INFORMATYKA EKONOMICZNA BUSINESS INFORMATICS 2(48) • 2018

ISSN 1507-3858
e-ISSN 2450-0003

Ewelina Wróbel
Uniwersytet Ekonomiczny we Wrocławiu
e-mail: ewelina.wrobel@ue.wroc.pl
ORCID: 0000-0002-3721-928X

HYBRYDYZACJA METODYK *KANBAN* I *SCRUM* JAKO NARZĘDZIE DOSTARCZANIA OPROGRAMOWANIA

HYBRIDIZATION OF KANBAN AND SCRUM METHODOLOGIES AS A TOOL OF SOFTWARE PROVIDING

DOI: 10.15611/ie.2018.2.08

Streszczenie: Celem artykułu jest wypracowanie metodyki, będącej odpowiedzią na zmieniające się otoczenie przedsięwzięcia informatycznego. Autorka zajęła się problemem badawczym, związanym z niedostosowaniem metodyki do specyfiki organizacji i wykonywanego projektu. Biorąc pod uwagę wieloletnie doświadczenie we wdrażaniu i utrzymaniu systemów informatycznych wspierających działanie przedsiębiorstw w różnych branżach i sektorach gospodarki opisała możliwości oraz słabe strony dwóch wybranych metodyk, a następnie dokonała ich hybrydyzacji, sięgając po najlepsze cechy obu. Efektem pracy są założenia początkowe nowej metodyki, które mogą być dostosowywane przez zespół projektowy do swoich indywidualnych potrzeb. Wypracowana metodyka jest elastyczna i pozwala na jej dalszy rozwój zgodnie z potrzebami zespołu oraz projektu.

Słowa kluczowe: *agile*, *kanban*, metodyki, *scrum*, zarządzanie projektem informatycznym, oprogramowanie.

Summary: The aim of the article is developing a methodology which is a response to the changing environment of an IT venture. The author addressed the research problem related to the maladjustment of the methodology to the specifics of the organization and the project. Given the many years of experience in implementing and maintaining IT systems supporting business operations in various industries and sectors of the economy, he described the possibilities and weaknesses of the two selected methodologies and then hybridized them, reaching for the best features of both. The result of the work are the initial assumptions of the new methodology, which can be adapted by the project team to their individual needs.

Keywords: agile, kanban, methodologies, Scrum, IT project management, software development.

1. Wstęp

Obecnie obserwuje się tendencję do pełnej informatyzacji przedsiębiorstw. Ze względu na szeroki zakres tego procesu należy rozdzielić obszar utrzymania systemu informatycznego od projektów wdrożeniowych [Flasiński 2006].

Projekty informatyczne są często skomplikowane i rozłożone w czasie, a duża dynamika w obszarze IT powoduje, że odchodzi się od projektowania i planowania całości prac związanych z wdrożeniem systemu przed realizacją przedsięwzięcia. Obserwowany od kilku lat w branży IT trend pokazuje, że firmy doceniają zalety iteracyjnego i przyrostowego dostarczania oprogramowania dla klienta. Pozwala to na ograniczenie ryzyka związanego z niepowodzeniem projektu.

Celem artykułu jest, poprzez wykorzystanie takich metod, jak przegląd literatury przedmiotu, studium przypadku i obserwacja zjawisk, analiza i porównanie dwóch wywodzących się z japońskich firm produkcyjnych metodyk: *kanban* i *scrum*, które pozwalają na przyrostowe dostarczanie oprogramowania do klienta, oraz zaproponowanie podejścia będącego hybrydą tych dwóch metodyk.

2. Opracowanie strategii informatyzacji przedsiębiorstwa – otoczenie systemu

Zaplanowanie procesu tworzenia projektu informatycznego polega na określeniu potrzeb i oczekiwanych rezultatów, zdefiniowanych w strategii informatyzacji przedsiębiorstwa [Wieczorkowski 2016]. Aby jednak przystąpić do takich działań, należy odpowiedzieć na kilka kluczowych pytań:

1. W jakiej branży działa przedsiębiorstwo oraz jaka jest dynamika jego biznesu?
2. Jaką część biznesu chcemy informatyzować? Które moduły wdrażane będą w pierwszej kolejności?
3. Co chcemy osiągnąć informatyzacją? Jakie są kluczowe cele, jakich oczekuje się rezultatów?
4. Czy potrzebujemy systemu dedykowanego[1] czy powielarnego[2]?
5. Czy budujemy własny zespół czy korzystamy z dostawcy zewnętrznego?

Po udzieleniu odpowiedzi na powyższe pytania można przejść do budowania strategii informatyzacji przedsiębiorstwa. Jest to plan działań, który cechuje się długoterminowością i kompleksowością.

Mając przygotowaną strategię, można wdrożyć plan w życie. W przypadku wyboru dostawcy zewnętrznego to on jest odpowiedzialny za opracowanie metodyki działania, choć rosnąca świadomość klientów pozwala na ich angażowanie w meto-

[1] System dedykowany (skastomizowany – *customized*) oznacza system dopasowany do specyfiki przedsiębiorstwa, w którym działa.

[2] System powielarny oznacza system sprzedawany na masową skalę. Jego architektura pozwala na jego wykorzystanie w różnych organizacjach.

dykach zwinnych jako partnera i właściciela produktu. Ogromną przewagą dostawców zewnętrznych jest jednak uczestnictwo w innych projektach, a co za tym idzie – znajomość branży i ograniczeń metod.

