Planowanie z użyciem metody PROBE

Jerzy R. Nawrocki <Jerzy.Nawrocki@put.poznan.pl>*
Bartosz Walter <Bartosz.Walter@cs.put.poznan.pl>
Adam Wojciechowski <Adam.Wojciechowski@put.poznan.pl>

Instytut Informatyki Politechniki Poznańskiej
60-965 Poznań, ul. Piotrowo 3A

Streszczenie

Metoda PROBE, będąca elementem *Personal Software Process*^{SM}, służy do szacowania rozmiaru i czasu budowy oprogramowania w przypadku realizacji systemu przez indywidualnego programistę. Metoda ta nie zawsze znajduje pełne zastosowanie przy planowaniu projektów realizowanych przez zespoły. Dlatego zaproponowano sposób wspomagania przydziału osób do zadań oparty o analizę danych historycznych zaczerpniętych z modelu PSP^{SM}, opisujących kompetencje poszczególnych członków zespołu.

1. Wprowadzenie

Planowanie jest tą częścią fazy strategicznej, która decyduje o powodzeniu projektu informatycznego zanim on się jeszcze rozpocznie. Rezultat planowania wpływa na cały przebieg projektu, jego koszt, ryzyko z nim związane, a przede wszystkim na sposób realizacji.

W każdym procesie programowym, zarówno klasycznym, opartym o obieg dokumentów , jak CMM (*Capability Maturity Model*) czy lekkim, który rezygnuje z większości artefaktów, jakim jest XP (*eXtreme Programming*), planowanie pozostaje istotnym czynnikiem powodzenia projektu.

* Praca wykonana w ramach projektu badawczego 43-364/01-BW

Model CMM[SM], zaproponowany przez Software Engineering Institute, przewiduje 5 poziomów dojrzałości procesu programowego. Na drugim poziomie (zarządzanym – *managed*; dawniej ten poziom nosił nazwę powtarzalnego – *repeatable*) jednym z kluczowych obszarów procesu (*Key Process Area*) jest właśnie planowanie oparte o dokumenty i ich wzorce.

Idea programowania ekstremalnego stanowi odpowiedź na problemy związane z wdrożeniem dobrze zdefiniowanego, klasycznego procesu budowy oprogramowania. Producenci oprogramowania nie nadążali za gwałtownie zmieniającymi się potrzebami rynku, a wprowadzone dokumenty wprowadzały istotny narzut na pracę włożoną w samą produkcję. Mimo tego, że XP odrzuca większość klasycznych technik związanych z klasycznym procesem programowym, planowanie pozostaje czynnością krytyczną, której nie wolno zaniedbać.

Modele procesu programowego opartego o indywidualną (*Personal Software Process*[SM]) i zespołową (*Team Software Process*[SM]) pracę zaproponował W. Humphrey [4, 5]. Oba opierają się na uproszczonych formularzach, które stanowią jednocześnie źródło informacji o procesie i prostą metodę jego oceny. Zarówno PSP[SM] jak i TSP[SM] definiują fazę planowania, która obejmuje następujące zagadnienia:

- czas konieczny do realizacji projektu,
- harmonogram prac,
- strukturę projektu (zespołu realizacyjnego),
- przydział zadań do osób.

TSP[SM], przeznaczony dla małych zespołów, jest rozwinięciem PSP[SM]; zakłada on, że każdy z członków zespołu pracuje stosuje PSP[SM], czyli planuje pracę we własnym zakresie. Oba modele wymagają kolekcjonowania danych dotyczących oszacowań i faktycznych wielkości osiągniętych w trakcie dotychczasowej pracy.

Humphrey do analizy statystycznej zebranych danych zaproponował metodę PROBE, która szacuje wysiłek indywidualnego programisty wymagany przy tworzeniu oprogramowania o przewidywanym rozmiarze. Metoda ta mocno zależy od charakterystyki pracy konkretnej osoby, natomiast wynik bezpośredniego przełożenia metody na planowanie pracy zespołu zależy od jakości i ilości danych zgromadzonych przez jego członków.

