

Estymacja projektów informatycznych

Tomasz Koszlajda
Instytut Informatyki Politechniki Poznańskiej
e-mail: Tomasz.Koszlajda@cs.put.poznan.pl

1. Wstęp

Uruchomienie dowolnego projektu informatycznego wymaga estymacji podstawowych parametrów tego projektu. Proces estymacji obejmuje cztery podstawowe etapy: estymacja rozmiaru tworzonego oprogramowania, estymacja wymaganego nakładu pracy wyrażonego na przykład w roboczomiesiącach, estymacja całkowitego czasu realizacji przedsięwzięcia i estymacja kosztu wyprodukowania oprogramowania. Rzetelna estymacja projektów informatycznych jest potrzebna podczas realizacji wszystkich faz projektu i powinna być ponawiana w wypadku uszczegóławiania bądź zmian specyfikacji konstruowanego produktu informatycznego. Estymacja jest niezbędna dla prawidłowego wypełnienia następujących zadań kierowników projektów:

- Ustalanie zasobów potrzebnych do pomyślnej realizacji projektu: liczby i wymaganych kwalifikacji pracowników, czasu realizacji, infrastruktury sprzętowo-programowej.
- Ustalanie poziomu i budżetu zatrudnienia.
- Określanie kompromisu między kosztem realizacji, a jakością i funkcjonalnością produktu.
- Identyfikacja ryzyka i ilościowe szacowanie jego wpływu na koszt i czas realizacji przedsięwzięcia.
- Monitorowanie realizacji projektu.
- Modyfikacja budżetu i harmonogramu w wypadku wystąpienia nieprzewidzianych zdarzeń.

Dokładna estymacja projektów informatycznych jest jednak utrudniona z powodu specyfiki produktów informatycznych, która polega na tak zwanej *niewidzialności* oprogramowania. Wiąże się ona z trudnością pomiaru takich własności oprogramowania jak jego wielkość, złożoność czy jakość. Dodatkowym utrudnieniem jest fakt, że estymacja projektu jest wymagana przed rozpoczęciem jego realizacji, kiedy specyfikacja tworzonego produktu informatycznego jest jeszcze bardzo ogólna i niejednoznaczna. Dlatego niezbędne jest stosowanie przez kierowników projektów właściwych metod i narzędzi, które pozwolą zwiększyć precyzję szacowania niezbędnego nakładu pracy i czasu konstrukcji oprogramowania, przy tych obiektywnie istniejących ograniczeniach.