W przypadku decyzji o wdrożeniu systemu przez własny zespół wytwórczy dobrą praktyką jest wybór doświadczonego właściciela produktu – osoby, która będzie odpowiadać za dobór metodyki i przygotowanie zespołu do pracy w niej.

3. Wybór metodyki

Największą popularnością w ostatnich latach cieszą się metodyki zwinne. Z raportu „The 12th annual state of Agile" wynika, że aż 30% ankietowanych przedsiębiorstw ma 5 lub więcej lat doświadczenia w pracy zwinnej[3]. Niewątpliwie jedną z cech metodyk zwinnych, która wpływa na ich rosnącą popularność, jest to, że pozwalają one na iteracyjne dostarczanie produktu, dzięki czemu możliwa jest szybka reakcja na ryzyko (związane z przerwaniem projektu – czas pracy rozliczany jest między klientem a dostawcą cyklicznie, na koniec każdej iteracji). Pozwalają one także na regularną kontrolę odbiorcy systemu odnośnie do spełnienia przez system oczekiwań klienta. W metodykach tradycyjnych nie ma takiej możliwości – produkt jest implementowany tylko w całości.

Metodyki zwinne (*agile*) obejmują [Abrahamsson i in. 2002; Wyrozębski 2011]:

- *kanban*,
- *scrum*,
- *lean*,
- *test driven development*,
- *dynamic system development method*,
- *extreme programming.*

Wielu dostawców oprogramowania, niezależnie od tego, w jakiej branży działają, z jakim klientem mają do czynienia oraz jakie zasoby posiadają, wybierając metodykę, podąża za modą. Jest to podejście niewłaściwe, ponieważ współczesny wachlarz metod wdrażania i utrzymania systemu pozwala na pełne wykorzystanie posiadanych zasobów, z uwzględnieniem otoczenia, w jakim projekt jest osadzony. W niniejszym artykule przedstawiono najbardziej popularne metodyki wdrażania[4] oprogramowania, czyli *kanban* i *scrum*, kładąc duży nacisk na zalety i ograniczenia wynikające z wyboru każdego z narzędzi.

---

[3] The 12th State of Agile Report, Collabnet Versionone, https://explore.versionone.com/state-ofagile/versionone-12th-annual-state-of-agile-report.

[4] Wdrażanie oprogramowania można podzielić na dwa podejścia. Wdrożenia *sensu largo* (od strategii, projektowania, programowania, testowania do uruchomienia systemu w przedsiębiorstwie) oraz wdrożenie *sensu stricto* (obejmujące jedynie instalację systemu w przedsiębiorstwie). Autor niniejszej pracy omawia metodyki, które mogą przyczynić się do wdrożenia systemu w obu tych ujęciach, jednak omówione zostaną cechy charakterystyczne dla wdrożeń *sensu largo*.

3.1. Charakterystyka metodyki *kanban*

Jedną z najbardziej popularnych metod wytwarzania oprogramowania zwinnego jest wykorzystanie metodyki *kanban*. Nie jest ona jednak stworzona na potrzeby przedsięwzięć informatycznych. Metodyka *kanban* powstała w latach czterdziestych XX wieku, w kolebce dużych, dobrze zorganizowanych przedsiębiorstw produkcyjnych – Japonii – w Toyota Motors Corporation [Krasiński 2013]. Rozwiązanie to stało się bardzo popularne dzięki uzyskanym przez Toyota Motors rezultatom. Po trzech latach od wdrożenia [Zamostny (red.), 2012]:

- zredukowano aż o 75% zapasy potrzebne do zapewnienia ciągłości produkcji, a co za tym idzie – doszło do znacznej redukcji wykorzystywanych przestrzeni magazynowych i związanych z ich utrzymaniem kosztów (pracownicy oraz utrzymanie budynków),
- o 95% zredukowano braki,
- doszło do 25-procentowego wzrostu produkcji, przy jednoczesnym ograniczeniu przestrzeni produkcyjnej o 10% (oraz ograniczeniu kosztów koniecznych do jej utrzymania).

Cechy systemu kontroli procesu *kanban*, które pozwoliły na jego wykorzystanie w przedsięwzięciach informatycznych, to [Ciesielski 2006]:

- redukcja zapasów,
- redukcja zaangażowania kapitału w wytworzenie produktu,
- wprowadzenie elastyczności produkcji,
- zachowanie gotowości do ciągłego wytwarzania produktu.

Jak można zauważyć, wymienione cechy są na tyle uniwersalne, że możliwe jest ich przełożenie i zastosowanie w projektach w obszarze IT, gdzie produktem jest wytwarzane oprogramowanie, a zapasem i kapitałem zatrudnione zasoby ludzkie.

Fundamentalne założenia metodyki *kanban* w IT to [Ordysiński 2014]:

1. Wizualizacja przepływu wykonywanej pracy, np. za pomocą tablicy (suchościeralnej lub wykorzystując karteczki). Przykładową tablicę do realizacji zadań związanych ze stworzeniem kartoteki egzaminów, w systemie działającym w szkole, przedstawia rys. 1. Odcieniami tła przedstawiony został obszar wykonawczy, jakiego dotyczy zadanie (dokumentacja, baza danych, grafika, programowanie). Innym wyróżnieniem (np. pogrubienie tekstu) wyróżnia się zadania priorytetowe, kluczowe do realizacji całości przedsięwzięcia informatycznego. Kolumny tablicy obrazują przepływ pracy (każda kolejna kolumna jest jednym z etapów niezbędnych do uznania zadania za zakończone).

