

Zastosowanie metody punktów funkcyjnych - wady i zalety

Ewa Magiera
Uniwersytet Śląski w Katowicach
e-mail: magiera@us.edu.pl

Abstrakt. Metoda punktów funkcyjnych została służy do szacowania oprogramowania zarówno w przypadku nowych projektów, jak i modernizacji oraz rozbudowy istniejących systemów. Metoda ta jest propagowana i aktywnie rozwijana przez the International Function Point Users Group. Celem artykułu jest przedstawienie metody.

1. Wstęp

Najczęściej stosowaną miarą w szacowaniu i ocenie oprogramowania jest obecnie metoda punktów funkcyjnych (the Function Point Analysis – FPA). Pierwsze zasady FPA zostały opracowane przez A. J. Albrechta i jego kolegów z IBM w połowie lat siedemdziesiątych. Następnie IBM zrzekł się wszelkich praw autorskich przekazując FPA do wspólnego korzystania przez wszystkich zainteresowanych. Powołano międzynarodową organizację użytkowników FPA (the International Function Point Users Group – IFPUG) z zarządem w Westerville, Ohio, USA. W ponad dwudziestu krajach całego świata istnieją podobne regionalne stowarzyszenia. Główne zadania IFPUG to opracowywanie i publikacja podręczników z kolejnymi wersjami FPA. FPA „żyje” i rozwija się wraz z technologiami informatycznymi.

Capers Jones [1] wskazuje na następujące mocne strony stosowania FPA:

- FPA jest stosowana bez względu na używany język programowania.
- FPA jest stosowana do szacowania całych systemów informatycznych lub tylko ich poszczególnych modułów.
- FPA jest stosowana do szacowania nowego softwaru jak i w przypadku modernizacji już pracującego.
- Wiele narzędzi programistycznych szacujących koszty i inne wskaźniki związane z realizacją projektów informatycznych bazuje na FPA.

Główne wady FPA to:

- Gwarancję osiągnięcia poprawnych rezultatów dają certyfikowani specjaliści FPA. Główne ośrodki certyfikacji znajdują się w USA, co dla polskich specjalistów i firm jest dużym ograniczeniem, przede wszystkim ze względu na duże koszty podróży, samych szkoleń i seminariów organizowanych przez IFPUG.
- Poprawne wyliczenie punktów funkcyjnych wymaga również dużo czasu, a co za tym idzie jest dość kosztowne.
- Automatyzacja procesu obliczenia punktów funkcyjnych może być obciążona nieznaną (dużą lub małą) dokładnością, jak na razie nie jest stosowana.
- Rezultaty obliczeń FPA w wypadku systemów o rozmiarze mniejszym niż 15 punktów funkcyjnych mogą być niereprezentatywne.
- Brak powiązań (konwersji) pomiędzy standardem IFPUG i różnymi wersjami FPA, które powstały na bazie standardu, jednak rozwijają się w różny sposób i w różnych kierunkach.

Jak na razie FPA jest najlepiej dopracowaną metodą pomiaru softwaru, jest ciągle rozwijana i dobrze opisana, oprócz dokumentacji opracowywanej przez IFPUG, jest tematem wielu książek, podręczników oraz artykułów.

W świecie jest grupa specjalistów posiadających dużą wiedzę i praktyczne doświadczenie w zakresie stosowania FPA zgodnego ze standardem IFPUG.

2. Analiza punktów funkcyjnych

Proces poprawnego stosowania analizy punktów funkcyjnych nie jest procesem trywialnym. Składa się na niego siedem następujących kroków:

2.1. Zdefiniowanie typu procesu liczenia punktów funkcyjnych

Można wyróżnić trzy typy:

- Pierwszy dla nowo powstających projektów, kiedy wszelkie oceny dokonuje się na podstawie wymagań funkcjonalnych przedstawionych przez końcowego użytkownika oraz wymagań co do konwersji danych.
- Drugi dotyczy przypadku modyfikacji istniejącej aplikacji, polegającej na zmianie funkcjonalności, tzn. usunięcie niektórych funkcji, zmiana lub dodanie nowych.
- Trzeci to pomiar istniejącej, pracującej aplikacji.

