Od procesów do oprogramowania: badania i praktyka

Redakcja naukowa:

Piotr Kosiuczenko
Michał Śmiałek
Jakub Swacha

Publikacja została dofinansowana

przez Ministra Nauki i Szkolnictwa Wyższego

w ramach programu związanego z realizacją zadań
upowszechniających naukę

(decyzja nr 530/P-DUN/2015 z dnia 27.01.2015)

POLSKIE TOWARZYSTWO INFORMATYCZNE

Od procesów do oprogramowania: badania i praktyka

Redakcja naukowa:

Piotr Kosiuczenko
Michał Śmiałek
Jakub Swacha

Warszawa 2015

Recenzenci

Zbigniew Banaszak, Włodzimierz Bielecki, Leszek Borzemski, Krzysztof Cetnarowicz, Zbigniew Czech, Włodzimierz Dąbrowski, Mariusz Flasiński, Janusz Górski, Adam Grzech, Piotr Habela, Bogumiła Hnatkowska, Zbigniew Huzar, Stanisław Jarząbek, Leszek Maciaszek, Jan Madey, Lech Madeyski, Karolina Muszyńska, Jerzy Nawrocki, Mirosław Ochodek, Aneta Poniszewska-Maranda, Piotr Poprawski, Mirosław Staroń, Andrzej Stasiak, Krzysztof Stencel, Zdzisław Szyjewski, Bartosz Walter, Andrzej Wasowski, Krzysztof Wnuk, Janusz Zalewski, Krzysztof Zielinski

Redakcja naukowa

Piotr Kosiuczenko, Michał Śmiałek, Jakub Swacha

Autorzy

Jakub Swacha – ROZDZIAŁ 1
Tomasz Protasowicki, Jerzy Stanik – ROZDZIAŁ 2
Jarosław Napiórkowski, Jerzy Stanik – ROZDZIAŁ 3
Bogusz Jeliński – ROZDZIAŁ 4
Rafał Wojszczyk – ROZDZIAŁ 5
Lech Tuzinkiewicz, Emilia Zakrawacz – ROZDZIAŁ 6
Bożena Śmiałkowska – ROZDZIAŁ 7
Monika Łobaziewicz – ROZDZIAŁ 8
Lucjan Pelc – ROZDZIAŁ 9
Dawid Karabin, Ziemowit Nowak – ROZDZIAŁ 10

Redakcja techniczna

Tomasz Klasa

Projekt okładki

Łukasz Piwowarski

ISBN 978-83-60810-73-6

Wydanie: I. Nakład: 200 egz.
Wydawca: Polskie Towarzystwo Informatyczne
Druk i oprawa: PPH ZAPOL, al. Piastów 42, 71-062 Szczecin

Spis treści

Wstęp

Inżynieria oprogramowania jako dyscyplina inżynierska powstała w latach sześćdziesiątych XX wieku w odpowiedzi na tzw. kryzys oprogramowania spowodowany znaczącym wzrostem złożoności oprogramowania. W literaturze pojawia się nawet teza, że tworzenie oprogramowania jest najbardziej złożoną działalnością intelektualną w historii ludzkości. Odpowiedzią na tą złożoność są nowe metody wytwarzania i pielęgnacji oprogramowania, nowe języki modelowania oraz nowe narzędzia wspomagające.

Niestety, wspomniany kryzys nigdy do końca nie został przezwyciężony, czego przyczyną jest stale rosnąca złożoność oprogramowania, a objawem częste informacje o niepowodzeniach projektów informatycznych. Powstające systemy oprogramowania często nie spełniają kryteriów jakościowych, a ponadto przekraczają przeznaczone na nie budżety i terminy. W związku z tym bardzo istotne stają się badania związane z metodami wytwarzania systemów informatycznych, optymalnego zarządzania cyklem życia oprogramowania oraz zapewnienia jego jakości. W praktyce inżynierskiej stosowane są różne techniki i technologie, których zadaniem jest wzrost wydajności w tworzeniu oprogramowania.

Niniejsza monografia przedstawia nowe wyniki badań i nowe praktyki dotyczące wytwarzania, eksploatacji i rozwoju oprogramowania. Podstawową osią monografii jest zarządzanie cyklem życia oprogramowania od wczesnego etapu kształcenia zespołów programistów do ostatecznego etapu zarządzania wytworzonym wcześniej kodem. Monografia eksponuje również kwestie zapewnienia jakości oprogramowania. Jest to bardzo istotny element całego cyklu życia oprogramowania. Warto podkreślić, że dobrej jakości system wymaga praktycznego zastosowania odpowiednich technik i procesów oraz odpowiedniego zarządzania całym cyklem wytwarzania oprogramowania. W ostatniej części monografii przedstawiamy wybrane zagadnienia w tym obszarze.

Część I monografii dotyczy zagadnień zarządzania cyklem życia oprogramowania. Rozdział 1 rozpoczyna się od przedstawienia wstępnego etapu tego cyklu, czyli kształcenia programistów. Przedstawiona tu jest jedna z ciekawych metod nauczania programowania polegająca na wykorzystaniu techniki gamifikacji. Gamifikacja polega na „tworzeniu doświadczeń przypomina-

jących gry w kontekstach nie będących grami". Jedną z jej kluczowych zalet stanowi pobudzanie motywacji wewnętrznej, co jest szczególnie istotne dla nauczania programowania, które często charakteryzuje się niewielką skutecznością. W rozdziale przytoczono definicje gamifikacji, opisano jej typowe elementy i omówiono przesłanki uzasadniające wykorzystanie gamifikacji w nauczaniu programowania. Kluczowa część rozdziału zawiera przegląd literatury na temat wykorzystania gamifikacji w inżynierii oprogramowania oraz przegląd dostępnych w Internecie zgamifikowanych kursów programowania i narzędzi wspomagających tego rodzaju naukę programowania.

Rozdziały 2 i 3 są poświęcone metodom zarządzania ryzykiem w cyklu życia oprogramowania. W rozdziale 2 dokonano przeglądu odpowiednich standardów zarządzania ryzykiem podczas wytwarzania systemów informatycznych. Z kolei w rozdziale 3 przedstawiono koncepcję metodyki analizy i zarządzania ryzykiem podczas wdrażania i eksploatacji takich systemów. W obydwu rozdziałach przedstawione zostały modele zarządzania ryzykiem oraz próby usystematyzowania tych zagadnień wraz z określeniem zaleceń dotyczących stosowania proponowanych metod w praktyce.

Rozdział 4 dotyczy zagadnienia zarządzania kodem źródłowym. Celem tego rozdziału jest przedstawienie mechanizmów utraty kontroli nad kodem źródłowym i wskazanie tych elementów kodu, które powinny być szczególnie chronione. Zawarto w nim także propozycję procesu zarządzania kodem źródłowym, zapewnienia jego dobrej jakości oraz pomiaru jego dojrzałości, w oparciu o doświadczenia dużej firmy telekomunikacyjnej.

Zarządzanie jakością kodu i zapewnienie jej to temat części II monografii. Rozdział 5 jest poświęcony jakości oprogramowania w kontekście wzorców projektowych. Wzorce projektowe są szeroko opisywane w literaturze; doczekały się różnych ujęć i wariantów. Pojawia się jednak problem weryfikacji poprawności ich implementacji. Celem rozdziału jest przedstawienie propozycji metody, która umożliwiałaby weryfikację poprawności implementacji struktury wzorców projektowych względem abstrakcyjnego modelu referencyjnego.

W rozdziale 6 zostały przedstawione zagadnienia jakości oprogramowania w kontekście jakości danych. Rozdział odpowiada na pytanie o mierzenie i ocenę jakość danych. W rozdziale zaproponowano odpowiedni model oceny, opracowany na podstawie normy jakości ISO 25012, oraz przedstawiono przykład wykorzystania tego modelu do oceny jakości przykładowego zbioru danych. Model ten uwzględnia zarówno techniczne, jak i biznesowe

kryteria akceptowalności danych pod względem jakościowym. Wzięto w nim pod uwagę zarówno perspektywę obiektywną, jak i subiektywną. Stosowanie modelu ułatwia odpowiednie narzędzie opracowane przez autorów rozdziału.

Rozdział 7 przedstawia również problemy zapewnienia jakości, ale w kontekście hurtowni danych. Brane są tu pod uwagę zmieniające się w czasie wymagania użytkowników takich hurtowni. Istotną rolę odgrywa możliwość oceny hurtowni pod względem użyteczności, funkcjonalności i przydatności do wspomagania działań organizacji przy zmiennych w czasie potrzebach informacyjnych, nowych źródłach zasilania danymi lub zmiennej strategii biznesowej organizacji. W rozdziale zaproponowano metodę oceny modelu danych, operacji dostępnych w hurtowni danych a także możliwości adaptacji hurtowni danych do nowych zmiennych potrzeb informacyjnych organizacji. Metoda jest oparta na ocenie wielokryterialnej i syntetycznej.

W ostatniej, III części monografii, zebrano kilka przykładowych zastosowań technologii inżynierii oprogramowania, podnoszących jakość powstającego oprogramowania. Rozdział 8 dotyczy zagadnienia standardów architektonicznych oprogramowania. Przedstawiono tutaj wyniki projektu badawczo–rozwojowego, którego celem jest opracowanie nowej platformy B2B wspomagającej zarządzanie procesami budowlanymi. Efektem jest powstanie standardu wielowarstwowego modelu architektonicznego dla tego typu systemów o nazwie OPTIbud. Założenia do budowy systemu B2B OPTIbud zostały oparte na koncepcji architektury otwartej, pozwalającej na integrację z systemami i aplikacjami, z których korzystają firmy budowlane.

Inne podejście do tworzenia modeli oprogramowania jest zaprezentowane w rozdziale 9. Zastosowano tutaj podejście formalne, oparte na sieciach Petriego. Zaprezentowano nowy typ takich sieci i przeprowadzono analizę jego praktycznych zastosowań. Rozdział przedstawia wyniki prac, które miały na celu zbadanie, czy sieć parametryzowana może zastąpić inny rodzaj sieci Petriego oraz wskazanie korzyści, które taka zamiana potencjalnie przynosi. Opracowane przykłady zastosowania sieci parametryzowanej zostały zaimplementowane w przemysłowym sterowniku PLC.

Monografię zamyka rozdział 10, który zajmuje się cechami jakościowymi aplikacji w obszarze ich wydajności. Dokonano tutaj analizy wydajności aplikacji webowych skonstruowanych przy pomocy różnych technologii opartych na języku Java. Badania polegały na analizie porównawczej dwóch aplikacji o identycznej funkcjonalności. Skoncentrowano się tutaj na pomiarach czasu renderowania widoków oraz czasu wykonywania kodu JavaScript.

Badania przeprowadzono uwzględniając różne rodzaje urządzeń klienckich, takich jak laptop, smartfon czy tablet, oraz różne modele dostępu sieciowego.

Piotr Kosiuczenko
Michał Śmiałek
Jakub Swacha

I. Zarządzanie tworzeniem oprogramowania

Rozdział 1

Gamifikacja w nauczaniu programowania: przesłanki i dostępne rozwiązania

1. Wprowadzenie

Wyniki licznych badań wskazują na trudność nauki programowania komputerów (patrz np. [33] i prace tam cytowane). W rezultacie wielu uczestników kursów programowania komputerów (także w ramach studiów wyższych) uzyskuje po ich zakończeniu jedynie niski poziom umiejętności programistycznych (zob. np. wyniki międzynarodowych badań wśród studentów informatyki [27]). Co więcej, ze względu na złożoność, rozległość i dynamiczny rozwój metod i języków programowania, dla profesjonalnego programisty nauka programowania nie kończy się wraz z jednorazowym ukończeniem kursu, lecz trwa zwykle przez całe życie. Przykładowo, według Leona E. Winslowa, potrzeba około dziesięciu lat, by początkujący programista stał się ekspertem [43] – w tym samym czasie dokonuje się często olbrzymia zmiana rodzaju oprogramowania, na które występuje na rynku zapotrzebowanie, związanych z tym technologii oraz adekwatnych do nich języków i bibliotek programistycznych. Aspekt ten dodatkowo zwiększa wartość rozwiązań prowadzących do uczynienia nauki programowania skuteczniejszą.

W niniejszym rozdziale podjęto zagadnienie możliwości wsparcia nauki programowania poprzez wykorzystanie gamifikacji. Jako że każde rozważania wymagają wcześniejszego uporządkowania pojęć, szczególnie gdy, tak jak w tym przypadku, są one stosunkowo nowe i bywają interpretowane na różne sposoby, drugi po niniejszym wprowadzeniu podrozdział poświęcono zdefiniowaniu pojęcia gamifikacji i opisaniu elementów, które zazwyczaj obejmuje praktyczna implementacja gamifikacji. W podrozdziale trzecim bliżej opisano przesłanki, które uzasadniają wykorzystanie gamifikacji w nauczaniu programowania.

Kolejny podrozdział poświęcono praktycznemu aspektowi wdrożenia gamifikacji w nauczaniu programowania. Omówiono w nim najpierw literaturę opisującą doświadczenia z wykorzystania gamifikacji w nauczaniu programowania i inżynierii oprogramowania, a następnie szereg dostępnych w Internecie zgamifikowanych rozwiązań edukacyjnych przeznaczonych do wspomagania nauki programowania. Rozdział kończy krótkie podsumowanie.

2. Gamifikacja

2.1. Pojęcie gamifikacji

Termin *gamification* pojawił się w języku angielskim w roku 2002 [4, s. 5]; na język polski tłumaczony jest zazwyczaj jako *gamifikacja*, choć pojawiły się także alternatywne przekłady: *grywalizacja* [42] i *gryfikacja* [20].

Mimo tak krótkiej historii, gamifikacja doczekała się już wielu, nie w pełni zbieżnych, definicji: w opartej na literaturze analizie Andrzeja Marczewskiego pojawia się ich aż trzydzieści, i nie jest to lista wyczerpująca [25]. Konkluzją wspomnianej analizy jest zwięzłe zdefiniowanie gamifikacji jako „tworzenie doświadczeń przypominających gry w kontekstach nie będących grami", choć zaproponowano także dłuższy wariant: „zorientowane na użytkownika zastosowanie elementów gier, mechaniki gier, projektów gier lub myślenia typowego dla gier w kontekstach nie będących grami w celu zaangażowania, motywowania, zmiany zachowania, rozwiązywania problemów, ułatwiania osiągania celów, czynienia zadań bardziej wesołymi oraz dla dodania zabawy" [25].

2.2. Elementy gamifikacji

Elementy gamifikacji to wykorzystywane w niej rozwiązania zaczerpnięte z gier. Istnieją różne typologie elementów gamifikacji (szczególnie rozwiniętą przedstawiono np. w pracy [24]), ze względu na stosunkową zwięzłość, w tym podrozdziale omówiona zostanie typologia zaproponowana przez Maika i Silvię Schachtów [35]. Jej opisanie ma na celu przybliżenie Czytelnikowi tego, co zwykle obejmuje gamifikacja, niemniej należy mieć na uwadze, że nie jest to typologia kompletna, i nie obejmuje wszystkich elementów, w tym także niektórych występujących w rozwiązaniach opisanych w dalszej części rozdziału.

Schachtowie wyróżnili dwie główne grupy elementów: (A) przeznaczone do bezpośredniej implementacji w grze (systemie zgamifikowanym) oraz (B) odnoszące się do emocji gracza (użytkownika systemu). Wskazali także drugie kryterium, za które obrali cel użycia danego elementu, którym może być: (1) ukazanie postępu gracza (użytkownika systemu); (2) zapewnienie graczowi (użytkownikowi systemu) informacji zwrotnej; oraz (3) zaangażowanie gracza (użytkownika systemu) w specyficzne zachowania [35].

Uwzględniając oba powyższe kryteria, do grupy A1 przypisali oni:

- punkty – wirtualne nagrody przypisane do ukończenia poszczególnych zadań i pozwalające graczowi (użytkownikowi systemu) wartościować możliwe kierunki działania (poprzez zróżnicowanie przez projektanta gry/systemu wielkości nagród punktowych przysługujących za różne rodzaje aktywności);
- premie – dodatkowe nagrody punktowe, wprowadzające element losowości (premie – niespodzianki) i/lub nagradzające specyficzną formę lub czas wykonania zadania;
- poziomy – odnotowujące przekroczenie przez gracza (użytkownika systemu) istotnych progów rozwoju; osiągnięcie przez gracza (użytkownika systemu) wyższego poziomu zazwyczaj odblokowuje dostęp do kolejnej puli trudniejszych zadań;

- wskaźnik postępu – pokazujący nie tylko dotychczasowy dorobek punktowy, ale i dystans dzielący gracza (użytkownika systemu) do kolejnego poziomu;
- osiągnięcia – zazwyczaj wizualizowane w formie odznak, które potwierdzają wykonanie przez gracza kluczowych dla osiągnięcia celu lub szczególnie trudnych zadań; ich zestaw tworzy swego rodzaju historyczny rysopis gracza (użytkownika systemu), kształtując jego reputację w społeczności (w dobrze zaprojektowanych systemach gracze reprezentujący różne podejścia będą zdobywali różne zestawy odznak).

Elementy zaliczone przez Schachtów do grupy A2 to:

- umówione terminy – za sprawą których gracz (użytkownik systemu) może uzyskać dostęp do nowych zadań lub zostać nagrodzony za ich wykonanie dokładnie w określonym czasie;
- znikające nagrody – ograniczające przedział czasowy, w jakim określone zadania dostępne są do wykonania, lub ich ukończenie jest nagradzane (premią lub w ogóle);
- licznik czasu – dobitnie informujący o nadchodzącym terminie zniknięcia nagrody;
- rankingi – ukazujące relatywny postęp gracza (użytkownika systemu) względem innych.

