PRACE NAUKOWE UNIWERSYTETU EKONOMICZNEGO WE WROCŁAWIU nr 55
Informatyka Ekonomiczna 13 2009

Beata Czarnacka-Chrobot, Andrzej Kobyliński
Uniwersytet Ekonomiczny we Wrocławiu

OCENA MIAR ZAKRESU PRODUKTU PROGRAMOWEGO

Streszczenie: Zasadniczym celem zarządzania przedsięwzięciami informatycznymi jest dostarczenie produktu programowego zgodnego z wymaganiami zleceniodawcy bez przekroczenia zaplanowanego czasu i budżetu. Osiągnięcie tego celu utrudnia wiele czynników, wśród których jednym z głównych jest brak jednoznacznej miary zakresu oprogramowania. Rozmiar produktu takiego przedsięwzięcia stanowi bowiem podstawę do estymacji jego pracochłonności, czasu i kosztów realizacji, a także jakości samego produktu. Dlatego brak takiej miary można uznać za jeden z głównych problemów inżynierii oprogramowania. Celem niniejszego artykułu jest zaprezentowanie oraz ocena różnych rodzajów miar zakresu produktu programowego.

Słowa kluczowe: inżynieria oprogramowania, produkt programowy, miara zakresu produktu, standardy ISO/IEC.

1. Wstęp

Podstawowym celem zarządzania przedsięwzięciem informatycznym jest terminowe dostarczenie produktu programowego, zgodnego z wymaganiami zleceniodawcy i bez przekroczenia zaplanowanego budżetu. Jego osiągnięcie utrudnia wiele czynników, wśród których jednym z zasadniczych jest brak jednoznacznej miary zakresu (rozmiaru) produktu programowego. Tymczasem „Wymiarowanie zakresu oprogramowania [...] jest tak ważne dla jego dostawców, jak pomiar powierzchni budynku dla jego wykonawcy. Wszystkie pozostałe dane, włączając w to nakłady pracy niezbędne do realizacji przedsięwzięcia, jego harmonogram i koszty, bazują na [...] zakresie produktu programowego" [Parthasarathy 2007, s. 149]. Co więcej, miary rozmiaru oprogramowania są wykorzystywane jako czynnik normalizujący w większości miar jakości produktu programowego [Fenton 2000, s. 92]. Dlatego brak jednoznacznej miary zakresu takiego produktu można uznać za jeden z zasadniczych problemów inżynierii oprogramowania, którą Institute of Electrical and Electronics Engineers (IEEE) definiuje jako dyscyplinę będącą „zastosowaniem systematycznego, zdyscyplinowanego, kwantyfikowalnego podejścia do rozwoju, obsługi i utrzymania oprogramowania" [*IEEE Std 610.12…* 1990], za: [Khelifi, Abran, Buglione

2004, s. 90]. Podejście kwantyfikowalne oznacza wszak, że wymiarowanie powinno stanowić jej cechę immanentną. Z powyższych powodów wynikają kilkudziesięcioletnie poszukiwania skutecznych rozwiązań w obszarze wymiarowania zakresu oprogramowania, które to poszukiwania w ostatnim czasie zaowocowały akceptacją przez międzynarodowe organizacje standaryzacyjne jednej z proponowanych koncepcji wraz z kilkoma opartymi na niej metodami.

2. Rodzaje miar zakresu produktu programowego

Zgodnie z definicją zaproponowaną w pracach Fentona [2000] i Zuse [1991], przez wymiarowanie w kontekście inżynierii oprogramowania należy rozumieć „proces, w którym atrybutom elementów świata rzeczywistego przydzielane są liczby lub symbole w taki sposób, aby charakteryzować te atrybuty według jasno określonych zasad. Jednostki przydzielane atrybutom w ramach tego procesu nazywane są miarą (*measure*) danego atrybutu" [Fenton 2000, s. 88-89]. Warto w tym miejscu zauważyć, że w literaturze przedmiotu pojęcie „miara" jest często utożsamiane z pojęciem „metryka". Podejście to nie jest do końca poprawne: wprawdzie każda metryka jest miarą, lecz nie każdą miarę można uznać za metrykę (musi być spełniona tzw. nierówność trójkąta) [Zuse 1991, s. 28-29].

Zakres produktu programowego bywa rozpatrywany w praktyce z perspektywy:

– długości programów składających się na produkt, mierzonej liczbą tzw. jednostek programowych,
– złożoności konstrukcyjnej produktu, wyznaczanej w tzw. jednostkach złożoności konstrukcyjnej,
– funkcjonalności produktu (złożoności funkcjonalnej), wyrażanej w tzw. jednostkach funkcjonalności.

W związku z powyższym do wymiarowania zakresu oprogramowania wykorzystuje się następujące rodzaje jednostek:

1) jednostki programowe[1] – w tym przede wszystkim linie kodu źródłowego (*source lines of code*), ale uwzględnia się także liczbę poleceń, liczbę zdań, liczbę zdań wykonywalnych, liczbę instrukcji języka maszynowego [Kobyliński 1991],

2) jednostki złożoności konstrukcyjnej produktu – obecnie wykorzystywane głównie w postaci punktów obiektowych (*object points*), zwanych także konstrukcyjnymi [Parthasarathy 2007, s. 155-156],

3) jednostki funkcjonalności produktu – w tym przede wszystkim punkty funkcyjne (*function points*) oraz powstałe na ich bazie warianty w postaci: pełnych punktów funkcyjnych (*full function points*), punktów charakterystycznych (*feature points*), punktów przypadków użycia (*use case points*) czy punktów internetowych (*Internet points*).