2.2. Identyfikacja zakresu analizy oraz określenie granic aplikacji

Poprzez zakres analizy rozumie się funkcjonalność, która podlega szacowaniu. IFPUG określa następujące reguły stosowane do wyznaczenia granic aplikacji:

- Wyznaczona granica wynika głównie z punktu widzenia i potrzeb użytkownika, tzn. użytkownik powinien zdefiniować zakres aplikacji, jej biznesową i użytkową funkcjonalność.
- Granice pomiędzy współpracującymi aplikacjami winny wynikać z ich funkcjonalności a nie z uwarunkowań technologicznych.
- Ustanowiona początkowo granica aplikacji jest niezależna od zakresu analizy, za wyjątkiem takich zmian funkcjonalności, których dodanie lub usunięcie spowoduje odpowiednio rozszerzenie lub redukcję granicy aplikacji.

IFPUG opublikował dodatkowe definicje i wskazówki do stosowania w wyznaczaniu granic aplikacji [2].

2.3. Wyliczenie liczby niezgodnionych punktów funkcyjnych dla wszystkich ILF i EIF

Na etapie tym należy zidentyfikować wszystkie logiczne zbiory danych aplikacji (ILF ang. internal logical files oraz EIF ang. external logical files) oraz oszacować ich kompletność. Następnie wyliczyć liczbę niezgodnionych punktów funkcyjnych dla wszystkich ILF i EIF.

ILF (an internal logical file) – to grupa logicznie powiązanych danych, wymaganych, określonych przez użytkownika lub danych kontrolnych utrzymywanych i działających w granicach danej aplikacji. ILF z reguły odpowiada encji w drugiej lub trzeciej postaci normalnej lub nazwie nadanej grupie powiązanych logicznie danych na diagramach przepływów. Dane kontrolne to dane niezbędne do sterowania procesami aplikacji.

EIF (an external interface file) – to określona przez użytkownika grupa logicznie powiązanych danych lub informacji kontrolnych odnoszących się do aplikacji, lecz utrzymywanych w granicach

innej aplikacji. EIF liczony dla jednej aplikacji musi być ILF'em w innej aplikacji. IFPUG zaleca stosowanie następujących reguł w określaniu czy grupa danych jest EIF:

- Są to dane lub informacje kontrolne, tworzące logiczną grupę danych i jednoznacznie identyfikowalne przez użytkownika.
- To grupa danych, która przekracza granice liczonej aplikacji.
- To grupa danych, która nie jest utrzymywana (pamiętana) w granicach danej aplikacji.
- Grupa danych, która jest ILF'em innej aplikacji.

Po zidentyfikowaniu wszystkich ILF i EIF dla każdego z nich należy wyznaczyć wszystkie tzw. RET'y i DET'y, których liczba decyduje o ilości niezgodnionych punktów funkcyjnych.

RET (a record element type) to podgrupa danych w ILF lub EIF, określona przez użytkownika, może być opcjonalna lub obowiązkowa.

Reguły wyznaczania RET:

- Licz jako RET każdą podgrupę danych ILF'a lub EIF'a.
- Jeśli nie można wydzielić żadnych podgrup należy każdy ILF i EIF policzyć jako jeden RET.

DET (a data element type) to unikalne, określone przez użytkownika, nie powtarzające się pole.

Reguły obliczania DET:

- Jako DET należy liczyć każde unikalne, zidentyfikowane przez użytkownika pole, będące elementem ILF'a lub EIF'a.
- Jeśli dwie aplikacje korzystają z tych samych ILF lub EIF ale odwołują się inaczej do podgrup danych, to liczbę DET należy liczyć stosownie do każdej aplikacji. Na przykład: jedna aplikacja wykorzystuje następującą grupę danych adresowych: kod pocztowy, miasto, ulica, nr domu, telefon, natomiast druga aplikacja odnosi się do tych danych jako do bloku, bez rozróżnienia poszczególnych pól. W pierwszym przypadku przypiszemy odpowiedniemu ILF - 5 DET, w drugim 1 DET.
- Każda grupa danych, która umożliwia relację z innym ILF lub EIF musi zostać policzony jako jeden DET.