Tablica zadań zawiera następujące kolumny:

1.1. Do realizacji (*to do* – lista zadań do wykonania, oczekująca na pobranie do realizacji.

1.2. W trakcie realizacji (*doing*) – lista zadań w trakcie wykonywania lub implementacji.

1.3. W trakcie weryfikacji/testów (*review/testing*) – lista zadań, które zostały wykonane oraz zostały pobrane do przeglądu kodu (*code review*) lub testów.

1.4. Gotowe (*ready*) – zadania zaakceptowane przez osoby wykonujące przegląd kodu oraz testy. Zadania te oczekują na instalację. Jeśli zadanie nie wymaga instalacji, sekcja ta jest pomijana.

1.5. Zakończone (*done*) – zadania zainstalowane.

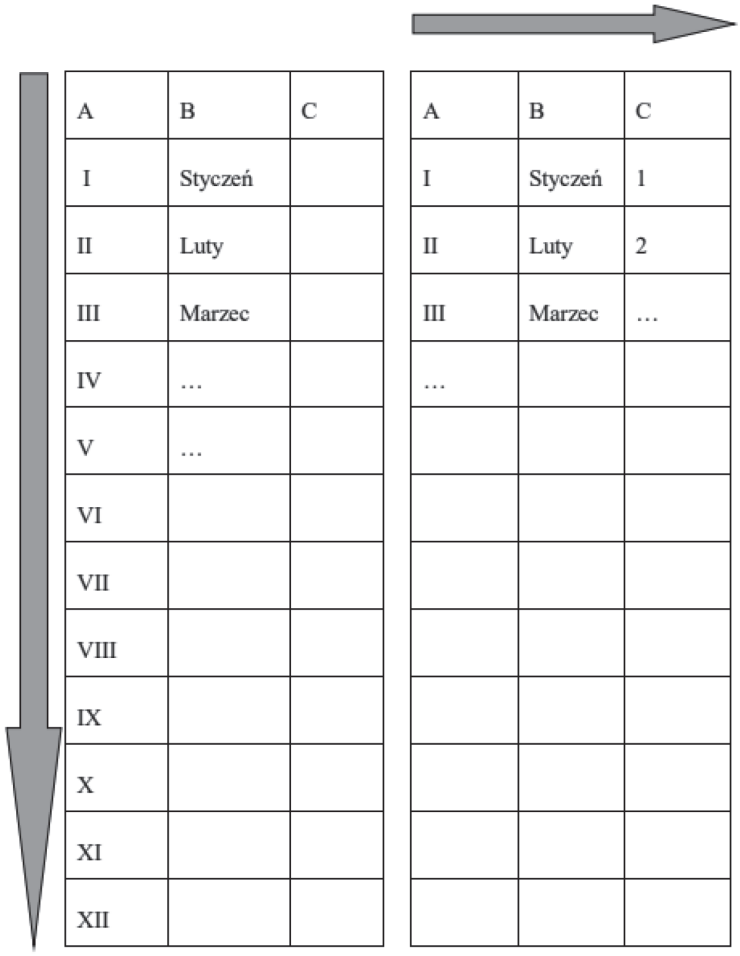
TO DO	**DOING**	**REVIEW/ TESTING**	**READY**	**DONE**
Do realizacji (1.1)	**W trakcie realizacji (1.2)**	**W trakcie weryfikacji/ testów (1.3)**	**Gotowe (1.4)**	**Zakończone (1.5)**
(3) Trzeci krok kreatora egzaminu	(2) Drugi krok kreatora egzaminu	(1) Pierwszy krok kreatora egzaminu	(4) Struktura egzaminu w bazie danych	(8) Uzupełnienie dokumentacji projektowej
(5) Funkcja masowego importu wyników egzaminu			(7) GUI kartoteki egzaminów	
(6) Masowy import wyników egzaminu			(9) GUI kreatora egzaminów	

Rys. 1. Przykładowa tablica wizualizująca zadania w metodyce *kanban*

Źródło: opracowanie własne.

2. Pomiar czasu dostarczania oraz jego stałe monitorowanie. Punkt ten ma na celu zwrócenie uwagi na „wąskie gardła" procesu oraz ich optymalizację. Pozwala na koncentrację zasobów tam, gdzie są one rzeczywiście potrzebne.

3. Redukcja pracy w toku (WIP – *Work in Progress*). W przedsięwzięciach informatycznych oznacza to koncentrację zespołu na minimalnej, możliwej do osiągnięcia liczbie zadań realizowanych w tym samym czasie. Pozwala to skrócić czas wykonania i dostarczenia zadania, które jest samo w sobie wartością dodaną dla projektu. Pozwala również na wyeliminowanie lub znaczną redukcję straty czasu, jaka jest ponoszona przez pracowników podczas przełączania skupienia między kontekstami realizowanych zadań. Najczęściej wykorzystywanym przez trenerów zarządzania projektami zadaniem, mającym na celu wizualizację strat podczas zmiany kontekstu, jest to polegające na wypełnieniu tabeli z rys. 2 (kolumnę A należy wypełnić kolejnymi liczbami rzymskimi, kolumnę B miesiącami, a kolumnę C liczbami arabskimi.