[1] Flasiński nazywa te jednostki miarami wolumenowymi (zob. [Flasiński 2006, s. 115]).

3. Jednostki programowe

Jednostki programowe w odniesieniu do oprogramowania stanowią miarę naturalną, łatwą w zastosowaniu i uniwersalną, tj. niezależną od kategorii mierzonego produktu, co niewątpliwie stanowi ich zaletę. Tyle tylko, że nie mierzą one ani złożoności programów, ani ich rozmiaru, a jedynie atrybut „długość programu", chociaż w odniesieniu do zakresu oprogramowania to one są najczęściej stosowane w praktyce [Fenton 2000, s. 89, 93; Parthasarathy 2007, s. 149]. Do ich podstawowych wad zalicza się:

– dowolność w definiowaniu jednostek – różnice między dwoma skrajnymi metodami obliczania liczby jednostek programowych, tzn. między metodą polegającą na uwzględnianiu wszystkich linii kodu a taką, w której uwzględnia się tylko linie wykonywane, nawet dla tego samego języka programowania mogą wyrażać się stosunkiem 5:1 [Fenton 2000, s. 91];
– brak możliwości bezpośredniego porównania programów napisanych w różnych językach oraz ignorowanie różnic między językami programowania – zarówno co do ich poziomu, jak i co do stopnia rozbudowania składni;
– możliwość obliczenia długości programu jedynie w odniesieniu do istniejącego już kodu, a ewentualna prognoza tego atrybutu, nawet w wykonaniu doświadczonych programistów, jest zwykle nietrafna i obciążona dużą dozą subiektywizmu; metody oparte na jednostkach programowych nie mogą więc być wykorzystywane w roli wiarygodnego i obiektywnego estymatora do planowania atrybutów przedsięwzięcia informatycznego we wczesnych fazach cyklu jego życia (np. w stadium analizy, a nawet projektowania);
– ostateczny wymiar programu nie w pełni odzwierciedla wykonane prace – często podczas realizacji powstają znaczne fragmenty kodu, które ostatecznie nie są umieszczane w produkcie finalnym (np. narzędzia opracowane wyłącznie na użytek powstającego produktu; alternatywne rozwiązania, które początkowo zaimplementowano, a później z nich zrezygnowano; fragmenty wspomagające testowanie itp.); szacuje się, że tego typu odrzucony kod odpowiada kilkudziesięciu procentom kodu oddawanego ostatecznie do użytku;
– zależność pracochłonności od długości programu, zdeterminowanej z kolei wykorzystywanym językiem programowania – w ten sposób wykorzystywany język programowania wpływa na koszty i czas realizacji produktu, które to atrybuty mogą się znacznie różnić nawet dla produktów o identycznym zbiorze funkcji i zbliżonych cechach jakościowych; jednostki programowe faworyzują przy tym produkty o nadmiarowej długości w porównaniu z produktami zwięzłymi;
– nieistotność dla zleceniodawcy przedsięwzięcia informatycznego – określenie wymagań zleceniodawcy w jednostkach długości programu nie ma bowiem dla niego najmniejszego znaczenia: nie pozwala na interpretację wyników w kontekście i języku biznesowym, przez co w żaden sposób nie sprzyja udziałowi klienta w procesie projektowania, a to z kolei stanowi jeden z zasadniczych

czynników powodzenia realizacji przedsięwzięcia informatycznego; miara ta ma więc znaczenie jedynie dla wykonawców oprogramowania;
– brak możliwości obiektywnego i wiarygodnego wyznaczenia na podstawie długości programu pracochłonności wstępnych stadiów przedsięwzięcia – nakłady pracy ponoszone w tych stadiach w żaden sposób nie zależą od wynikowej długości programu, a od wymagań użytkownika;
– brak zgodności z ekonomiczną definicją produktywności – określany jako paradoks metryk programowych (paradoks produktywności) [Jones 1995], który powoduje, że faworyzują one języki programowania o mniejszej wydajności (niższego poziomu); w przypadku zastosowania wydajniejszego języka programowania, zmniejszającego koszty całkowite przedsięwzięcia, jednostki programowe wskazują bowiem na wzrost kosztów jednostkowych, a zatem na wzrost produktywności, podczas gdy powinno być dokładnie odwrotnie[2];
– brak możliwości porównania rzeczywiście zrealizowanego zakresu produktu z zakresem wymaganym przez zleceniodawcę przedsięwzięcia.

Z powyższego wynika, że jednostki programowe nie odzwierciedlają prawdziwych możliwości produktu programowego: program o większej liczbie takich jednostek może dostarczać zleceniodawcy mniej wymaganych przez niego funkcji i cech, a kosztować więcej, jak również programy o bardzo zbliżonych możliwościach mogą się znacznie różnić długością, a zatem kosztami i czasem realizacji. Toteż nie są one właściwe ani z punktu widzenia szacowania rozmiaru produktu programowego, a przez to pracochłonności jego rozwoju, ani nawet z perspektywy pomiaru tych atrybutów. W związku z tym podjęto prace zmierzające do poszukiwania sposobów wymiarowania zakresu oprogramowania z innych perspektyw niż długość programu. Jedną z takich perspektyw stanowi złożoność konstrukcyjna produktu programowego.