Elementy zaliczone przez Schachtów do grupy A3 to:

- współpraca w ramach społeczności – wymagana do realizacji zadań niemożliwych do wykonania przez jedną osobę;
- wiralność – nagradzająca graczy (użytkowników systemu) za skuteczne zapraszanie do uczestnictwa w grze (zgamifikowanym systemie) innych osób.

Schachtowie nie wskazali ani jednego elementu, który miałby przynależeć grupie B1, i tylko pojedynczy element grupy B2: informacja kaskadowa – polegający na przekazywaniu graczowi (użytkownikowi systemu) informacji w niewielkich porcjach, tak by zapewnić jak najlepsze jej zrozumienie. Z kolei dla grupy B3 wymienili:

- zazdrość – wzbudzająca w graczach (użytkownikach systemu) chęć zdobycia tego, co posiadają już inni (wymaga pewnego poziomu widoczności osiągnięć innych graczy);
- lęk przed utratą – stanowiący osobny powód do kontynuowania rozgrywki w celu uniknięcia straty dotychczasowych zdobyczy;
- darmowy lunch – nagradzający graczy (użytkowników systemu) za osiągnięcia innych graczy (z oczywistych powodów projektanci powinni korzystać z tego elementu z umiarem);
- epickość – nadająca działaniom i sukcesom graczy (użytkowników systemu) aury wielkości i wyjątkowości wyrastającej ponad szarą codzienność.

3. Przesłanki dla wykorzystania gamifikacji w nauczaniu programowania

Dostrzeżenie trudności nauki programowania komputerów przez międzynarodowe środowisko naukowe zaowocowało podjęciem wysiłków badawczych nad zidentyfikowaniem przyczyn i znalezieniem środków zaradczych dla tego problemu. Badania te przyniosły z biegiem lat liczne rezultaty. Wśród osiągnięć dotyczących identyfikacji przyczyn należy wymienić, m.in., sklasyfikowanie typowych błędów pojęciowych popełnianych przez studentów[1] [15], zdiagnozowanie barier w nauczaniu programowania [34], czy zidentyfikowanie skutecznych sposobów radzenia sobie przez studentów z zastojami w nauce [26]. Z kolei dążąc do zmniejszenia poziomu trudności związanego z nauką programowania, zaproponowano, m.in., uwzględnienie preferencji studentów przy wyborze nauczanego języka programowania [29], bardziej dopasowane do specyfiki przedmiotu metody [36] i techniki [12] nauczania, dostosowane do potrzeb edukacyjnych środowiska programistyczne [40] i łatwiejsze w odbiorze materiały dydaktyczne [19], a wreszcie całościowe uwzględnienie szeregu takich rozwiązań w realizowanym programie nauczania [37].

Kolejnym kierunkiem badań było zwiększenie zaangażowania studentów w naukę programowania. Wyniki dotychczasowych badań wskazują na pozytywną korelację miedzy wewnętrzną motywacją studentów a skutecznością nauczania ich programowania ([2] i prace tam cytowane). Aby wzbudzić ten rodzaj motywacji próbowano sięgać po różne rodzaje środków, nawet tak oryginalne jak poszerzenie nauczanych treści o elementy innych przedmiotów [9], czy włączenie w naukę programowania tańca [16].

Istotne miejsce w tej galerii środków zajmuje wykorzystanie gier. Podstawowym sposobem zrobienia tego jest posłużenie się grami jako tematem projektów programistycznych realizowanych przez studentów [39]. Wskazano zalety takiego rozwiązania, zarówno w przypadku zastosowania go na samym początku kursu programowania [21] (w celu rozpoczęcia nauki w kontekście znanym studentom i interesującym dla większości studentów), jak i w dalszej jego części [37] – by podtrzymać zainteresowanie nauką programowania w momencie, gdy poziom trudności podejmowanych zagadnień znacząco rośnie, a jednocześnie można pełniej wykorzystać szeroki zakres problemów związanych z programowaniem gier (m.in. zaawansowane algorytmy i struktury danych niezbędne do implementacji elementów mechaniki gier, programowanie efektów graficznych i dźwiękowych, sztuczna inteligencja, optymalizacja wydajnościowa programów), jako że studenci osiągnęli już poziom podstawowy umiejętności programistycznych.

Zupełnie nowe możliwości na tym polu otwiera jednak gamifikacja, która wybiórczo korzystając z rozwiązań znanych z gier, wspiera motywację wewnętrzną, umożliwiając pokonywanie wyzwań (także współpracując w grupie), zaspokajanie ciekawości, a także wzbogacenie nauki o aspekt fantastyczny, a także dając studentom poczucie sprawowania kontroli, zadowolenie z sukcesu, bogactwa, pozycji, czy sławy, którego może im brakować w życiu codziennym (por. [23]). Wspomnieć też należy o będących podstawowymi elementami gamifikacji punktach, odznaczeniach i

[1] Większość omawianych w tym rozdziale wyników dotyczy badań realizowanych w kontekście akademickim. Dla zachowania spójności i przejrzystości wywodu do końca niniejszego rozdziału osoba będąca odbiorcą kursu określana będzie konsekwentnie terminem *student*. Należy mieć jednak na uwadze, że tak rezultaty omawianych badań, jak i zawarte w niniejszym opracowaniu wnioski, w przeważającym stopniu odnoszą się także do innych niż akademicki kontekstów nauki programowania (szkoły niższego szczebla, szkolenia zawodowe, kursy dla amatorów, itd.).

rankingach. Traktowane są one głównie jako bodziec pobudzający do współzawodnictwa, a zatem oddziaływujący na motywację zewnętrzną, której pozytywnego wpływu na naukę programowania jak dotąd nie potwierdzono. Jednakże stanowią także doskonałe mierniki postępu w nauce, dostarczając wielce wartościowej informacji tak osobie prowadzącej kurs, jak i samemu studentowi, a jak pokazują to rezultaty badań J.H. Junga i współpracowników, dostępność informacji zwrotnej o efektywności własnych działań może prowadzić do poprawy ich rezultatów [14].

4. Wykorzystanie gamifikacji w nauczaniu programowania

4.1. Przegląd literatury na temat wykorzystania gamifikacji w nauczaniu programowania i inżynierii oprogramowania

Jakkolwiek wykorzystywanie gamifikacji w edukacji staje się coraz bardziej popularne [11], przypadki zastosowania jej przy nauczaniu programowania są jeszcze nieliczne. Tym samym wielką wartość poznawczą dla dydaktyków zainteresowanych wprowadzeniem elementów gamifikacji w prowadzonych przez siebie kursach programowania stanowią dostępne publikacje opisujące jednostkowe doświadczenia zespołów, które takiej implementacji już dokonały.

Swapneel Sheth i współpracownicy opracowali prototypową platformę HALO, w której proces testowania oprogramowania wzbogacili o elementy gamifikacji, dodając w szczególności warstwę fabularną, zadania, których ukończenie prowadzi do zdobycia punktów, a te z kolei podwyższają z czasem poziom gracza, a także osiągnięcia (tytuły przyznawane użytkownikom). Studenci, którzy zdecydowali się korzystać z platformy wykazali znaczący postęp w nauce [36].

Cen Li z zespołem wykorzystali elementy gamifikacji, m.in. takie jak wyzwania (w postaci zadań programistycznych), punkty, poziomy i rankingi do zwiększenia zaangażowania studentów informatyki w korzystanie z opracowanego przez nich środowiska wspomagającego wspólną naukę PeerSpace, uzyskując pozytywne wyniki (przykładowo, grupa poddana eksperymentowi wygenerowała niemal trzykrotnie więcej wypowiedzi na forum dyskusyjnym aniżeli grupa kontrolna) [22].

Paul Neve i współpracownicy dokonali rozszerzenia używanego na Uniwersytecie Kingston środowiska nauczania programowania NoobLab o wybrane elementy gamifikacji (m.in. odznaki przyznawane za pomyślne ukończenie zadań, punkty i ranking wewnątrzgrupowy), co spotkało się z pozytywną oceną uczestniczących w eksperymencie studentów [30].

Antti Knutas z zespołem wykorzystali zgamifikowany system dyskusji online w celu zmotywowania studentów kursu programowania do wzajemnej pomocy. Udało się w ten sposób zwiększyć współpracę między studentami, zmniejszyć czas oczekiwania na odpowiedzi na pytania zadawane przez studentów i generalnie poprawić efektywność komunikacji w grupie [17].

Jenilyn L. Agapito i współpracownicy postawili sobie za cel opracowanie zgamifikowanej metody oceny studentów podstaw programowania, a plonem ich pracy jest system Xiphias obejmujący m.in. osiągnięcia i odznaki, tablicę wyników i przekazywanie informacji zwrotnej [1].

Krzysztof Jassem i Bartosz Piskadło wykorzystali w kursie inżynierii oprogramowania opracowywany przez siebie zgamifikowany system CyberAcademy, obejmujący m.in. takie elementy gamifikacji, jak: wyzwania, warstwa fabularna,

wskaźniki postępu, wirtualne dobra i odznaki. Został on z zainteresowaniem przyjęty przez studentów: mając do wyboru klasyczną i zgamifikowaną formę kursu, ponad 80% spośród nich wybrało tę drugą opcję [13].

Johanna Pirker i współpracownicy wprowadzili elementy gamifikacji (obejmujące m.in. punkty, odznaki i rankingi) do opartego na systemie Moodle kursu wyszukiwania informacji, który obejmował jednak zadania programistyczne dotyczące algorytmów indeksowania i wyszukiwania. Spotkało się to z umiarkowanie pozytywną oceną studentów, którym spodobał się system punktacji, rankingi podzieliły ich na dwie zbliżone wielkością grupy o sprzecznych opiniach, natomiast odznaki uznali w większości za nieciekawe i nieistotne [32].

Niepowodzeniem natomiast zakończyła się gamifikacja kursu inżynierii oprogramowania przeprowadzona przez Kay Berkling i Christopha Thomasa na uniwersytecie Badenii-Wirtembergii. Mimo wykorzystania m.in. takich elementów gamifikacji jak punkty, poziomy, wyzwania, natychmiastowa informacja zwrotna i rankingi, pod koniec semestru tylko 13% studentów uczestniczących w eksperymencie zdołało przerobić materiał niezbędny do zdania egzaminu [3].

Warto zauważyć, że z uwagi na trudność implementacji gamifikacji bez informatycznego jej wspomagania (wynikającą z konieczności m.in. weryfikacji licznych reguł, precyzyjnego zliczania punktów, prowadzenia rankingów, czy dostarczania błyskawicznej informacji zwrotnej), większość opisanych doświadczeń obejmowało opracowanie nowego systemu wspomagającego nauczanie lub poszerzenie dotychczas wykorzystywanego o wybrane elementy gamifikacji. Z pewnością wskazuje to na potrzebę opracowania narzędzia, które w kompleksowy sposób wspomagałoby przeprowadzenie gamifikacji kursu programowania. Prace nad spełniającym takie wymagania rozwiązaniem są prowadzone przy udziale niniejszego autora [38].

4.2. Przegląd dostępnych w Internecie zgamifikowanych kursów programowania i narzędzi wspomagające naukę programowania

Alternatywą dla samodzielnej gamifikacji własnego kursu programowania jest sięgnięcie po gotowe rozwiązania dostępne w Internecie. Najprostszy sposób to wykorzystanie w całości istniejących zgamifikowanych kursów programowania. Wiąże się to jednak z wieloma ograniczeniami, w szczególności brakiem możliwości swobodnego wyboru przez prowadzącego kurs nie tylko nauczanych treści i ich formy (w tym nawet języka, w którym prowadzony jest kurs), ale i elementów gamifikacji i sposobu ich implementacji.

Gotowe zgamifikowane kursy internetowe można także wykorzystać jako pomoce dydaktyczne jedynie wspomagające i uzupełniające główny proces nauki, i to niezależnie od tego, czy zamierza się dokonać gamifikacji całego własnego kursu, czy też nie: w tym pierwszym przypadku niezbędna jest agregacja osiągnięć w poszczególnych wykorzystywanych komponentach w celu ustalenia oceny ogólnej studenta, w drugim zaś mogą być one traktowane tak jak lektury nadobowiązkowe: jako zalecany studentom środek na poszerzenie ich wiedzy i umiejętności, przy pozostawieniu decyzji o skorzystaniu z niego samym studentom.

Zanim omówione zostaną rozwiązania zaprojektowane w duchu gamifikacji, warto zauważyć, że pewne narzędzia wspomagające naukę programowania, nawet historycznie starsze od samego terminu gamifikacja, posiadają jej wybrane elementy. Przykładowo, w niektórych systemach automatycznej oceny poprawności programów,

jak np. polskim Sphere Online Judge [18] znaleźć można np.: wyzwania (lista otwartych zadań programistycznych), umówione terminy (turnieje ograniczone czasowo), natychmiastową informację zwrotną (wynik automatycznej oceny poprawności zgłoszonego rozwiązania), punkty (wyrażające liczbę poprawnych i niepoprawnych zgłoszeń) oraz oparte na nich rankingi.

Rozwinięciem koncepcji systemów automatycznej oceny poprawności programów o elementy dydaktyczne (w postaci kompendium podstaw programowania w języku C# i zasad programowania kontraktowego) oraz ograniczone wsparcie dla nauczycieli (poprzez możliwość podglądu osiągnięć własnych uczniów) stanowi platforma Pex4Fun udostępniona w roku 2010 przez Microsoft Research [44].

Wynikiem dalszego rozwijania platformy Pex4Fun i jej faktycznym następcą jest platforma CodeHunt [41], w której zadania programistyczne zorganizowano w sektory i poziomy o rosnącej trudności (nadając całemu kursowi formę gry). Zwiększono także interaktywność procesu weryfikacji poprawności zgłaszanych rozwiązań, w szczególności dostarczając studentowi bardziej szczegółowych wskazówek w zakresie koniecznych poprawek. W rezultacie, o ile Pex4Fun miała charakter zestawu zagadek programistycznych, o tyle CodeHunt stała się swego rodzaju samouczkiem programowania.

Taką formę od swego zarania miała Code School [7]. Na platformie dostępne są kursy specjalistyczne obejmujące m.in. języki programowania (Ruby, JavaScript), technologie webowe (HTML, CSS), platformy systemowe (iOS), czy narzędzia użyteczne przy wytwarzaniu oprogramowania (Git). Code School stawia przed uczestnikami wyzwania, których ukończenie daje im punkty, które nie tylko służą uzyskaniu przez uczestnika wyższego poziomu i związanych z nim odznaczeń, ale które także mogą być wykorzystane jako środek płatniczy do zakupu podpowiedzi, pomocnych w rozwiązaniu trudnych zadań.

Najbardziej popularną spośród omawianych platform jest Codecademy – już w kwietniu 2014 r. miała ona ponad 24 miliony użytkowników [8]. W chwili pisania tych słów na platformie dostępne były kursy podstaw programowania w kilku językach oraz co najmniej dwadzieścia kilka kursów specjalistycznych (dotyczących np. posługiwania się specyficznymi interfejsami programistycznymi). Jej elementy gamifikacji obejmują m.in. nagradzanie punktami za poprawne wykonane ćwiczenia oraz odznakami za ukończenie określonych kursów lub lekcji, liczbę kolejnych dni nauki, czy osiągnięcie określonej liczby punktów.

Jakkolwiek od niedawna Codecademy współpracuje z nauczycielami, udostępniając im materiały edukacyjne oraz umożliwiając śledzenie postępów uczniów, oryginalnie pomyślana była jednak jako miejsce zdobywania i doskonalenia umiejętności przez osoby pracujące i poszukujące pracy w branży programistycznej. Platformą stworzoną z myślą o wspieraniu nauczania programowania w szkołach jest natomiast Code Avengers [5], gdzie treść kursów przygotowana została z myślą o nastolatkach jako odbiorcach i ilustrowana jest grafiką utrzymaną w konwencji komiksowej.

Do jeszcze młodszych dzieci adresowany jest kurs programowania udostępniony przez Khan Academy. Do zawartych w nim elementów gamifikacji zaliczyć należy „Mapę Wiedzy" pokazującą w atrakcyjny graficznie sposób zestaw zagadnień do opanowania przez użytkownika, punkty energii uzyskiwane za ukończenie zadań, odznaki przyznawane użytkownikowi za osiągnięcie kolejnych poziomów, i którymi może on pochwalić się na swojej stronie profilowej w portalu Facebook, cele (zestawy zadań do wykonania) oraz różnego typu wskaźniki postępu [28].

O ile wymienione wyżej platformy charakteryzują się brakiem warstwy fabularnej – a w najlepszym razie szczątkową jej formą – o tyle dostępny także w języku polskim kurs programowania Code Combat [6] jest w zasadzie grą przygodową (a zatem stanowi połączenie gamifikacji i nauki opartej na grach), w której gracz steruje bohaterem wydając mu polecenia będące komendami języka programowania. Na szczególną uwagę zasługuje tryb rozgrywki dla dwóch graczy („Dungeon Arena"), w którym każdy z nich steruje bohaterem i sługami jednej z walczących ze sobą stron, zdobywając punkty, decydujące o pozycji w rankingu. Warto wspomnieć, że jako jedyna spośród omówionych dotąd platform, Code Combat rozwijana jest jako oprogramowanie o otwartym kodzie źródłowym.