A	B	C
I	Styczeń	
II	Luty	
III	Marzec	
IV	…	
V	…	
VI		
VII		
VIII		
IX		
X		
XI		
XII		

A	B	C
I	Styczeń	1
II	Luty	2
III	Marzec	…
…		

Rys. 2. Zadanie mające na celu obliczenia straty czasu podczas zmiany kontekstu zadań

Źródło: opracowanie własne.

Pierwsza część zadania (tabela z lewej strony) polega na uzupełnianiu kolejnych kolumn. Drugi etap zadania to wypełnianie tabeli poprzez uzupełnianie kolejnych wierszy (tabela z prawej strony). To ćwiczenie pokazuje, że czas potrzebny na realizację pierwszej części jest znacznie dłuższy niż czas przeznaczony na wykonanie części drugiej. Zjawisko, jakie tu zachodzi, to właśnie przełączanie kontekstu, czyli upraszczając – każda kolumna to oddzielny kontekst. Osoba, która wypełnia kolejne kolumny, realizuje zadanie w ramach jednego kontekstu i przechodzi do kolejnego. Uzupełniając kolejne wiersze, mamy do czynienia ze stratą czasu wynikającą ze zmiany kontekstu (poprzez konieczność przypominania sobie wcześniej ustalonej sekwencji i jej kroku, jaki w danym momencie następuje).

Kanban jest metodą elastyczną i uniwersalną, ma jednak pewne ograniczenia. Jednym z nich jest zasada ograniczenia WIP (pracy w toku). Co do zasady reguła ta pozwala na minimalizację strat czasu, spowodowanych zmianą kontekstu, i dla wielu przedsiębiorstw przyniesie wymierną korzyść. W sytuacjach szczególnych jednak zasada ta może być ograniczeniem. Jedną z takich sytuacji jest praca polegająca na wsparciu użytkowników końcowych systemu, gdzie czas reakcji (oczekiwanie na odpowiedź) może być wykorzystywany do przyjęcia do realizacji i rozpoznania kolejnego zadania (podczas kiedy zadanie wcześniejsze nie zostało jeszcze ukończone). To tylko jeden z przykładów koniecznego zastosowania modelu wielozadaniowości (multitaskingu). Inne to zależność od infrastruktury (dostępność środowisk deweloperskich i testowych[5]) oraz zależności między zadaniami (konieczność zaangażowania innych osób w realizację zadań).

3.2. Charakterystyka metodyki *agile-scrum*

Agile jest to zbiór metodyk i ram postępowania wytwarzania oprogramowania, oparty na zasadach określonych w Agile Manifesto, czyli spisie zasad polegających na przedkładaniu[6]:

1) ludzi i interakcji nad procesy i narzędzia,
2) działającego oprogramowania nad rozbudowaną dokumentację,
3) współpracy i partnerskich relacji z klientem nad negocjacje i umowy,
4) reagowania na zmiany nad realizację założonego wcześniej planu.

Najpopularniejszym zbiorem ram postępowania (*framework*) wykorzystania zwinnego podejścia do wytwarzania oprogramowania, jedocześnie najczęściej wykorzystywanym w przedsiębiorstwach wytwarzających oprogramowanie (*Software house*), jest *scrum*. Określenie to oznacza grupę zawodników rugby, kotłującą się wokół piłki, gdy wznawiany jest mecz. Metaforą meczu jest określona rama czasowa, podczas której wykonywane będą zaplanowane w ramach jednej tablicy zadania, piłką i boiskiem jest lista zadań do realizacji, zaplanowana na daną iterację czasową, a zawodnicy to po prostu zespół pracujący nad realizacją planu.

Podobnie jak metodyka *kanban,* tak i *scrum* wywodzi się z sektora produkcyjnego (w pierwszym opracowaniu służyć miał branży motoryzacyjnej, komputerowej, fotograficznej i drukarskiej). Nie zastosowano wtedy jednak nazwy *scrum*. Pierwszy raz została ona użyta przez Jeffa Sutherlanda i Kena Schwabera, którzy na konferencji OOPSLA[7] w Austin zaprezentowali wizję tej metody [Koszlajda 2010].

---

[5] Środowiska deweloperskie i testowe to środowiska dostawcy oprogramowania lub zespołu wdrażającego oprogramowanie wewnętrznie. Ich zadaniem jest odseparowanie procesu dostarczania oprogramowania od środowiska produkcyjnego, służą do weryfikacji i testów wprowadzanych w systemie zmian.

[6] http://agilemanifesto.org/iso/pl/manifesto.html.

[7] *Object-oriented programming systems*, *languages and applications*.

Scrum zakłada określenie ról w przedsięwzięciu:

1. Właściciel produktu – PO (*Product Owner*), to osoba odpowiedzialna za biznesową płaszczyznę kształtu gotowego produktu.

2. *Scrum master* – osoba odpowiedzialna za przebieg procesu, zgodnie z zasadami określonymi w metodzie.

3. Członek zespołu – jedna z osób bezpośrednio realizująca zadania. Zespoły z założenia powinny być kilkuosobowe, a każdy z członków powinien mieć kompetencje do realizacji wszystkich zadań w sprincie.

Wdrożenie *scrum* jako ram postępowania w organizacji projektu informatycznego wymaga zastosowania kluczowych elementów, takich jak [Dingsøyr, Dyba, Brede Moe 2015; Schwaber, Sutherland 2013]:

1. Sprint – ramy czasowe (stosunkowo niewielkie, ich trwanie określa zespół podczas planowania, najczęściej jest to tydzień lub dwa).

