Zeszyty Naukowe Uniwersytet Ekonomiczny w Krakowie

3 (939)

ISSN 1898-6447
Zesz. Nauk. UEK, 2015; 3 (939): 35–50
DOI: 10.15678/ZNUEK.2015.0939.0303

Krzysztof Woźniak
Katedra Procesu Zarządzania
Uniwersytet Ekonomiczny w Krakowie

Pomiar sprawności działania w zwinnych metodykach projektowania oprogramowania*

Streszczenie

W artykule przedstawiono teoretyczno-metodologiczne rozważania dotyczące sprawności działania zespołów programistów realizujących projekty z wykorzystaniem zwinnych (*agile*) metodyk projektowania złożonych systemów informatycznych. Metodyki te we współczesnej literaturze informatycznej opisywane są jako uniwersalne rozwiązanie wszelkich problemów organizacyjnych, przed którymi stoją kierownicy dużych zespołów pracujących nad złożonymi projektami informatycznymi.

Artykuł stanowi próbę odpowiedzi na pytanie, w jakim zakresie zwinne metodyki tworzenia oprogramowania spełniają przesłanki klasycznej metodologicznej sprawności działania. Stawiana jest teza, że nawet w przypadku ich stosowania, należy podejmować próby operacjonalizacji podstawowych parametrów opisujących sprawność i efektywność działania zespołów programistów. Umożliwi to kierownikom podejmowanie skutecznych decyzji, opierających się na faktach, a nie na przypuszczeniach i przeczuciach.

Słowa kluczowe: sprawność, pomiar, projekty informatyczne, podejmowanie decyzji.

* Niniejszy artykuł przygotowano w ramach projektu badawczego. Projekt został sfinansowany ze środków Narodowego Centrum Nauki przyznanych na podstawie decyzji nr DEC-2011/03/B/HS4/03585.

1. Wprowadzenie

Rozważania w niniejszym artykule dotyczą stosowania zwinnych (*agile*) metod i technik przy realizacji złożonych projektów informatycznych. Mają one charakter teoretyczno-metodologiczny i stanowią próbę odniesienia klasycznej dla nauki organizacji i zarządzania problematyki sprawności działania do zasadniczo odmiennego podejścia do procesu pracy stosowanego współcześnie przez zespoły programistów. Metodyki zwinne przedstawiane są w literaturze dotyczącej zarządzania projektami informatycznymi jako uniwersalne rozwiązanie wszelkich problemów, przed którymi stoją twórcy oprogramowania. Należy w związku z tym odpowiedzieć na pytanie, w jakim zakresie zwinne metodyki tworzenia oprogramowania spełniają przesłanki klasycznej metodologicznej sprawności działania oraz w jaki sposób można dokonać pomiaru sprawności programistów wykorzystujących je w codziennej pracy.

2. Charakterystyka zwinnych metodyk tworzenia oprogramowania

Metodyki zwinne, zwane także prowizorycznymi lub elastycznymi, stanowią zestaw praktyk i technik szczegółowych charakterystycznych dla dziedziny realizacji złożonych projektów informatycznych. Ich powstanie było poprzedzone wystąpieniem wielu problemów w projektach realizowanych z wykorzystaniem klasycznych metodyk zarządzania. Najczęściej pojawiały się one w trakcie realizacji przedsięwzięć o szerokim zakresie prac (dotyczących realizacji znacznej liczby tzw. funkcjonalności oprogramowania) i wymagających twórczego rozwiązywania problemów trudnych do przewidzenia na etapie planowania projektu. Dynamiczny rozwój zastosowań zwinnych metodyk zarządzania projektami rozpoczął się w 2001 r., od powstania Manifesto for Agile Software Development stanowiącego wizję zmiany w podejściu do tworzenia oprogramowania. Dokument ten był reakcją środowiska informatyków na wady klasycznego podejścia do zarządzania projektami, nastawionego na formalizację, jasno sprecyzowane cele i budżety, sztywno przypisane zadania i zakresy odpowiedzialności. Treść manifestu zwinnego tworzenia oprogramowania formułuje najważniejsze zalecenia metodologiczne dla projektów informatycznych [*Manifest…* 2013]: „wytwarzając oprogramowanie i pomagając innym w tym zakresie, odkrywamy lepsze sposoby wykonywania tej pracy. W wyniku tych doświadczeń przedkładamy:

– ludzi i interakcje ponad procesy i narzędzia,
– działające oprogramowanie ponad obszerną dokumentację,
– współpracę z klientem ponad formalne ustalenia,

– reagowanie na zmiany ponad podążanie za planem.

Doceniamy to, co wymieniono wyżej po prawej stronie, jednak bardziej cenimy to, co znajduje się po lewej".

Z treści manifestu wynika, że chociaż jego twórcy doceniają tradycyjne strukturyzowane podejście do projektowania systemów informatycznych (procesy, dokumentację, formalizację, planowanie), to większy nacisk kładą na ludzi, praktyczny rezultat, współpracę oraz elastyczne reagowanie na zmiany wymagań klienta, zakresu i specyfikacji projektu. Zasadnicze cele i zasady, na jakich opierają się zwinne metodyki tworzenia oprogramowania to:

– elastyczność i adaptacyjność projektowania względem dynamicznie zmieniających się potrzeb i oczekiwań klienta (tj. zwinność),

– tworzenie wartościowych i innowacyjnych rozwiązań zarówno dla firmy, jak i klientów na każdym etapie projektowania,

– zmniejszanie kosztów dzięki skróceniu czasu wytwarzania,

– koncentracja na członkach zespołu projektowego powodująca wzrost motywacji wśród pracowników i zmniejszenie stresu,

– ścisła współpraca z klientem, częste bezpośrednie kontakty,

– proste procedury i samoorganizujące się zespoły,

– satysfakcja klienta osiągana dzięki szybkiemu i regularnemu dostarczaniu wartościowego produktu,

– minimalizacja ryzyka niepowodzenia projektu.