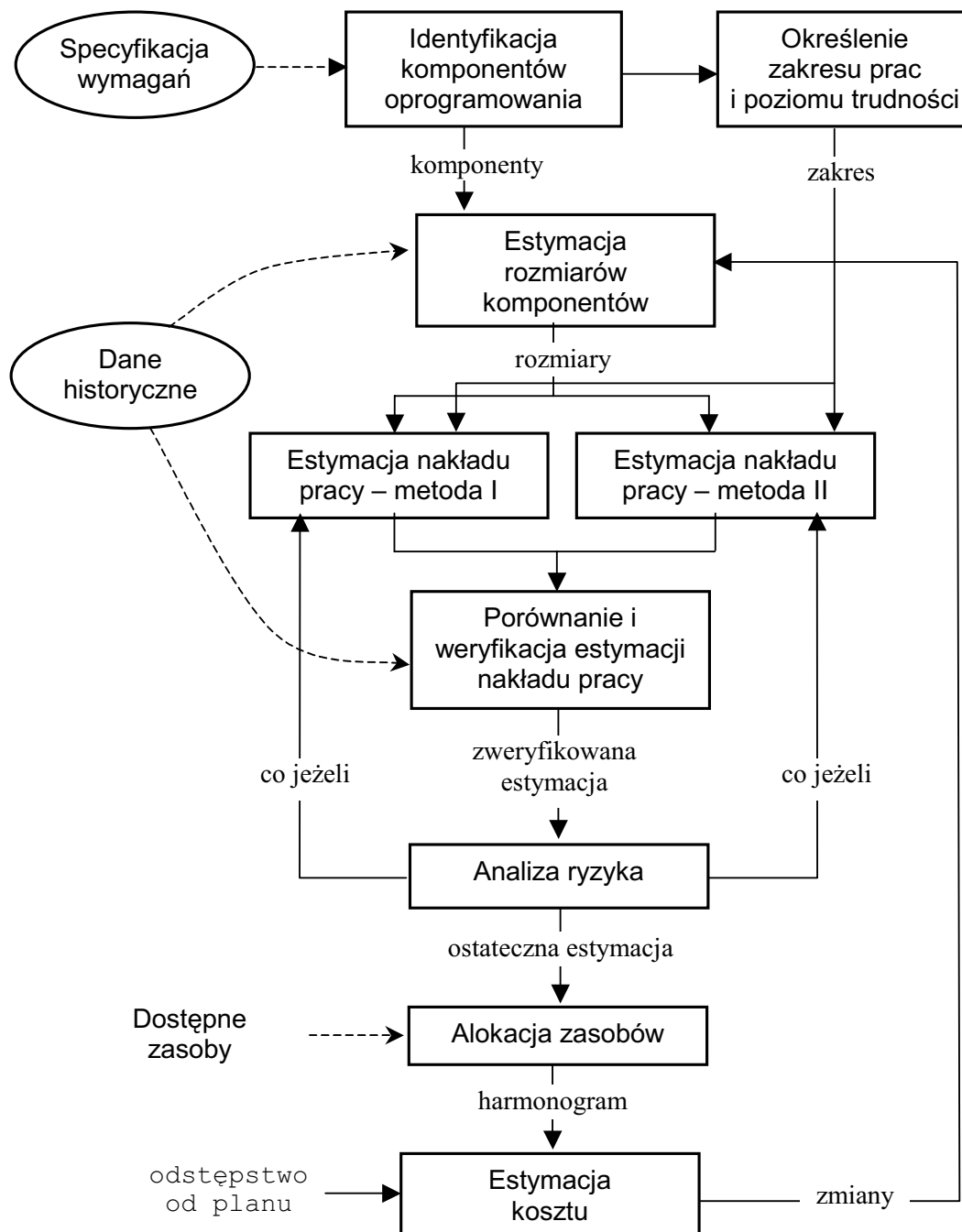
Dziedzina inżynierii oprogramowania dostarcza wielu propozycji miar własności oprogramowania oraz metod i modeli estymacji projektów informatycznych. Proponowane miary oprogramowania opisują różne aspekty wielkości oprogramowania: rozmiar programów źródłowych – mierzony liczbą linii kodu źródłowego (SLOC), funkcjonalność programów – mierzona za pomocą tzw. punktów funkcyjnych lub architekturę programów – mierzona tzw. punktami konstrukcyjnymi (ang. object points). Wielkość oprogramowania jest podstawowym parametrem wejściowym dla parametrycznych modeli estymacji nakładu pracy i czasu realizacji. Proponowane model są oparte na solidnych podstawach matematycznych.

Powstały one na podstawie analizy zależności zachodzących między wielkością oprogramowania, nakładem pracy i czasem realizacji.

Również rynek informatyczny oferuje od wielu lat narzędzia programowe implementujące wypracowane przez inżynierię oprogramowania modele estymacji. Najbardziej znane programy wspierające proces estymacji to: COCOMO (dostępne są zarówno wersje darmowe, jak i wersje komercyjne dostarczane przez różnych producentów, np.: COSTAR, REVIC, COSTMODL), SLIM, SoftCost i PRICE-S.

2. Proces estymacji

Proces estymacji obejmuje osiem kroków, których kolejność przedstawiono na rysunku poniżej. Proces zostanie uruchomiony, jeżeli zgromadzono wystarczająco kompletne i dokładne wymagania określające konstruowany produkt informatyczny.