2. SP (*Story Points*) – są to punkty określające „trudność" wykonania zadania (na co składają się między innymi takie elementy, jak: czasochłonność, podatność na błędy, trudność opanowania technologii, w jakiej ma być wykonane zadanie).

3. Backlog produktu (*product backlog*) – lista zadań do realizacji.

4. Kontrakt scrumowy – zbiór zasad zaakceptowanych przez wszystkich członków zespołu *scrum*, określających, w jakim zakresie zespół wykonuje zadania.

Proces zgody z ramami postępowania *scrum* zakłada pracę iteracyjną – przyrostową, z uwzględnieniem elementów koniecznych do pełnego wdrożenia *scrum* w organizacji, takich jak:

1. Planowanie sprintu – przed każdym sprintem odbywa się planowanie polegające na stworzeniu, na podstawie wcześniejszych doświadczeń, planu realizacji zadań wyestymowanych w SP. Tak przygotowany plan powinien obejmować zadania do realizacji na cały sprint (którego czas trwania można dostosowywać do bieżących potrzeb projektu, najczęściej są to jednak około 1 tydzień lub 2 tygodnie). Do planu sprintu powinny trafić zadania, które przeszły proces opisany w punkcie 2. Efektem planowania jest sprint backlog, czyli lista zadań do realizacji w sprincie. Podczas spotkania określa się także cel sprintu.

2. *Product backlog refinement* – spotkanie mające na celu dbanie o jakość backlogu produktu, w ramach którego zespół wytwórczy rozmawia o realizacji zadania, planuje i projektuje jego wykonanie, a także estymuje zadanie w SP.

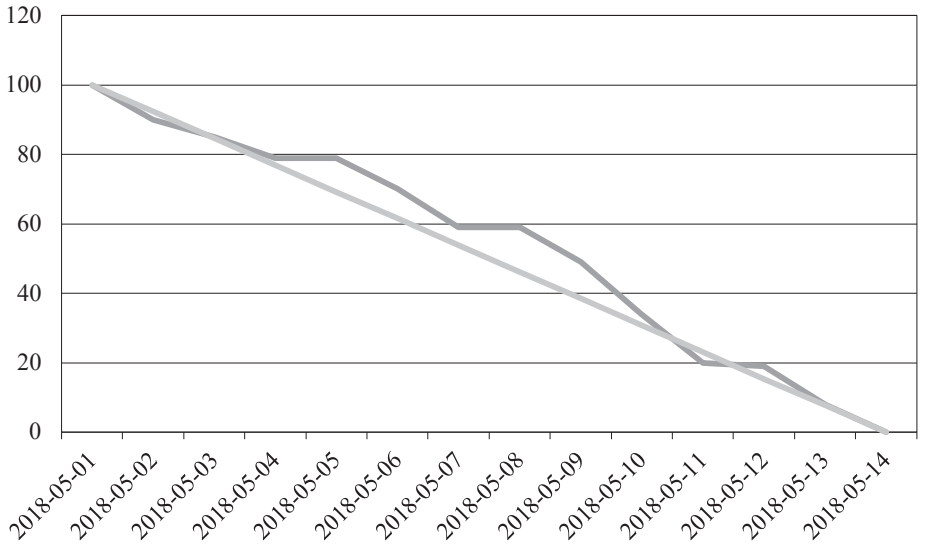
3. *Sprint review* – spotkanie, którego celem jest przedstawienie wykonanych w ramach sprintu zadań w formie wartości dodanej dla klienta (np. działający moduł lub oprogramowany obszar aplikacji). Podczas tego spotkania zespół weryfikuje, czy cel sprintu został osiągnięty.

4. Retrospektywa lub retro (*retrospection*) – podczas tego spotkania zespół rozmawia o minionym sprincie, wyciąga wnioski oraz aktualizuje kontrakt scrumowy tak, aby jego treść odpowiadała założeniom poczynionym przez zespół.

5. *Daily scrum* – codzienne, krótkie spotkanie mające na celu określenie etapu realizacji bieżących zadań i sposobu, w jaki przybliżają nas one do osiągnięcia celu sprintu.

W zasadach postępowania zgodnych ze *scrum* wykorzystuje się również wizualizację zadań. Tablica reprezentująca *sprint backlog* może wyglądać tak samo jak w przypadku tablicy wykorzystywanej w metodyce *kanban* (rys. 1), jednak z zastrzeżeniem, że zadania w sekcji *to do* są planowane na cały sprint przed jego rozpoczęciem, a nie jak w przypadku metody *kanban* pobierane z backlogu produktu na bieżąco, w ramach aktualnych potrzeb. Zaletą pracy w *scrum* jest dokładne przygotowywanie zadań na sprint, co pozwala na dostarczenie na koniec cyklu wycinka całości produktu, który będzie możliwy do odbioru przez klienta i będzie spójną wartością dodaną całości systemu.

Zespoły pracujące zgodnie z ramami postępowania *scrum* obrazują swoją pracę przez wykorzystanie wykresu spalania (*burndown*) (rys. 3).



Rys. 3. Przykładowy wykres spalania obrazujący postępy w realizacji planu

Źródło: opracowanie własne.