Do najbardziej znanych zwinnych metodyk tworzenia oprogramowania można zaliczyć opisane w dalszej części tekstu: metodykę *extreme programming*, metodykę SCRUM oraz Feature Driven Development.

Extreme programming to jedna z częściej stosowanych zwinnych metodyk projektowania i realizacji systemów informatycznych. Najlepiej sprawdza się w sytuacjach, gdy kierownictwo przedsiębiorstwa i klienci nie mają do końca sprecyzowanych oczekiwań wobec finalnego produktu informatycznego i chcą uczestniczyć w procesie tworzenia, zmieniając swoje wymagania w trakcie powstawania systemu. Projekt realizowany z wykorzystaniem tej metodyki nie posiada precyzyjnego planu i sztywno zdefiniowanego celu. Składa się na niego nieprzerwany ciąg zmian w specyfikacji, testowania różnych prototypów, dostosowywania do zmieniających się wymagań klienta. Z tych powodów często określa się tę metodykę mianem twórczego bałaganu.

W metodyce SCRUM prace zespołu projektowego zorganizowane są wokół przygotowanego przez zleceniodawcę zestawu funkcjonalności, jakie musi spełniać gotowy produkt (system informatyczny). Praca podzielona jest na wiele ok. 30-dniowych przedziałów – segmentów – czasowych (*sprints*), których wynikiem jest fragment systemu realizujący jakąś użyteczną dla klienta funkcję.

W sytuacji, gdy zrealizowane zostaną wszystkie ustalone funkcjonalności (moduły systemu), zleceniodawca ma możliwość rozszerzenia prac o dodatkowe moduły.

W pierwszej kolejności realizowane są funkcjonalności o najwyższym priorytecie, mające największy wpływ na wartość zarówno dla klienta (użytkownika systemu), jak i wynik finansowy. Czas realizacji poszczególnych zadań szacowany jest na bieżąco i następnie grupowane są one w segmenty będące podprojektami całego przedsięwzięcia [Schwaber 2002, s. 99–112].

Zespół projektowy jest odpowiedzialny za samoorganizację i wykonanie poszczególnych zadań w określonym czasie. Kierownik zespołu (tzw. mistrz scruma) organizuje codzienne spotkania, jednak jego rola polega głównie na motywowaniu i inspirowaniu zespołu do efektywnej pracy oraz podsumowaniu rezultatów osiągniętych poprzedniego dnia. Każdy segment projektu kończy się spotkaniem podsumowującym (*Sprint Review Meeting*), na którym prezentowany jest finalny rezultat pracy zespołu nad danym segmentem systemu informatycznego.

Metodyka Feature Driven Development (FDD) zakłada maksymalne uproszczenie realizacji złożonych projektów informatycznych poprzez wydzielenie z ogólnej specyfikacji tworzonego systemu informatycznego cech cząstkowych (funkcjonalności, *features*) i skupienie wysiłków zespołu programistów na realizacji poszczególnych części systemu. Narastający proces realizacji złożonego systemu sprzyja osiągnięciu dobrych rezultatów, gdyż każda zrealizowana funkcjonalność systemu (cel cząstkowy) stanowi dodatkowy czynnik motywujący dla zespołu programistów. Podstawowe założenia metodyki FDD to [Highsmith 2002, s. 151]:

– realizowanie złożonych systemów informatycznych wymaga skalowalnego i elastycznego procesu opisanego w czytelnych procedurach,

– dobrze zdefiniowane procesy realizowane są w sposób nawykowy, dzięki czemu członkowie zespołu projektowego mogą skupić się na kluczowych działaniach,

– najlepsze rezultaty osiąga się dzięki krótkim cyklom działań, nastawionym na wcześniej zdefiniowane cechy i funkcjonalności systemu,

– z punktu widzenia kierowników metodologia jest uosobieniem prostoty, gdyż składa się z pięciu procesów: stworzenia modelu ogólnego (10% czasu projektu), budowy listy cech/funkcjonalności (4%), planowania funkcjonalności (2%), projektowania oraz realizacji funkcjonalności (77%).

Metodyki zwinne dzięki zwiększeniu elastyczności i ograniczeniu biurokracji umożliwiają szybkie osiąganie głównych celów projektów informatycznych (uzyskanie oprogramowania realizującego użyteczną funkcję). Trudno jest jednak je zastosować w realizacji projektów innego typu (np. budowlanych). Głównym obszarem ich stosowania są projekty wymagające znacznego twórczego zaangażowania wysoko wykwalifikowanych pracowników, wysokiego poziomu zespołowej interakcji oraz projekty, dla których nie określono precyzyjnie celów oraz

wymagań w odniesieniu do ostatecznego rezultatu i dla których wymagania te zmieniają się w trakcie realizacji prac projektowych i wdrożeniowych.