4. Jednostki złożoności konstrukcyjnej produktu

Fenton [2000, s. 93-94] podkreśla, że zaproponowano setki miar złożoności konstrukcyjnej produktu programowego, z których większość ogranicza się do kodu programu. Najwcześniej pojawiły się takie jednostki jej pomiaru, jak np. liczba ścieżek elementarnych w grafie przepływu sterowania programu, liczba wystąpień operatorów i operandów w programie czy liczba węzłów (przecięć) w programie. Nieco później zwrócono uwagę nie tylko na złożoność przepływów sterowania, ale również i przepływów danych w programie czy głębokość zagnieżdżenia. Pojawiły się również liczne miary złożoności konstrukcyjnej oprogramowania, opierające się na jego wymaganych elementach składowych (np. modułach, procedurach, przepływach informacji między modułami)[3]. Natomiast za najpopularniejsze obecnie podej-

[2] Odpowiedni przykład zamieszczono w pracy Czarnackiej-Chrobot [2001, s. 27-55].

[3] Szczegółowy opis tego typu miar, jak również obszerne badanie dotyczące korelacji między miarami znaleźć można w pracy Kobylińskiego [1991].

ście do jej wymiarowania Parthasarathy [2007, s. 155-156] uważa miarę bazującą na punktach obiektowych (konstrukcyjnych), które w odróżnieniu od dalej przedstawionych jednostek funkcjonalności produktu przypisywane są nie jego elementom funkcjonalnym, a konstrukcyjnym. Za pomocą tej jednostki dokonuje się pomiaru oprogramowania poprzez wyznaczenie liczby wyświetlanych ekranów, tworzonych raportów i modułów programowych opracowywanych w języku trzeciej generacji w celu uzupełnienia kodu w języku generacji czwartej (zob. [Flasiński 2006, s. 117; Sommerville 2003, s. 514]). Te elementy konstrukcyjne, w tym podejściu nazywane obiektami, są klasyfikowane ze względu na różny poziom ich złożoności poprzez przypisanie im odpowiedniej liczby punktów obiektowych.

Zaproponowane jednostki złożoności konstrukcyjnej oprogramowania również spotkały się z silną krytyką [Fenton 2000, s. 94]. Wobec tych ograniczających się do kodu programu, uznawanych za zbyt uproszczone, stosuje się wszystkie powyżej przedstawione zarzuty dotyczące jednostek programowych. Dlatego w praktyce takie jednostki nie stanowią lepszej podstawy do wyznaczania atrybutów przedsięwzięcia informatycznego niż jednostki programowe, tym bardziej że obliczanie wartości tych miar jest stosunkowo trudne, a korelacja między jednostkami programowymi a jednostkami konstrukcyjnymi jest bardzo silna [Kobyliński 1991]. Kolejnym propozycjom zarzucano uwzględnianie jedynie poszczególnych, szczegółowych aspektów złożoności, w dużym stopniu uzależnionych od stosowanej technologii, co również nie daje właściwej podstawy do prognoz. Natomiast w odniesieniu do miar złożoności konstrukcyjnej oprogramowania, opierających się na jego wymaganych elementach składowych, to można je wprawdzie wykorzystać w stadium projektowania, a zatem jeszcze przed przystąpieniem do kodowania, jednak przedsięwzięcie na tym etapie charakteryzuje się już takim stopniem zaawansowania, że na podjęcie racjonalnej decyzji inwestycyjnej przez jego zleceniodawcę jest za późno. Także wobec punktów obiektowych wysuwa się poważny zarzut: pracochłonność wyprowadzana na ich podstawie ogranicza się jedynie do etapów kodowania i testowania, co prowadzi do błędnych szacunków kosztów i czasu realizacji dla całego cyklu życia projektu [Parthasarathy 2007, s. 158-159].

Z powyższego wynika, że jednostki złożoności konstrukcyjnej produktu programowego nie sprawdzają się w roli wiarygodnego i obiektywnego estymatora na etapie analizy cyklu życia przedsięwzięcia informatycznego, a zatem nie pozwalają na planowanie kluczowych jego atrybutów. Poszczególne ich warianty mają także inne wady, które zdecydowanie ograniczają ich przydatność we właściwym wymiarowaniu atrybutów projektu. Jedną z nich, odnoszącą się do wszystkich jednostek złożoności konstrukcyjnej i niewystarczająco sygnalizowaną w literaturze przedmiotu, jest pomijanie punktu widzenia zleceniodawcy przedsięwzięcia. Te miary zakresu produktu mają bowiem dla niego znaczenie drugorzędne: o ile złożoność konstrukcyjna stanowi bardzo istotny atrybut dla wykonawcy, o tyle z perspektywy jego odbiorcy priorytetowym celem produktu jest faktyczne zaspokojenie jego potrzeb. Oznacza to, że to wymagany zbiór funkcji i cech produktu powinien determinować

jego złożoność konstrukcyjną – nie zaś odwrotnie. W efekcie nie sprzyjają one zaangażowaniu zleceniodawcy w przedsięwzięcie, co wiąże się z istotnym ryzykiem niepowodzenia.

5. Jednostki funkcjonalności produktu

Jednostki służące do wymiarowania zakresu produktu z punktu widzenia jego funkcjonalności nie wykazują ograniczeń właściwych jednostkom programowym i jednostkom złożoności konstrukcyjnej oprogramowania, chociaż można mieć zastrzeżenia co do ich uniwersalności, której stopień jest różny w zależności od bazowej jednostki funkcjonalności, jak również w odniesieniu do stosunkowo dużego stopnia skomplikowania opartych na nich metod wymiarowania.