Pionowa oś na wykresie oznacza liczbę SP do realizacji przez zespół w ramach sprintu, pozioma zaś jest osią czasu (od dnia rozpoczęcia do dnia zakończenia sprintu). Na wykres nanoszona jest linia, która oznacza oczekiwany spadek SP pozostałych do realizacji w danym dniu oraz na bieżąco uzupełniana jest linia oznaczająca rzeczywiste wartości pozostałych do realizacji SP.

Jednak *agile* przy wykorzystaniu *scrum*, jak każda metodyka, nie jest narzędziem, które może sprawdzić się we wszystkich przedsiębiorstwach. Wadą *scrum* jest ograniczona możliwość reakcji na dużą dynamikę projektu. W przypadku zmian priorytetów w trakcie realizacji zadań nie można zmienić backlogu sprintu. W takich sytuacjach właściciel produktu podejmuje decyzję o przerwaniu sprintu

oraz zaplanowaniu kolejnej iteracji. Powoduje to stratę czasu poświęconego na kolejne spotkanie.

Inną cechą ram postępowania *scrum*, która może spowodować nieużyteczność w niektórych projektach, jest odejście od estymowania zadań w ujęciu czasowym. Zespoły scrumowe wykorzystują SP. Z jednej strony pozwala to na skuteczną ocenę ryzyka dostarczenia danego zadania, jednak potrzebny jest czas i znajomość projektu przez członków zespołu, aby trafnie określić podaż SP na podstawie podaży czasu pracy, jaką członkowie zespołu dysponują.

Dużym zagrożeniem dla wdrożenia metodyki *agile* w przedsiębiorstwie jest też niechęć samego zespołu. W przypadku kiedy członkowie zespołu czują, że inicjatywa nie wychodzi od nich, obowiązkowe do pełnego zastosowania ram postępowania *scrum* spotkania mogą być odbierane jako strata czasu. Przeprowadzenie transformacji w sposobie pracy z metodyk klasycznych na *scrum* wymaga więc dużych miękkich umiejętności zarządczych osób decyzyjnych i trenerów odpowiedzialnych za przygotowanie zespołu do pracy zgodnie z zasadami metody. Niezbędna jest tutaj umiejętność słuchania pracowników oraz ich pełne zaangażowanie we wdrożenie.

Kolejną z cech *scrum*, która powoduje, że nie jest to narzędzie dostosowane do pracy wszystkich zespołów, jest odejście od specjalizacji. Jednym z fundamentalnych założeń tego sposobu pracy jest to, że każdy z członków zespołu ma kompetencje do realizacji wszystkich zadań w sprincie. Oznacza to tyle, że nie sprawdzi się on w zespołach o dużym stopniu specjalizacji (gdzie w zespole współpracują ze sobą jednocześnie programiści, analitycy i testerzy). Aby choć w części wyeliminować wady obu prezentowanych metodyk, autor proponuje ich hybrydyzację, co zostanie przedstawione w punkcie 4.

4. Hybrydyzacja metodyk

Nie istnieją narzędzia bezwzględnie idealne. W projektach informatycznych narzędzie powinno być dobrane do potrzeb danego projektu. Nie ma dwóch takich samych przedsiębiorstw, nie ma dwóch takich samych projektów i nie ma dwóch identycznych zespołów. Wszystkie te różnice powodują, że niemożliwe jest stworzenie ram postępowania w projekcie, które sprawdzą się dla wszystkich uwarunkowań społeczno-gospodarczych [Cockburn 2002, s. 10-12].

Wiele organizacji stosuje tzw. kastomizację metodyki, aby znaleźć rozwiązanie „szyte na miarę". Aby dostosować metodykę do swoich potrzeb, w pierwszej kolejności należy określić cechy otoczenia przedsięwzięcia. W niniejszym artykule zaproponowano hybrydyzację metodyk *scrum* i *kanban*, przy założeniu następujących cech charakterystycznych projektu:

1. Duża zmienność i dynamika projektu, częsta zmiana wymagań, konieczność szybkiej reakcji na zmiany.
2. Zespół jest dostawcą systemu, współpracuje z klientem zewnętrznym.

3. Doświadczony zespół o dużym stopniu specjalizacji jego członków (programiści, analitycy, testerzy, kierownik projektu (PM – *Project Manager*)).

4. Obowiązek pełnienia asysty technicznej (*helpdesk*) przez wszystkich członków zespołu; wymagany jest szybki czas reakcji na zgłoszenie od klienta.

Biorąc pod uwagę powyższe oraz omówione wcześniej cechy prezentowanych ram postępowania, można zaproponować hybrydową metodykę, która powinna cechować się następującymi regułami postępowania:

1. Wizualizacja pracy w formie tablicy elektronicznej. Elektroniczna tablica ma przewagę nad tablicą tradycyjną, polegającą na jej automatycznej aktualizacji podczas realizacji standardowych zadań związanych z raportowaniem czasu pracy oraz planowania. Przy wykorzystaniu wielu z dostępnych na rynku narzędzi ogranicza się czas wykorzystywany na utrzymywanie tablicy tradycyjnej. Ponadto możliwe jest udostępnienie tablicy elektronicznej klientowi, który może na bieżąco śledzić postępy w pracach i określać priorytety dla zadań. Różnica między skastomizowaną metodą a oryginalnymi ramami postępowania, z którego miałaby czerpać inspiracje, polega na tym, że zadania są planowane na cały sprint (jak w *scrum*), jednak możliwe jest wycofywanie zadań z sekcji *to do* na potrzeby pobrania innych zadań z backlogu produktu (jak w metodyce *kanban*).