3. Model oceny sprawności działania w realizacji złożonych systemów informatycznych

Sprawność działania stanowi zagadnienie rozpatrywane w wielu różnych aspektach. Jest ono często definiowane jako zdolność osiągania określonych przez człowieka celów (zamierzeń) poprzez świadome lub częściowo świadome zachowanie się. Sprawca działania, dzięki swojemu postępowaniu, nie tylko przyczynia się do osiągnięcia pożądanych skutków i jest tego świadom, ale wcześniej podjął decyzję, że będzie daną czynność wykonywał. Działający ukierunkowuje się na cel i w związku z tym jego zachowanie ma charakter dowolny, gdyż mógł podjąć inną decyzję (np. o niewykonaniu działania lub wykonaniu innego działania). Uwzględniając powyższe założenia, analityczny model działania osób realizujących projekty informatyczne powinien obejmować:

1) sprawcę działania – projektanta, programistę, analityka systemowego, autora dokumentacji, testera itp. W sensie prakseologicznym zawsze sprawcą działania jest człowiek, maszyny bowiem nie działają, ale funkcjonują. Osoby realizujące projekty informatyczne zorganizowane są najczęściej w zespoły, których praca jest splotem działań wszystkich zaangażowanych pracowników. Sprawność prac zespołowych jest warunkowana sprawnością działań poszczególnych ich członków oraz sposobem zorganizowania pracy zespołowej;

2) cel działania – rozumiany jako wewnętrzny lub częściej zewnętrzny impuls skłaniający sprawcę do działania. Celem działania jest stan rzeczy, który będąc pod jakimś względem cenny (pożądany) dla działającego wyznacza kierunek i strukturę działania. Działanie zmierza do uzyskania lub utrzymania zamierzonego przez sprawcę stanu. Precyzyjne określenie celu działania jest konieczne, ale nie zawsze w pełni możliwe. Podstawowe kategorie celów działań wynikają z podstawowych ich rodzajów [Kieżun 1977]: dotyczą czasu, tj. są zależne od okresu, w jakim przewiduje się wykonanie działania, odnoszą się do doniosłości (cenności) rezultatu działania dla organizacji;

3) wynik działania – stanowi skutek, rezultat zamierzony lub niezamierzony (strata, niepowodzenie), wyrób lub wytwór. W realizacji złożonych systemów informatycznych wynikiem działania są wszelkie elementy składające się na ten system, w szczególności oprogramowanie, ale także struktury baz danych, opisy procesów przetwarzania, projekty interfejsu użytkownika, dokumentacja. Znajomość zamierzonego wyniku działania ma duże znaczenie praktyczne przy planowaniu prac, jak i w trakcie ich realizacji. Jeśli wartość finansowa wyniku działania przewyższa

jego koszty, to można stwierdzić, że działanie jest korzystne. Jeśli wartość wyniku jest mniejsza od kosztów, to działanie powoduje stratę netto. Należy zwrócić uwagę, że dla projektów informatycznych wycena wartości wyniku działania zarówno przed, jak i po jego realizacji może być problematyczna;

4) środki i metody – czyli sposoby wykorzystane do osiągnięcia wyniku działania. Środki działania w znaczeniu prakseologicznym często interpretowane są jako cele pośrednie. Rzeczy, którymi sprawcy posługują się w swoim działaniu, nazywa się zasobami materialnymi. Zasoby niezbędne do realizacji działania można udostępnić sprawcom w różnym zakresie i w różnej kolejności. Dobór sprawców, zasobów, środków działania (celów pośrednich) oraz kolejności działań to inaczej sposób lub metoda działania. Istota rozważań prakseologicznych polega na określeniu takiego sposobu działania, który w sposób optymalny przyczyni się do osiągnięcia zakładanych rezultatów;

5) warunki – czyli czasowe i przestrzenne okoliczności działania. Istotny wpływ mają tutaj także zagadnienia związane z kontekstem działania oraz wpływem czynników zewnętrznych względem sprawcy. W przypadku projektów informatycznych do warunków działania można zaliczyć lokalizację, możliwość pracy zdalnej, dostęp do szybkich łącz internetowych, możliwość elastycznego dopasowania czasu pracy do specyficznych potrzeb pracownika;

6) aparaturę – tj. wszelkie narzędzia, maszyny oraz urządzenia techniczne wykorzystane do osiągnięcia celu działania. W przypadku projektów informatycznych aparatura obejmuje komputery oraz środki łączności wraz z całą infrastrukturą pozwalającą na zapewnienie bezpieczeństwa tworzonego kodu (zabezpieczenie przed usunięciem, nieautoryzowanym dostępem, modyfikacją);

7) materiał – zasób, tworzywo, surowiec przetwarzany przez sprawcę z wykorzystaniem aparatury i za pomocą środków i metod, celem osiągnięcia wyniku działania. W przypadku projektów informatycznych mamy do czynienia ze specyficznym materiałem, którym jest informacja zapisana nie tylko w postaci oprogramowania, ale także dokumentacji systemowej, specyfikacji wymagań użytkowych, raportach z testów oprogramowania. Specyfika projektów informatycznych wynika z tego, że najczęściej nie dotyczą przetwarzania zasobów materialnych, ich rezultatem również jest produkt niematerialny (kwestia kosztu nośnika, na którym zapisane jest wynikowe oprogramowanie, jest pomijana z uwagi na stosunkowo niewielką wartość);

8) uzasadnienie działania – racjonalne uzasadnienie przyczyn podjęcia działania oraz intencje sprawcy. Ten element prakseologicznego modelu działania programistów nie zawiera specyficznych dla tej dziedziny cech. Większość sprawców, działa z pobudek ekonomicznych, chociaż można zwrócić uwagę na dużą popularność tzw. modelu *open source*, w którym programiści pracują, nie pobierając wynagrodzenia.