Drugim elementem istotnym przy planowaniu projektu jest przyporządkowanie zadań do osób. Celem takiego przyporządkowania jest minimalizacja kosztu (czasu) realizacji przy możliwym zbalansowaniu obciążenia osób zaangażowanych w projekt. Niestety, PROBE pomimo nacisku na kolekcjonowanie danych, nie proponuje sposobu przydziału, pozostawiając tę kwestię liderowi zespołu.

Celem tego artykułu jest zaproponowanie metody wykorzystania danych historycznych w metodzie PROBE, tak aby uwzględnić dane zebrane przez różnych członków zespołu i na ich podstawie planować pracę grupy. Ponadto zostanie zaproponowana idea opartego o dane historyczne przydziału zadań do osób w projektach zespołowych.

Drugi rozdział jest poświęcony wykorzystanym metodom szacowania: metodzie PROBE i jej zastosowaniu w PSP^{SM} oraz Wideband-Delphi. Kolejny rozdział przedstawia proponowaną metodę przydziału zadań do osób w metodzie PROBE przy projektach zespołowych z uwzględnieniem danych historycznych. Rozdział czwarty stanowi podsumowanie przedstawionych rozwiązań, formułuje wnioski oraz cele do dalszej pracy.

2. Proste metody szacowania

2.1. Metoda Probe

Metoda PROBE została zaproponowana przez W. Humphreya jako część indywidualnego procesu budowy oprogramowania PSP^{SM} [4].

PROBE (*PROxy Based Estimation*) służy do szacowania wysiłku wymaganego przy tworzeniu wybranych komponentów (*proxies*) programowych na podstawie zebranych danych historycznych. Idea oparcia estymacji o znormalizowany komponent (którym w przypadku PROBE jest obiekt) umożliwia stworzenie intuicyjnego elementu systemu, którego rozmiar można łatwo oszacować metodami niestatystycznymi.

Metoda PROBE składa się z dwóch zasadniczych kroków:

- znalezienia współczynników regresji liniowej B_0 i B_1 pomiędzy szacowanym i faktycznym rozmiarem na zbiorze programów w danych historycznych; dzięki temu tej możliwe znalezienie zależności między tymi dwiema wielkościami w dotychczas wytworzonym oprogramowaniu
- oszacowania wysiłku (czasu) potrzebnego do stworzenia oprogramowania o szacowanym rozmiarze

Oba kroki zakładają wykorzystanie regresji liniowej; jednak jest stosowanie jest obwarowane warunkami: zbiór danych historycznych musi być odpowiednio liczny (zawiera co najmniej 3 punkty) oraz dane te muszą spełniać zależność

$$r^2 = 0.5$$

czyli muszą być ze sobą wystarczająco skorelowane.

Niestety, nie zawsze te warunki są spełnione, szczególnie w przypadku szacowania wysiłku na podstawie rozmiaru oprogramowania. Wówczas możliwe są trzy warianty:

- A (najlepszy): zbiór danych historycznych zawiera co najmniej 3 pary szacowany rozmiar – faktyczny czas, dla których spełnione jest podane równanie
- B: wymagana relacja zachodzi między faktycznym rozmiarem a faktycznym czasem realizacji
- C (najgorszy): nie jest spełniony żaden z powyższych warunków; w tym nie stosuje się regresji liniowej, tylko oblicza szacowany wysiłek na podstawie średniego rozmiaru oprogramowania tworzonego w jednostce czasu.

Oszacowanie uzyskane w każdym z przypadków jest estymatą punktową, która nie zawiera informacji o jakości tego oszacowania. Aby to osiągnąć, oblicza się za pomocą dwustronnego rozkładu *t* Studenta przedział ufności, w którym z zadanym prawdopodobieństwem znajduje się estymowana wielkość.

Metoda PROBE dzięki prostemu aparatowi matematycznemu i stosunkowo niewielkiej ilości koniecznych danych historycznych stanowi rozwiązanie dla pojedynczych programistów. W przypadku stosowania w większych zespołach pojawia się problem szacowania wysiłku, jaki muszą poświęcić poszczególni członkowie zespołu oraz jak podzielić pracę między nich. Sugerowane rozwiązanie zostanie zaprezentowane w rozdziale 3.