2. Dzielenie się wiedzą. Reguła ta obejmuje wykorzystanie idei backlog refinementu, podczas którego członkowie zespołu wspólnie planują i projektują sposób realizacji zadań. Należy jednak uwzględnić charakterystykę zespołu. Jak opisane zostało w punkcie 3 założeń projektu informatycznego, omawiany zespół cechuje się dużym poziomem specjalizacji. Oznacza to, że angażowanie analityków i testerów w projektowanie realizacji rozwiązania przez programistów byłoby stratą zasobów. Zamiast tego można wyznaczyć osoby odpowiedzialne za spójność systemu, optymalność rozwiązań oraz testowalność[8] rozwiązania w formie historyjki użytkownika (*user story*[9]). Każda z tych osób, w ramach swojego czasu, przygotowywałaby plan realizacji zadań, a w następnym kroku przedstawiałaby go osobom o tej samej specjalizacji. Podczas takiego spotkania mają oni możliwość zgłoszenia własnych sugestii czy wątpliwości. Spotkanie całego zespołu – *backlog refinement* – ograniczałoby się do przedstawienia właścicielowi produktu, przez wyznaczone osoby odpowiedzialne, całościowego planu realizacji zadania, od etapu analizy przez implementację aż do testów. Nie oznacza to oczywiście całościowego odcięcia pozostałych członków zespołu od wiedzy tak wytypowanych architektów. Dobrą praktyką jest przeznaczenie określonego czasu każdego członka zespołu na rozwój kompetencji, także między specjalnościami. Programiści mogą uczyć testerów automatyzacji. Te-

[8] Testowalność systemu oznacza podział całości zakresu projektu na mniejsze zadania, których przetestowanie podczas testów akceptacyjnych klienta UAT (*User Acceptance Tests*) będzie wykonalne.

[9] *User stories* – historyjki użytkownika pozwalające na podział całości zakresu projektu na mniejsze zadania, przynoszące wartość dodaną. Mają najczęściej formę „Ja jako… (rola w systemie) chcę… (potencjalne działanie systemu), aby… (potrzeba biznesowa). Kryterium akceptacji będzie… (kiedy uznam zadanie za wykonane)".

sterzy mogą uczyć analityków działania systemu z perspektywy użytkownika. Analitycy mogą pomagać testerom i programistom lepiej zrozumieć intencje użytkownika docelowego. Wypracowanie „zespołowego planu rozwoju kompetencji" pozwala na czerpanie korzyści z poszerzającej się wiedzy pracowników, ale także wpływa na morale zespołu, zaspokojenie potrzeby samorealizacji i zdobywania wiedzy.

3. Wykorzystanie roli *Project Managera*. Z definicji zespoły pracujące zgodnie z zasadami *scrum* są samoorganizujące się[10]. Niestety, nie każdy projekt pozwala na taki tryb pracy. W badanym przedsiębiorstwie konieczna jest rola osoby decyzyjnej, która będzie połączeniem scrum mastera oraz kierownika projektu. Osoba ta, z jednej strony, powinna dokonywać ustaleń z klientem (dotyczących terminów oraz priorytetów). Z drugiej zaś – powinna też dbać o poczynione założenia związane z nową, hybrydową metodyką.

4. Zastosowanie *sprint review* do przeprowadzenia testów odbiorowych. Pozwala ograniczyć czas poświęcony na wsparcie (*helpdesk*). W celu realizacji tego punktu organizuje się telekonferencję (z udostępnieniem pulpitu) lub też zaprasza przedstawicieli klienta, wykonujących testy odbiorowe. Podczas takiego spotkania, które ma na celu przedstawienie właścicielowi produktu efektu sprintu, można rozwiać wątpliwości oraz odpowiedzieć na pytania związane z działaniem nowych funkcji systemu.

5. Wprowadzenie roli specjalisty ds. wsparcia użytkownika końcowego. Jak wcześniej wspomniano, zasada redukcji WIP (pracy w toku) jest co do zasady dobrym rozwiązaniem, aby ograniczyć straty czasu związane ze zmianą kontekstu. Jednak w przypadku, gdy istnieją zadania w projekcie wymagające umiejętności pracy wielowątkowej, warto stworzyć stanowisko dla osoby, która taką umiejętność ma. Nie jest to łatwe zadanie i wymaga dużego nakładu pracy ze strony działu HR odpowiedzialnego za rekrutacje. Przygotowanie zestawu zadań, pozwalających na wyłonienie wśród kandydatów osoby z potencjałem do pracy w warunkach wielozadaniowości, nie jest zadaniem łatwym. Stworzenie takiej roli pozwala jednak ograniczyć WIP tam, gdzie to możliwe, przy ograniczeniu realizacji zadań związanych ze wsparciem klienta przez jedną przeznaczoną do tego osobę.

Przedstawione reguły postępowania to tylko początkowy proponowany kształt nowej metodyki zwinnej. Małe rozbudowanie pozwala na ograniczenie niechęci członków zespołu do wprowadzanych zmian. Na koniec każdego cyklu (sprintu) powinna być organizowana retrospektywa. Pozwala to na zrozumienie pracowników oraz daje im pole do wskazania wąskich gardeł procesu i wspólnej pracy nad wykorzystywaną metodą, ponieważ ludzie w zespole, dzięki swojemu doświadczeniu, dysponują ogromnym potencjałem i są skarbnicą wiedzy o mankamentach procesu i możliwościach wprowadzanych usprawnień.