Konstrukcja analitycznego modelu oceny sprawności programistów wymaga określenia podstawowych cech wartościujących działanie, w szczególności w zakresie specyficznych dla tej dziedziny działalności. Kryteriami ogólnymi wykorzystywanymi do wartościowania sprawności działania są najczęściej skuteczność, korzystność, ekonomiczność, wydajność, racjonalność. Działanie, które w niniejszym modelu analitycznym oceniane jest jako sprawne, powinno wykazywać pewne minimalne zakładane poziomy przyjętych wyżej cech (kryteriów wartościujących).

Podobne do opisywanego wyżej analityczne ujęcie sprawności działań prezentuje T. Kotarbiński, według którego sprawnością jest efektywność, wydajność, ale także skuteczność oraz jakość. Zgodnie z tym ujęciem, działa się tym sprawniej, im działanie to bliższe jest posiadania w sobie wszystkich walorów dobrej roboty, i to w jak najwyższym wymiarze [Kotarbiński 1982, s. 117]. Walorami dobrej roboty w przypadku projektowania złożonych systemów informatycznych jest przede wszystkim zestaw cech powstałego systemu informatycznego, które powinny być w jak największym stopniu zbieżne z wymaganiami klienta zamawiającego ten system.

Analogiczne ujęcie zagadnienia sprawności prezentuje J. Zieleniewski [1976, s. 305], wskazując, że sprawność rzeczywista ma miejsce, gdy działanie i jego rezultaty odpowiadają właściwym w danej sytuacji kryteriom sprawności. Z tej definicji wynika, że każdorazowo kryteria oceny sprawności działania powinny być dopasowane do specyfiki danej dziedziny oraz potrzeb informacyjnych i decyzyjnych kierownictwa.

Skuteczność określa stopień realizacji założonego celu działania. Miara ta jest stopniowalna, tj. działanie może być: w pełni skuteczne (zamierzony cel został w pełni osiągnięty), częściowo skuteczne (cel nie został w pełni osiągnięty), nieskuteczne (cel nie został osiągnięty), bezskuteczne (działanie pozostaje w sferze zamiaru), obojętne (działanie nie związane z celem), przeciwskuteczne (działanie oddaliło osiągnięcie zamierzonego celu).

Korzystność odnosi się do różnicy pomiędzy wynikiem użytecznym (W) a kosztami działania (K). Działanie może być korzystne ($W > K$), obojętne ($W = K$) oraz niekorzystne ($W < K$). Wykorzystanie miary korzystności wymaga wartościowego ujęcia ekonomicznego wyniku działania oraz jego kosztów, co często jest trudne lub niemożliwe w przypadku złożonych projektów informatycznych lub działań obarczonych znacznym ryzykiem niepowodzenia.

Dokładność działania oznacza stopień zgodności zrealizowanego produktu do wcześniej zaplanowanego wzoru. W przypadku realizacji złożonych systemów informatycznych dokładność może odnosić się zarówno do wyników przetwarzania danych w zrealizowanym systemie, jak i zgodności funkcjonalnej systemu z jego specyfikacją wynikającą z analizy wymagań klienta.

Prostota jest to przeciwieństwo zawiłości i działania złożonego. Według T. Kotarbińskiego [1982], sposób działania jest prostszy, jeśli składa się na niego mniej czynności i jeśli te czynności są łatwiejsze do wykonania i jasno ze sobą powiązane. Cecha ta ma szczególne znacznie w przypadku projektów informatycznych, w których każdy kolejny poziom złożoności wielokrotnie zwiększa ryzyko wystąpienia błędów w oprogramowaniu i utrudnia ich usunięcie. Zawiły kod programu komputerowego uniemożliwia także współpracę w zespołach programistów, gdy nie mogą oni zrozumieć kodu stworzonego przez swoich kolegów.

Niezawodność jest cechą odnoszącą się do funkcjonowania rezultatu działania i zgodnie z jego przeznaczeniem. Pokrewnym terminem jest także solidność produktu wytworzonego. Należy stwierdzić, że niezawodność systemu informatycznego stworzonego przez programistę stanowi z reguły najważniejszy element wymagań klientów wobec takiego systemu. Niezawodność i solidność mogą także odnosić się do podmiotu działającego i charakteryzować wykonawców działania. Solidny i niezawodny wykonawca, to ktoś na kim można polegać, zarówno jeżeli chodzi na terminowość zrealizowanych prac, jak i ich funkcjonalny rezultat.

Racjonalność stanowi uniwersalne ujęcie sprawności działania. Dotyczy nie tylko sposobu lub rezultatu działania, ale także całego procesu czynności składającego się na działanie w sensie rzeczowym i metodologicznym. Działanie jest sprawne w znaczeniu metodologicznym, gdy podmiot działania w granicach posiadanej wiedzy i możliwości przewidywania skutków swoich działań uczynił wszystko, aby maksymalizować ich wynik. Sprawność rzeczywista może być mniejsza na skutek nieprzewidzianych okoliczności lub działania siły wyższej.