Podstawową jednostkę funkcjonalności produktu programowego, w oparciu o którą rozwinięto wszystkie inne niżej wymienione jednostki, stanowią punkty funkcyjne. Twórcą ich koncepcji, a zarazem koncepcji wymiarowania funkcjonalnego oprogramowania, był Albrecht, który już pod koniec lat 70. ubiegłego stulecia zdefiniował punkty funkcyjne jako „liczbę bezwymiarową, którą uważamy za efektywną relatywną miarę wartości funkcji dostarczanych naszemu klientowi" [Albrecht 1979, s. 83-92]. Uznał on bowiem, że zakres produktu powinno się wymiarować w jednostkach istotnych dla jego odbiorcy, czyli z punktu widzenia jego użytkowej funkcjonalności, a nie z perspektywy długości programów, która w praktyce nie ma dla odbiorcy większego znaczenia. Dodatkowo założył on, iż jednostki te powinny być niezależne od wykorzystywanego języka programowania. W efekcie Albrecht stworzył metodę znaną jako Function Points Analysis, opartą na pomiarze i estymacji funkcjonalności wymaganej przez zleceniodawcę. Stąd grupę metod, która powstała na skutek rozwoju tego podejścia, określa się mianem metod wymiarowania rozmiaru funkcjonalnego (*functional size measurement methods*).

Punkty funkcyjne to zatem **miara biznesowej funkcjonalności produktu programowego** – zarówno tej, która ma być dostarczona zleceniodawcy w celu zaspokojenia jego wymagań, jak i tej, która rzeczywiście została zrealizowana. Jako że nie ma możliwości pomiaru funkcjonalności w sposób bezpośredni, jednostki te wyprowadza się w oparciu o pewne mierzalne charakterystyki, decydujące o zakresie oprogramowania: „Stanowią one ilościową reprezentację [...] funkcji [...] i danych, które działają wspólnie jako aplikacja komputerowa, [...] bazując na programowych komponentach odpowiadających wymaganiom" [Parthasarathy 2007, s. 154]. Opisywane jednostki uwzględniają więc perspektywę odbiorcy produktu, co w szczególności wyraża się w:

- wymiarowaniu zakresu oprogramowania z punktu widzenia atrybutu o pierwszorzędnym znaczeniu dla zleceniodawcy, mianowicie jego funkcjonalności[4];

[4] O pierwszorzędnym znaczeniu funkcjonalności dla użytkowników aplikacji świadczą wyniki badania wspomnianego przez Kobylińskiego [2005, s. 112].

- możliwości wyznaczenia wymaganej funkcjonalności na bazie specyfikacji wymagań już w stadium analizy cyklu życia przedsięwzięcia, a na tej podstawie wystarczająco obiektywnego i wiarygodnego zaplanowania kluczowych jego atrybutów, co pozwala zleceniodawcy na:
  - podjęcie racjonalnej decyzji o przystąpieniu do informatycznego przedsięwzięcia inwestycyjnego o przewidywanej funkcjonalności, kosztach i czasie lub o zaniechaniu takiej inwestycji,
  - wybór wariantu przedsięwzięcia (konstrukcja produktu od podstaw, zakup produktu gotowego i jego kastomizacja, doskonalenie produktu istniejącego lub jedynie jego utrzymanie),
  - porównanie i ocenę konkurencyjnych ofert dostawców produktów programowych pod kątem oferowanej funkcjonalności, kosztów jej dostarczenia oraz czasu realizacji, czemu sprzyja także to, iż zakres produktu wyrażony w takich jednostkach nie zależy od metodyki realizacji przedsięwzięcia ani od stosowanej technologii;
- stosunkowo łatwym kontrolowaniu przebiegu przedsięwzięcia przez zleceniodawcę; w oparciu o wyznaczoną wymaganą funkcjonalność nawet klient bez wiedzy i doświadczenia w projektowaniu systemów oprogramowania i wykorzystywanej technologii może monitorować działania związane z realizacją kontraktu; wynika to z faktu, iż tak wyrażony zakres produktu nie tylko jest niezależny od zastosowanej technologii, ale także uwzględnia wszystkie standardowe stadia cyklu życia oprogramowania, a zatem można wyznaczyć nakłady niezbędne do realizacji poszczególnych jego etapów;
- możliwości oceny funkcjonalności rzeczywiście zrealizowanej i porównania jej z funkcjonalnością wymaganą przez zleceniodawcę w celu stwierdzenia, w jakim stopniu zleceniobiorca wywiązał się ze swoich zobowiązań oraz ustalenia faktycznej należności za dostarczony produkt;
- zgodności z ekonomiczną definicją produktywności, a zatem w prawidłowości uzyskanych na ich bazie wskaźników z perspektywy ekonomicznej; oznacza to, że w przypadku zmniejszenia kosztów całkowitych przedsięwzięcia, wynikającego z wykorzystania wydajniejszego języka programowania, zastosowanie jednostek funkcjonalności wskazuje na spadek kosztów jednostkowych, a zatem na wzrost produktywności[5].

Z powyższego wynika, iż jednostki funkcjonalności służą do oceny zakresu produktu na bazie kryteriów nie tylko dostrzeganych przez ich odbiorców bez potrzeby angażowania ich w szczegóły techniczne i implementacyjne, ale także postrzeganych przez nich jako istotne. W związku z tym stosowanie jednostek funkcjonalnych sprzyja ściślejszej partycypacji zleceniodawcy w przedsięwzięciu, co zwiększa prawdopodobieństwo precyzyjnego wyspecyfikowania wymagań i akceptacji produktu, a to z kolei świadczy o powodzeniu przedsięwzięcia.

[5] Odpowiedni przykład zamieściła Czarnacka-Chrobot [2001, s. 27-55].