---

[10] Zespoły samoorganizujące się nie mają lidera – przełożonego, który wyznaczałby im zadania. W tym modelu to członkowie zespołu wspólnie decydują o rozdysponowaniu zadań między poszczególne osoby. Szczególnie popularnym modelem wykorzystania samoorganizacji zespołów są turkusowe organizacje.

5. Zakończenie

Obserwując rynek branży informatycznej, można zauważyć, że wiele przedsiębiorstw zwraca uwagę na trendy i próbuje dostosować projekt do wykorzystywanego narzędzia. Powoduje to utratę podstawowej zasady – narzędzie ma służyć realizacji celów i zadań, a nie być celem samym w sobie. Wiele z organizacji, które próbowało wdrożyć książkową formę metodyki, powróciło do starych, tradycyjnych technik. Powodem tego jest przede wszystkim niechęć członków zespołu, związana z narzucaniem im rozwiązań niedostosowanych do otoczenia, w jakim pracują. Brak ich zaangażowania w proces przemiany organizacyjnej i wyboru metodyki negatywnie wpływa na morale zespołu. Podejście zakładające aktywny udział wszystkich pracowników, nie tylko w wybór metodyki, ale także w jej dostosowanie do potrzeb, powoduje, że czują się oni bardziej zaangażowani i odpowiedzialni za efekt końcowy przedsięwzięcia.

Wiele z dostępnych na rynku rozwiązań z obszaru metodyk zarządzania i organizacji projektu jest definiowane przez sztywne ramy postępowania, które odnoszą się do modelowych projektów. Każde odstępstwo od normy w kwestii cech charakterystycznych przedsięwzięcia informatycznego powoduje, że wypracowana metodyka nie znajduje zastosowania. Branża informatyczna rozwija się na tyle dynamicznie, że nie można zakładać statycznego obrazu żadnego z jej obszarów.

Zaproponowana w niniejszym artykule hybrydyzacja metodyk *kanban* i *scrum* pozwala na kastomizację wypracowanych sposobów pracy i organizacji przedsięwzięcia na potrzeby konkretnego projektu i zespołu tak, aby każdy z jego członków mógł brać czynny udział w osiągnięciu rozwiązania „szytego na miarę" projektu. Hybryda, która powstała między innymi w wyniku obserwacji poczynionych w pracy zawodowej autora, różni się od klasycznych metodyk formą. Nie obejmuje ona sztywnych ram postępowania, a jedynie wskazuje kierunki, które pozwolą na samodzielne wypracowanie sposobu organizacji własnej pracy przez zespół. Dzięki temu możliwe jest zastosowanie tej metodyki w każdym, nawet najbardziej nietypowym dynamicznym i turbulentnym środowisku, w którym organizowane jest przedsięwzięcie informatyczne.

Literatura

Abrahamsson P., Salo O., Ronkainen J., Warsta J., 2002, *Agile Software Development Methods. Review and Analysis*, https://www.vtt.fi/inf/pdf/publications/2002/P478.pdf (dostęp: 15.05.2018).

Anderson D., 2010, *Kanban i Scrum – jak uzyskać najlepsze z obu*, C4Media Inc.

Ciesielski M., 2006, *Instrumenty zarządzania logistycznego*, Polskie Wydawnictwo Ekonomiczne, Warszawa.

Cockburn A., 2002, *Learning from Agile Software Development – Part One*, Cross Talk, The Journal of Defense Software Engineering, https://pdfs.semanticscholar.org/1666/0d9dc45aa2ab9631fa9cec480f108957d4ea.pdf (dostęp: 16.05.2018).

Dingsøyr T., Dyba T., Brede Moe N., 2015, *Agile Software Development*, Springer, Heidelberg–Dordrecht–London–New York.

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

Koszlajda A., 2010, *Zarządzanie projektami IT – przewodnik po metodykach*, Helion, Gliwice.

Krasiński M., 2013, *Możliwość zastosowania metodyki Kanban w zarządzaniu projektami*, Nauki o Zarządzaniu, 1(14).

Li J., 2012, *Agile Software Development*, Technische Universitt Berlin, Berlin https://www.snet.tu-berlin.de/fileadmin/fg220/courses/WS1112/snet-project/agile-softwaredevelopment_li.pdf (dostęp: 20.05.2018).

Ordysiński T., 2014, *System Kanban w administracji publicznej*, Roczniki Kolegium Analiz Ekonomicznych, Szkoła Główna Handlowa, nr 33, Warszawa.

Schwaber K., Sutherland J., 2013, *Scrum Guide*, http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-pl.pdf (dostęp: 10.06.2018).

The 12th State of Agile Report, Collabnet Versionone, dostęp 2018-05-10, https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report.

Wieczorkowski J., 2016, *Strategia informatyzacji i analiza przedwdrożeniowa a cykl życia oprogramowania standardowego*, Zeszyty Naukowe Politechniki Częstochowskiej, Zarządzanie, nr 23 (t. 1).

Wyrozębski P., 2011, *Zwinne metodyki zarządzania projektami*, [w:] *Metodyki zarządzania projektami*, Bizarre, Warszawa.

Zamostny B. (red.), 2012, *Optymalizacja produkcji w czasie kryzysu*, Wydawnictwo Wiedza i Praktyka, Warszawa.