2.2. Metoda Wideband-Delphi

Metoda Wideband-Delphi (delficka), zaproponowana przez B. Boehma, jest zupełnie innym podejściem do szacowania rozmiaru oprogramowania niż PROBE. Jest ona szczególnie przydatna w sytuacjach, gdy posiadane doświadczenie nie jest usystematyzowane w postaci zbioru danych historycznych.

Metoda delficka opiera się o szacowanie rozmiaru przez ekspertów, którzy na podstawie specyfikacji oprogramowania dekomponują je na części, których rozmiar każdy z nich szacuje niezależnie od innych osób . Istotą szacowania są spotkania, na których osoby szacujące dyskutują o systemie (ale nie o estymatach!), zgłaszają uwagi, a następnie przekazują swoje oszacowania prowadzącemu, który je zbiera i zestawia, jednak bez identyfikowania autorów poszczególnych wyników. Następuje dyskusja, a potem po raz kolejny każdy

ekspert dokonuje oceny rozmiaru. Ta procedura jest powtarzana do czasu osiągnięcia założonego przedziału rozbieżności.

Elementem, który decyduje o skuteczności metody delfickiej, jest anonimowość oszacowań: dzięki temu nie istnieje psychologiczny przymus obrony własnego zdania i eksperci łatwiej dokonują rewizji swoich estymat. Współdzielenie wielu ocen daje efekt synergii, która często przynosi dobre rezultaty.

Wadą metody delfickiej jest duża zależność od indywidualnych cech osób szacujących, a także prowadzącego dyskusję. Umiejętność kierowania uwagi ekspertów na system i jego cechy, a nie konkretne oceny, decyduje o wzajemnym zrównoważeniu tendencji w indywidualnych wynikach i – w efekcie – skuteczności szacowania.

3. Metoda przydziału zadań do osób

Obie przedstawione metody stosują dekompozycję wyłącznie do celu uzyskania lepszych oszacowań rozmiaru i wysiłku. Jednak planowanie wymaga również m.in. wykonania harmonogramu prac oraz przydzielenia zadań do osób.

TSP^{SM}, proces zespołowej budowy oprogramowania, zakłada dokonywania podziału zadań w trakcie negocjacji i uzgodnień wewnątrz zespołu. Podejście takie, mimo swej prostoty, może przynieść w efekcie nierównomierne obciążenie członków zespołu pracą oraz przydział zadań do osób daleki od optymalnego.

W praktyce członkowie zespołu mają często przydzielone na stałe role, które pełnią w projektach. W przypadku osób nie zajmujących się produkcją oprogramowania takie przypisanie jest uzasadnione: zarządzanie konfiguracją czy nadzór jakości jest zwykle pomocniczą rolą w projekcie i nie narusza istotnie podziału pracy. Programiści jednak często pracują zamiennie z wykorzystaniem różnych technologii, co powoduje, że nie mogą specjalizować się w jednej technologii czy wykonywać wyłącznie zadania jednego typu.

Powstaje potrzeba liczbowego, opartego o statystykę sposobu oszacowania kosztu przydziału (czasu realizacji systemu) poszczególnych zadań do osób oraz metody znajdowania przydziału dającego w sumie najlepsze (najtańsze) rozwiązanie.

Wprowadzenie TSP^{SM} wymaga, aby każdy z członków zespołu posiadał doświadczenie w pracy z PSP^{SM}. Oznacza to, że każdy z nich posiada zebrane dane historyczne dotyczące rozmiaru, czasu realizacji i oszacowań z wykonanych projektów.

3.1. Ocena kompetencji

Pierwszym elementem proponowanej metody jest ocena kompetencji programistów na podstawie danych historycznych. Programowanie obecnie nie oznacza wyłącznie umiejętności tworzenia kodu w danym języku, ale przede wszystkim znajomość techologii, algorytmiki czy tematyki problemu. W związku z tym ocenianie średniej produktywności programisty na podstawie wszystkich zrealizowanych projektów może być niewystarczające; konieczne staje się wyróżnienie cech charakterystycznych, profili tych części systemów (komponentów), i szacowanie wysiłku potrzebnego do realizacji oprogramowania właśnie dla konkretnego profilu, a nie dla uśrednionego przypadku Wymaga to wprawdzie dużej bazy danych historycznych pochodzącej od wielu osób, obejmujących różne profile, ale można ją uzupełnić poprzez krótkie sesje testowe: przygotowanie zadań próbnych i pomiar metryk dla wszystkich członków zespołu.