Również z perspektywy wykonawców oprogramowania metody oparte na jednostkach funkcjonalnych charakteryzują się dużą użytecznością, jako że pozwalają na:

- analizę, ocenę i porównanie atrybutów przedsięwzięć zrealizowanych i produktów będących ich efektem, nawet jeśli wykorzystywane były różne języki programowania; wynika to z faktu, iż pomiar zakresu produktu w jednostkach funkcjonalności wykazuje niezależność technologiczną; działania te z kolei umożliwiają:
  - określenie produktywności właściwej dla danej organizacji projektującej (zespołu projektowego), co stanowi podstawę szacowania pracochłonności, kosztów i czasu realizacji przyszłych przedsięwzięć, a także działań doskonalących i związanych z utrzymaniem gotowych produktów,
  - wyznaczenie rozkładu pracochłonności i kosztów w czasie oraz w poszczególnych stadiach cyklu życia przedsięwzięcia,
  - określenie wielkości zmiany zakresu produktu wynikającej z modyfikowania wymagań przez zleceniodawcę w trakcie trwania projektu, co powinno skutkować odpowiednią korektą ocen estymacyjnych,
  - wyciągnięcie wniosków dotyczących wpływu konkretnych decyzji i czynności na przedsięwzięcie;
- analizę, ocenę i porównanie parametrów działania zespołów projektowych, co pozwala na:
  - obserwację i ocenę produktywności bieżącej pracy projektantów oraz sygnalizowanie sytuacji niezgodnych z planem,
  - rozpoznanie i przedstawienie trendów produktywności zespołów projektowych,
  - określenie czynnika wykorzystania czasu pracy (udziału czasu pracy netto, czyli czasu pracy faktycznie poświęconego na działania projektowe, w czasie pracy brutto, tj. w czasie pracy możliwym do wykorzystania),
  - wykazanie, czy zasoby będące w dyspozycji zespołów projektowych są wykorzystywane w sposób wydajny.

Badania na temat obiektywności i wiarygodności szacowania zakresu produktu programowego w jednostkach funkcjonalności we wczesnych fazach cyklu życia przeprowadziła organizacja International Benchmarking Standards Group – ISBSG [*The ISBSG Report*… 2005, s. 5-6]. Badania były prowadzone dwutorowo: dotyczyły zarówno prognozy tak wyrażonego zakresu jedynie na bazie uzyskanego w stadium analizy modelu danych, jak i jego metodycznej kalkulacji na bazie specyfikacji wymagań. Instytucja ta uznała, iż zakres produktu wyrażony w jednostkach funkcjonalności w obu przypadkach jest szacowany w sposób wystarczająco obiektywny i trafny w zestawieniu z obliczeniami dokonanymi w oparciu o produkt finalny – spełnione są bowiem warunki konieczne takiego wymiarowania[6]. Przy tym oceny esty-

[6] W celu uznania metody za wiarygodną stosuje się kryterium spełniające następujący warunek: dla co najmniej 80% przypadków metoda umożliwia uzyskanie takich ocen estymacyjnych, które nie przekraczają założonego względnego błędu równego 30% uzyskanej w rzeczywistości wartości –

macyjne wyprowadzone na podstawie specyfikacji wymagań charakteryzują się większą wiarygodnością w porównaniu z uzyskanymi na bazie samego modelu danych: w pierwszym przypadku 70% ocen szacunkowych było nie mniejszych niż faktyczny zakres produktu, w drugim zaś współczynnik ten wynosił 62% takich ocen. W obu przypadkach założono dopuszczalny błąd szacunku jedynie na poziomie ±10%. Odnośnie do obiektywności estymacji zakresu produktu przy wykorzystaniu jednostek funkcjonalności stwierdzono, że dwóch niezależnych specjalistów otrzymuje wyniki różniące się jedynie o ±10%, jednakże pod warunkiem właściwego wyspecyfikowania wymagań [International Software… 2008]. Wnioski płynące z badań ISBSG potwierdza także m.in. Parthasarathy [2007, s. 292], stwierdzając obiektywność i relatywnie wysoki poziom wiarygodności szacowania zakresu produktu programowego w oparciu o punkty funkcyjne.

Możliwość wyznaczenia zakresu produktu w jednostkach funkcjonalności we wczesnej fazie realizacji przedsięwzięcia oraz niezależność tak wyrażonego zakresu od technologii realizacji to zdaniem Fentona [2000, s. 98] podstawowe powody, dla których jednostki te są coraz szerzej używane do wyznaczenia pracochłonności w kontraktach na opracowanie oprogramowania, jak również coraz częściej zastępują jednostki programowe przy określaniu produktywności projektantów i niezawodności programów.

Wymiarowanie zakresu produktu w oparciu o jednostki funkcjonalności nie jest jednak pozbawione wad. Wobec metody bazującej na punktach funkcyjnych w jej pierwotnej formie opracowanej przez Albrechta kierowano przede wszystkim zarzut braku uniwersalności. Pojawiły się bowiem wątpliwości, czy za pomocą takich jednostek można właściwie zmierzyć zakres wszystkich kategorii produktów programowych. O ile raczej nie budziła (i nie budzi) kontrowersji adekwatność wymiarowania funkcjonalności systemów oprogramowania wspomagających zarządzanie, o tyle w przypadku produktów charakteryzujących się niewielką liczbą wejść i wyjść, a jednocześnie wysoką wewnętrzną złożonością przetwarzania[7], wymiarowanie funkcjonalności jest uważane za niekompletne.

Skutkiem było pojawienie się propozycji stanowiących mutacje opisywanych jednostek, których zadaniem było rozszerzenie możliwości wymiarowania funkcjonalności produktu – przede wszystkim na systemy czasu rzeczywistego. W połowie lat 80. ubiegłego wieku Jones zaproponował podejście bazujące na punktach charakterystycznych – za jego pomocą można dodatkowo wymiarować złożoność algorytmów[8]. Natomiast dekadę później na University of Quebec w Montrealu opracowano

ewentualnie 75% przypadków *versus* względny błąd równy 25% lub 70% przypadków *versus* względny błąd równy 20% bądź 60% przypadków *versus* względny błąd równy 10% (zob. np. [Abran, Robillard 1993, s. 81]).