1. **Identyfikacja komponentów oprogramowania.** Ma ona na celu ustalenie produktów, dla których w kolejnych krokach mają być wykonane estymacje. Wyodrębnienie różnych komponentów może być wynikiem wymagań użytkownika lub jest wynikiem decyzji kierownika projektu. Podział może mieć charakter funkcjonalny lub dotyczyć kolejnych wersji tego samego produktu. Podział funkcjonalny pociąga za sobą konieczność uwzględnienia dodatkowych kosztów integracji.
2. **Estymacja rozmiarów komponentów.** Nakład pracy jest funkcją rozmiaru produktu. Tak więc krok ten ma na celu wyznaczenie wartości wejściowych niezbędnej dla określenia nakładu pracy. Wybór jednej z metod estymacji rozmiaru budowanego oprogramowania zależy od kilku czynników: klasy wielkości i złożoności konstruowanego produktu, doświadczeń wykonawcy w realizacji podobnych projektów, przewidywanych narzędzi informatycznych do budowy oprogramowania, dziedziny zastosowania, itp.. Uzyskanie dobrego oszacowania rozmiaru komponentów jest bardzo istotne, ze względu na dużą czułość kolejnych kroków estymacji od wyników uzyskanych w tym kroku.
3. **Określenie zakresu prac i poziomu trudności.** Wiarygodność oszacowań nakładu pracy, czasu realizacji i kosztu projektu jest mocno zależna od poprawnego określenia przewidywanego zakresu prac, to jest identyfikacji etapów, które pojawią się w cyklu życia projektu oraz technicznej trudności projektu. Problemy w ustaleniu technicznej trudności projektu wynikają z kłopotów w ilościowym opisie trudności. Tymczasem zależność między poziomem trudności, a niezbędnym nakładem pracy ma charakter eksponentialny.
4. **Estymacja nakładu pracy dla każdego z komponentów.** Popularnych jest kilka metod estymacji nakładu pracy niezbędnego dla realizacji określonego w poprzednim etapie zakresu prac w celu zbudowania komponentu o oszacowanym rozmiarze. Metody te pozwalają na uwzględnienie w procesie estymacji dodatkowych czynników mających wpływ na wymagany nakład pracy. Niektóre z tych metod pozwala również odwoływać się, za pomocą metod statystycznych na przykład regresji, do charakterystyki już zrealizowanych projektów.
5. **Weryfikacja estymacji nakładu pracy.** Dla zwiększenia wiarygodności estymacji należy stosować równolegle kilka niezależnych metod. Dodatkowo poszczególne estymacje mogą być przeprowadzane przez różnych wykonawców.
6. **Analiza ryzyka.** Uzyskane oszacowania muszą zostać poddane analizie ryzyka. Powinna ona obejmować określenie wpływu potencjalnych zagrożeń, polegających na przykład na obciążeniu budżetu, skróceniu harmonogramu lub niedoszacowania rozmiaru tworzonego oprogramowania, na szacowany nakład pracy.
7. **Alokacja zasobów.** Dla ustalenia harmonogramu prac niezbędne jest przypisanie do projektu jego bezpośrednich wykonawców. Czasy realizacji poszczególnych zadań zależą od dostępnych zasobów kadrowych.
8. **Estymacja kosztu.** Gotowy harmonogram jest podstawą do wyznaczenia kosztu przedsięwzięcia i przyrównania go z dostępnym budżetem. W przypadku niedopasowania kosztów do budżetu może okazać się, że do projektu należy wprowadzić stosowne zmiany. Zmiany wszystkich estymowanych parametrów mogą mieć miejsce również w trakcie realizacji projektu, w wyniku zmiany specyfikacji wymagań lub odstępstwa realizacji projektu od planu.

3. Miary wielkości oprogramowania

Powszechnie wyróżnia się trzy podstawowe miary wielkości oprogramowania: wielkość źródłowej postaci programów, złożoność funkcjonalna i złożoność konstrukcyjna. Żadna z tych miar nie jest doskonała; wszystkie obarczone są określonymi wadami.

- Miarą wielkości źródłowej postaci programów jest liczba fizycznych linii kodu źródłowego (SLOC – Source Lines of Code). Brane pod uwagę są linie zawierające instrukcje i deklaracje. Linia zawierająca kilka instrukcji lub deklaracji jest brana pod uwagę tylko raz. Linie puste, bądź zawierające tylko komentarze są pomijane. Istnieje kilka odmian tej podstawowej propozycji. Na przykład alternatywą jest liczenie logicznych linii kodu. W tej sytuacji kilka linii fizycznych zawierających pojedynczą instrukcję są traktowane jako jedna jednostka. Sposób liczenia może być również uzależniony od konkretnego języka programowania.

Zalety

Podstawową zaletą miary SLOC jest jej prostota. Dodatkowo jest ona również bardzo dokładna. Po zakończeniu projektu można precyzyjnie wyliczyć rozmiar oprogramowania. Miara ta jest wykorzystywana przez większość parametrycznych modeli estymacji.

Wady

Lista wad tej miary jest bardzo długa. Jest ona trudna do zastosowania podczas etapu planowania, gdy dysponujemy jedynie ogólną specyfikacją oprogramowania. Nie uwzględnia ona złożoności funkcjonalnej i konstrukcyjnej oprogramowania. Jest zależna od jakości oprogramowania. Trudno jej używać podczas negocjacji z użytkownikiem.