Platforma Flippi [10] to środowisko wspomagania nauczania programowania z wewnętrzną siecią społecznościową oparte na koncepcji fiszek i dostosowanego do specyfiki nauki programowania algorytmu wyznaczania powtórek. Wykorzystuje ono liczne elementy gamifikacji, m.in. takie jak:

- punkty (uzyskiwane za opanowanie fiszek, przy czym liczba przyznanych punktów zależy od regularności w przestrzeganiu terminów powtórek) oraz ukończone wyzwania,
- wyzwania (odnoszące się do różnorakiej aktywności na platformie, np. dodawania kursów, czy poszerzania grona znajomych),
- informacja zwrotna – obejmująca różne aspekty aktywności na platformie (np. ukończone wyzwania, czy aktywność w sieci społecznościowej);
- poziomy rozwoju – osiągane wraz ze zgromadzeniem odpowiedniej liczby punktów;
- śledzenie postępu – graficznie ukazujące, jak duży dystans dzieli użytkownika od osiągnięcia kolejnego poziomu rozwoju;
- awatary przedstawiające żartobliwie ukazane zwierzątka wraz z krótkimi satyrycznymi historyjkami, specyficznymi dla każdego z nich; awatary nie są wybierane przez użytkownika, ale zmieniają się wraz z osiąganiem przez niego kolejnych poziomów rozwoju (od lamy – stanowiącej nawiązanie do słowa „lamer", oznaczającego w slangu informatycznym słabego programistę – do, mającego oczywiste konotacje, lwa), co oznacza, że pełnią we Flippi rolę zbliżoną do odznaczeń;
- ranking – ukazujący listę użytkowników uporządkowaną według posiadanej w danej chwili liczby punktów.

5. Podsumowanie

Gamifikacja stanowi jeszcze dość nowy obszar wiedzy, a jej praktyczne stosowanie budzi pewne kontrowersje (zob. np. [3] i prace tam cytowane). Niemniej przypisywany jej potencjał do zwiększania motywacji wewnętrznej stanowi wystarczającą przesłankę do podjęcia prób wykorzystywania jej w nauczaniu, szczególnie przedmiotów tak trudnych jak programowanie.

Choć zastosowanie elementów gamifikacji w nauczaniu programowania nie daje gwarancji jego powodzenia, niemal wszystkie takie próby opisane w literaturze zakończyły się przynajmniej częściowym sukcesem. W Internecie dostępnych jest

także szereg zgamifikowanych rozwiązań wspomagających naukę programowania, gotowych do wykorzystania bez potrzeby projektowania i implementacji własnych. Wszystko to daje przesłanki zachęcające do sięgania po to rozwiązanie przez osoby i instytucje zajmujące się nauczaniem programowania.

Literatura

[1] Agapito, J.L., Casano, J.D.L., Martinez, J.C. Xiphias: A Competitive Classroom Control System to Facilitate the Gamification of Academic Evaluation of Novice C++ Programmers. W: *Proceedings of the International Symposium on Computing for Education* (s. 9–15). PSITE, Boracay, Filipiny, 2014.

[2] Bergin, S., Reilly, R. The influence of motivation and comfort-level on learning to program. W: *Proceedings of the 17th Workshop of the Psychology of Programming Interest Group*, University of Sussex, Brighton, Wielka Brytania, 2005.

[3] Berkling, K., Thomas, C. Gamification of a Software Engineering course and a detailed analysis of the factors that lead to it's failure. W: *International Conference on Interactive Collaborative Learning* (s. 525–530), IEEE, Kazań, Rosja, 2013.

[4] Burke, B. *Gamify: how gamification motivates people to do extraordinary things*, Gartner, Brookline, USA, 2014.

[5] *Code Avengers*, http://www.codeavengers.com.

[6] *Code Combat*, http://codecombat.com.

[7] *Code School*, http://www.codeschool.com.

[8] Colao, J. J. With 24 Million Students, Codecademy Is Bigger Than You Thought, *Forbes*, 23.4.2014, http://www.forbes.com/sites/jjcolao/2014/04/23/with-24-million-students-codecademy-is-bigger-than-you-thought, 2014.

[9] Corney, M.W., Teague, D.M., Thomas, R.N. Engaging students in programming. W: *12th Australasian Computing Education Conference* (s. 63–72), ACS, Brisbane, Australia, 2010.

[10] Florczak, R. *Flippi*, http://florczak.unixstorm.org, 2014.

[11] Hanus, M.D., Fox, J. Assessing the effects of gamification in the classroom: A longitudinal study on intrinsic motivation, social comparison, satisfaction, effort, and academic performance. *Computers & Education* **80** (2015), 152–161.

[12] Hu, M., Winikoff, M., Cranefield, S. Teaching Novice Programming Using Goals and Plans in a Visual Notation. W: *Proceedings of the Australasian Computing Education Conference* (s. 43–52), ACS, Darlinghurst, Australia, 2012.

[13] Jassem, K., Piskadło, B. On the development of an open-source system for introducing gamification in higher education. W: *EDULEARN14 Proceedings* (s. 1739–1747). IATED, Barcelona, Hiszpania, 2014.

[14] Jung, J.H., Schneider, C., Valacich, J. Enhancing the Motivational Affordance of Information Systems: The Effects of Real-Time Performance Feedback and Goal Setting in Group Collaboration Environments. *Management Science* **56**,4 (2010), 724–742.

[15] Kaczmarczyk, L.C., Petrick, E.R., East, J.P., Herman, G.L., Identifying student misconceptions of programming. W: *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (s. 107–111), ACM, Nowy Jork, USA, 2010.

[16] Katai, Z., Toth, L. Technologically and artistically enhanced multi-sensory computer-programming education. *Teaching and Teacher Education* **26**, 2 (2010), 244-251.

[17] Knutas, A., Ikonen, J., Nikula, U., Porras, J. Increasing collaborative communications in a programming course with gamification: a case study. W: *Proceedings of the 15th International Conference on Computer Systems and Technologies* (s. 370–377). ACM, Nowy Jork, USA, 2014.

[18] Kosowski, A., Małafiejski, M., Noiński, T., Pomykalski, P. Zintegrowany system do automatycznej oceny rozwiązań oraz prowadzenia zajęć laboratoryjno-projektowych: Sphere Online Judge. *Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej. Technologie Informacyjne*, 7:523–528, 2005.

[19] Lahtinen, E., Ala-Mutka, K., & Järvinen, H.M. A study of the difficulties of novice programmers. W: *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (s. 14–18), ACM, Nowy Jork, USA, 2005.

[20] Laskowski, M. Wykorzystanie czynników grywalizacyjnych w tworzeniu aplikacji użyteczności publicznej. *Nierówności społeczne a wzrost gospodarczy* **36**, (2013), 23–30.

[21] Leutenegger, S., Edgington, J. A games first approach to teaching introductory programming. *ACM SIGCSE Bulletin* **39**,1 (2007), 115–118.

[22] Li, C., Dong, Z., Untch, R.H., Chasteen, M.: Engaging Computer Science Students through Gamification in an Online Social Network Based Collaborative Learning Environment. *International Journal of Information and Education Technology* **3**, 1 (2013), 72–77.
[23] Malone, T.W., Lepper, M.R. Making Learning Fun: A Taxonomy of Intrinsic Motivations for Learning. W: Snow, R. E., Farr, M. J. (red.), *Aptitude, Learning, and Instruction. Volume 3: Conative and Affective Process Analyses* (s. 223–253). Lawrence Erlbaum Associates, Hillsdale, USA, 1987.
[24] Marache-Francisco, C., Brangie, E. Process of Gamification. From the Consideration of Gamification to its Practical Implementation. W: *CENTRIC 2013: The Sixth International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services* (s. 126–131), IARIA, Wenecja, Włochy, 2013.
[25] Marczewski, A. *Defining gamification – what do people really think?*. http://www.gamified.uk/2014/04/16/defining-gamification-people-really-think, 2014.
[26] McCartney, R., Eckerdal, A., Moström, J.E., Sanders, K., Zander, C., Successful students' strategies for getting unstuck. W: *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, ACM, Nowy Jork, USA, 2007.
[27] McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B.-D., Laxer, C., Thomas, L., Utting, I., Wilusz, T. A multinational, multi-institutional study of assessment of programming skills of first-year CS students, *ACM SIGCSE Bulletin* **33**, 4 (2001), 125–140.
[28] Morrison, B.B., DiSalvo, B. Khan academy gamifies computer science. W: *Proceedings of the 45th ACM technical symposium on Computer science education* (s. 39–44). ACM, Nowy Jork, USA, 2014.
[29] Muszyńska, K., Swacha, J. Python and C#: a comparative analysis from students' perspective. *Annales UMCS, Informatica* **11**,1 (2011), 89–101.
[30] Neve, P., Livingstone, D., Hunter, G., Edwards, N., Alsop, G. More than just a game: Improving students' experience of learning programming through gamification. W: *STEM Annual Conference 2014* (Presentation online: http://www.heacademy.ac.uk/sites/default/files/resources/COMP-224-P.pdf), Higher Education Academy, Edynburg, Wielka Brytania, 2014.
[31] Paspallis, N. A Gamification Platform for Inspiring Young Students to Take an Interest in Coding. W: *Information Systems Development: Transforming Organisations and Society through Information Systems (ISD2014 Proceedings)*. University of Zagreb, Faculty of Organization and Informatics, Varaždin, Chorwacja, 2014.
[32] Pirker, J., Riffnaller-Schiefer, M., Gütl, C. Motivational Active Learning – Engaging University Students in Computer Science Education. W: *2014 Conference on Innovation & Technology in Computer Science Education*. Uppsala, Szwecja, 2014.
[33] Robins, A., Rountree, J., Rountree, N. Learning and teaching programming: A review and discussion. *Computer Science Education* **13**,2 (2003), 137–172.
[34] Rogerson, C., Scott, E. The Fear Factor: How It Affects Students Learning to Program in a Tertiary Environment, *Journal of Information Technology Education* **9** (2010), 147-171.
[35] Schacht, M., Schacht, S. Start the Game: Increasing User Experience of Enterprise Systems Following a Gamification Mechanism. W: A. Maedche, A. Botzenhardt, L. Neer (red.), *Software for People* (s. 181–199). Springer, Berlin/Heidelberg, Niemcy, 2012.
[36] Sheth, S.K., Bell, J.S., Kaiser, G.E. *Increasing Student Engagement in Software Engineering with Gamification*. Columbia University Academic Commons, http://hdl.handle.net/10022/AC:P:15273, 2012.
[37] Swacha, J. Nowe rozwiązania w dydaktyce programowania komputerów. W: M. Kowalski, A. Olczak (red.), *Edukacja w przebiegu życia. Od dzieciństwa do starości* (s. 115-126), Impuls, Kraków 2010.
[38] Swacha, J., Baszuro, P. Gamification-based e-learning platform for computer programming education. W: Reynolds, N., Webb, M. (red.), *Learning while we are connected. Volume 1: Research papers* (s. 122–130). Wydawnictwo Naukowe Uniwersytetu Mikołaja Kopernika, Toruń, 2013.
[39] Sweedyk, E., de Laet, M., Slattery, M.C., Kuffner, J. Computer games and CS education: why and how. W: *Proceedings of the 36th SIGCSE technical symposium on Computer* science education, ACM SIGCSE, St. Louis, USA, 2005.
[40] Tang, T., Rixner, S., Warren, J. An environment for learning interactive programming. W: *Proceedings of the 45th ACM technical symposium on Computer science education* (s. 671–676). ACM, Nowy Jork, USA, 2014.
[41] Tillmann, N., de Halleux, J., Xie, T., Bishop, J. Code Hunt: Gamifying Teaching and Learning of Computer Science at Scale. W: *Proceedings of the first ACM conference on Learning @ Scale* (s. 221–222). ACM, Nowy Jork, USA, 2014.
[42] Tkaczyk, P. *Grywalizacja. Jak zastosować mechanizmy gier w działaniach marketingowych*. Helion, Gliwice 2012.
[43] Winslow, L.E. Programming pedagogy – A psychological overview. SIGCSE Bulletin **28**,3 (1996), 17–22.

[44] Xie, T., de Halleux, J., Tillmann, N., Schulte, W. Teaching and Training Developer-Testing Techniques and Tool Support. W: *Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications* (s. 175–182). ACM, Nowy Jork, USA, 2010.

Rozdział 2

Metodyka kształtowania ryzyka w cyklu rozwojowym systemu informatycznego

1. Wprowadzenie

W literaturze spotyka się wiele różnych definicji pojęcia ryzyka, np.

"Ryzyko to skumulowany efekt prawdopodobieństwa niepewnych zdarzeń, które mogą korzystnie lub niekorzystnie wpłynąć na realizację projektu" [6]
"Ryzyko jest możliwością wystąpienia zdarzenia, które będzie miało niepożądany wpływ na daną organizację i jej systemy informatyczne "[1].

Na potrzeby niniejszej pracy jak i również na potrzeby zaproponowanej metodyki kształtowania ryzyka, ryzyko systemu informatycznego definiowane jest jako zagrożenie iż:

- zastosowana metoda i/lub technika w procesie inżynierii oprogramowania,
- zastosowana technologia informatyczna niezależnie od jej rodzaju i skali działalności,
- zastosowana metodyka zarządzania i/lub kształtowania ryzyka:
  - nie spełnia wymogów i/lub zaleceń inżynierii oprogramowania,
  - nie spełnia wymogów biznesowych,
  - nie zapewnia odpowiedniej jakości i bezpieczeństwa,
  - nie została odpowiednio wdrożona i/lub nie działa zgodnie z przyjętymi założeniami.

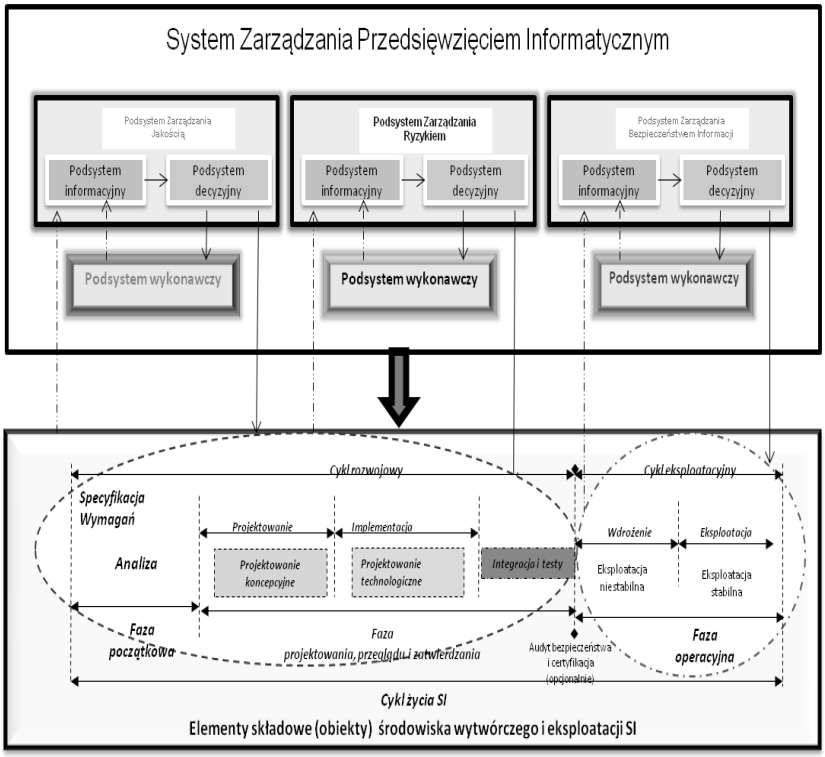
W związku z powyższym ryzyko związane z przedsięwzięciami informatycznymi lub w cyklu życia systemu informatycznego rozpatrywane jest w podziale na:

- ryzyko projektowe odnoszące się do zagrożeń występujących w **cyklu rozwojowym ,**
- ryzyko eksploatacyjne odnoszące się do zagrożeń występujących w **cyklu eksploatacyjnym** - badanie funkcjonującego systemu informatycznego w celu oszacowania poziomu zagrożeń związanych z jego eksploatacją ,

a w ramach tych cykli na różne kategorie związane z realizacją następujących procesów:

- proces wytwórczy, związany z wytwarzaniem oprogramowania systemu,
- proces zarządczy, związany z prowadzeniem projektu informatycznego,
- proces zarządzania jakością,
- proces zarządzania bezpieczeństwem lub zapewniania bezpieczeństwa, związany z utrzymywaniem atrybutów bezpieczeństwa systemu informatycznego.

Systemowy model zarządzania przedsięwzięciem informatycznym przedstawia rysunek 1.



Rysunek 1.Ogólny model zarządzania przedsięwzięciem informatycznym. Źródło: Opracowanie własne.

System Zarządzania Przedsięwzięciem Informatycznym to połączenie podsystemów, procesów, procedur i praktyk działania stosowanych w *cyklu życia systemu informatycznego* w celu wykreowania i wdrożenia *metodyki prowadzenia przedsięwzięcia informatycznego*, która może być bardziej skuteczna w osiąganiu założonych celów - rezultatu końcowego przedsięwzięcia. Pod pojęciem cyklu życia systemu informatycznego należy rozumieć określoną koncepcję rozłożenia w czasie czynności odbywających się podczas pracy nad opracowaniem i wyprodukowaniem systemu informatycznego oraz jego użytkowania - eksploatacji. Podstawowe elementy składowe cyklu życia systemu (jak to pokazuje rysunek 1) to *cykl rozwojowy* i *cykl eksploatacyjny*.

W ujęciu modelowym w **Systemie Zarządzania Przedsięwzięciem Informatycznym,** dla każdego z wyszczególnionych aspektów zarządzania przedsięwzięciem: jakością, ryzykiem, bezpieczeństwem informacji, wyróżniono dwa podsystemy:

1. ***Wykonawczy***, który stanowią siły i środki realizujące procesy wykonawcze;
2. ***Zarządzania***, który realizuje procesy informacyjno-decyzyjne, decydujące o sposobie zapewniania wymaganego poziomu odpowiednio: jakości ryzyka, bezpieczeństwa, poszczególnym obiektom Systemu Informatycznego (SI) przez podsystem wykonawczy.