[7] Za takie uważa się np. systemy czasu rzeczywistego, systemy operacyjne, systemy wbudowane, oprogramowanie telekomunikacyjne, systemy sterowania procesami, programy naukowe o skomplikowanych algorytmach matematycznych, systemy eksperckie, CAD (Computer Aided Design).

[8] Informacje o tej metodzie można znaleźć na stronach internetowych Software Productivity Research: http://www.spr.com/products/feature.shtm i http://www.spr.com/products/choosing.shtm.

sposób wymiarowania funkcjonalności oprogramowania w oparciu o pełne punkty funkcyjne, rozwijany później przez organizację COSMIC (Common Software Measurement International Consortium). Ten sposób wymiarowania okazał się użyteczny zarówno dla aplikacji biznesowych, jak i dla systemów uwarunkowanych czasowo. Inne jednostki funkcjonalności produktu, tj. punkty przypadków użycia i punkty internetowe, chociaż podobnie jak punkty charakterystyczne i pełne punkty funkcyjne wywodzą się z koncepcji zaproponowanej przez Albrechta, mają dużo bardziej ograniczone zastosowanie: ta pierwsza jednostka jest dedykowana tylko dla produktów uzyskiwanych przy zastosowaniu obiektowych technik projektowania, druga natomiast – dla projektów stron internetowych.

Kolejnym argumentem krytycznym, podnoszonym wobec wymiarowania zakresu produktu programowego na bazie jednostek funkcjonalności, jest duży stopień jego skomplikowania, odnoszący się zarówno do trudności semantycznych, jak i do dużej liczby kroków niezbędnych do jego realizacji, w związku z czym wymaga ono zastosowania specjalnych metod. Argument ów przestaje być po części aktualny, jeżeli zważy się, że istnieją liczne narzędzia, które takie wymiarowanie wspomagają (zob. np. [Czarnacka-Chrobot 2005]). Jednak poziom złożoności reguł metod opartych na takich jednostkach powoduje, iż narzędzia te nie są w stanie wspomagać wszystkich operacji niezbędnych do prawidłowego ich użycia, toteż duża część tego procesu pozostaje nadal operacją niezautomatyzowaną, wymagającą znacznego doświadczenia w ich stosowaniu. Jednakże produkty programowe są z natury rzeczy skomplikowane, dlatego trudno oczekiwać, że metoda wymiarowania ich zakresu będzie jednocześnie skuteczna i prosta. Zatem mimo że koszty wymiarowania zakresu oprogramowania w oparciu o jednostki funkcjonalności mogą faktycznie nie być małe, chociaż i tak są niskie w stosunku do olbrzymich sum wydatkowanych na przedsięwzięcia informatyczne[9] – to powinny być potraktowane jako inwestycja w doskonalenie procesu budowy oprogramowania.

Wobec zaakceptowania kilku metod wymiarowania opartych na jednostkach funkcjonalności przez międzynarodowe organizacje standaryzacyjne ISO (International Organization for Standardization) i IEC (International Electrotechnical Commission) przestaje być aktualny wysuwany wobec takich jednostek zarzut dotyczący subiektywizmu pomiaru – jednym z podstawowych warunków ich uznania przez te organizacje była bowiem jego obiektywizacja[10]. Niemniej jednak w kilku podejściach bazujących na tych jednostkach kalkulacja zakresu uzależniona jest od wag arbitralnie ustalonych przez organizacje je rozwijające. Tymczasem nie ma dowodów na to, iż takie wagi powinny mieć charakter uniwersalny i obowiązywać w

[9] Czas poświęcony na wyznaczenie zakresu funkcjonalnego produktu przez menedżera zakresu (*scope manager*) ocenia się na mniej niż 1% czasu trwania cyklu wytwórczego (zob. [Morris 2001, s. 35]).

[10] Dlatego niektóre z metod wymiarowania rozmiaru funkcjonalnego oprogramowania nie zostały zaakceptowane przez ISO/IEC w tej części, gdzie subiektywności jak dotąd wyeliminować się nie udało.

stosunku do wszystkich kategorii produktów programowych, a także iż będą się sprawdzać przy nowych obszarach aplikacyjnych.

5. Normy ISO/IEC dotyczące wymiarowania zakresu funkcjonalnego produktu programowego

W ostatnich latach nastąpił duży postęp w standaryzacji wymiarowania produktów programowych. Wyrazem tego są m.in. międzynarodowe normy dotyczące wymiarowania zakresu funkcjonalnego oprogramowania opublikowane wspólnie przez ISO i IEC. Zbiór zasad wymiarowania rozmiaru funkcjonalnego produktu programowego zawarto w sześcioczęściowej normie ISO/IEC 14143 [1998-2007][11]. Wszystkie jej części, podobnie jak niżej wymienione standardy ISO/IEC dotyczące poszczególnych metod wymiarowania takiego rozmiaru, są zgodne z normą ISO/IEC 15939 [2007], wyznaczającą ogólne procedury dla procesu wymiarowania oprogramowania. Struktura standardu ISO/IEC 14143 jest następująca:

Część 1. Ma charakter definicyjny i przez to stanowi podstawę koncepcyjną wymiarowania rozmiaru funkcjonalnego produktu programowego. Z tego powodu stanowi bazę dla pozostałych części. Identyfikuje ona wspólne cechy metod FSM (Functional Size Measurement) i określa zestaw obligatoryjnych wymagań ogólnych, które takie metody muszą spełniać. Opracowano ją z myślą o wykorzystaniu przez osoby zaangażowane w nabywanie, rozwój, użytkowanie, wspomaganie, utrzymanie i audyt oprogramowania, jak i w rozwój metod FSM.