Specyfika pracy programistów, a w szczególności jej twórczy charakter wymaga zmodyfikowanego podejścia do oceny sprawności wykonywanych przez nich działań. Działaniem jest bowiem nie tylko świadome poruszanie mięśniami (pisanie na klawiaturze), lecz także myślenie o tym jak rozwiązać konkretny problem z nim związany. Sprawne myślenie może wpływać pozytywnie na podejmowane przez działającego decyzje prowadzące do osiągnięcia zakładanych rezultatów. Proces podejmowania decyzji poprzedza podjęcie działania i jest związany z wieloma procesami mentalnymi, które także należy uznać za działanie, dlatego także im można przypisać cechę sprawności.

Do przykładowych procesów myślowych będących wyrazem twórczego działania można zaliczyć:

– opracowanie sposobów klasyfikacji i identyfikacji procesów,

– opracowanie udoskonaleń organizacyjnych i technicznych, które pozwalają dokładniej poznać badane produkty i procesy,

– tworzenie i rozwój baz danych i baz wiedzy,

– zaprojektowanie czegoś, co nie było dotychczas znane jako wytwór człowieka, czyli innowacja w sensie podstawowym,

– uczynienie mierzalnym czegoś dotychczas niemierzalnego, doskonalenie narzędzi pomiarowych,

– sformułowanie ogólnych praw dotyczących pewnego stanu rzeczywistości, budowa teorii naukowych,

– wykrycie nieznanych faktów i zależności empirycznych, stawianie i weryfikacja hipotez,

– zrozumienie tego, co nie było dotychczas zrozumiałe, analiza i wyjaśnianie zjawisk zachodzących w środowisku.

Praca twórcza, namysł, zastanawianie się itp. stanowią niezwykle złożony obszar analizy. Konsekwencją tego jest trudność w ocenie sprawności pracy twórczej przez kierowników zlecających wykonanie tej pracy. Jej efekty można oceniać po zakończeniu zaplanowanych etapów i przedstawieniu przez wykonawcę (sprawcę działania) rezultatów. Ekonomiczne rezultaty pracy twórczej, tj. przychody i zyski z wdrożenia jej efektów, są często odległe w czasie od zakończenia samego działania. Na obecną chwilę nie da się w pełni analizować samego procesu pracy twórczej, który zachodzi w mózgu działającego człowieka. Problem ten nie występuje dla prac manualnych czy zmechanizowanych, gdzie poszczególne etapy procesu pracy mogą być zaobserwowane, zmierzone i poddane analizie.

4. Kryteria sprawności działania w zwinnych metodykach projektowania systemów informatycznych

Ocena sprawności zrealizowanych i zaplanowanych do realizacji działań ze względu na różnorodne kryteria jest obecnie jednym z najważniejszych zadań kierownika w każdej organizacji. Decyduje ona o efektywności podejmowanych decyzji kadrowych i rezultatach osiąganych przez zespoły pracownicze.

Sprawność może oznaczać stopień realizacji danego celu lub, jeżeli miernik osiągnięcia celu nie jest stopniowalny, będzie miała charakter binarny (tzn. cel został osiągnięty lub nie). W przypadku projektów informatycznych trudno jest jednakże precyzyjnie określić jeden cel. Często projekt obejmuje wiązkę celów związanych z różnymi funkcjonalnościami systemu realizowanymi na potrzeby zleceniodawcy projektu.

Cele działania podlegają hierarchizacji. Można wyróżnić cele główne, pośrednie i podrzędne. Właściwe określenie priorytetów (celów głównych) i ich podział na cele cząstkowe (pośrednie) pozwala na osiąganie większej sprawności w działaniu jednostek oraz całych zespołów pracowniczych. Cele mogą być określane indywidualnie oraz dla całych zespołów. W przypadku celów zespołowych mogą one być im narzucane przez kierownictwo i system zarządzania organizacji, ale także mogą być wynikiem spontanicznych działań i przemyśleń członków zespołu. Należy

podkreślić zasadniczą rolę wcześniejszego sformułowania odpowiednich celów dla osiągnięcia sukcesu zarówno jednostek, jak i całych organizacji.

Cele, które nie zostały wyraźnie sformułowane, są niejasne, wieloznaczne, często nie zostały zapisane lub określone w sposób dający się zmierzyć, nie spełniają swojej roli motywacyjnej i utrudniają osiąganie wysokiego poziomu sprawności działania. W przypadku projektów informatycznych realizowanych z wykorzystaniem zwinnych metodyk jednoznaczne formułowanie celów może być utrudnione. Wynika to z braku jednoznacznego zdefiniowania projektu i funkcjonalności ostatecznego produktu programistycznego. Można przyjąć, że w hierarchicznym ujęciu najistotniejszy jest cel ekonomiczny firmy realizującej zlecenie na potrzeby klienta, ale trudno jest ten cel przełożyć na konkretne cele poszczególnych programistów i zespołów.

Jeżeli w organizacji występują działania, których wyniki nie mogą być skwantyfikowane, ocena odbywa się na podstawie ich opisu i porównania do działań wykonywanych przez innych pracowników wykonujących podobne funkcje. Ocenę pracy zespołów i jednostek w firmach informatycznych przeprowadza się na podstawie różnorodnych kryteriów, takich jak np.: poziom zaangażowania w prace zespołowe, wysiłek intelektualny, stopień odpowiedzialności za wynik ostateczny, jakość kodu, poziom wykształcenia, doświadczenie zawodowe, umiejętność rozwiązywania problemów, odporność na stres, poziom samoorganizacji, ilość błędów w kodzie.