Analiza posegregowanych danych powinna dać dużo większą korelację między oszacowaniami i faktycznymi rezultatami w przypadku, gdy jedna osoba realizuje wiele systemów o bardzo podobnym profilu. Daje to dwie korzyści:

- przedział ufności jest dużo mniejszy, czyli oszacowania są dokładniejsze,
- możliwa jest liczbowa, oparta o dane historyczne ocena wydajności poszczególnych członków zespołu przy realizacji zadań o określonym profilu

Zgromadzenie danych o kompetencjach pracowników, popartych wartościami liczbowymi, pozwala precyzyjniej szacować czas potrzebny do budowy całego systemu (a nie tylko części wykonanej w oparciu o jeden profil problemu), a także efektywnie przyporządkowywać zadania do osób najbardziej kompetentnych w realizacji poszczególnych zadań.

3.2. Wyznaczanie kosztu

Jeżeli zebrane dane historyczne pozwalają na ocenę produktywności programistów w odniesieniu do profili oprogramowania, można przystąpić do przydziału poszczególnych zadań do osób. Ponieważ każde zadanie, na można podzielić system, jest scharakteryzowane przez profil (czyli estymatę czasu wymaganego do realizacji oprogramowania o danym rozmiarze) oraz szacowany rozmiar, znany jest także szacowany czas, jaki musi poświęcić każdy programista na tworzenie każdego komponentu.

W ten sposób powstaje macierz o wymiarach $i \times j$, w którym i oznacza liczbę dostępnych programistów, natomiast j – liczbę komponentów; elementy macierzy c_{ij} wyznaczają szacowany koszt realizacji komponentu j przez osobę i.

Zakładamy przy tym, że jedna osoba może realizować jedno niepodzielne zadanie.

W istocie przedstawiony problem przydziału jest zadaniem programowania zerojedynkowego, w którym zmienne x_{ij} przyjmują wartości 1, gdy osoba j wykonuje zadanie i lub 0 w przeciwnym przypadku, przy funkcji celu minimalizującej koszt realizacji wszystkich zadań.

Przy nałożonych w ten sposób ograniczeniach zadanie może zostać rozwiązane przy użyciu tzw. metody węgierskiej [2], która znajduje optymalne rozwiązanie (tj. rozwiązanie o minimalnym koszcie) przydziału właśnie dla zadań niepodzielnych, przy założeniu że i = j. Zaletą metody węgierskiej jest wielomianowa złożoność obliczeniowa.

Zadanie to można też rozwiązać metodami ogólniejszymi, np. simpleksu, co osłabia nałożone założenia, ale zwiększa złożoność obliczeniową (w ogólnym przypadku problem programowania liniowego jest NP-zupełny).

Wadą powyższych metod jest konieczność posiadania dużej i kompletnej bazy danych historycznych. Dane te muszą spełniać następujące warunki

- być odpowiednio liczne (minimum 3 punkty dla każdego profilu dla każdego użytkownika),
- zachowywać wysoką korelację między rozmiarem a czasem realizacji dla tych zbiorów,
- opisywać wszystkie zdefiniowane profile programów dla wielu użytkowników.

Te trzy warunki są zwykle trudne do spełnienia. Najtrudniejsze do realizacji wydaje się warunek trzeci, czyli zgromadzenie bazy danych dla wielu profili dla wielu użytkowników. Stwarza to problemy szczególnie przy szacowaniu systemów o nowym profilu, dla którego nie można wyznaczyć szacowanej pracochłonności. W takiej sytuacji są możliwe dwa rozwiązania:

- określić podobieństwo nowego zadania do dotychczas realizowanych; wówczas profil można zdefiniować jako zbiór atrybutów przyjmujących wartości z pewnego zakresu, a podobieństwo dwóch profili – jako korelację między nimi. To podejście wymaga jednak prac w kierunku wyznaczenia sposobu określania profili.
- określić czasochłonność metodami niestatystycznymi, np.Wideband-Delphi

Z tych dwóch metod prostsze wydaje się zastosowanie tej drugiej. Wymaga to częściowej rezygnacji z szacowania liczbowego, jednak wyniki obu z nich mogą w znacznym stopniu się pokrywać, szczególnie gdy szacowania dokonują osoby doświadczone.