Cykl, który opisuje realizację skomplikowanych procesów, dzielony jest na mniejsze jednostki nazywane zwykle *etapami*. Podział cyklu na etapy jest korzystny, ponieważ:

- ułatwia rozwiązanie problemu - etapy obejmują mniejszy zakres zadań i czynności (niż cały cykl) i dzięki temu są łatwiejsze do wykonania i nadzoru. W razie potrzeby, etapy można podzielić na jeszcze mniejsze jednostki nazywane w niniejszym opracowaniu fazami (np. etap analizy będzie się składał z faz: pozyskiwania wymagań, formułowania wymagań, formułowania specyfikacji), które mogą być niekiedy realizowane równolegle;
- umożliwia precyzyjniejsze określenie stopnia ukompletowania projektu;
- ułatwia kontrolę jakości prowadzonych prac - każdy etap może zakończyć się odpowiednimi przedsięwzięciami oceny i kontroli;

Cykl życia systemu informatycznego rozpoczyna się wraz z początkową identyfikacją potrzeb, poprzez planowanie, badania, projektowanie, produkcję, ocenę, użytkowanie, serwisowanie i ostateczne wycofanie produktu informatycznego (cykl rozpoczyna się na użytkowniku i na nim kończy). Cykl życia systemu informatycznego rozumiany jest tutaj, jako ewolucja w czasie tego systemu od koncepcji do wycofania z użycia.
Prawidłowo określone rozbicie cyklu rozwojowego na etapy:

- zwiększa prawdopodobieństwo poprawnego wykonania projektu w zaplanowanym terminie i budżecie oraz
- umożliwia zaproponowanie lepszej - skuteczniejszej i/lub kompleksowej metodyki analizy i zarządzania ryzykiem systemu, biorąc pod uwagę różne kategorie czynników ryzyka występujące na wszystkich jego wyróżnionych etapach, zarówno rozwoju jak i eksploatacji.

W niniejszym artykule opisane są tylko zagadnienia związane z ryzykiem w cyklu rozwojowym systemu informatycznego. Aby efektywnie zarządzać ryzykiem w cyklu rozwojowym systemu informatycznego należy w możliwie obiektywny sposób wyznaczać poziom tego ryzyka na każdym etapie procesu wytwórczego. Istnieje obecnie wiele standardów (norm, metodyk i metod) oceny ryzyka systemów informatycznych, ale zdaniem autorów żaden z nich nie jest jednak standardem uniwersalnym, nadającym się do analizy i zarządzania ryzykiem związanego z

projektowaniem i użytkowaniem zarówno prostego systemu informatycznego w małej firmie, jak też złożonych systemów informacyjnych dużej międzynarodowej organizacji (korporacji, federacji). Dodatkowo żaden z wykorzystywanych obecnie standardów czy metod analizy ryzyka systemów informatycznych nie bierze w sposób bezpośredni holistyczny i kompleksowy pod uwagę jednocześnie wpływu grup czynników jak czynniki ilościowe, jakościowe, bezpieczeństwa, ekonomiczne czy socjotechniczne, co z jednej strony stanowi o sile obecnie stosowanych podejść poprzez ich ukierunkowanie na wybrane aspekty bezpieczeństwa i jakości zaradzania przedsięwzięciem informatycznym, a z drugiej strony jest ich słabością poprzez daleko idące uproszczenie przyjmowanych modeli.

Celem niniejszego artykułu jest próba skonstruowania (w wersji uproszczonej) spójnej metodyki kształtowania ryzyka systemu informatycznego, uwzględniającej zarówno czynniki procesu wytwórczego oprogramowania, środowiska realizacji projektu oraz zarządzania przedsięwzięciem informatycznym.

Poziom ryzyka oszacowany w fazie rozwoju systemu informatycznego stanowi wielkość wejściową do metody zarządzania ryzykiem w fazie eksploatacji, której opis wykracza poza ramy niniejszego artykułu.

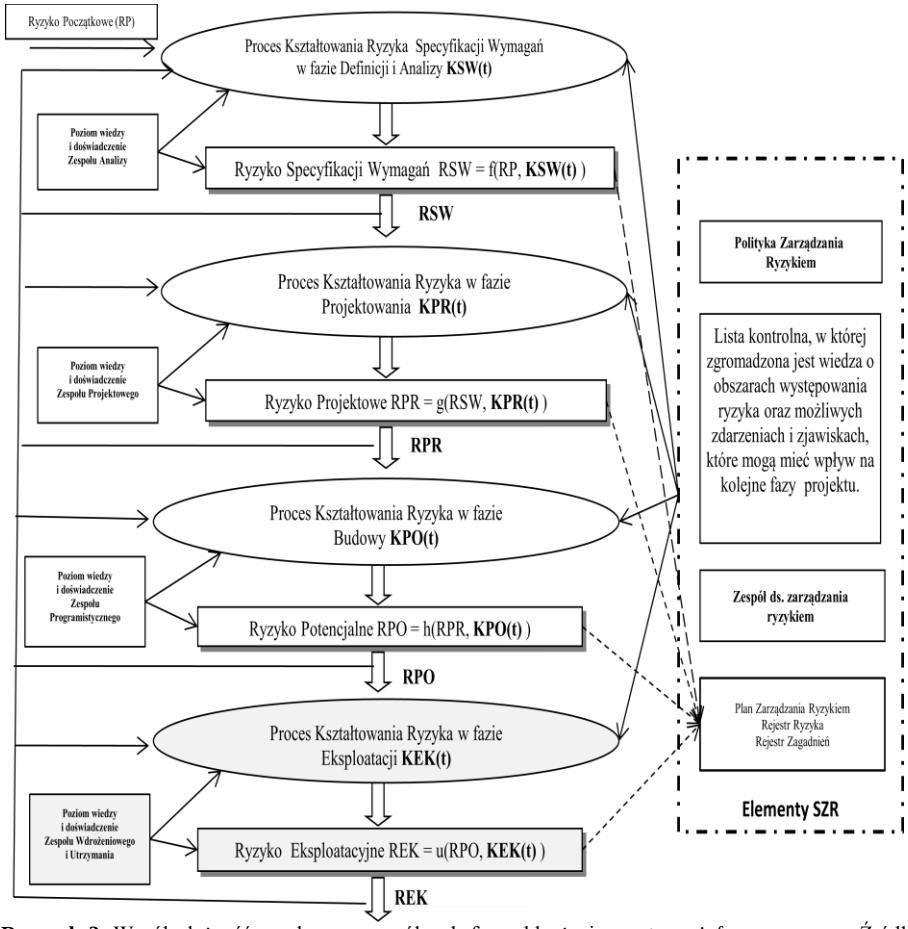
2. Koncepcja metodyki kształtowania ryzyka w cyklu rozwojowym systemu

Kształtowanie ryzyka w procesie wytwórczym systemu informatycznego obejmuje nie tylko sam produkt informatyczny lecz również proces zarządzania i proces wytwórczy, które prowadzą do realizacji tego produktu. Określone ryzyko systemu informatycznego jest wynikiem nie tylko przeprowadzonych testów końcowych wytworzonego systemu lecz jest rezultatem odpowiednich "działań" prowadzonych na wszystkich etapach tworzenia systemu informatycznego. Już podczas analizy i projektowania powinny zostać podjęte środki mające na celu redukcję ryzyka do poziomu akceptowalnego oraz uzyskanie systemu, który spełnia wszystkie wymagania użytkownika, a co się z tym wiąże stopień zadowolenia użytkownika powinien świadczyć o jakości systemu informatycznego. Odpowiednie działania powinny być również prowadzone w trakcie wytwarzania, instalowania, jak i eksploatacji wytworzonego systemu informatycznego.

Wytwarzany system informatyczny realizuję koncepcję i rozwiązania definiowane na wszystkich etapach jego powstawania. Współzależność ryzyka odpowiadającego poszczególnym etapom/fazom cyklu wytwarzania systemu dla podejścia klasycznego przedstawiono na rysunku 2.

Ponieważ zakresy prac oraz sposoby ich realizacji dla wyróżnionych etapów, faz, warstw lub poziomów nie są jednakowe dlatego zasadne jest rozróżnienie ryzyka odpowiadającego poszczególnym fazom cyklu życia systemu. Zatem wyróżniać można (w przypadku wyboru podejścia klasycznego):

- ryzyko specyfikacji wymagań dla fazy definicji i analizy (RSW),
- ryzyko projektowe dla fazy projektowania (RPR),
- ryzyko potencjalne dla fazy wytwarzania /budowy (RPO),
- ryzyko eksploatacyjne dla fazy wdrażania i eksploatacji (REK).



Rysunek 2. Współzależność ryzyka poszczególnych faz cyklu życia systemu informatycznego. Źródło: Opracowanie własne

Między tymi poziomami ryzyka poszczególnych faz wytwarzania systemu informatycznego istnieją następujące zależności:

- REK = u(RPO, KEK(t)),
- RPO = h(RPR, KPO(t)),
- RPR = g(RSW, KPR(t)),
- RWS = f (RP, KSW(t)),
- RP = w(f(g(h(u(RPO, KEK(t)),KPO(t)), KPR(t)), KSW(t)))

gdzie:

KSW(t) - proces kształtowania ryzyka specyfikacji wymagań,
KPR(t) - proces kształtowania ryzyka projektowego,
KPO(t) - proces kształtowania ryzyka i potencjalnego,
KEK(t) - proces kształtowania ryzyka eksploatacyjnego.

Ryzyko projektowe (RPR) systemu informatycznego narzucone przyjętym rozwiązaniem konceptualnym (funkcjonalno - konstrukcyjnym) i akceptowalnym ryzykiem specyfikacji wymagań (RSW) przekształca się w procesie produkcji oprogramowania (implementacji) w ryzyko potencjalne (RPO), a to z kolei w procesie wdrażania i eksploatacji zmienia się w ryzyko eksploatacji obserwowane przez użytkownika (REK).

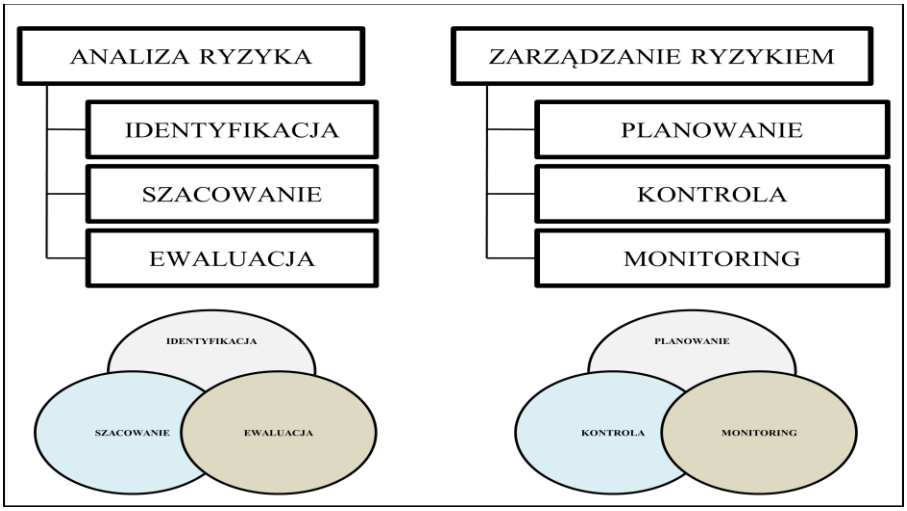
Jest oczywiste, że osiągnięcie wymaganego poziomu bezpieczeństwa, sprawności i racjonalności eksploatacji w dużej mierze zależy od poziomu ryzyka i jakości rozwiązań przyjętych w trakcie analizy, projektowania, budowy i wdrażania systemu (prace studialne, modelowanie pojęciowe i logiczne, projektowanie, prace programowe i prace wdrożeniowe). Rozwiązania systemowe powinny zapewniać możliwość względnie swobodnej zmiany własności eksploatacyjnych wybranych elementów systemu i jego bezpieczeństwa funkcjonalnego. Powinny być one dostosowane do zmieniających się w krótkich odstępach czasu warunków, w których działa system, oraz do zmieniających się w dłuższych okresach potrzeb użytkowników.

Kształtowanie ryzyka w procesie wytwórczym SI to realizacja zadań, których bardzo często nie ma w harmonogramie projektu, a które są konieczne do osiągnięcia celu projektu. Metodyka kształtowania ryzyka jest to zbiór "zapisów" poszczególnych kroków działań zorientowanych na ryzyko, metod, technik i narzędzi niezbędnych do prawidłowego zrealizowania procesu transformacji wymagań jakościowych użytkownika i zagrożeń początkowych projektu na działający system informatyczny. Metodyka dostarcza wzorców, formularzy, wskazówek inżynierskich, itp., według których należy realizować zadania przedsięwzięcia informatycznego. Wymienione składniki tworzą środowisko pracy kierownika przedsięwzięcia informatycznego. Istotnym elementem tego środowiska są narzędzia, które wspomagają realizację procesu kształtowania ryzyka i jakości systemu informatycznego. Poziom ryzyka w cyklu rozwojowym systemu informatycznego jest rezultatem nałożenia się na siebie ryzyk wszystkich faz cyklu życia systemu. Każda faza cyklu życia systemu powoduje "wniesienie" i/lub "przeniesienie" pewnego poziomu ryzyka w ryzyko szczątkowe końcowego produktu, którym jest system informatyczny.

Metodyka kształtowania ryzyka z praktycznego punktu widzenia jest systemem metod, technik i działań zmierzających do zidentyfikowania, oszacowania i reagowania na zagrożenia pojawiające się w poszczególnych fazach cyklu życia projektu tak, aby:

- procesy kształtowania ryzyk w poszczególnych fazach cyklu życia systemu "dawały" wyniki akceptowalne dla kierownika projektu,
- cele projektu zostały osiągnięte.

Proces kształtowania ryzyka w każdej fazie składa się z następujących grup prac - rysunek 3.



Rysunek 3. Grupy prac składających się na analizę ryzyka i zarządzanie ryzykiem. Źródło: Opracowanie własne

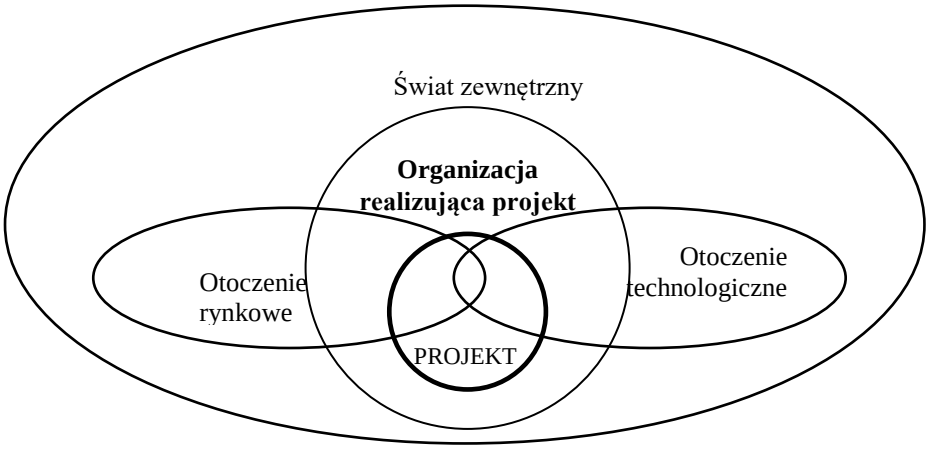
Przytaczane opisy analizy ryzyka (zaczerpnięte ze standardów np. metodyki PRINCE2), naszym zdaniem, nie wyczerpują ani problemu analizy ryzyka, ani problemu zarządzania ryzykiem. Krótki opis podstawowych elementów metodyki kształtowania ryzyka znajduje się w rozdziale 3.

Bardzo ważnym elementem w procesie kształtowania ryzyka jest określenie:

- Źródeł ryzyka - kategorii możliwych sytuacji ryzyka (np. działania uczestników, niewłaściwe szacunki, zmiany w zespole), które mogą wpływać na projekt negatywnie lub pozytywnie. Lista źródeł powinna być kompletna (wyczerpująca), tzn. powinna obejmować wszystkie zidentyfikowane zdarzenia, niezależnie od częstotliwości, prawdopodobieństwa wystąpienia lub skali zagrożeń. Najpowszechniejsze źródła ryzyka to: zmiany w założeniach lub zmiany wymagań i zakresu projektu, błędy wykonania, niezrozumienie lub pominięcie istotnych elementów projektu, źle zdefiniowane lub zrozumiane role i odpowiedzialność, złe oszacowania, niekompetencja lub brak doświadczenia zespołu, zależność od zdarzeń i zespołów zewnętrznych. Należy ponadto szczególnie dokładnie ocenić ryzyka dla sytuacji, gdzie stwierdza się ograniczoną liczbę zasobów i gdzie takie zasoby są w pełni obciążone, albo mogą stać się niedostępne. W grupie zadań o wysokim ryzyku są zwłaszcza te z wieloma poprzednikami (im więcej zależności ma zadanie, tym większe prawdopodobieństwo jego opóźnienia) oraz zadania z długimi czasami realizacji albo z dużą różnorodnością zasobów (prawdopodobieństwo nieścisłości oszacowania takich zadań jest z reguły dość duże).
- Objaw ryzyka – objawy ryzyka to zespół wydarzeń lub faktów potwierdzających (niekoniecznie bezpośrednio) rzeczywiste zaistnienie danego ryzyka. Dobrze jest wskazać takie objawy, które pozwolą sygnalizować nadchodzące zagrożenie z wyprzedzeniem – zanim ono

nastąpi, czyli tzw. objawów wyprzedzających, określanych po angielsku jako triggers = = cyngle. „Cyngle" są wskaźnikami ujawniającymi, że dane zagrożenie już nastąpiło lub niedługo się wydarzy. Cyngle uruchamiają odpowiednie procedury, podobnie jak cyngiel broni uruchamia pocisk. Identyfikację cyngli ułatwia dyskusja z ludźmi, którzy są najbardziej odpowiedzialni za spowodowanie zagrożeń i z ludźmi którzy najbardziej je odczują. Należy przy tym określić osoby odpowiedzialne za monitorowanie ryzyk i śledzenie poziomu cyngli.