Jednym z głównych kryteriów oceny sprawności pracy jest ekonomiczność działania. Działanie jest ekonomiczne, jeżeli stosunek wyniku użytecznego do kosztu poniesionego na realizację tego działania jest większy od jedności. Działanie jest obojętne ze względu na ekonomiczność, jeżeli stosunek ten jest równy jedności, działanie jest nieekonomiczne, jeżeli stosunek ten jest mniejszy niż jedność. Miarą ekonomiczności działania jest więc stosunek pomiędzy nakładem i rezultatem, a nie wartość bezwzględna rezultatu [Zieleniewski 1976, s. 230].

W przypadku projektów informatycznych realizowanych z wykorzystaniem metodyk zwinnych niejednokrotnie trudno jest odnieść nakład pracy (i koszt tej pracy) do konkretnego rezultatu. Często kod powstający w ramach jednego projektu jest wykorzystywany w innym projekcie, niejednokrotnie także realizowane są funkcjonalności, które nie znajdują zastosowania w finalnym systemie informatycznym. Tradycyjny rachunek ekonomiczny traktowałby takie wydatki jako koszty nieekonomiczne, jednakże często umiejętności i doświadczenie, które programiści zdobywają w trakcie takiej pracy procentują w przyszłości i przyczyniają się do lepszej pracy przy innych projektach.

Innym kryterium oceny sprawności jest skuteczność pracowników. Skuteczność nie zawsze jest warunkiem pozytywnej oceny pracy – pozytywny wynik pomiaru sprawności mogą bowiem spowodować „pozytywne niespodzianki" mogące towarzyszyć działaniom nieskutecznym. Mimo to w takich przypadkach

działania nieskuteczne mogą być korzystne i ekonomiczne, należą jednak do wyjątków. W typowych więc przypadkach odpowiednio wysoki stopień skuteczności jest warunkiem korzystności i ekonomiczności działań. Kryterium skuteczności ma szczególnie istotne znaczenie w przypadku zwinnych metodyk tworzenia oprogramowania. Skuteczność, tj. osiągnięcie użytecznego funkcjonalnego kodu programu, jest nadrzędnym celem sformułowanym w manifeście zwinnego oprogramowania. Mniejsze znaczenie ma tutaj dobra organizacja pracy i precyzyjna dokumentacja, często także występują „pozytywne niespodzianki", gdy powstaje kod i funkcjonalność, która nie była zaplanowana, ale spełnia potrzeby klienta zamawiającego produkt informatyczny.

Jedną z metod ograniczania niepożądanych strat związanych z działaniami jest dobre zorganizowanie pracy i zastosowanie metod i narzędzi wspomagających wykonywane zadania. Metody te mają mniejsze znaczenie w zwinnych metodykach tworzenia oprogramowania, gdzie większy nacisk kierowany jest na dostosowanie się do zmian. Praca dobrze zorganizowana charakteryzuje się następującymi cechami [Pszczołowski 1982, s. 112–113]:

– działanie poprzedza ustalenie wyraźnego i ściśle określonego celu działania,

– wykonany został szczegółowy i dokładny przegląd wszystkich warunków i środków, za pomocą których wyznaczony cel może być osiągnięty,

– utworzony został dokładny plan działań zmierzających do celu, przy zastosowaniu najbardziej odpowiednich środków w najlepszych warunkach,

– plan został skrupulatnie zrealizowany, bez odkładania istotnych działań na później,

– skontrolowano osiągnięte wyniki przez porównanie ich z zamierzonym celem, wyciągnięto wnioski z poszczególnych etapów i wprowadzono poprawki dotyczące celu, warunków, środków, planu działania, jego realizacji i kontroli.

Mimo że wymienione wyżej cechy dobrze zorganizowanego działania znajdują się w opozycji do głównych założeń metodyk zwinnego tworzenia oprogramowania, to kierownicy chcący podnosić sprawność działań zespołów programistów powinni rozważyć chociażby częściowe ich wdrożenie do praktyki. Sprawność działania jest jednym z wielu istotnych zagadnień będących w obszarze zainteresowań każdego kierownika.

5. Pomiar sprawności działania programistów

Pomiar jest podstawowym instrumentem pozyskiwania informacji wykorzystywanej przez kierowników do podejmowania decyzji, jednakże w praktyce przedsiębiorstw (jak również innych organizacji) unikają oni ilościowego ujęcia wielu badanych przez siebie zjawisk. Mierzenie badanych zjawisk i procesów, od czasów prekursora metody indukcyjnej F. Bacona [1955], stanowi podstawową metodę

pozyskiwania prawdziwej wiedzy o świecie. W swojej koncepcji odszedł on od dominującego wcześniej wśród filozofów założenia, że wiedza o świecie jest absolutną, niezależną od badacza prawdą, zaś jego zadaniem jest jej odkrycie drogą logicznego myślenia, poprzez wyjście od uznanego za niepodważalne zestawu aksjomatów. F. Bacon uważał, że rolą badacza jest pozyskiwanie jak największej liczby danych, rejestracja obserwacji i wyników eksperymentów, a następnie na ich podstawie dochodzenie do wniosków w zakresie zasad i praw rządzących zachodzącymi w naturze zjawiskami.