Część 2. Zawiera reguły weryfikacji zgodności nowo proponowanej (kandydującej) metody FSM z obligatoryjnymi zasadami zawartymi w części definicyjnej omawianego standardu. Ustanawia również schemat takiej weryfikacji.

Część 3. Obejmuje opis procesów służących obiektywnej i spójnej weryfikacji różnych aspektów metod FSM (np. wykorzystywanej procedury), tak aby umożliwić użytkownikom wybór metody najlepiej dostosowanej do ich potrzeb.

Część 4. Zawiera zbiór referencji przeznaczony do weryfikacji metod FSM z podziałem na ich odniesienie do aplikacji biznesowych, systemów czasu rzeczywistego i programów naukowych. Jest ona użyteczna przy ocenie skuteczności tego rodzaju metod dla odmiennych kategorii oprogramowania rozwijanych w różnych środowiskach.

Część 5. Obejmuje opis klas oprogramowania, tzw. domen funkcjonalnych, dla których deklaruje się możliwość wykorzystania metod FSM. Zawiera także charakterystykę procesu określania takiej domeny na podstawie zbioru wymagań funkcjonalnych sformułowanych przez użytkownika.

Część 6. Obejmuje reguły wyboru odpowiedniej metody FSM dla określonej domeny funkcjonalnej spośród tego typu metod uznanych przez ISO/IEC. Wskazu-

[11] Części 3-5 są opublikowane jako raporty techniczne (Technical Report – TR), które zgodnie z oczekiwaniami powinny z biegiem czasu przybrać formę tzw. pełnego międzynarodowego standardu. W rozumieniu ISO raport techniczny reprezentuje zwykle obszar nadal rozwijany, przez co ma charakter mniej rygorystyczny.

je także na sposoby wykorzystania koncepcji wymiarowania rozmiaru funkcjonalnego w celu wspomagania zarządzania projektami rozwoju i utrzymania oprogramowania.

Wśród metod uznanych przez ISO i IEC za zgodne z zasadami wymiarowania rozmiaru funkcjonalnego opisanymi w normie ISO/IEC 14143 znajdują się obecnie:

– metoda punktów funkcyjnych IFPUG (International Function Point Users Group), w części zasadniczej, tj. dotyczącej wymiarowania rozmiaru funkcjonalnego, zaakceptowana w normie ISO/IEC 20926 [2003],
– metoda punktów funkcyjnych w wersji Mk II (Mark II) rozwijanej przez UKSMA (United Kingdom Software Metrics Association), również w części zasadniczej uznana w normie ISO/IEC 20968 [2002],
– metoda punktów funkcyjnych NESMA (Netherlands Software Metrics Association), w części dotyczącej wymiarowania rozmiaru funkcjonalnego objęta normą ISO/IEC 24570 [2005],
– metoda pełnych punktów funkcyjnych w wersji opracowanej przez COSMIC (Common Software Measurement International Consortium), w pełni uznana w standardzie ISO/IEC 19761[2003],
– metoda wymiarowania zakresu funkcjonalnego oprogramowania w wersji rozwijanej przez FiSMA (Finnish Software Measurement Association), zawarta w normie ISO/IEC 29881[2008].

Dotychczasowe doświadczenia pokazują, że w celu zaakceptowania określonej metody FSM przez ISO/IEC musi ona przejść weryfikację trwająca od 2 do 4 lat.

6. Wnioski

Z przedstawionych rozważań wynika, że wśród miar zakresu produktu programowego największą użytecznością, pomimo sygnalizowanego braku doskonałości, charakteryzują się jednostki funkcjonalności produktu. Spełniają one większość wymagań stawianych miarom zakresu, chociaż niektóre z nich budzą kontrowersje z punktu widzenia stopnia uniwersalności; również sposób obliczania ich wartości trudno uznać za prosty. Niemniej jednak jednostkom programowym oraz złożoności konstrukcyjnej brakuje cech dużo bardziej podstawowych: możliwości wystarczająco obiektywnej i wiarygodnej estymacji zakresu produktu stosunkowo wcześnie w cyklu życia przedsięwzięcia w sposób niezależny od wykorzystywanej technologii i uwzględniający perspektywę istotną dla zleceniodawcy. Jednostki funkcjonalności, jako miara priorytetowego dla klienta atrybutu produktu, sprzyjają udziałowi zleceniodawcy w działaniach projektowych, co z kolei zwiększa prawdopodobieństwo sukcesu podczas realizacji przedsięwzięcia. Dodatkowo dają one możliwość uchwycenia przyrostu wymagań zleceniodawcy w trakcie cyklu produkcyjnego. Niebagatelnym ułatwieniem jest powszechny dostęp do ogólnych danych historycznych

wskazujących na zależność nakładów pracy, kosztów i czasu realizacji od zakresu produktu wyrażonego właśnie w jednostkach funkcjonalności[12].