Źródła zagrożeń (ryzyk) dla przedsięwzięcia informatycznego są bardzo liczne i wynikają nie tylko z jego wyjątkowości i niepowtarzalności. Specyfikację źródeł ryzyka ułatwia ich lokalizacja w różnych obszarach środowiska i otoczenia projektu. Projekt wykonywany jest przez zespół, stanowiący część grupy kilku organizacji. Organizacje te działają z kolei w określonych otoczeniach rynkowych, biznesowych i technologicznych, a całość jest osadzona w świecie zewnętrznym, rys.4.



Rysunek 4. Schemat środowiska realizacji projektu. Źródło: Opracowanie własne

Zagrożenia tego projektu mogą pojawić się w każdym elemencie jego środowiska, przy czym najłatwiej je dostrzec i kontrolować, jeśli pojawiają się w najbliższym otoczeniu projektu. Ze względu na rodzaj przesłanek można wyróżnić:

- *ryzyko właściwe*, czyli takie, które można prognozować w oparciu o prawo wielkich liczb, czyli dotyczące zjawisk niepewnych, ale mających znaną i opisaną historię i przez to podlegających opisowi probabilistycznemu – klęski żywiołowe, choroby, wahania kursów walut, awarie sprzętu, itp.,
- *ryzyko subiektywne* - wynikające z niekompetencji człowieka dokonującego analizy i podejmującego decyzje,
- *ryzyko obiektywne* – wynikające z nieprzewidywalności przyszłych zdarzeń, np. odkrycia naukowe, nowe technologie, przewroty polityczne, itp.

W projekcie powinno się kontrolować ryzyko właściwe i eliminować ryzyko subiektywne. Na zagrożenia wynikające z ryzyka obiektywnego nie mamy na ogół wpływu i zagrożenia tego rodzaju pozostają na ogół poza kontrolą zespołu projektowego. Rolą kierownika projektu jest odpowiednie postępowanie, polegające na podejmowaniu wyważonych decyzji, uwzględniających dostępną wiedzę o

ryzykach projektu oraz metodach zarządzania ryzykami. Zaleca się także wyodrębnić w zespołach projektowych osoby zajmujące się tym zakresem jako właściciele ryzyka, wspomagający pracę Kierownika Projektu.

3. Opis podstawowych elementów metodyki kształtowania ryzyka

W niniejszym rozdziale zostały scharakteryzowane następujące podstawowe elementy:

- Systemu Zarządzania Ryzykiem,
- Grupy prac składających się na analizę ryzyka i zarządzanie ryzykiem,
- Zakres odpowiedzialności za realizację Polityki zarządzania ryzykiem w Projekcie.

3.1. Podstawowe elementy Systemu Zarządzania Ryzykiem

System zarządzania ryzykiem - ta część całościowego systemu zarządzania, odnosząca się do podejmowanie działań mających na celu rozpoznanie, ocenę i sterowanie ryzykiem oraz kontrolę podjętych działań. Celem zarządzania jest ograniczanie ryzyka oraz zabezpieczanie się przed jego skutkami.

Polityka Zarządzania Ryzykiem ma na celu wdrożenie wytycznych w całym procesie wytwórczym dla osiągnięcia założonych celów strategicznych projektu. Główne elementy jakie powinny zostać określone w dokumencie to: poziom ryzyka tolerowanego i dopuszczalnego, progi tolerancji na ryzyko, procedury eskalacji, role i odpowiedzialność, opis procesu zarządzania ryzkiem, wskaźniki pozwalające na wczesne ostrzeganie, narzędzia i techniki wspomagające proces, kiedy zarządzanie ryzykiem powinno być stosowane, raportowanie, budżet, zasady zapewnienia jakości, metoda corocznych przeglądów. Opis Procesu Zarządzania Ryzykiem natomiast wskazuje kroki i działania, konieczne do wdrożenia zarządzania ryzykiem w organizacji. Dokument ten powinien składać się z: ról i odpowiedzialności, kroków w procesie, narzędzi i technik, wzorów dokumentów, wskaźników wczesnego ostrzegania, sposoby definiowania rezerwy, określenia właścicieli rezerw.

Plan Zarządzania Ryzykiem, którego celem jest opisanie działań dla obszaru czynności, związanych z zarządzaniem ryzykiem, czyli opisu działań, opisu ról i odpowiedzialności, procesów, skal do oceny prawdopodobieństwa i skutków, definicji skali prawdopodobieństwa, definicji skali skutku, wartości oczekiwanej, bliskości, kategorii reakcji na ryzyko, wymaganego budżetu, narzędzi i techniki, wzorów dokumentów, wskaźników wczesnego ostrzegania, harmonogramu działań związanego z zarządzaniem ryzykiem, raportowania, listy kontrolne i zagadnienia wymagające uwagi.

Rejestr Ryzyka, to dokument, w którym następuje gromadzenie i przechowywanie informacji zidentyfikowanych zagrożeń lub okazji. Składa się z: identyfikatora ryzyka, kategorii ryzyka, przyczyny ryzyka, charakteru ryzyka, efektu ryzyka,

Prawdopodobieństwa, spodziewanego skutku, wartości oczekiwanej, bliskości, kategorii reakcji na ryzyko, reakcji na ryzyko, ryzyko rezydualne, statusu ryzyka, właściciela ryzyka, właściciela reakcji związanego z ryzykiem.

Rejestr Zagadnień to zapisywanie informacji dotyczących zagadnień, które wystąpiły i wymagają działań. Identyfikator zagadnienia, kategoria zagadnienia, opis zagadnienia, wpływ zagadnienia, wymagane działania, planowana data rozpoczęcia działań, data, kiedy działania wdrożono, właściciel zagadnienia, właściciel działań związanych z zagadnieniem.

Procesy planowania - zbiór czynności, które mają na celu z jednej strony skłonienie menedżera projektu do przygotowania i zorganizowania procesu zarządzania ryzykiem, z drugiej zaś mają one doprowadzić do powstania infrastruktury organizacyjnej, której zadaniem będzie podjęcie działań zmierzających do:

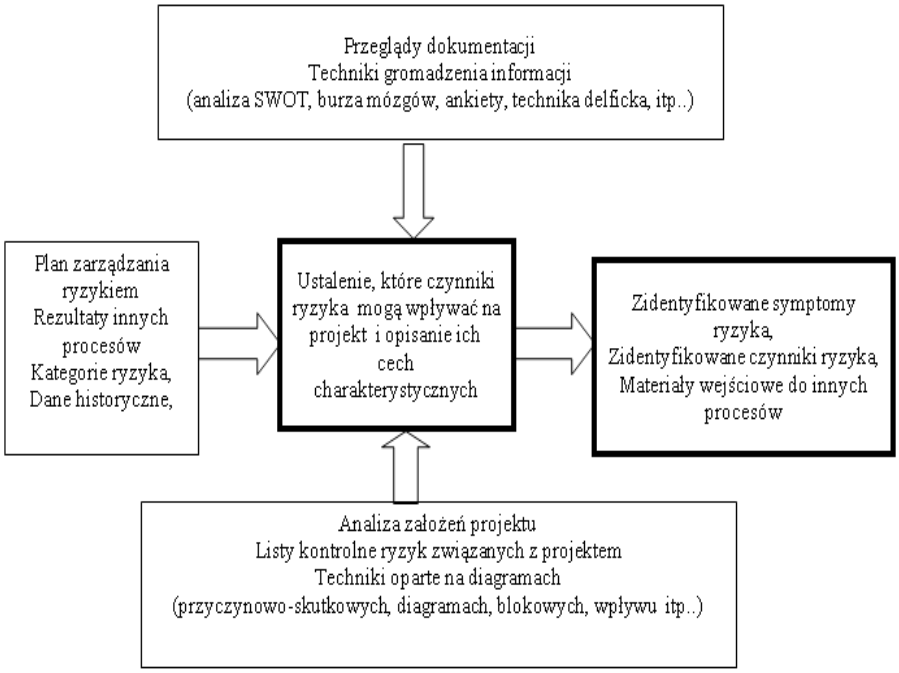
- izolowania i zmniejszenia ryzyka,
- eliminowania ryzyka (jeśli jest to możliwe i uzasadnione),
- przygotowania alternatywnych sposobów działania,
- określenia rezerw czasowych i pieniężnych w celu zabezpieczenia przed zagrożeniami mogącymi pojawić się podczas planowania i wykonywania prac projektowych.

Procesy monitorowania i kontroli ryzyka - dostarczają informacji niezbędnych do podejmowania właściwych decyzji projektowych, pozwalających wyprzedzić niekorzystne zdarzenia i ograniczyć ich wpływ na proces wytwórczy systemu informatycznego. Monitorowanie, niezależnie od typu ryzyka jest procesem ciągłym. Składa się on zawsze z następujących etapów:

- identyfikacja ryzyka,
- ocena ryzyka,
- podjęcie działań prewencyjnych,
- ocena skuteczności podjętych działań,
- optymalizacja procesów niwelowania ryzyka.

3.2. Grupy prac składających się na analizę ryzyka i zarządzanie ryzykiem

Identyfikacja ryzyka - są to działania, których celem jest wykrycie źródeł ryzyka, a następnie ich usystematyzowanie według przyjętych kategorii. Identyfikacja ryzyka jest procesem iteracyjnym, wykonuje się ją wielokrotnie zarówno podczas planowania, jak i w trakcie realizacji procesu wytwórczego. Ogólny schemat procesu identyfikacji ryzyk przedsięwzięcia przedstawiono na rysunku 5.

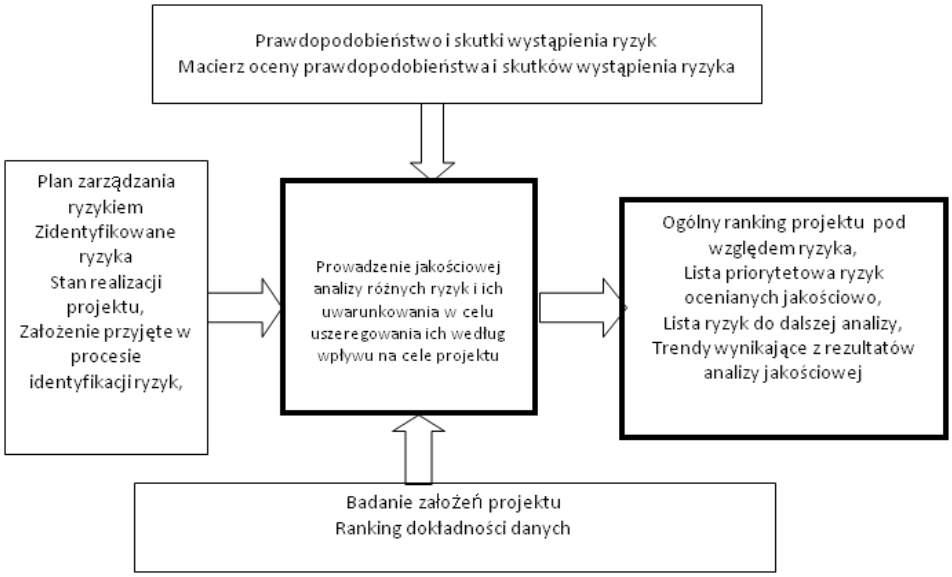


Rysunek 5. Ogólny schemat procesu identyfikacji ryzyk projektu. Źródło: Opracowanie własne.

Ocena ryzyka - są to działania związane z przeprowadzeniem jakościowej i ilościowej analizy ryzyka oraz ocena ryzyka. Głównym zadaniem jakościowej analizy ryzyka jest oszacowanie wielkości prawdopodobieństwa i skutków zaistnienia zidentyfikowanych uprzednio ryzyk. Na tym etapie wykonuje się hierarchizację zidentyfikowanych niebezpieczeństw według ich potencjalnego wpływu na proces realizacji przedsięwzięcia. Uzyskane wyniki będą stanowiły podstawę do dalszego planowania reakcji na niekorzystne zjawiska.
Zidentyfikowane ryzyka aktywne i odroczone powinny zostać poddane filtracji w procesie analizy jakościowej w celu ustalenia, które z nich mogą zostać zaakceptowane ze względu na znikome prawdopodobieństwo wystąpienia i potencjalne zagrożenia oraz które z nich i w jakiej kolejności powinny być przedmiotem dalszej analizy i zaplanowania reakcji na ryzyko.
W szczególności, wyniki analizy jakościowej stanowią podstawę podjęcia decyzji, które z czynników ryzyka:

- wymagają dalszego monitorowania, z czasowym zawieszeniem planowania i realizacji odpowiedzi na ryzyko,
- zostaną wyeliminowane w wyniku ograniczenia lub zmiany zakresu projektu,
- zostaną przetransferowane na rzecz uczestnika projektu zdolnego do jego przejęcia; ryzyko może być także przejęte przez inwestora lub przeniesione na ubezpieczyciela,
- wymagają zaplanowanego przeciwdziałania, podjętego przez zarządzających projektem.

Schemat procesu jakościowej analizy ryzyka przedstawiono na rysunku 6.

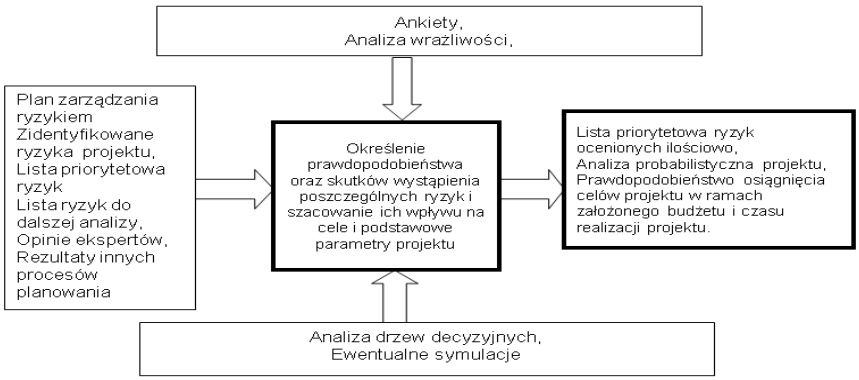


Rysunek 6. Schemat procesu analizy ilościowej ryzyk. Źródło: Opracowanie własne

Proces ilościowej analizy ryzyka ma na celu określenie wymiernych wartości wielkości prawdopodobieństwa oraz skutków wystąpienia zdarzeń niekorzystnych zarówno dla poszczególnych czynności projektu, jak i dla całego przedsięwzięcia. Pomiar taki pozwala ustalić w wielkościach wymiernych szansę osiągnięcia celów projektu, określić poziomy niezbędnych rezerw, przeprowadzić szczegółową analizę typu, „co-jeśli". Ilościowa analiza ryzyka często jest poprzedzana badaniami jakościowymi. W przypadku projektów o znacznej wartości lub w przypadku identyfikacji ryzyk o znacznym wpływie na możliwość osiągnięcia ustalonych celów projektu, analiza jakościowa ryzyka projektu powinna zostać uzupełniona o analizę ilościową, polegającą na numerycznym określeniu prawdopodobieństwa oraz skutków wystąpienia poszczególnych ryzyk. W szczególności, analiza ilościowa ryzyk ma na celu:

- określenie prawdopodobieństwa osiągnięcia określonego celu projektu,
- ustalenie rezerw czasowych i kosztowych, które mogą potrzebne, jako rekompensata skutków wystąpienia poszczególnych ryzyk.

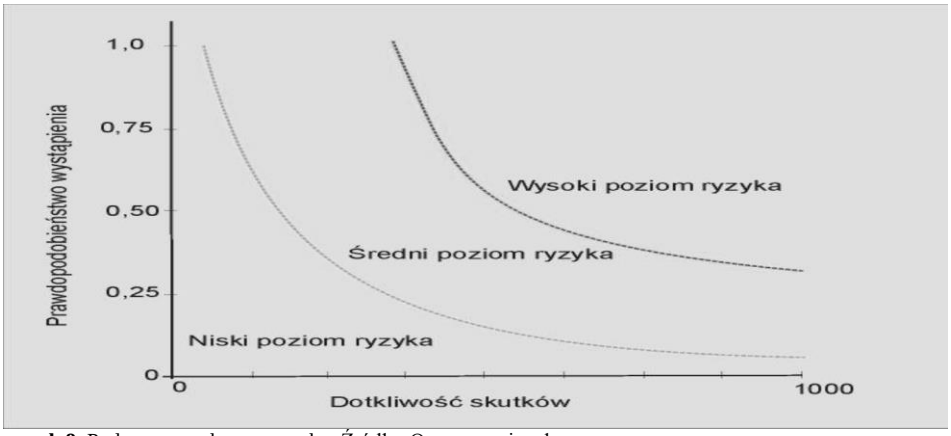
Schemat procesu ilościowej analizy ryzyka przedstawiono na rysunku 7.



Rysunek 7. Schemat procesu analizy ilościowej ryzyk. Źródło: Opracowanie własne.