Ponadto miara ta jest zależna od stosowanego języka programowania, co ma zarówno pozytywne jak i negatywne implikacje.

- Miarą złożoności funkcjonalnej oprogramowania są tzw. punkty funkcyjne. Istnieje kilka odmian tej miary, ale ich ogólne własności są takie same. Punkty funkcyjne są powiązane z określonymi typami funkcjonalności oprogramowania. W metodzie punktów funkcyjnych wyróżnia się pięć abstrakcyjnych klas funkcjonalności:
 - wprowadzania zewnętrznych danych do programu (**wejście**),
 - wyprowadzania danych na zewnątrz programu (**wyjście**),
 - wewnętrznych struktur danych (**pliki**),
 - komunikacją z programami i danymi zewnętrznymi (**interfejs**),
 - interakcją nie związaną z przepływem danych (**zapytania**).

Każdy wyodrębniony fragment oprogramowania musi być zaklasyfikowany do jednej z tych klas. Dodatkowo należy mu przypisać jeden z trzech poziomów złożoności: niski, średni lub wysoki. Z poszczególnymi klasami i poziomami trudności skojarzone są określone stałe wagi. Poniżej przedstawiono przykładową klasyfikację elementów abstrakcyjnego oprogramowania oraz wyliczenie sumarycznej złożoności systemu informatycznego składającego się z dwóch modułów obejmujących aplikacje o różnej funkcjonalności.

Moduł	Wejścia			Wyjścia			Pliki			Interfejsy			Zapytania			Suma punktów funkcyjnych
	nisk.	śr.	wys.	n	ś	w	n	ś	w	n	ś	w	n	ś	w	
	3	4	6	4	5	7	7	10	15	5	7	10	3	4	6	
Moduł 1	1x3	1x4			1x5			10x10						5x4	1x6	138
Moduł 2		2x4			2x5	2x7		5x10						5x4		102
Razem	3	12			15	14		150						40	6	240

Wyliczona suma punktów funkcyjnych dla danego oprogramowania jest następnie korygowana dla uwzględnienia kilkunastu czynników korygujących, które oddają techniczną złożoność oprogramowania, np. praca w trybie on-line, rozproszenie, złożoność przetwarzania danych, itp.. Docelowa złożoność funkcjonalna systemu jest obliczana według wzoru:

$$\text{Skorygowane FP} = [0,65 * \Sigma(\text{czynników korygujących}) / 100] * \text{Nieskorygowane FP}$$

Zaproponowano przeliczenia miary punktów funkcyjnych i linii kodu źródłowego. Przykładowe przeliczenia zawarto w poniższej tabeli.

Język programowania	Liczba linii kodu przypadających na jeden punkt funkcyjny
<i>Assembler</i>	320
<i>Basic</i>	107
<i>C</i>	128
<i>C++</i>	53
<i>COBOL</i>	107
<i>Delphi</i>	29
<i>Eiffel</i>	21
<i>Java</i>	53
<i>Język naturalny</i>	3200
<i>Oracle Developer/2000</i>	23
<i>SQL</i>	13
<i>Visual Basic 5</i>	29
<i>4GL</i>	20

Zalety

Punkty funkcyjne nie są ograniczone do kodu programu i są niezależne od języków programowania. W przeciwieństwie do linii kodu źródłowego są łatwe do określenia na wczesnych etapach projektu. Są niezależne od jakości konstrukcji programu. Metoda ta jest wspierana przez międzynarodową grupę użytkowników - IFPUG.

Wady

Podstawową wadą metody punktów funkcyjnych jest pewna subiektywność ich przypisywania. Stąd wynika trudność automatyzacji tej metody. Ponadto metoda ta jest przypisana do predefiniowanego typu funkcjonalności. Wymaga ona adaptacji do nowych klas zastosowań.

- Miarą złożoności konstrukcyjnej są tzw. punkty konstrukcyjne. Miara ta jest podobna do punktów funkcyjnych. Jednak w odróżnieniu od punktów funkcyjnych, punkty konstrukcyjne przypisuje się nie elementom funkcjonalnym oprogramowania, lecz elementom konstrukcyjnym. Wyróżnia się trzy klasy takich elementów związanych ze środowiskiem narzędzi CASE: ekrany, raporty oraz moduły utworzone za pomocą języków 3GL.