W dziedzinie organizacji i zarządzania zalecenia empiryzmu w zaczęły się upowszechniać pod koniec XIX w. Staje się to widoczne w pracach A. Smitha [2007] i F.W. Taylora [1965], którego obecnie uznajemy za twórcę koncepcji naukowego zarządzania. Od czasów F.W. Taylora rozwój metod i narzędzi w dziedzinie badań naukowych (w szczególności w dziedzinie nauk przyrodniczych: fizyce, biologii itp.) postępował bardzo szybko. Współczesny poziom rozwoju technologii jest w dużej części efektem rozwoju nauki oraz zastosowania jej dokonań w praktyce.

Można wysunąć twierdzenie, że zaawansowanie współczesnych organizacji, wyrażające się m.in. wzrostem produktywności, wydajności pracy i efektywności działania, jest efektem rozwoju metod i technik, które po raz pierwszy zostały systematycznie opisane w pracach A. Smitha i F.W. Taylora. Dla praktyki zarządzania kluczowe znaczenie miał i ma nadal pomiar efektywności działania oraz identyfikacja wszelkich zjawisk i czynników mogących mieć na tę efektywność wpływ. Począwszy od ustalenia optymalnej wielkości łopat używanych przez robotników w badaniach F.W. Taylora po analizę wpływu oświetlenia na wydajność pracowników w klasycznych już badaniach E. Mayo [1945] zawsze zasadnicze znaczenie dla badacza miało ustalenie poprzez obserwację i pomiary ilościowych poziomów charakteryzujących interesujące go zjawiska. W praktyce działania wielu współczesnych organizacji dostrzegalny jest ciągły rozwój różnorodnych narzędzi pomiarowych, metod i technik rejestracji oraz analizy danych.

Kwantyfikacji i ocenie sprawności powinny podlegać jedynie działania kluczowe dla organizacji. Są to działania wywierające największy wpływ na osiągane przez nią efekty ekonomiczne oraz zdolność do rozwoju. Sprawna realizacja tych działań stanowi często tzw. kluczowy czynnik sukcesu organizacji, gdyż w istotny sposób wpływa na osiągane przez nią cele. Działania te należy zidentyfikować poprzez przeprowadzenie procedury badawczej polegającej na zebraniu i rejestracji pochodzących z różnorodnych źródeł (wywiadów, dokumentacji, obserwacji procesu) informacji dotyczących realizowanych procesów wytwórczych i procesów zarządzania. W trakcie badań wykorzystuje się różnorodne metody i techniki identyfikacji, metody analizy z obszaru badań organizatorskich oraz doskonalenia systemów zarządzania.

W przypadku projektów informatycznych realizowanych z wykorzystaniem elastycznych metodologii tworzenia oprogramowania do kluczowych działań należy zaliczyć:

– tworzenie specyfikacji funkcjonalnej modułów,
– budowanie specyfikacji interfejsu użytkownika,
– projektowanie struktur baz danych,
– kodowanie – programowanie,
– testowanie oprogramowania,
– tworzenie dokumentacji użytkownika i in.

Poszczególne zadania realizowane są przez wykwalifikowanych specjalistów, jednakże w przypadku metodyk zwinnych występuje szczególny nacisk na etap programowania. Często struktury baz danych i elementy interfejsu użytkownika są dodawane lub modyfikowane przez programistę w trakcie kodowania i zachodzi wtedy konieczność aktualizacji dokumentacji projektowej *ex post*. W przypadku dużych zespołów programistów może to powodować problemy polegające na błędach uniemożliwiających integrację i poprawne działanie modułów oprogramowania utworzonych z wykorzystaniem różnych struktur danych.

Stanowisko programisty jest typowym stanowiskiem obecnym w przedsiębiorstwach tworzących złożone systemy informatyczne. Główną funkcją pracownika jest programowanie, utrzymanie aktualności dokumentacji, testowanie i współpraca przy tworzeniu kodu złożonego systemu informatycznego. Zasadnicze kompetencje programisty związane są ze znajomością jednego lub więcej języków programowania pozwalających rozwiązywać zadane problemy algorytmiczne. Podstawowe zadania pracownika na tym stanowisku to interpretacja założeń projektowych zdefiniowanych przez analityków systemu i udokumentowanych w specyfikacji funkcjonalnej systemu i projekcie struktur baz danych, napisanie działającego kodu programu w zadanym języku programowania (stosowanych w danym przedsiębiorstwie), testy działania programu, testy integracji programu, tj. współpracy programu z innymi modułami systemu lub bibliotekami kodu, analiza i ocena jakości wyników generowanych przez napisany program.

Tabela 1. Przykładowe kryteria oceny sprawności na stanowisku programisty

Wyszczególnienie	Wartość normatywna
Liczba efektywnych linii kodu stworzonych przez programistę	2000 linii tygodniowo
Średni czas identyfikacji i usunięcia błędu w kodzie systemu	2,5 h
Liczba testów systemu związanych z korygowaniem błędu w kodzie	10 tygodniowo
Średni czas realizacji jednego modułu funkcjonalnego	5 dni
Liczba aktualizacji dokumentacji systemowej tworzonego oprogramowania	8 miesięcznie

Źródło: opracowanie własne.

O sprawności pracownika na tym stanowisku decyduje jakość napisanego kodu, szybkość rozwiązywania stawianych przed nim problemów programistycznych, liczba wykonanych przebiegów testowych, liczba błędów i poprawek w napisanym kodzie. Przykładowe wskaźniki pomiaru sprawności pracownika na tym stanowisku przedstawiono w tabeli 1.

