczyć czynnik złożoności technicznej należy określić wpływ każdego z czynników w skali od 0 do 5, a następnie wyliczyć wartość **TCF** z poniższego wzoru:

gdzie:

wi – waga i-tego czynnika

impacti - ocena wpływu i-tego czynnika w skali od 0 do 5

**2.4.3 Aktorzy**

W klasyfikacji aktorów [3, 4, 12, 16] identyfikujemy wszystkich aktorów, którzy będą wchodzić w interakcje z systemem. Wyszczególniono 3 grupy złożoności aktorów. Każdej grupie przypisano wagę liczbową, mówiącą, jak bardzo dany aktor wpływa na ostateczny wynik estymacji. Grupy złożoności aktorów oraz wagi przedstawiono w Tab. 5.

Tab. 5. Grupy złożoności aktorów

|  |  |
| --- | --- |
| **Złożoność aktora** | **Waga** |
| Prosty | 1 |
| Średni | 2 |
| Złożony | 3 |

Źródło: [14]

Aktor o złożoności **prostej** – aktor komunikujący się z systemem przez API

Aktor o złożoności **średniej** – aktor komunikujący się z systemem przez protokół, lub stanowiący źródło danych

Aktor **złożony** – aktor komunikujący się z systemem przez graficzny interfejs użytkownika

W etapie klasyfikacji aktorów wyliczamy **nieskorygowaną wagę aktorów** (UAW - Unadjusted Actors Weight) z poniższego wzoru:

gdzie:

ni – liczba aktorów zakwalifikowana do i-tej grupy

wi – waga dla i-tej grupy

**2.4.4 Przypadki użycia**

W klasyfikacji przypadków użycia [3, 4, 12, 16] identyfikujemy wszystkie specyfikacje przypadków użycia. Wyszczególniony 3 grupy złożoności przypadków użycia. Każdej grupie przypisano wagę liczbową, mówiącą, jak bardzo dany przypadek użycia wpływa na ostateczny wynik estymacji. Grupy złożoności przypadków użycia oraz wagi przedstawiono w Tab. 6.

Tab. 6. Grupy złożoności przypadków użycia

|  |  |
| --- | --- |
| **Złożoność przypadku użycia** | **Waga** |
| Prosty | 5 |
| Średni | 10 |
| Złożony | 15 |

Źródło: [14]

Przypadek użycia jest złożoności **prostej** gdy:

- posiada prosty interfejs użytkownika

- operuje na pojedynczej encji bazy danych

- zawiera 3 lub mniej kroków w scenariuszu

- implementacja obejmuje mniej niż 5 klas

Przypadek użycia jest złożoności **średniej** gdy:

- posiada średnio złożony interfejs użytkownika

- operuje na przynajmniej 2 encjach bazy danych

- zawiera od 3 do 7 kroków w scenariuszu

- implementacja obejmuje od 5 do 10 klas

Przypadek użycia jest **złożony** gdy:

- posiada złożony interfejs użytkownika

- operuje na przynajmniej 3 encjach bazy danych

- zawiera 7 lub więcej kroków w scenariuszu

- implementacja więcej niż 10 klas

W etapie klasyfikacji przypadków użycia wyliczamy **nieskorygowaną wagę przypadków użycia** (UUCW - Unadjusted Use Cases Weight) z poniższego wzoru:

gdzie:

ni – liczba przypadków użycia zakwalifikowana do i-tej grupy

wi – waga dla i-tej grupy

**2.4.5 Wyznaczenie pracochłonności**

Po obliczeniu nieskorygowanych wag aktorów oraz nieskorygowanych wag przypadków użycia możliwe jest obliczenie **nieskorygowanych punktów przypadków użycia** [3, 4, 12, 16] (UUCP - Unadjusted Use Case Points) poprzez zsumowanie wspomnianych wartości.