Oszacowanie złożoności konstrukcyjnej wymaga identyfikacji wszystkich elementarnych konstrukcji składających się na dany system informatyczny i przypisania im określonej wagi reprezentującej złożoność pojedynczego elementu. Dla ekranów i raportów zaproponowano jednoznaczną metodę określania klasy złożoności w zależności od wielkości ekranów i raportów oraz liczby bazowych relacji. Dla obliczenia sumarycznej złożoności należy zsumować wszystkie punkty konstrukcyjne.

Metoda ta przewiduje dodatkowo możliwość uwzględnienia operacji przystosowania istniejących elementów konstrukcyjnych.

Zalety i wady tej metody są analogiczne dla metody punktów funkcyjnych. Dodatkową jej zaletę stanowi silny związek ze środowiskiem CASE.

4. Metody estymacji

Wyróżnia się sześć podstawowych metod estymacji projektów informatycznych, które pokrótce scharakteryzowano poniżej. Wybór jednej z nich wynika z preferencji, doświadczeń i umiejętności stosujących je firm lub osób oraz specyfiki danego projektu.

1. **Estymacja przez analogię.** Estymacja projektu odbywa się na podstawie doświadczeń wykonawcy w realizacji podobnych systemów. Niezbędne do tego informacje dotyczące zakończonych projektów są zbierane i przechowywane w specjalnych katalogach. Informacje te muszą obejmować specyfikę projektów pozwalającą na ustalenie podobieństw i różnic między poszczególnymi projektami oraz dane dotyczące ostatecznego rozmiaru, nakładu pracy, czasu realizacji i kosztu.
2. **Metoda bottom-up.** Estymacja jest wykonywana dla składowych jednostek komponentów oprogramowania i poszczególnych etapów ich realizacji: projektu, kodowania, testowania, itd., za pomocą jednej lub kilku pozostałych metod. Następnie wyniki poszczególnych estymacji są sumowane.
3. **Modele parametryczne.** Estymacja jest wykonywana na podstawie funkcji wiążących rozmiar oprogramowania i inne dodatkowe czynniki kosztu z nakładem pracy i czasem realizacji. Wyróżnia się cztery podstawowe klasy modeli parametrycznych, dysponujące wspierającymi je grupami użytkowników i producentami. Poszczególne modele różnią się przyjętą bazą matematyczną. Modele te pozwalają zazwyczaj na ich kalibrację na podstawie bazy estymowanych i zrealizowanych projektów.
4. **PERT.** W metodzie tej estymacje są wykonywane na podstawie rozkładu prawdopodobieństwa dla najgorszego l , najlepszego h i najbardziej prawdopodobnego przypadku m nakładu pracy i czasu wykonania, przez użycie następującej formuły:

$$\text{Nakład pracy} = (l+4m+h)/6$$

Wartości l, m i h są wyznaczane za pomocą metody lub metody Delphi.

5. **Metoda top-down.** Estymacja jest wykonywana dla komponentu jako całości na podstawie jego ogólnych własności odwołując się do dotychczasowych doświadczeń. Wyniki estymacji są następnie dzielone między elementy składowe komponentu.
6. **Metoda Delphi.** Dla realizacji tej metody jest niezbędny zespół ekspertów. Eksperti równoległe estymują podstawowe parametry projektu. Wyniki estymacji są następnie konfrontowane. Proces estymacji przebiega dalej iteracyjnie, aż do uzgodnienia wyników

Metoda	Zalety	Wady
Analogia	Bazuje na doświadczeniu	Brak doświadczeń dla projektu nowego typu Podobieństwa różnych projektów mogą być mylące
Bottom-up	Estymacja na poziomie szczegółów jest dokładniejsza Uwzględnia specyfikę projektu Stabilna	Ułatwia przeoczenie czynników związanych z poziomem całego systemu Jest kosztowna
Modele parametryczne	Obiektywne i powtarzalne Bierze pod uwagę wiele czynników Uwzględnia wpływ skali projektu	Trudna jest kalibracja metod Część czynników ma charakter subiektywny Nie obejmuje wszystkich dziedzin zastosowań
PERT	Pozwala ograniczyć ryzyko	Trudne do uzyskania dobre dane wejściowe
Top-down	Wspiera widzenie systemu jako całości Efektywna	Ułatwia przeoczenie detali Nie wspiera szczególnych przypadków
Delphi	Równoległe stosowanie alternatywnych estymacji	Ograniczona jakością ekspertów

W praktyce (nie tylko polskiej) stosowane są również inne metody estymacji, które jednak nie są zalecane. Na przykład podczas przetargów na realizację oprogramowania, stosowana jest powszechnie metoda polegająca na określeniu ceny i czasu realizacji na poziomie niższym od konkurencji, abstrahując od wielkości i złożoności oprogramowania.