Interpretując przedstawione w tabeli 1 mierniki sprawności działania, kierownik może wskazać kierunki doskonalenia dla programistów realizującego te działania. Należy jednak pamiętać, że pewne ich wartości mogą być związane z czynnikami niezależnymi od pracowników. Przykładowo czas identyfikacji i usunięcia błędu w kodzie może być uzależniony od stopnia złożoności błędu i powiązania błędu z algorytmami realizowanymi w innych modułach systemu, stworzonych przez innego programistę. Pracownicy o wyższych kwalifikacjach będą otrzymywać do wykonania moduły o wyższym poziomie złożoności i przez to narażeni będą na większą liczbę potencjalnych błędów, dłuższy też będzie czas niezbędny do ich realizacji. Podobnie niejednoznacznym przy interpretacji wskaźnikiem będzie liczba linii kodu, która w dużym stopniu uzależniona jest od zastosowanego języka programowania oraz stosowanego przez programistę stylu tworzenia programu (dobry programista potrafi rozwiązać dany problem za pomocą mniejszej liczby operacji, tj. krótszego kodu). Mimo wskazanych trudności w interpretacji, kwantyfikacja przedstawionych powyżej mierników sprawności pozwala na dokonywanie porównań pomiędzy programistami oraz porównań w czasie, dzięki czemu można zidentyfikować trendy i kierunki zmian w sprawności ich działań oraz odpowiednio wcześniej zareagować na zmiany niekorzystne. Dzięki temu kierownicy mogą podejmować efektywne decyzje personalne przekładające się na skuteczność całego zespołu programistów zatrudnionych do realizacji złożonego systemu informatycznego.

6. Podsumowanie

Wzrost wymagań stawianych przed firmami informatycznymi przez klientów i innych interesariuszy powoduje, że zwiększenie sprawności działania zespołów programistów staje się imperatywem dla podejmowania wszelkich działań kierowniczych. Na sprawność działania jednostek i zespołów w różnym stopniu wpływają: struktura organizacyjna, stosowane narzędzia motywacji i kontroli, stosowana metodyka zarządzania projektami, kompetencje kierownictwa i pracowników, ergonomiczne uwarunkowania stanowiska pracy i strategia firmy. Identyfikacja czynników mających największy wpływ na sprawność wymaga stałego monitorowania efektów pracy, co wiąże się z koniecznością jej kwantyfikacji. Konieczność ilościowego ujęcia sprawności pracy dodatkowo wzmaga

też fakt wykorzystania mierników sprawności do porównań zarówno pomiędzy pracownikami, ale też zespołami lub całymi organizacjami. Porównania te są następnie podstawą podejmowania strategicznych decyzji rozwojowych we wszystkich obszarach funkcjonalnych przedsiębiorstwa. Zasadniczym błędem, popełnianym przez kierowników jest przyjęcie założenia, że jeżeli jakieś zjawisko trudno jest zmierzyć, to najlepiej zignorować jego istnienie (z reguły uznaje się go jako czynnik niewywierający istotnego wpływu wynik pomiaru). Założenie to może prowadzić jednakże do błędnych decyzji, podejmowanych na podstawie niepełnej informacji o procesach zachodzących w przedsiębiorstwie. Słuszna wydaje się w związku z tym teza, że nawet w przypadku stosowania zwinnych metodologii realizacji projektów informatycznych, należy podejmować próby operacjonalizacji podstawowych parametrów opisujących sprawność i efektywność działania zespołów programistów. Umożliwi to kierownikom podejmowanie skutecznych decyzji opierających się na faktach, a nie przypuszczeniach i przeczuciach.

Literatura

Bacon F. [1955], *Novum Organum*, PWN, Warszawa.
Highsmith J. [2002], *Agile Software Development Ecosystems*, Addison Wesley, Indianapolis.
Kieżun W. [1977], *Podstawy organizacji i zarządzania*, KiW, Warszawa.
Kotarbiński T. [1982], *Traktat o dobrej robocie*, Ossolineum, Wrocław.
Manifest zwinnego tworzenia oprogramowania, http://agilemanifesto.org/iso/pl/ (dostęp: 14.07.2013).
Mayo E. [1945], *The Social Problems of an Industrial Civilization,* Graduate School of Business Administration, Harvard University.
Pszczołowski T. [1978], *Mała encyklopedia prakseologii i teorii organizacji*, Ossolineum, Wrocław–Gdańsk.
Schwaber K. [2004], *Agile Project Management with Scrum*, Microsoft Press, Washington.
Smith A. [2007], *Badania nad naturą i przyczynami bogactwa narodów*, t. I i II, Wydawnictwo Naukowe PWN, Warszawa.
Taylor F.W. [1967], *The Priniciples of Scientific Management*, W.W. Norton & Company, Inc., New York.
Zieleniewski J. [1976], *Organizacja zespołów ludzkich. Wstęp do teorii organizacji i kierowania*, PWN, Warszawa.

Measuring Efficiency in Agile Software Design Methodologies

The paper presents theoretical and methodological considerations in matters of efficiency in software project development teams which use agile methodologies. In the contemporary literature, those methodologies are described as a universal solution to

all organisational problems faced by managers of large teams working on complex IT projects.

This article attempts to answer the question of the extent to which agile software development methodologies fulfill the conditions of the classical methodological efficiency. I suggest that even when using agile design methodology, managers should attempt to operationalise the basic parameters describing the efficiency and effectiveness of the team. This will allow them to make effective decisions based on facts, not assumptions and hunches.

Keywords: efficiency, performance measurement, IT projects, decision-making.