Następnym krokiem jest obliczenie **punktów przypadków użycia** (UCP – Use Case Points) ze wzoru:

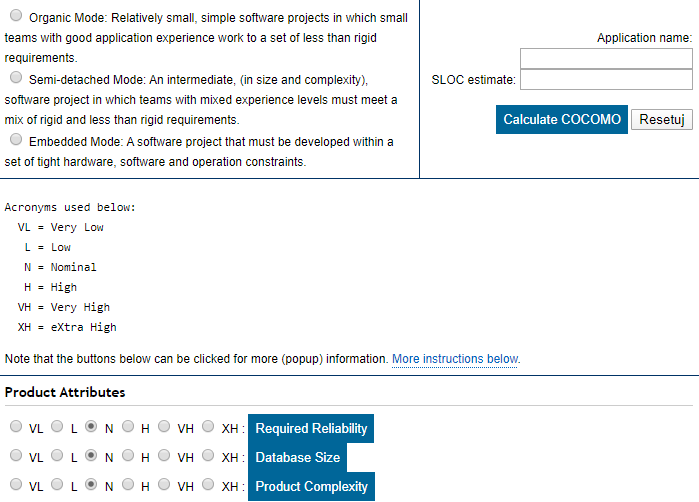
Ostatnim krokiem w metodzie Use Case Points jest przeliczenie punktów przypadków użycia na wartości pracochłonności w osobomiesiącach. Dokonuje się tego poprzez pomnożenie UCP przez **współczynnik pracochłonności** (PF - Productivity Factor). Współczynnik ten przekształca jeden punkt przypadku użycia na ilość godzin pracy ludzkiej i wynosi od 15 do 30 roboczogodzin.

**Rozdział 3. Charakterystyka istniejących rozwiązań**

**3.1 Kalkulator COCOMO.**

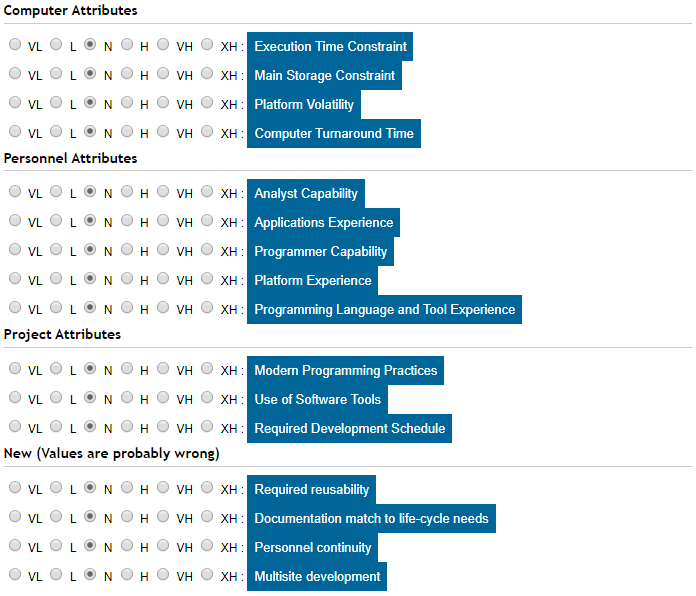
Kalkulator COCOMO [13] bazuje na metodzie zaproponowanej przez Dr. Berry Boehm w 1981, na podstawie której powstała później metoda COCOMO II. Kalkulator ten został zaimplementowany w postaci aplikacji webowej, jego interfejs został przedstawiony na Rys. 1. Przeprowadzenie procesu oceny pracochłonności polega na wybraniu, za pomocą przycisków, wartości czynników istotnych w algorytmie metody COCOMO, a następnie wpisanie w odpowiednie pole nazwy projektu oraz szacowaną ilość linii kodu źródłowego. Po wciśnięciu przycisku „Calculate COCOMO” zostaje wyświetlona tabela przedstawiająca obliczone wartości m.in. pracochłonność, czas trwania projektu czy proponowaną obsadę etatów. Przykładowa tabela została przedstawiona na Rys. 2.