Na podstawie wyznaczonej liczby RET i DET dla każdego ILF i EIF szacuje się poziom funkcjonalnej kompletności zgodnie z poniższymi zasadami:

Liczba RET	DET	1-19	20-50	51 i więcej niż 51
1		niski	niski	średni
2-5		niski	średni	wysoki
6 i więcej niż 6		średni	wysoki	wysoki

Na podstawie wyznaczonego poziomu kompletności funkcjonalnej możemy przypisać odpowiednią liczbę niezgodnionych punktów funkcyjnych zgodnie z poniższymi zasadami,

dla ILF:

Poziomu kompleksowości funkcjonalnej	Liczba surowych punktów funkcyjnych
--------------------------------------	-------------------------------------

niski	7
średni	10
wysoki	15

dla EIF:

Poziomu kompleksowości funkcjonalnej	Liczba surowych punktów funkcyjnych
niski	5
średni	7
Wysoki	10

Odpowiednio wylicza się liczbę niezgodnionych punktów funkcyjnych dla każdego ILF i EIF, suma daje stosowny wynik.

2.4. Wyliczenie liczby niezgodnionych punktów funkcyjnych dla wszystkich funkcji transakcyjnych

Zidentyfikowanie tzw. funkcji transakcyjnych (ang. external inputs, external outputs, external inquiries) oraz ich kompletności, a następnie, odpowiednio wyliczenie liczby surowych punktów funkcyjnych.

EI (external inputs) – to proces elementarny, któremu poddawane są dane lub dane kontrolne przychodzące spoza granic aplikacji. Podstawowym celem EI jest działanie na/z jednym lub więcej ILF zmieniając jego dane lub/i zmiana zachowania systemu.

EO (external outputs) – to proces elementarny, który wysyła dane lub dane kontrolne poza granice aplikacji. Podstawowym celem EO jest prezentacja informacji użytkownikowi w procesie wyszukiwania, odzyskiwania informacji. Proces powinien zawierać przynajmniej formułę lub wzór matematyczny wyliczający wartość danych lub generować (tworzyć) wyprowadzane dane. EO może również działać na/z jednym lub więcej ILF zmieniając dane lub/i zmiana zachowania systemu.

EQ (external inquiry) – to proces elementarny, który wysyła dane lub dane kontrolne poza granice aplikacji. Podstawowym celem EQ jest prezentacja informacji użytkownikowi poprzez wyszukanie danych lub danych kontrolnych z ILF lub EIF, ale bez korzystania z formuł i wzorów matematycznych oraz tworzenia nowych danych. W trakcie działania EQ nie może nastąpić modyfikacja ILF i zmiana zachowania systemu.

Aby zidentyfikować wszystkie EI, EO, EQ należy każdy proces elementarny aplikacji poddać analizie co do jego podstawowej celowości i zgodnie z przedstawioną niżej zależnością określić typ funkcjonalny:

Funkcja:	Transakcyjny typ funkcyjny:		
	EI	EO	EQ
Zmiana zachowania systemu.	główny	możliwy	niedozwolony
Modyfikacja jednego lub więcej ILF	główny	możliwy	niedozwolony
Prezentacja informacji użytkownikowi	możliwy	główny	główny

Każdy proces elementarny musi być jednoznacznie określony i może być liczony tylko raz. Więcej, bardziej szczegółowych informacji o klasyfikacji procesów elementarnych można znaleźć w [2].

Przypisanie każdemu EO, EI, EQ odpowiedniej liczby punktów funkcyjnych zależy od liczby FTR (ang. a file type referenced) i DET (ang. a data element type).