Z analizy proponowanych od szeregu lat podejść do problematyki wymiarowania zakresu produktu programowego wynika, że mimo dynamicznych zmian zachodzących w inżynierii oprogramowania, w tym obszarze dąży się nie tyle do opracowania nowych sposobów wyrażania tego zakresu, co raczej podejmuje się próby rozwijania takich sposobów postępowania, dla których podstawę stanowią właśnie jednostki funkcjonalności produktu. W efekcie istnieje kilka odmian takich jednostek, a niektóre z metod służących do ich kalkulacji dojrzewają i z upływem czasu uzyskują coraz większą obiektywność i wiarygodność. Obecnie już na tyle dużą, że w ostatnich latach nie tylko samo podejście do wymiarowania produktu programowego z perspektywy jego funkcjonalności, ale także część z takich metod znalazła uznanie międzynarodowych organizacji standaryzacyjnych, które wymagają spełnienia tych warunków. Z tych przede wszystkim powodów punkty funkcyjne zostały uznane przez Gartner Group za najwłaściwszą miarę zakresu aplikacji [Gartner Research 2002].

Literatura

Abran A., Robillard P.N., *Reliability of function points productivity models for enhancement projects (a field study)*, Conference on Software Maintenance 1993-CSM-93, Montreal, IEEE Computer Society Press, Los Alamitos 1993.

Albrecht A.J., *Measuring application development productivity*, Proceedings of IBM Application Development Symposium, Monterey, CA., October 14-17, 1979.

Czarnacka-Chrobot B., *Porównanie metod pomiaru i szacowania projektów informatycznych – jednostki programowe a jednostki umowne*, [w:] *Efektywność zastosowań systemów informatycznych*, t. I, Materiały konferencyjne XIII Szkoły Górskiej PTI, red. J.K. Grabara, J.S. Nowak, Wydawnictwo Naukowo-Techniczne, Warszawa 2001.

Czarnacka-Chrobot B., *Automatyzacja wymiarowania projektów IT. Jakie narzędzie wybrać?*, „Teleinfo" 2005, nr 25.

Fenton N.E., *Zapewnienie jakości i metryki oprogramowania*, [w:] *Inżynieria oprogramowania w projekcie informatycznym*, red. J. Górski, Mikom, Warszawa 2000.

Flasiński M., *Zarządzanie projektami informatycznymi*, Wydawnictwo Naukowe PWN, Warszawa 2006.

Gartner Research, *Function Points Can help Measure Application Size*, Research Notes SPA-18-0878, November 2002.

IEEE Std 610.12-1990: IEEE Standard Glossary of Software Engineering Terminology, The Institute of Electrical and Electronics Engineers Inc., New York, NY, 1990.

International Software Benchmarking Standards Group (ISBSG), http://www.isbsg.org/Isbsg.Nsf/weben/Functional%20Sizing%20Methods (21.04.2008).

[12] Na przykład najnowsza wersja repozytorium International Software Benchmarking Standards Group (ISBSG) zawiera takie dane dla ponad 4 tys. przedsięwzięć informatycznych (zob. *The ISBSG Special*… 2005, s. 1).

ISO/IEC 14143 Information Technology – Software measurement – Functional size measurement – Part 1-6, ISO, Geneva 1998-2007.
ISO/IEC 15939:2007 Systems and software engineering – Measurement process, ISO, Geneva 2007.
ISO/IEC 19761:2003 Software engineering – COSMIC-FFP – A functional size measurement method, ISO, Geneva 2003.
ISO/IEC 20926:2003 Software engineering – IFPUG 4.1 Unadjusted functional size measurement method – Counting practices manual, ISO, Geneva 2003.
ISO/IEC 20968:2002 Software engineering – Mk II Function Point Analysis – Counting practices manual, ISO, Geneva 2002.
ISO/IEC 24570:2005 Software engineering – NESMA functional size measurement method version 2.1 – Definitions and counting guidelines for the application of Function Point Analysis, ISO, Geneva 2005.
ISO/IEC 29881:2008 Information Technology – Software and systems engineering – FiSMA 1.1 functional size measurement method, ISO, Geneva 2008.
Jones T.C., *What are function points?*, Software Productivity Research Inc., Burlington 1995.
Khelifi A., Abran A., Buglione L.*, A System of References for Software Measurements with ISO 19761 (COSMIC-FFP)*, 14th International Workshop on Software Measurement (IWSM), DASMA Metrik Kongress, Konigs Wusterhausen, Shaker-Verlag, Magdeburg 2004.
Kobyliński A., *Zagadnienia złożoności programów komputerowych*, praca doktorska, SGH, Warszawa 1991.
Kobyliński A., *Modele jakości produktów i procesów programowych*, OW SGH, Warszawa 2005.
Morris P., *Functional Size Metrics*, Total Metrics, October 2001.
Parthasarathy M.A., *Practical Software Estimation: Function Point Methods for Insourced and Outsourced Projects*, Addison Wesley Professional, 2007.
Software Metrics: A Rigorous Approach, ed. N.E. Fenton, Chapman & Hall, London 1991.
Software Productivity Research: http://www.spr.com/products/feature.shtm (20.12.2008).
Software Productivity Research: http://www.spr.com/products/choosing.shtm (20.12.2008).
Sommerville I., *Inżynieria oprogramowania*, WNT, Warszawa 2003.
The ISBSG Report: Software Project Estimates – How accurate are they?, ISBSG, Hawthorn VIC, Australia, January 2005.
The ISBSG Special Analysis Report: Early Lifecycle Software Estimation, ISBSG, Hawthorn VIC, Australia, September 2005.
Zuse H., *Software Complexity. Measures and Methods*, Walter de Gruyter, Berlin – New York 1991.

SOFTWARE SIZE MEASURES ASSESSMENT

Summary: The main aim of software project is to deliver a product which meets the client requirements on time and within a budget. There are many factors which make it difficult, but one of the most important is the lack of an unequivocal software size measure. Software size measurement is a base for project effort, time, costs and product quality estimation. That is why the lack of that measure can be recognized as one of the main software engineering problems. The aim of this article is to present and assess the different kinds of software size measures.