Rys. 1. Kalkulator COCOMO



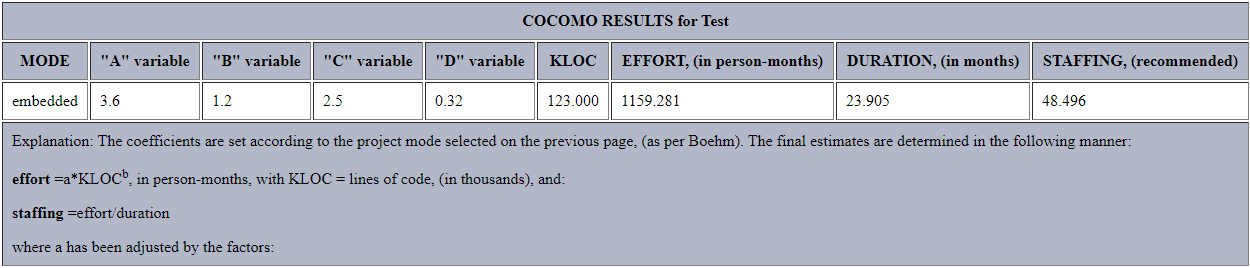
Źródło: [13]

Rys. 1 (c.d.). Kalkulator COCOMO



Źródło: [13]

Rys. 2. Tabela obliczonych wartości przy użyciu kalkulatora COCOMO



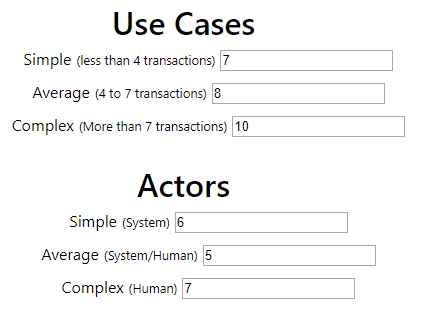
Źródło: [13]

**3.2 Kalkulator USE CASE POINTS.**

Kalkulator do obliczenia pracochłonności [15] wykorzystuje metodę USE CASE POINTS. Kalkulator ten został zaimplementowany w postaci aplikacji webowej, jego interfejs został przedstawiony na Rys. 3. Proces przeprowadzenia oceny pracochłonności polega na wpisaniu w odpowiednie pola wartości czynników istotnych w algorytmie metody USE CASE POINTS. Wartości pracochłonności wyrażone w osobogodzinach zostają przeliczane na bieżąco oraz wyświetlone w części zatytułowanej ‘Hours’. Przykładowy wynik kalkulacji przedstawiono na Rys. 4.

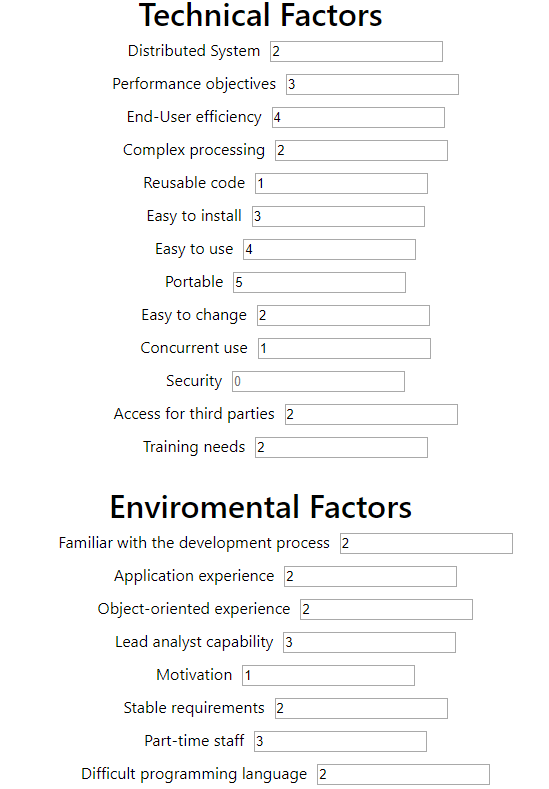
Podstawowym problemem tego rozwiązania jest fakt, że w części czynników środowiska i czynników technicznych dopuszcza wartości z przedziału nienależącego do <0, 5>, gdy w metodzie USE CASE POINTS wartości tych czynników ocenia się właśnie od 0 do 5 [3]. Kolejnym problemem jest problem czytelności. Aplikacja przedstawia wyniki pracochłonności w sposób niezbyt przyjazny dla użytkownika.