Szacowanie ryzyka (co wiąże się z danym ryzykiem, jak często występują i jakie maja efekty dane ryzyka, jak znaczące jest dla nas ryzyko) – określenie prawdopodobieństwa zagrożeń i dotkliwości ich skutków.
Rezultat:

- czynniki oceny – precedens; zasoby i umiejętności; czas, koszty i jakość; prawdopodobieństwo wystąpienia zdarzenia ryzyka, skutek wystąpienia zdarzenia ryzyka
- macierz wagi zdarzeń ryzyka i/lub Podstawowe obszary ryzyka – spójny sposób powiązania prawdopodobieństwa i skutków wystąpienia danego zdarzenia ryzyka np.
- hierarchia ryzyk - lista ryzyk uporządkowana według ich wpływu(wagi) z uwzględnieniem ustalonych priorytetów, która pomoże stwierdzić, czy dane zdarzenie ryzyka są warte uwagi.



Rysunek 8. Podstawowe obszary ryzyka. Źródło: Opracowanie własne.

Planowanie reakcji na ryzyko - jest to proces opracowywania wariantów postępowania dotyczących czynności zmniejszających zagrożenia i zwiększających potencjalne korzyści dla sformułowanych celów projektowych. Plan reakcji na ryzyko to kluczowy etap procesu zarządzania ryzykiem, gdyż opracowuje się w nim metody reagowania na zdarzenia korzystne i niekorzystne. Skuteczność planowania reakcji na ryzyka zadań zagrożonych ma bezpośredni wpływ na wzrost lub spadek ryzyka realizacji całego projektu. Planowane reakcje muszą być proporcjonalne do skutków wystąpienia niekorzystnych zjawisk, likwidować (lub niwelować) wpływy danego zagrożenia w sposób kosztowo efektywny oraz być realizowane terminowo. Do najbardziej popularnych strategii zalicza się:

- unikanie ryzyka - polega na takiej modyfikacji planów realizacji projektu, by zlikwidować dane ryzyko (niestety nie można w praktyce wyeliminować wszystkich zdarzeń, z którymi wiążą się niebezpieczeństwa) albo korzystnie zmienić uwarunkowania z nim związane,
- transfer ryzyka - to działanie polegające na przeniesieniu skutków wystąpienia ryzyka na inny podmiot. Działanie to jest najskuteczniejsze w obszarze finansów, wiąże się ono zazwyczaj z koniecznością wypłacenia premii podmiotowi przyjmującemu ryzyko (np. ubezpieczenie na wypadek włamania do firmy),
- łagodzenie ryzyka - to najpowszechniejsza ze wszystkich strategii reagowania na ryzyko. Proces ten polega na podejmowaniu określonych działań prowadzących do zmniejszenia prawdopodobieństwa lub skutków ryzyka,
- akceptacja ryzyka - polega na przyjęciu i udźwignięciu wszelkich konsekwencji wynikających z ewentualnego wystąpienia niekorzystnego zjawiska. Jest to świadoma decyzja osób zarządzających ryzykiem, by nie wprowadzać żadnych zmian w planie projektu związanych z wystąpieniem danego niekorzystnego zjawiska. Istnieją dwa podstawowe typy akceptacji ryzyka: aktywna i pasywna. Pasywna akceptacja polega na przyjęciu ryzyka bez podejmowania jakichkolwiek działań w celu rozwiązania problemów jakie się z nim wiążą. Natomiast aktywna akceptacja polega na pogodzeniu się z ryzykiem, ale wymaga stworzenia specjalnego planu działania w razie wystąpienia niekorzystnego zdarzenia, a w niektórych przypadkach tzw. planu odwrotu,
- plan awaryjny - buduje się tylko dla zidentyfikowanych ryzyk, które mogą pojawić się w trakcie realizacji projektu, wcześniejsze opracowanie planu awaryjnego może w sposób istotny obniżyć koszty działań podejmowanych w reakcji na wystąpienie danego niekorzystnego zjawiska.

Kontrola ryzyka - jest to proces wdrożenia planu zarządzania ryzykiem, nieustannej obserwacji i nadzorowaniu zidentyfikowanych ryzyk, identyfikacji nowo powstałych zagrożeń oraz systematycznego oceniania skuteczności podejmowanych działań prewencyjnych. Celem monitorowania ryzyka jest ustalenie czy:

- wdrożono zgodnie z planem strategie reakcji na ryzyka,

- działanie podejmowane w ramach realizacji planów reakcji na ryzyko skutkują oczekiwanymi rezultatami,
- przyjęte założenia projektu są aktualne,
- podczas realizacji projektu nie doszło do zmian w szczególnym i ogólnym poziomie ryzyka,
- wystąpiły czynniki wyzwalające zidentyfikowane ryzyka,
- wystąpiły nowe ryzyka nierozpoznane uprzednio.

3.3. Zakres odpowiedzialności za realizację polityki zarządzania ryzykiem w procesie wytwórczym systemu informatycznego

Odpowiedzialność za prawidłowy przebieg procesu zarządzania ryzykiem ponoszą:

Zespół Nadzoru Projektu – odpowiada za zarządzanie ryzykiem na poziomie strategicznym poprzez:
- kształtowanie i wdrażanie polityki ds. zarządzania ryzykiem,
- wybór działań kontrolujących ryzyko w związku z występującymi ryzykami (unikanie, ograniczanie, transfer, akceptacja),
- wyznaczenie właścicieli ryzyk, na poziomie strategicznego zarządzania ryzykiem.

Komitet Sterujący – za zarządzanie ryzykiem na poziomie strategicznym, a w szczególności za:
- nadzór nad działaniami Kierownika Projektu w zakresie realizacji polityki zarządzania ryzykiem,
- okresowy monitoring poziomu ryzyka projektu oraz składanie raportów w tym zakresie Zespołowi Nadzoru Projektu,
- identyfikację, analizę i ocenę ryzyk kluczowych w podległych jednostkach organizacyjnych z punktu widzenia realizacji celów operacyjnych w danym obszarze,
- opracowywanie i wdrażanie mechanizmów kontrolnych,
- utworzenie i aktualizację rejestrów ryzyka,
- zapewnienie zgodności działania z obowiązującą polityką zarządzania ryzykiem,
- archiwizację dokumentów dotyczących realizacji polityki zarządzania ryzykiem,
- współpracę i utrzymywanie kontaktów z Zespołem Nadzoru Projektu lub Zespołem ds. zarządzania ryzykiem,
- zapewnienie, by wykonawcy byli świadomi wagi procesu zarządzania ryzykiem,
- zapewnienie wszystkim podległym pracownikom możliwości formalnego zgłaszania zmian w zakresie identyfikowanych przez nich ryzyk lub innych istotnych problemów,
- identyfikację potrzeb szkoleniowych dotyczących zarządzania ryzykiem,

Zespół ds. zarządzania ryzykiem lub Kierownik Projektu odpowiada za monitoring i przegląd systemów zarządzania ryzykiem w Projekcie, w szczególności za:

- analizowanie wyników działań podejmowanych przez Kierowników Faz cyklu życia projektu w celu identyfikacji ryzyka, analizy ryzyka oraz stosowania mechanizmów kontrolnych dotyczących ryzyka,
- przeprowadzanie oceny ryzyka, wykorzystując Rejestry ryzyka, sporządzone podczas identyfikacji ryzyka,
- identyfikację, analizę i ocenę ryzyka na poziomie strategicznym,
- dokonywanie we współpracy z Rektorem przeglądu działań podejmowanych w związku z występującym ryzykiem.

Kierownik projektu odpowiedzialny jest za zarządzanie ryzykiem na poziomie projektu. W szczególności powinien:

- identyfikować ryzyko, które jest właściwe dla charakteru projektu,
- dokonywać oceny (prawdopodobieństwo plus skutki wywołane danym ryzykiem) i hierarchizacji zidentyfikowanych ryzyk,
- projektować i wdrażać działania naprawcze redukujące ryzyko do poziomu akceptowalnego,
- archiwizować dokumentację z zakresu zarządzania ryzykiem w projekcie,
- w przypadku wystąpienia ryzyka, które może wywołać zagrożenie realizacji projektu, Kierownik projektu powinien o tym fakcie powiadomić właściwego przełożonego.

Koordynator ds. zarządzania ryzykiem odpowiada za:

- opracowanie i aktualizowanie polityki zarządzania ryzykiem,
- przedstawianie Zespołowi ds. zarządzania ryzykiem rejestrów ryzyka wraz z określonymi działaniami w celu poprawy systemu zarządzania ryzykiem,
- prowadzenie bieżącego monitoringu realizacji zadań dotyczących zarządzania ryzykiem i koordynowanie działań osób odpowiedzialnych za realizację,
- ocenę sposobu identyfikacji i zarządzania ryzykiem przez Kierowników poszczególnych zespołów projektowych oraz zapewnienie komunikacji pomiędzy wyżej wymienionymi osobami a Zespołem ds. zarządzania ryzykiem,
- odbieranie wyników działań podejmowanych w celu identyfikacji ryzyka, analizy ryzyka oraz stosowania mechanizmów kontrolnych dotyczących ryzyka,
- prowadzenie rejestru ryzyka oraz dokumentacji dotyczącej zarządzania ryzykiem,

Pozostali uczestnicy projektu są zobowiązani znać i przestrzegać politykę zarządzania ryzykiem, a w zakresie swoich kompetencji są zobowiązani do:

- monitorowania poziomu ryzyka w miejscu pracy,
- informowania przełożonych o wszelkich zdarzeniach, które mogą doprowadzić do negatywnych skutków w Projekcie, w tym o potencjalnych nowych ryzykach lub istotnych zmianach poziomu ryzyka.

4. Podsumowanie

Kształtowanie ryzyka to system metod i działań zmierzających do obniżenia ryzyka do poziomu akceptowalnego, przy uwzględnieniu kosztów działania oraz zabezpieczenia się w racjonalny sposób przed jego skutkami.

Proces kształtowania ryzyka we wszystkich fazach cyklu życia systemu informatycznego obejmuje grupy prac związane z analizą ryzyka i zarządzaniem ryzykiem. Analiza ryzyka obejmuje identyfikowanie i ocenę ryzyka oraz reagowanie na nie; proces zarządzania ryzykiem obejmuje ryzyko występujące we wszystkich procesach decyzyjnych i na każdym szczeblu zarządzania projektem.

Zarządzanie ryzykiem zapobiega nieoczekiwanym sytuacją; pozwala skoncentrować się na budowie właściwego rozwiązania już przy pierwszym podejściu; zapobiega pojawianiu się problemów, a w przypadku wystąpienia – ich eskalacji; pomaga w realizowaniu celów programu.

Ryzyko ograniczane powinno być na każdym etapie cyklu życia systemu poprzez zaprojektowanie i wdrożenie właściwych mechanizmów kontrolnych ustalonych na podstawie wyników monitoringu poziomu ryzyka oraz jego oceny, jak również podejmowanych działań mających na celu zmniejszenie skutków zaistniałych zdarzeń negatywnych, tj. planów awaryjnych.

Prowadzenie rejestru ryzyka i rejestru zagadnień, pozwala na usystematyzowanie zagrożeń i okazji, co w konsekwencji prowadzi do "lepszego" monitorowania ryzyka, eskalowania go na wyższe szczeble, jak również podejmowanie działań w odpowiednim czasie.

Głównym czynnikiem ryzyka jest brak znajomości możliwości i ograniczeń TI oraz traktowanie jej jako niezależnej od kontekstu organizacyjnego. Ryzyko zwiększają twórcy SI i dostawcy wyposażenia i oprogramowania oferując bez niezbędnej diagnozy potrzeb zaawansowana TI oraz kompleksowe SI.

Zagrożeń nie można uniknąć ani w pełni wyeliminować, ale można i należy zmniejszać ryzyko poprzez odpowiednie jego kształtowanie na wszystkich etapach cyklu życia przedsięwzięcia informatycznego. Za minimum, jakie można wskazać, trzeba przyjąć opracowanie polityki zarządzania ryzykiem i systemu zarządzania ryzykiem uwzględniającego czynniki ryzyka i powodzenia zamierzenia.

Literatura

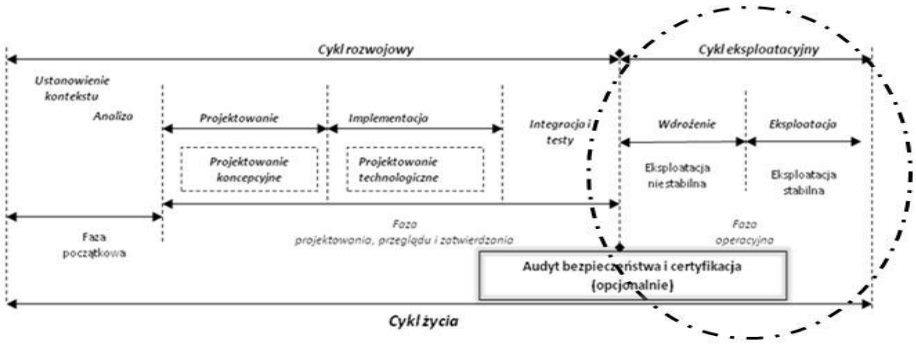
[1] Gleim I., CIA, Review - Part II-Conducting the Internal Engagement Certified Internal Auditors Gainesville FL, 2004
[2] Korczowski A., "Zarządzanie ryzykiem w projektach informatycznych. Teoria i praktyka", Wyd. Helion, Gliwice 2009
[3] Liderman K., „Analiza Ryzyka i ochrona informacji w systemach komputerowych”, Wyd. Naukowe PWN, Warszawa 2008
[4] Łuczak. J., „Metody szacowania ryzyka – kluczowy element systemu zarządzania bezpieczeństwem informacji ISO/IEC 27001”, Zeszyty Naukowe Akademia Morska w Szczecinie, 2009.
[5] Molski M., Łacheta M., „Przewodnik audytora systemów informatycznych”, Wyd. Helion, Gliwice 2007
[6] Prichard C.L., "Zarządzanie ryzykiem w projektach - teoria i praktyka", WIG-PRESS, Warszawa,2002.
[7] Ryba M., "Analiza i zarządzanie ryzykiem systemów informatycznych", Wykład firmy Ernst &Young, stan z 05.04.2005

Rozdział 3

Metodyka kształtowania ryzyka w cyklu eksploatacyjnym systemu informatycznego

1. Wprowadzenie

Ryzyko związane z systemami informatycznymi należy rozpatrywać w dwóch zupełnie różnych, lecz zależnych ujęciach. W pierwszym z nich, jest ono postrzegane jako ryzyko projektowe odnoszące się do zagrożeń występujących w cyklu rozwojowym systemu. Drugim podejściem jest badanie już wytworzonego i funkcjonującego systemu informatycznego w celu oszacowania poziomu zagrożeń występujących w fazie operacyjnej związanych z audytami informatycznymi, np. bezpieczeństwa i ewentualną certyfikacją oraz jego eksploatacją.
Miejsce fazy operacyjnej w cyklu życia systemu informatycznego pokazano na rysunku 1.



Rysunek 1. Miejsce fazy operacyjnej w cyklu życia systemu informatycznego. Źródło: Opracowanie własne.

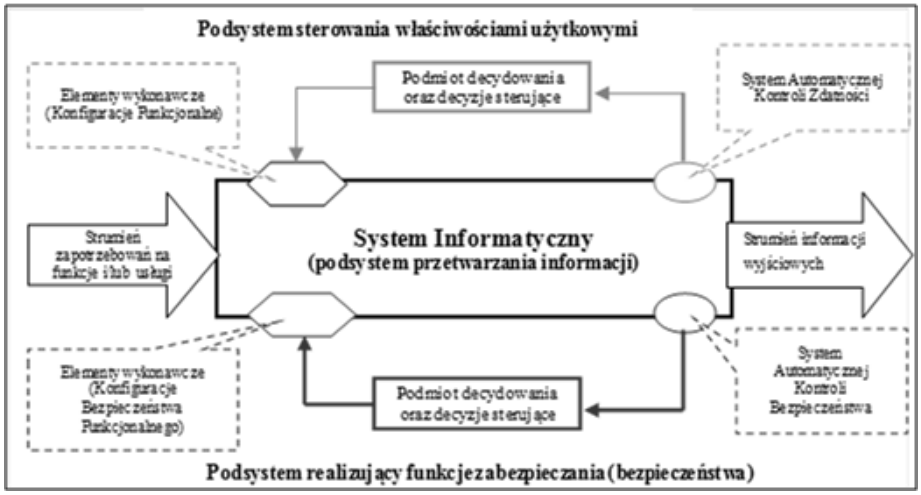
Na potrzeby niniejszej pracy jak i również na potrzeby zaproponowanej metodyki kształtowania ryzyka w fazie eksploatacji przyjęto, że faza operacyjna systemu informatycznego obejmuje następujące etapy:

- etap audytowania (audyt bezpieczeństwa i certyfikacji),
- etap wdrażania - eksploatacja niestabilna zwana często również jako faza przejścia lub faza eksperymentalno - użytkowa,
- etap eksploatacji - eksploatacja stabilna zwana często również jako faza użytkowania i doskonalenia.

Kształtowanie ryzyka w fazie eksploatacji systemu informatycznego praktycznie sprowadza się do analizy i zarządzania ryzykiem z punktu widzenia:

- sterowania jego właściwościami użytkowymi (aspekt jakości świadczenia usług przez SI na rzecz użytkownika końcowego),
- utrzymania bezpieczeństwa funkcjonalnego (aspekt zapewniania podstawowych atrybutów bezpieczeństwa w odniesieniu do kluczowych zasobów organizacji - podstawowe procesy biznesowe i zasoby informacyjne).

Graficzną ilustrację SI, z punktu widzenia sterowania jego właściwościami użytkowymi i utrzymania bezpieczeństwa zasobów informacyjnych przedstawiono na rysunku 2.