FTR to:

- ILF czytany lub modyfikowany przez EQ, EO, EI,
- EIF z którego czytane są informacje.

Definicja DET została przedstawiona wcześniej.

Reguły liczenia FTR są następujące:

- Licz każdy modyfikowany ILF jako jeden FTR.
- Każdy czytany ILF oraz EIF licz jako jeden FTR.
- Każdy modyfikowany i czytany plik licz tylko raz.

Na podstawie wyznaczonej liczby FTR i DET szacuje się poziom funkcjonalnej kompletności zgodnie z poniższymi zasadami

dla EI:

Liczba FTR	DET	1-4	5-15	16 i więcej niż 16
0 - 1		niski	niski	średni
2		niski	średni	wysoki
3 i więcej niż 3		średni	wysoki	wysoki

dla EO i EQ:

Liczba FTR	DET	1-5	6-19	20 i więcej niż 20
0 - 1		niski	niski	średni
2-3		niski	średni	wysoki
4 i więcej niż 4		średni	wysoki	wysoki

Na podstawie wyznaczonego poziomu kompletności funkcjonalnej możemy przypisać odpowiednią liczbę niezgodzonych punktów funkcyjnych zgodnie z poniższymi zasadami,

dla EI i EQ:

Poziomu kompletności funkcjonalnej	Liczba surowych punktów funkcyjnych
niski	3
średni	4
wysoki	6

dla EO:

Poziomu kompletności funkcjonalnej	Liczba surowych punktów funkcyjnych
niski	4
średni	5
wysoki	7

2.5. Obliczenie współczynnika dopasowania wartości VAF (ang. the value adjustment factor)

Współczynnik dopasowania wartości VAF oparty jest na 14 generalnych charakterystykach GSC (ang. general system characteristic), których zadaniem jest oszacowanie funkcjonalności liczonej aplikacji. GSC to zbiór 14 pytań, które pozwalają na całkowite oszacowanie złożoności systemu. Dokładny opis wszystkich 14 charakterystyk można w raz z opisem ocen można znaleźć w [2].

Aby otrzymać VAF, należy zgodnie z [2] wykonać następujące kroki.

- Oszacuj każdą z 14 charakterystyk, wynik należy umieścić na skali o zakresie od 1 do 5, co odpowiada określeniu tzw. stopnia wylwu DI (ang. the degree of influence).
- Oblicz całkowity stopień wpływu TDI (ang. the total degree of influence) dodając otrzymane w punkcie 1 wyniki.
- Oblicz VAF wstawiając TDI do równania:

$$VAF=(TDI*0.01)+0.65$$

2.6. Wylczenie końcowej wartości punktów funkcyjnych

Ostatnim krokiem w analizie punktów funkcyjnych jest końcowe obliczenie liczby punktów funkcyjnych zależne od przedstawionego w pkt.1 typu liczenia punktów funkcyjnych.

W przypadku projektowania, instalowania i uruchamiania aplikacji po raz pierwszy rozważyć następujące elementy:

- Funkcjonalność aplikacji wynikające z wymagań przedstawionych przez użytkownika.
- Funkcjonalność wynikająca z wymogu konwersji danych.
- VAF.

Do obliczenia całkowitej liczby punktów funkcyjnych należy użyć następującej reguły:

$$DFP=(UFP+CFP)*VAF$$

gdzie:

- DFP (od ang. development project function point count) – całkowita liczba punktów funkcyjnych dla nowego projektu,

- UFP (ang. unadjusted function point) – niezgodniona liczba punktów funkcyjnych dla funkcjonalności aplikacji, dostępnej dla użytkownika końcowego po instalacji,
- CFP (ang. conversion function point) - niezgodniona liczba punktów funkcyjnych wynikająca z konwersji danych,
- VAF – współczynnik dopasowania wartości, uzyskany zgodnie z zasadmi przedstawionymi w pkt. 2.5.