5. Parametryczne modele estymacji projektów

W zależności od zastosowanej bazy matematycznej wyróżnia się cztery podstawowe typy modeli parametrycznych:

- Modele oparte na regresji – przykładem jest model *COCOMO*,
- Modele fenomenologiczne – przykładem jest model *SLIM*,
- Modele heurystyczne – przykładem jest model *PRICE-S*,
- Modele hybrydowe – przykładem jest model *SoftCost*.

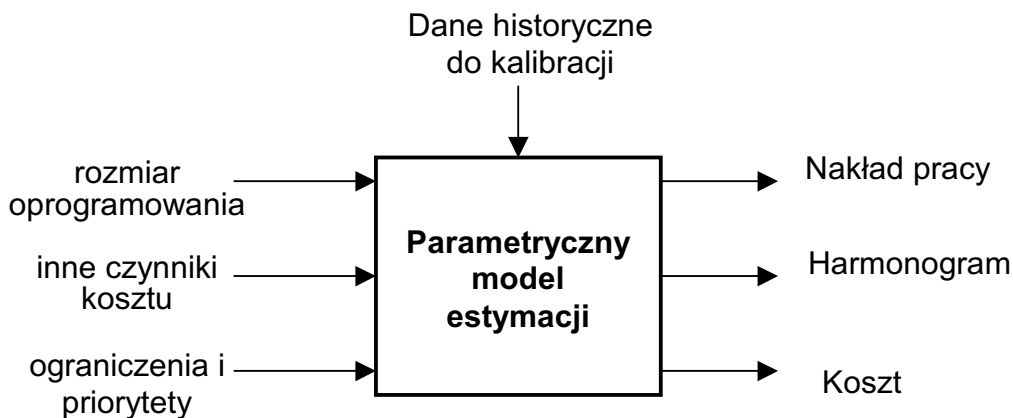
Jednak wszystkie te modele bazują na podobnej matematycznej formule wiążącej rozmiar produktu informatycznego wyrażonej za pomocą jednej z przedstawionych miar, z nakładem pracy niezbędnym do jego realizacji. Formuła ta wygląda następująco:

$$\text{Nakład pracy} = \alpha * (\text{rozmiar})^\beta$$

gdzie:

- α jest pewną zmienną reprezentującą zbiór czynników wpływających na pracochłonność przedsięwzięcia,
- β określa nieliniowy wpływ skali wielkości oprogramowania na wymagany nakład pracy.

Ogólny schemat działania modeli parametrycznych został przedstawiony na poniższym rysunku.



Poszczególne modele różnią się listą czynników, których wpływ jest uwzględniany w wyznaczaniu wartości parametru α oraz poziomem czułości nakładu pracy na poszczególne czynniki. Różnią się one również sposobem kalibracji poszczególnych metod uwzględnianej przez parametr β . Przykładowo wyjściowy wzór w metodzie COCOMO II stosowany podczas wczesnego etapu projektu, gdy dysponujemy jedynie specyfikacją wymagań na produkt informatyczny przyjmuje następującą postać:

$$PM = 2,45 * \prod_{i=1}^6 (EM_i) * size^{0,91 + 0,01 * \sum_{j=1}^5 SF_j}$$

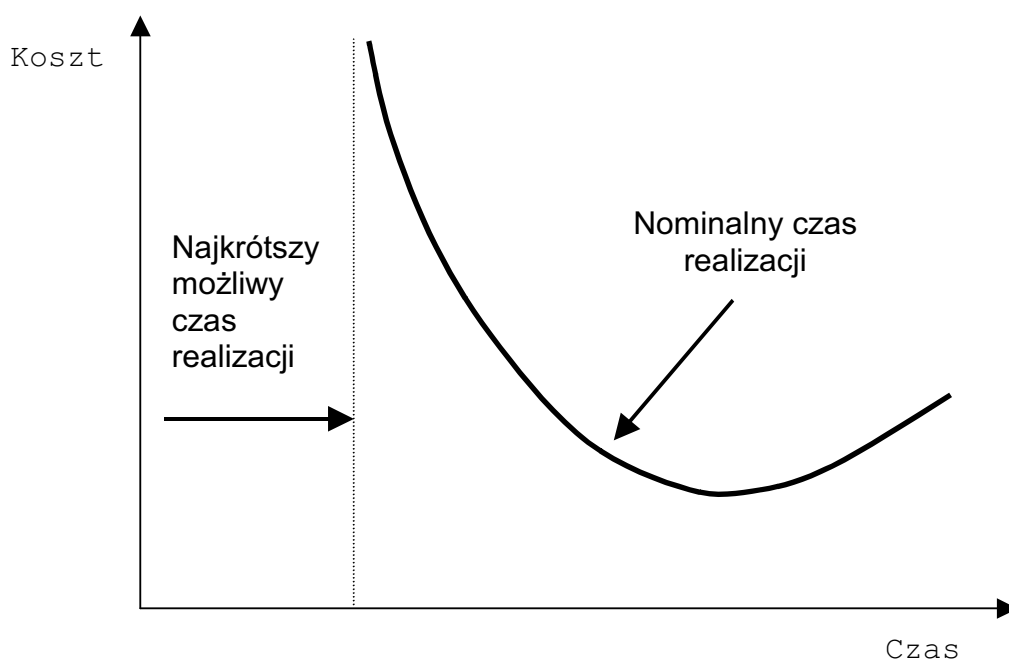
gdzie:

- **PM** oznacza pracochłonności przedsięwzięcia szacowaną w osobomiesiącach;
- **size** oznacza rozmiar budowanego systemu informatycznego w standardowych liniach kodu źródłowego;
- **EM_i** i **SF_j** są współczynnikami korygującymi wymagany nakład pracy o następującym znaczeniu;

SF ₁	Typowość problemu
SF ₂	Elastyczność procesu budowy
SF ₃	Ryzyko wyboru architektury
SF ₄	Spoistość zespołu
SF ₅	Opanowanie technologii

EM ₁	Niezawodność i złożoność produktu
EM ₂	Wielokrotne zastosowanie/universalność rozwiązań
EM ₃	Złożoność platformy sprzętowo-programowej
EM ₄	Kwalifikacje zespołu wykonawców
EM ₅	Doświadczenie wykonawcy
EM ₆	Siła stosowanych narzędzi programistycznych

Prezentowane modele oprócz szacowania nakładu pracy i czasu realizacji projektu, umożliwiają również szukanie zadawalającego kompromisu między tymi dwoma wielkościami. Zależność między tymi parametrami jest silnie nieliniowa. Skracanie czasu realizacji, a również wydłużanie go powyżej pewnej wartości granicznej, wymaga zwiększenia nakładu pracy. Najkrótszy możliwy czas pracy określa wartość, której nie można przekroczyć niezależnie od ponoszonych nakładów. Zależność tę przedstawiono na poniższym rysunku.