Rysunek 2. Ilustracja SI z punktu widzenia sterowania jego właściwościami użytkowymi i poziomem bezpieczeństwa funkcjonalnego. Źródło: Opracowanie własne.

Na rysunku tym wyróżniono trzy istotne elementy:

1. System Informatyczny, jako obiekt podlegający sterowaniu i doskonaleniu,
2. Podsystem sterowania właściwościami użytkowymi, w celu utrzymania jego funkcjonalności i jakości,
3. Podsystem realizujący funkcje zabezpieczania, w celu utrzymania wymaganego poziomu bezpieczeństwa podstawowych zasobów organizacji (kluczowych procesów biznesowych i zasobów informacyjnych).

Celem niniejszego artykułu jest próba skonstruowania (w wersji uproszczonej) metodyki kształtowania ryzyka w cyklu eksploatacyjnym (w fazie operacyjnej) systemu informatycznego uwzględniającej:

- poziom ryzyka szczątkowego - "resztkowego" z fazy implementacji,
- projekt mechanizmów bezpieczeństwa (systemu zabezpieczeń) opracowany na etapie implementacji lub wdrażania SI,

- zaimplementowane mechanizmy bezpieczeństwa w fazie budowy (implementacji) systemu,
- czynniki ryzyka pochodzące z procesu użytkowania i obsługiwania,
- czynniki ryzyka związane z procesem doskonalenia systemu.

Wyżej wymienione elementy stanowią wielkość wejściową do procesu zarządzania ryzykiem w fazie eksploatacji, której opis jest zamieszczony w kolejnych podrozdziałach niniejszego artykułu.

2. Koncepcja metodyki kształtowania ryzyka w fazie operacyjnej

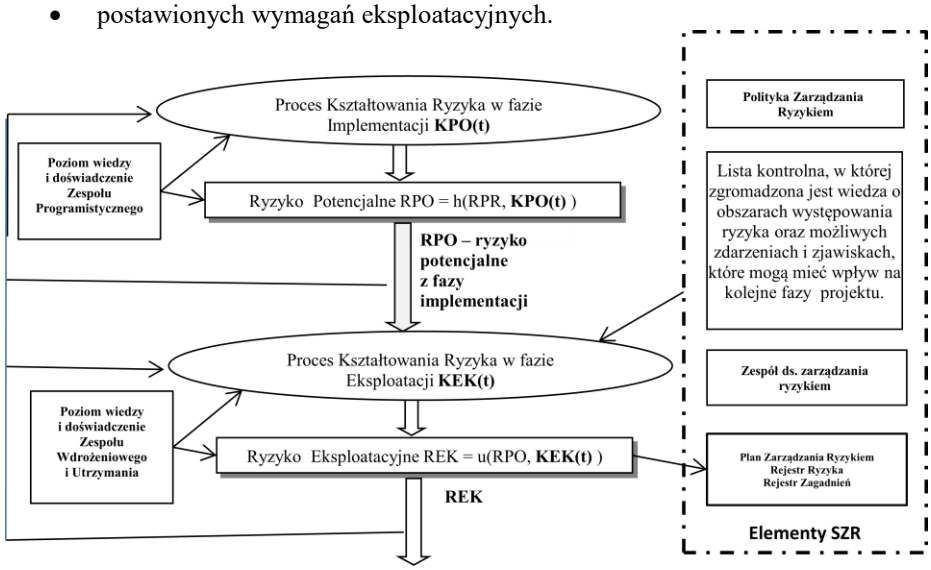
Proponowana w niniejszej pracy koncepcja metodyki kształtowania ryzyka w fazie operacyjnej zakłada, że zastosowany w niej proces kształtowania ryzyka [rys.3], bazuje na metodach i technikach analizy i zarządzania ryzykiem w podziale na następujące kategorie - atrybuty bezpieczeństwa SI:

- Użyteczność, w kategorii tej wyróżnia się ryzyko (czynniki ryzyka), iż zebrana informacja nie jest wykorzystywana lub okazuje się nieprzydatna, bądź też terminowość otrzymanej i opracowanej informacji jest niewystarczająca.
- Integralność, w kategorii tej wyróżnia się ryzyko (czynniki ryzyka), iż wykorzystywane dane i programy nie są wolne od błędów, nie zapewniają poprawności i kompletności informacji lub nie przedstawiają wiernie zdarzeń biznesowych; w szczególności wyróżnia się czynniki związane z utratą integralności w obszarach procesu, programu i danych.
- Poufność, w kategorii tej wyróżnia się ryzyko (czynniki ryzyka), iż:
    - użytkownicy systemu otrzymują prawa dostępu do systemu i danych, których nie potrzebują bądź otrzymać nie powinni,
    - dostęp do systemu lub danych , uznanych za poufne uzyskany zostanie w sposób nieautoryzowany.
- Dostępność, w kategorii tej wyróżnia się ryzyko (czynniki ryzyka), iż nie będzie zapewniany oczekiwany poziom dostępu do systemu i danych lub zachwiana zostanie zdolność systemu do przetwarzania danych na potrzeby kluczowych procesów biznesowych.
- Adekwatność, w kategorii tej wyróżnia się ryzyko (czynniki ryzyka), że kluczowe procesy informatyczne nie zapewniają w sposób efektywny odpowiedniego wsparcia dla kluczowych procesów organizacji.

Powyższe kategorie są rozszerzeniem o elementy ryzyka (atrybutów bezpieczeństwa informacji), zdefiniowanych w standardzie typu norma ISO/IEC 27002.

Ryzyko eksploatacyjne (REK) jest rezultatem:

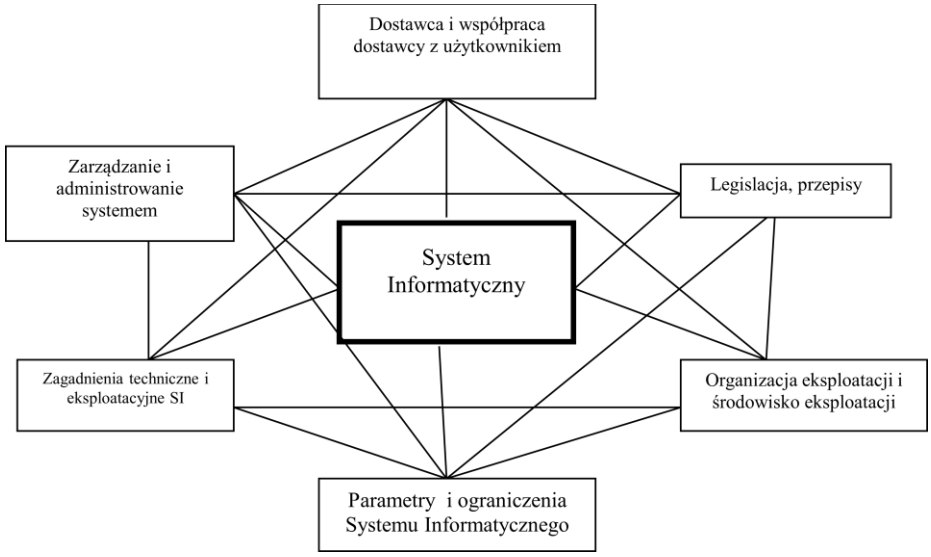
- wyboru procesu kształtowania ryzyka w fazie eksploatacji,
- poziomu ryzyka potencjalnego (RPO - ryzyko szczątkowe po zrealizowaniu fazy implementacji i/lub fazy wdrażania),
- wyboru warunków eksploatacji systemu informatycznego
- postawionych wymagań jakościowych i bezpieczeństwa informacji,
- postawionych wymagań eksploatacyjnych.



Rysunek 3. Współzależność ryzyka fazy implementacji i fazy eksploatacji w cyklu życia systemu informatycznego. Źródło: Opracowanie własne

Czynniki wpływające na ryzyko eksploatacyjne można podzielić na [rys.4]:

- czynniki związane z warunkami eksploatacji,
- czynniki związane z organizacją eksploatacji (tzn. z organizacją i jakością obsługi technicznej, diagnostyką, kwalifikacjami personelu obsługującego),
- czynniki związane z środowiskiem kształtowania ryzyka eksploatacyjnego.



Rysunek 4. Ilustracja powiązań obszarów czynników ryzyka w kontekście eksploatacji SI. Źródło: Opracowanie własne.

Osiągnięcie odpowiedniego poziomu ryzyka eksploatacyjnego zależy zarówno od rozwiązań przyjętych w trakcie implementacji systemu, jak i od poziomu organizacji eksploatacji, czyli przebiegu procesów użytkowania i obsługiwania elementów składowych systemu. Najlepiej zaprojektowany i oprogramowany system może nie przynieść spodziewanych efektów, jeżeli organizacja eksploatacji jest nieracjonalna i powoduje zwiększenie prawdopodobieństwa uszkodzenia systemu.
W czasie eksploatacji system informatyczny powinien być poddawany stałej obserwacji i ocenie z punktu widzenia czynników ryzyka mających wpływ na:

- funkcjonalność i sprawność działania,
- szeroko rozumianą jakość i efektywność,
- bezpieczeństwo funkcjonalne i informacyjne.

Przeprowadzane oceny powinny dać odpowiedź na cztery pytania:

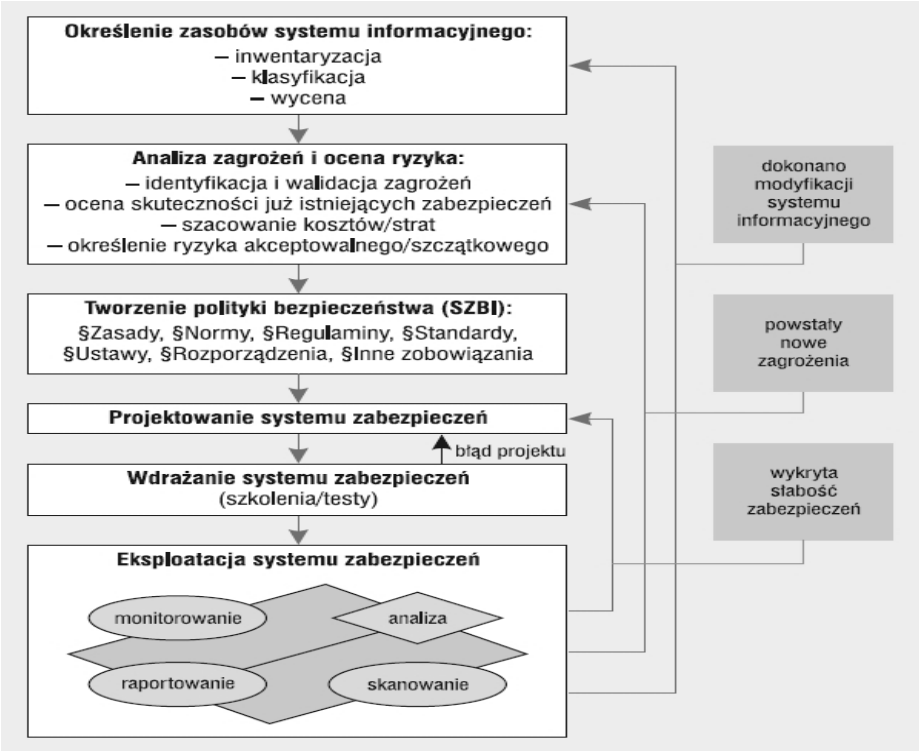
- czy system jest skuteczny, sprawny i ekonomiczny?
- czy system jest bezpieczny?
- co i jak można zmienić, usprawnić w systemie?
- czy wprowadzenie odpowiednich zmian w systemie będzie opłacalne?.

Wytworzony system informatyczny realizuje swoje cele i rozwiązania zdefiniowane we wcześniejszych fazach procesu jego powstawania. Aktualność tych rozwiązań po niezbędnym czasie na wytworzenie systemu, w dużej mierze zależy od inwencji autorów oraz zastosowanych metod i technik pracy, w tym również dotyczących kształtowania ryzyka. Oczywiste jest jednak, że system informatyczny będzie funkcjonował pewien czas, po którym konieczne będą zmiany i modyfikacje,

które najczęściej wynikają ze zmian w obiekcie użytkowania systemu lub z nowych koncepcji użytkownika. Konieczny więc jest etap doskonalenia systemu informatycznego, tak aby cały czas spełniał zmieniające się oczekiwania użytkownika. W związku z tym należy inaczej spojrzeć na sam proces tworzenia i eksploatacji systemu informatycznego, stosowane metody i techniki pracy i kształtowania ryzyka. Bardzo ważna staje się:

- bezpieczeństwo systemu informatycznego, a w zasadzie projekt systemu zabezpieczeń i jego implementacja [rysunek 4] , który decydująco wpływa na ryzyko i związane z nim bezpieczeństwo systemu informatycznego,
- jakość systemu oprogramowania, które decydująco wpływa na jakość i bezpieczeństwo całego systemu.

W tym momencie nasuwa się wniosek, że wybór odpowiedniej metodyki kształtowania ryzyka (metody, technik i narzędzi) w fazie operacyjnej systemu informatycznego ma nie tylko wpływ na bezpieczeństwo systemu czy sprawność procesu eksploatacji, ale również na jakość wytworzonego produktu.



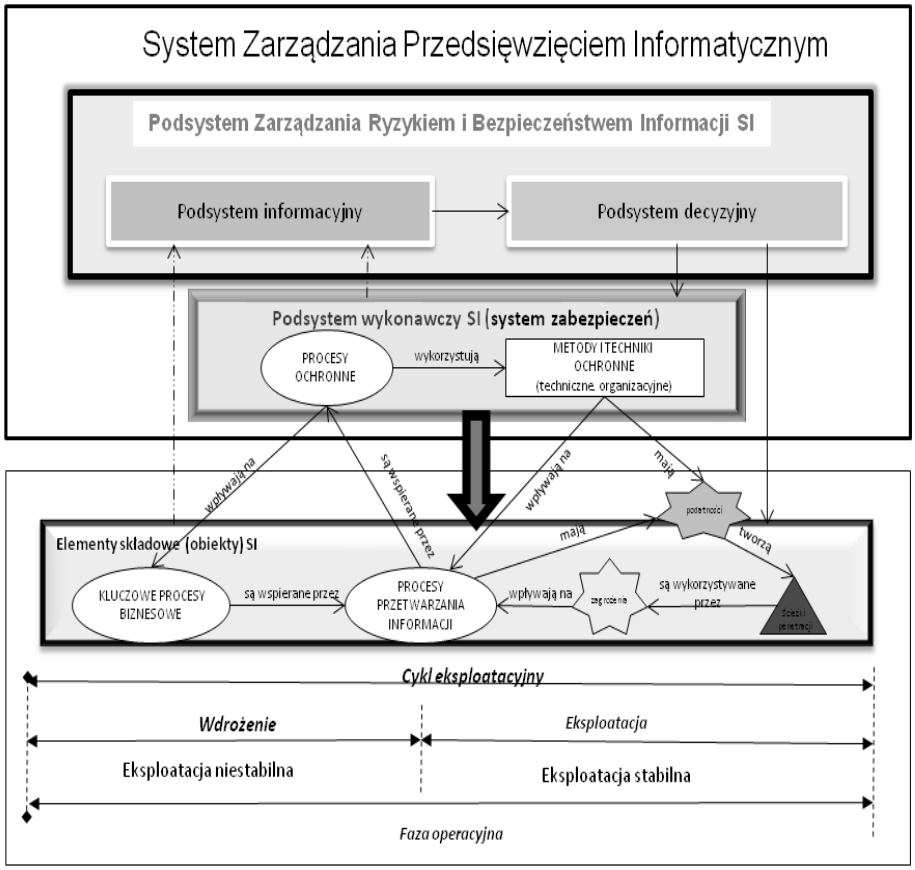
Rysunek 5. Przykładowy cykl życia systemu zabezpieczeń. Źródło: Opracowanie własne.

Ponieważ osiągnięcie odpowiedniego poziomu ryzyka eksploatacyjnego zależy przede wszystkim od rozwiązań przyjętych w trakcie budowy systemu zabezpieczeń, więc w dalszej części artykułu ten element został uszczegółowiony. Pozostałe elementy metodyki nie zostały scharakteryzowane w niniejszej pracy.

3. Charakterystyka procesu projektowania i implementacji systemu zabezpieczeń

Projektowanie systemu zabezpieczeń (mechanizmów bezpieczeństwa) dla systemu informatycznego to bardzo ważny etap w cyklu życia systemu ochrony, w szczególności bezpieczeństwa jego danych.