W przypadku modyfikacji funkcjonalności aplikacji (pkt. 2b) wzór na końcowe obliczenie liczby punktów funkcyjnych ma następującą postać:

$$\mathbf{EFP = [(ADD+CHGA+CFP)*VAFA]+(DEL*VAFB)}$$

gdzie:

- EFP (od: the enhancement project function point) – końcowa wartość punktów funkcyjnych w wypadku modyfikacji funkcjonalności aplikacji.
- ADD – niezgodniona liczba punktów funkcyjnych odzwierciedlająca te funkcje, które zostaną dodane w procesie modyfikacji aplikacji,
- CHGA - niezgodniona liczba punktów funkcyjnych liczona dla modyfikacji istniejących funkcji,
- CFP (ang. conversion function point) - niezgodniona liczba punktów funkcyjnych wynikająca z konwersji danych,
- VAFA – współczynnik dopasowania wartości, uzyskany zgodnie z zasadami przedstawionymi w pkt. 5., liczony po modyfikacji aplikacji,
- DEL - niezgodniona liczba punktów funkcyjnych odzwierciedlająca funkcjonalność elementów, które zostaną usunięte w wyniku modyfikacji aplikacji,
- VAFB – współczynnik dopasowania wartości, uzyskany zgodnie z zasadami przedstawionymi w pkt. 2.5., liczony przed modyfikacją aplikacji.

W przypadku obliczania ilości punktów funkcyjnych dla pracującej aplikacji, należy użyć następującej formuły:

$$\mathbf{AFP=AD*VAF}$$

gdzie:

- AFP - końcowa wartość punktów funkcyjnych,
- AD - niezgodniona liczba punktów funkcyjnych wynikająca z funkcjonalności aplikacji, dostępnej dla użytkownika końcowego,
- VAF – współczynnik dopasowania wartości, uzyskany zgodnie z zasadami przedstawionymi w pkt. 2.5., liczony dla danej aplikacji.

Aby oszacować funkcjonalność aplikacji po modyfikacji, należy zastosować następującą formułę:

$$\mathbf{AFP=[(UFPB+ADD+CHGA)-(CHGB+DEL)]*VAF}$$

gdzie:

- AFP - końcowa wartość punktów funkcyjnych,
- UFPB - niezgodniona liczba punktów funkcyjnych wynikająca z funkcjonalności aplikacji, przed modyfikacją,
- ADD – niezgodniona liczba punktów funkcyjnych odzwierciedlająca te funkcje, które zostaną dodane w procesie modyfikacji aplikacji,
- CHGA - niezgodniona liczba punktów funkcyjnych modyfikacji istniejących funkcji,
- liczona po modyfikacji aplikacji,
- CHGB - niezgodniona liczba punktów funkcyjnych wynikająca z modyfikacji istniejących funkcji,
- Liczba odpowiada funkcjonalności zmodyfikowanych elementów przed procesem modyfikacji,
- DEL - niezgodniona liczba punktów funkcyjnych odzwierciedlająca te funkcje, które zostaną usunięte w wyniku modyfikacji aplikacji,
- VAFA – współczynnik dopasowania wartości, uzyskany zgodnie z zasadami przedstawionymi w pkt. 2.5., liczony dla aplikacji po przeprowadzonej modyfikacji.

3. Podsumowanie

Metoda punktów funkcyjnych nie jest powszechnie stosowana w Polsce. Jest natomiast metodą sprawdzoną w świecie, gdzie szacowanie softwaru jest zjawiskiem powszechnym, pozwala na efektywne zarządzanie projektami informatycznymi oraz kosztami wynikającymi z ich realizacji.

Bibliografia

1. Capers Jones: Software Assessments, benchmark and best practces, Addison Wesley, 2000, ISBN 0-201-48542-7
2. Function Point Counting Practices Manual. Release 4.1, The International Function Point Users Group, (www.ifpug.org)
3. Garmus D., Herron D.: Function Point Analysis. Measurement Practices for Successful Software Projects, Addison Wesley , 2001, ISBN 0-201-69944-3