Rys. 3. Kalkulator USE CASE POINTS



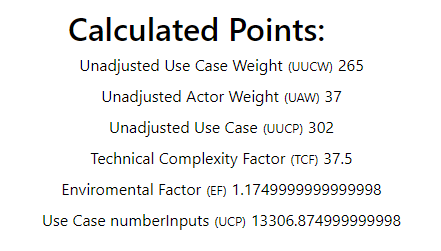
Źródło: [15]

Rys. 3 (c.d.). Kalkulator USE CASE POINTS



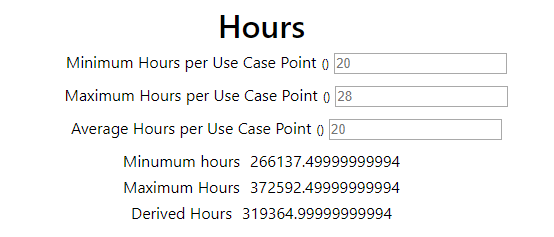
Źródło: [15]

Rys. 3 (c.d.). Kalkulator USE CASE POINTS



Źródło: [15]

Rys. 4.



Źródło: [15]

**Bibliografia**

[1] [Kucharski, M.: Pomiar rozmiaru i pracochłonności oprogramowania komputerowego, 2006, s 105-110](http://yadda.icm.edu.pl/baztech/contributor/aee41e8962cc1ce51aafd4bc62a2db44)

[2] Weichbroth P., Orłowski C.: Przegląd miar oceny oprogramowania. [w:] Chmielarz W., Parys T.(Red.): Technologie informacyjne dla społeczeństwa. Wyższa Szkoła Ekonomiczno-Informatyczna,Exit, Warszawa 2009, s. 275-284.

[3] https://en.wikipedia.org/wiki/Use\_Case\_Points

[4] https://michalwolski.pl/2009/10/metoda-punktw-przypadkw-uzycia/

[5] https://nesma.org/2015/05/software-size-measures-and-their-use-in-software-project-cost-estimation/

[6] https://pl.wikipedia.org/wiki/Metryka\_oprogramowania

[7] https://pl.wikipedia.org/wiki/Metoda\_delficka

[8] https://octigo.pl/na-kiedy-skonczysz-ten-projekt-latwe-zaplanowanie-harmonogramu-projektu/

[9] https://www.researchgate.net/publication/4362086\_Effects\_of\_software\_process\_maturity\_on\_COCOMO\_II's\_effort\_estimation\_from\_CMMI\_perspective

[10] https://researchgate.net/publication/243482239\_Cocomo\_ii\_model\_definition\_manual

[11] https://sites.google.com/site/rafalpetryniakcourses/przedmioty/metrologia-informatyczna/cwiczenia-i-projekt/szacowanie-pracochlonnosci-1

[12] https://sites.google.com/site/rafalpetryniakcourses/przedmioty/metrologia-informatyczna/cwiczenia-i-projekt/szacowanie-pracochlonnosci-2

[13] https://strs.grc.nasa.gov/repository/forms/cocomo-calculation/

[14] https://tutorialspoint.com/estimation\_techniques/estimation\_techniques\_use\_case\_points.htm

[15] http://ucp-calculator.appspot.com/

[16] http://wazniak.mimuw.edu.pl/images/3/30/Zio-13-wyk.pdf

diagram klas, aktywności, koncepcje, pomysł, diagramy url baza danych

wytyczne dla pracy dla dyplomantów