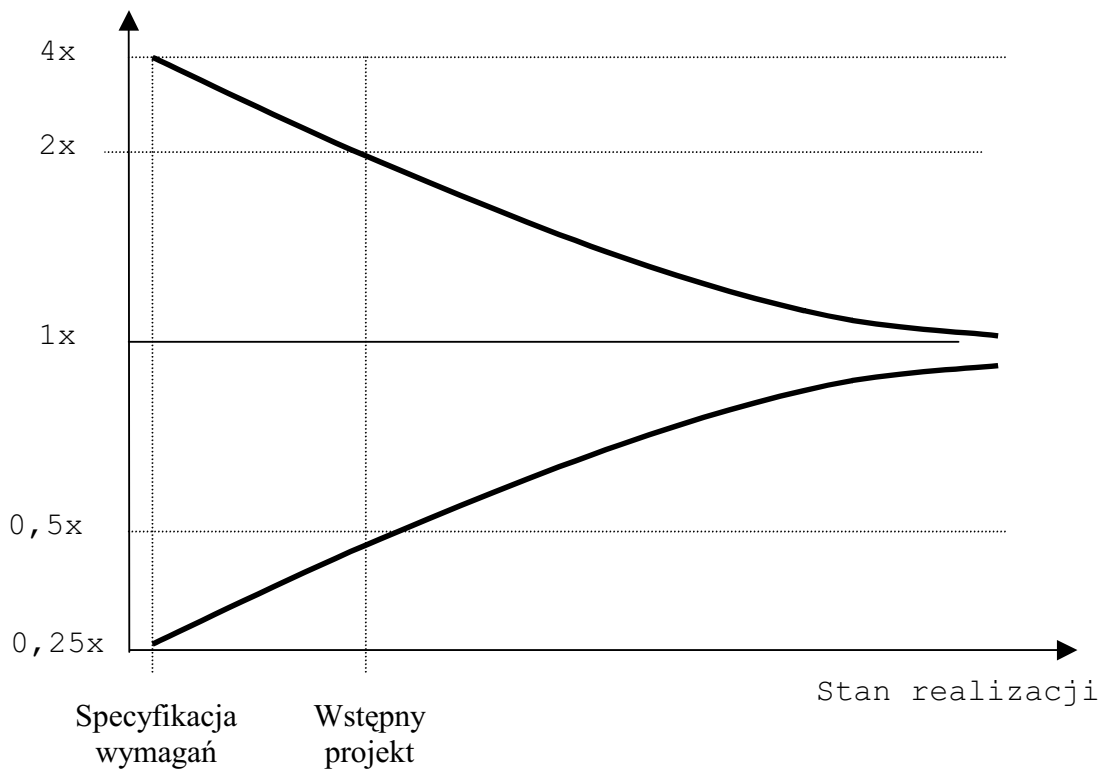


Poniżej przedstawiono trzy różne estymacje tego samego programu o szacowanym rozmiarze równym 75 KSLOC dla trzech wybranych czasów realizacji. Przykład ten ilustruje jak nieprawdziwe jest przekonanie o proporcjonalnej zależności między zwiększeniem miesięcznego nakładu pracy, a skróceniem czasu realizacji.

	Plan nominalny	Plan o najkrótszym czasie realizacji	Plan o najniższym koszcie
Nakład pracy [roboczomiesiące]	40	97	14
Czas realizacji [miesiące]	12,4	10	16,2
Koszt	605 000 \$	1 479 000 \$	212 000 \$
Maksymalna liczba wykonawców	4,8	14,6	1,3
Średnia liczba	3,2	9,8	0,9

Należy mieć świadomość, że poziom ufności przedstawionych metod estymacji niezależnie od wyrafinowania wykorzystywanego aparatu matematycznego, jest w początkowej fazie projektu bardzo niski. Wraz z uszczegóławianiem projektu poziom ufności estymacji rośnie. Zależność poziomu ufności estymacji od stanu zaawansowania projektu ilustruje poniższy wykres.

nakład pracy,
rozmiar



6. Podsumowanie

Estymacja projektów informatycznych jest dojrzałą technologią. Dziedzina ta oferuje wiele dobrych rozwiązań problemu estymacji. Za rozwiązaniami teoretycznymi nadążają rozwiązania praktyczne. Rynek informatyczny oferuje kilkanaście programów wspierających stosowanie proponowanych przez teorię metod. Publikowane są liczne przykłady pomyślnego zastosowania programów i metod, które umożliwiają ich weryfikację i dalsze udoskonalanie. Aktywnych jest kilka organizacji i grup użytkowników wspierających poszczególne rozwiązania i narzędzia programowe. Powstają również standardy porządkujące stosowane definicje pojęć i metodyki.

Z różnych powodów rozwiązania te z trudem przyjmują się na polskim rynku informatycznym. Prawdopodobnymi przyczynami tej niedobrej sytuacji jest brak opanowania tych technik, nieuzasadniony optymizm warstwy kierowniczej, nieumiejętność uczenia się na własnych błędach oraz niechęć do wydłużania procesu planowania i ponoszenia dodatkowych

nakładów finansowych związanych z ich stosowaniem. Wydaje się, że jest to jednym z istotnych powodów kłopotów w realizacji dużych przedsięwzięć informatycznych.

Dziedzina estymacji projektów informatycznych ciągle się rozwija, starając się dopasować do nowych rodzajów narzędzi informatycznych i nowych klas zastosowań produktów informatycznych. Nowe trendy są związane z objęciem projektów realizowanych z wykorzystaniem technik obiektowych, wzmocnienia technik wielokrotnego użycia kodu oraz integracją z narzędziami typu CASE.

Literatura:

- Boehm, Barry, *Software Engineering Economics*, Prentice-Hall, 1981
- Putnam, Lawrence, Myers Ware, *Measures for Excellence: Reliable Software on Time, Within Budget*, Yordon Press, 1992
- Marciniak, John, *Encyclopedia of Software Engineering*, John Wiley & Sons, Inc.1994
- Humphrey, Watts, *A Discipline for Software Engineering*, Addison-Wesley, 1995