W efekcie cała zaproponowana metoda szacowania składa się z 5 kroków:

1. przydziału zadań nie związanych z produkcją oprogramowania, dla których istnieje ścisły podział ról w zespole
2. podziału całego systemu na komponenty programowe o zdefiniowanych profilach (zgodnie z przyjętą przez zespół definicją profilu) i oceny rozmiaru każdego komponentu
3. wyznaczenia za pomocą metody PROBE macierzy szacowanej pracochłonności każdego komponentu dla każdego programisty; jeżeli nie są spełnione warunki stosowanie regresji liniowej, można zastosować np. metodę delficką.
4. wyznaczenia za pomocą metody węgierskiej optymalnego przydziału zadań do osób
5. ewaluacji otrzymanego rozwiązania, traktując je jako punkt wyjścia do ostatecznego przydziału; w szczególności można próbować osłabić założenia, jakie nakłada metoda węgierska, zrównoważyć obciążenie pracowników oraz uwzględnić ograniczenia kolejnościowe między realizacją poszczególnych części systemu.

Opisana procedura wymaga wielu danych liczbowych zgromadzonych w poprzednich systemu, ale pozwala na oparte o statystykę znalezienie optymalnego przydziału. W przypadku zespołów realizujących oprogramowanie przez dłuższy czas może okazać się pomocą przy podziale systemu na komponenty oraz ocenie kompetencji pracowników w konkretnych dziedzinach.

4. Podsumowanie

Przedstawiona metoda może znaleźć zastosowanie w zespołach stosujących procesy PSP i TSP. Opiera się ona o proste i łatwe w uzyskaniu dane oraz wspomaga planowanie pracy zespołu. Pomaga w opracowaniu optymalnego przydziału zadań opartego o statystyczną analizę danych statystycznych oraz liczbowy opis kompetencji członków zespołu.

Wadą jest przede wszystkich ilość niezbędnych do skutecznego szacowania danych. Ponadto zastosowanie metody węgierskiej wymusza nałożenia silnych założeń, które utrudniają modelowanie rzeczywistych sytuacji decyzyjnych. Dlatego metoda ta powinna być stosowana przede wszystkim jako wsparcie decyzji do ostatecznego przydziału zadań, natomiast finalną decyzję musi podjąć lider zespołu.

Doskonalenie metody wymaga podjęcia prac w kilku kierunkach. Należy zdefiniować pojęcie profilu komponentu i zidentyfikować jego atrybuty. Ważne jest także modelowanie podobieństwa profili, co umożliwi szacowanie wysiłku nowych systemów na podstawie innych zrealizowanych dotychczas. Drugim elementem prac są możliwości skalowania metody PROBE, czyli szacowania

pracochłonności oprogramowania o innym rozmiarze niż systemy, dla których istnieją dane historyczne. Ważna jest także odpowiedź na pytanie, czy jest możliwe szacowanie wysiłku na podstawie danych zebranych przez innych członków zespołu, co zmniejszy zależność od kompletności bazy danych.

Liczbowe określenie podstawowych parametrów projektu na etapie planowania jest przy ocenie jego jakości. Zaproponowana metoda stanowi proste rozwinięcie PROBE dla systemów realizowanych przez niewielkie zespoły zbierające i wykorzystujące dane historyczne.

Bibliografia

[1]. K. Beck, M. Fowler "Planning Extreme Programming", Addison-Wesley, 2000

[2]. J. Błażewicz, W. Cellary, R. Słowiński, J. Węglarz "Badania operacyjne dla informatyków", PWN, Warszawa XXX

[3]. Z. Hellwig "Elementy rachunku prawdopodobieństwa i statystyki matematycznej", PWN, Warszawa 1995

[4]. W. S. Humphrey "Discipline for Software Engineering", Addison-Wesley, 1995

[5]. W. S. Humphrey "Introduction to the Team Software Process^{SM}", Addison-Wesley, 1999