Celem etapu projektowania jest zaprojektowanie procesów ochronnych dla procesów przetwarzania informacji wspierającej krytyczne procesy biznesowe [rys.5]

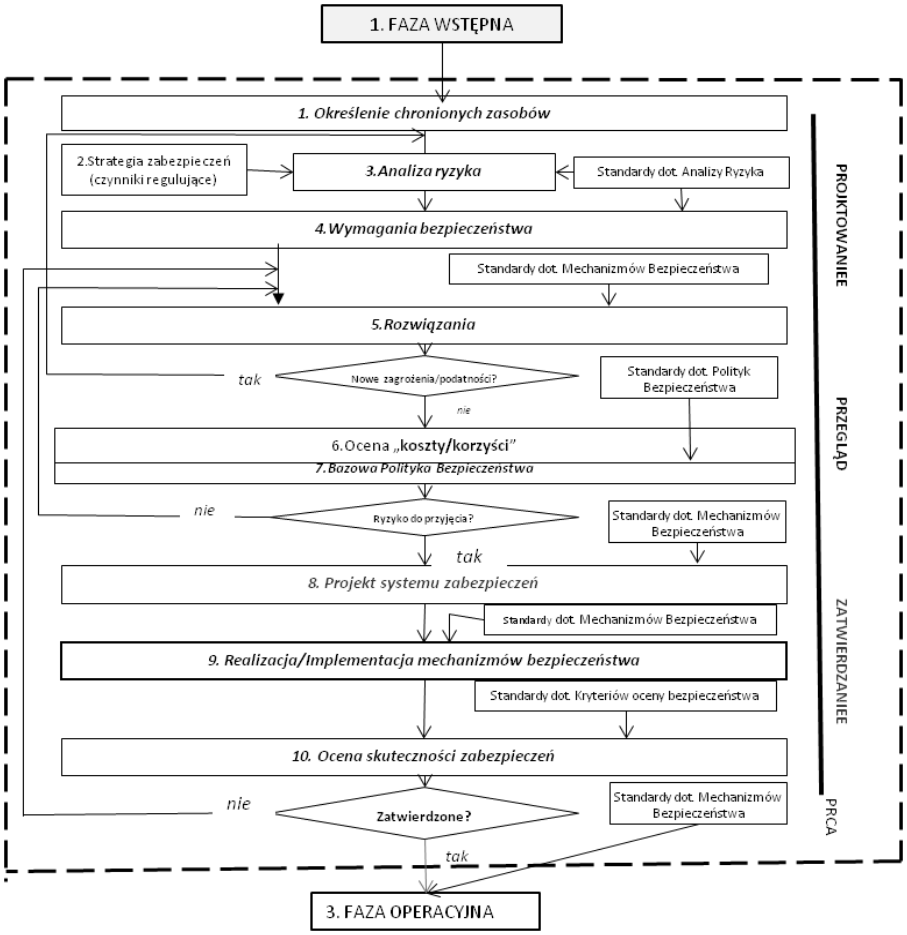


Rysunek 6. Ilustracja systemu informatycznego z punktu widzenia procesów ochronnych, procesów biznesowych i mechanizmów bezpieczeństwa. Źródło: Opracowanie własne.

Podstawowym zadaniem systemu zabezpieczeń jest zapewnienie podstawowych atrybutów bezpieczeństwa informacji, ponieważ ma ona wpływ na przebieg procesów biznesowych.

Cykl życia systemu zabezpieczeń obejmuje następujące kroki [rys 7].:

1. Faza wstępna,
2. Faza projektowania, przeglądu i zatwierdzania,
3. Faza operacyjna.



Rysunek 7. Podstawowe fazy cyklu życia systemu zabezpieczeń. Źródło: Opracowanie własne.

W ramach fazy wstępnej wykonywane są następujące podstawowe działania i czynności:

A. Na etapie ustanawiania kontekstu:

1. Wyłania się interdyscyplinarną grupę ekspertów (rzeczywisty przekrój wszystkich stron) sterującą całym procesem: analizy, projektowania, realizacji i obsługi zabezpieczeń w obszarze chronionym (np. w obszarze ochrony danych statystycznych).
2. Określenia się zakres procesu zarządzania bezpieczeństwem informacji, tak aby zapewnić, że przy szacowaniu ryzyka uwzględniono wszystkie odnośne aktywa.
3. Wyznacza się podstawowe kryteria (kryteria oceny ryzyka, kryteria skutków, kryteria akceptowania ryzyka) potrzebne do procesu zarządzania:
   - bezpieczeństwem w obszarze ochrony danych statystycznych,

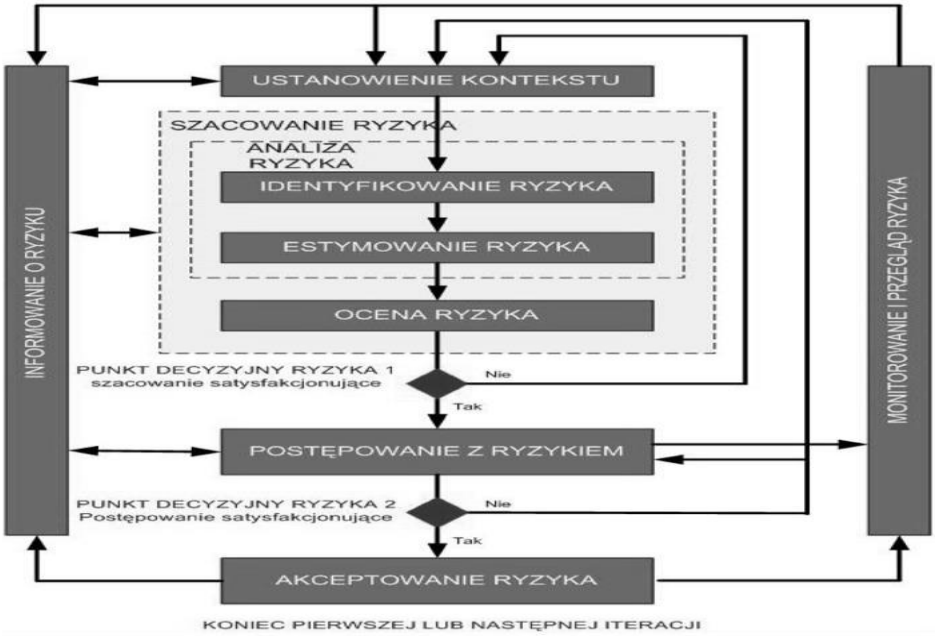
- przedsięwzięciem projektowania i budowy systemu mechanizmów bezpieczeństwa w celu zapewnienia tajności, integralności, niezaprzeczalności i dostępności przetwarzanej w chronionych systemach informacyjnych (np. baz danych statystycznych).

B. Na etapie analizy:

1. Identyfikowanie procesów kluczowych (krytycznych).
2. Określenie dla zidentyfikowanych procesów ich wrażliwości na zakłócenia we wspierających je procesach przetwarzania informacji.
3. Dla procesów krytycznych zidentyfikowanie wspierających je procesów przetwarzania informacji w systemach teleinformatycznych.
4. Dla każdego z procesów zidentyfikowanych jako krytyczne ustalić zbiór zagrożeń, które mogą doprowadzić do utraty tajności, integralnosci, niezaprzeczalności i dostępności przetwarzanych w nich informacjach.
5. Dla zidentyfikowanych procesów zidentyfikować zasoby w nich wykorzystywane.
6. Dla każdego zasobu z każdego procesu zidentyfikować podatności, które mogą być wykorzystane przez zagrożenia.
7. Na podstawie zbioru zasobów i zbioru podatności wyznaczyć ścieżki penetracji dla procesów wspierających.
8. Zidentyfikowanie ograniczeń w zakresie potencjalnych decyzji projektowych wynikających np. z przepisów prawnych, budżetu przeznaczonego na projekt i wykonanie systemu zabezpieczeń.

W ramach fazy projektowania, przeglądu i zatwierdzania wykonywane są następujące podstawowe działania i czynności:

1. Określenie chronionych zasobów - Określenie zasobów (aktywów), które podlegają ochronie i określenie poziomu ich zabezpieczeń. Należy wyróżnić dwa rodzaje aktywów:
   - Aktywa podstawowe: Procesy i działania biznesowe oraz Informacje;
   - Aktywa wspierające (na których opierają się podstawowe elementy z zakresu) wszystkich rodzajów: Sprzęt, Oprogramowanie, Sieć, Personel, Siedziba, Struktura organizacyjna.
2. Strategia zabezpieczeń:
   - Nakreślenie obszarów i celów zabezpieczeń,
   - Zdefiniowanie zakresu zabezpieczeń,
   - Utworzenie oficjalnego słownika pojęć dotyczących bezpieczeństwa.
3. Analiza ryzyka - przeprowadzona zgodnie z zaleceniami normy ISO/IEC 27005:2011 Technika informatyczna - Techniki bezpieczeństwa - Zarządzanie Bezpieczeństwem Informacji - Analiza ryzyka. Zakres działań przedstawia rysunek 8.



Rysunek 8. Proces zarządzania ryzykiem w bezpieczeństwie informacji. Źródło: Norma ISO/IEC 27005.

4. Wymagania bezpieczeństwa i Proponowane rozwiązania:
   - Wymagania bezpieczeństwa są określane po wykonaniu analizy ryzyka i przeanalizowaniu problemów natury prawnej,
   - Proponowane rozwiązania mają na celu spełnienie określonych wcześniej wymagań bezpieczeństwa,
   - Usługi i mechanizmy bezpieczeństwa mogą być wybrane z biblioteki technik ochrony informacji.
5. Ocena „koszty i korzyści" i Bazowa Polityka Bezpieczeństwa:
   - Opracowanie Rozwiązań - Zaproponowane rozwiązania są oceniane pod kątem kosztów i korzyści.
   - Opracowanie Bazowej Polityki Bezpieczeństwa - opracowanie aktualnego zbioru praw, zasad, reguł i sposobów, które regulują zarządzanie, przetwarzanie, użycie, zabezpieczenie rozpowszechnienie informacji i zasobów systemu informacyjnego.
   - Określenie Budżetu - Koszty określające wymagany/pożądany poziom bezpieczeństwa nie powinny przekraczać zysków z zastosowanych usług bezpieczeństwa.
6. Projekt zabezpieczeń - jest ostatecznym wynikiem działań przeprowadzanych w fazie projektowania, przeglądu i zatwierdzania. Zawiera:
   - Bazową Politykę Bezpieczeństwa (BPB),
   - Wyniki analizy ryzyka (Raport Analizy Ryzyka),

- Lista procesów ochronnych dla procesów przetwarzania informacji wspierających krytyczne procesy biznesowe,
- Dobór środków ochronnych (zwanych również mechanizmów bezpieczeństwa, zabezpieczeń) – do procesów ochronnych.

Zakres działań dla Fazy operacyjnej - eksploatacji przedstawia się następująco:

I. Wytwarzanie:
- Instalacja sprzętu i oprogramowania w systemie informacyjnym oraz systemów ochrony fizycznej i technicznej,
- Rekonfiguracja sieci/systemów na potrzeby systemu zabezpieczeń,
- Spisanie polityki, planu i procedur bezpieczeństwa teleinformatycznego,
- Opracowanie planów zapewniania ciągłości działania,
- Wdrożenie SZ i szkolenia.

II. Testowanie:
- Audyt bezpieczeństwa,
- Ocena na zgodność z ustalonym standardem,
- Testy penetracyjne.

III. Eksploatacja:
- Nadzór i kontrola w zakresie bezpieczeństwa (w tym audyty wewnętrzne i różnicowa analiza ryzyka),
- Likwidacja lub bezpieczeństwa,
- Szkolenie podstawowe,
- Przeglądy i aktualizacje dokumentacji bezpieczeństwa,
- Testowanie planów zapewniania ciągłości działania.

4. Podsumowanie

Problemy kształtowania ryzyka w cyklu życia systemów informatycznych, na które dotychczas nie zwracano zbyt dużej uwagi, stały się ostatnio przedmiotem badań zakrojonych na szeroką skalę. Ryzyko w bezpieczeństwie systemów informatycznych zależało od doświadczeń i umiejętności zarówno zespołu wykonawczego jak i użytkownika końcowego . Gdy koszt sprzętu komputerowego maleje, a koszty wytwarzania i wdrażania systemów informatycznych rosną (zwłaszcza relatywnie wysokie są koszty pielęgnacji i doskonalenia systemów), sprawa jakości, bezpieczeństwa i kształtowania ryzyka nabrała odpowiedniego wymiaru. Nowe metody i/lub metodyki wytwarzania systemów informatycznych np. podejście zwinne i metodyka SCRUM, kładą nacisk na tworzenie systemów wysokiej jakości i bezpieczeństwa.

Zarządzanie ryzykiem w systemach informatycznych na etapie ich eksploatacji jest obecnie fundamentem umożliwiającym podejmowanie skutecznych decyzji dotyczących ich bezpieczeństwa. Właściwy dobór mechanizmów bezpieczeństwa jest

szczególnym rodzajem przedsięwzięcia i oznacza złożone działanie, wielopodmiotowe, przeprowadzone na podstawie:

1. Wdrożonej technologii informatycznej w firmie,
2. Wyników analizy ryzyka i planu postępowania z ryzykiem.
3. Zatwierdzonej specyfikacji wymagań bezpieczeństwa.
4. Bazowej polityki bezpieczeństwa.

Podstawową fazą w cyklu życia systemu zabezpieczeń jest faza, przeglądu i zatwierdzania. Celem tej fazy jest doskonalenie procesów ochronnych dla procesów przetwarzania informacji wspierającej krytyczne procesy biznesowe. Podstawowe czynności procesu doskonalenia procesów ochronnych obejmują:

1. Dobór środków zarządzania bezpieczeństwem (struktur, dokumentów i procedur) – rozwiązania organizacyjne.
2. Dobór technicznych i programowych środków ochronnych do podatności.
3. Dobór zabezpieczeń sieciowych.
4. Zaprojektowanie sposobu zastosowania mechanizmów bezpieczeństwa (architektury rozwiązań).
5. Ocena ryzyka szczątkowego.
6. Przeprojektowanie systemu zabezpieczeń lub przejście do realizacji/implementacji mechanizmów bezpieczeństwa.

Podstawowe dokumenty fazy projektowania, przeglądu i akredytacji obejmują :

1. Strategia Zabezpieczeń (SZ),
2. Zweryfikowana Analiza Ryzyka (ZAR), na podstawie której określa się Wymagania Bezpieczeństwa,
3. Specyfikacja wymagań bezpieczeństwa (SWB),
4. Zaproponowane warianty rozwiązań spełniających wymagania (WR),
5. Ocena „koszty / zyski” dla zaproponowanych rozwiązań (OW).
6. Bazowa Polityka Bezpieczeństwa (BPB) dla przyjętego rozwiązania.
7. Dokumentacja projektu zabezpieczeń (DPZ).
8. Zaakceptowane dokumenty wynikowe.

Ze względu na specyficzny sposób realizacji fazy początkowej i fazy projektowania, przeglądu i akredytacji systemu zabezpieczeń, można zaryzykować stwierdzenie, że są one sterowane analizą ryzyka. Implikuje to m.in. posiadanie odpowiednich kwalifikacji przez zespół projektujący system zabezpieczeń – wiedza i doświadczenie z zakresu inżynierii oprogramowania i/lub inżynierii systemów mogą być w takim przedsięwzięciu niewystarczające.

Systematyczne powtarzanie analizy ryzyka pozwala na ustalenie wytycznych dotyczących rozwoju systemu zabezpieczeń oraz weryfikację skuteczności dotychczasowych działań ochronnych. Przeprowadzenie analizy ryzyka i zaprojektowanie systemu zabezpieczeń są podstawą do określenia optymalnego ekonomicznie poziomu ryzyka w systemie informatycznym.

Literatura

[1] Gleim I., CIA, Review - Part II-Conducting the Internal Engagement Certified Internal Auditors Gainesville FL, 2004
[2] Korczowski A., "Zarządzanie ryzykiem w projektach informatycznych. Teoria i praktyka", Wyd. Helion, Gliwice 2009
[3] Liderman K., „Analiza Ryzyka i ochrona informacji w systemach komputerowych", Wyd. Naukowe PWN, Warszawa 2008
[4] Łuczak. J., „Metody szacowania ryzyka – kluczowy element systemu zarządzania bezpieczeństwem informacji ISO/IEC 27001", Zeszyty Naukowe Akademia Morska w Szczecinie, 2009.
[5] Molski M., Łacheta M., „Przewodnik audytora systemów informatycznych", Wyd. Helion, Gliwice 2007
[6] Prichard C.L., "Zarządzanie ryzykiem w projektach - teoria i praktyka", WIG-PRESS, Warszawa,2002.
[7] Ross A., Inżynieria zabezpieczeń, Wydawnictwo Naukowo-Techniczne, warszawa 2005
[8] Ryba M., "Analiza i zarządzanie ryzykiem systemów informatycznych", Wykład firmy Ernst &Young, stan z 05.04.2005
[9] Information Systems Audit and Control Association Standard 050.050.030 - IS Auditing Guideline-Use of Risk Assesmenting in Audit Planning ISACA,2000
[10] Intrusion Detection Methodologies Demystified Enterrasys Networks Whitepaper, 2003

Rozdział 4

Zarządzanie kodem źródłowym systemów informatycznych

1. Wstęp

Paradoksalnie, choć kod źródłowy determinuje funkcjonowanie systemów informatycznych, często nie jest postrzegany jako istotny element aktywów przedsiębiorstw użytkujących te systemy i nie jest zarządzany w sposób jawny. W skrajnych przypadkach, być może wskutek niewystarczającej wiedzy zamawiających systemy informatyczne, umowy nie regulują szczegółowo kwestii praw własności do kodu źródłowego. W ten sposób zamawiający mimowolnie zapewniają dostawcy oprogramowania monopol na dokonywanie w nim zmian. W Polsce taką nieroztropność poznaliśmy na przykładzie decyzji urzędników ZUS-u.

Oczywiście zarządzanie kodem źródłowym sprowadza się nie tylko do aspektów prawnych i ochrony przed zagubieniem lub utratą nad nim kontroli. Konieczna jest także dbałość o jakość kodu źródłowego oraz podnoszenie „dojrzałości" procesu przemiany kodu źródłowego w działające systemy (ang. software process improvement). Wpływa to korzystnie na efektywność funkcjonowania departamentów IT w zakresie rozwoju i utrzymania systemów oraz na relacje z dostawcami, także w sensie pozycji negocjacyjnej. O tym w szczegółach poniżej.

Prawdopodobnie najtrudniejszym wyzwaniem stojącym przed menedżerami pragnącymi uporządkować powyższy obszar jest uzasadnienie rentowności niezbędnych zmian organizacyjno-technicznych. Konstruowanie uzasadnień biznesowych w obszarze zarządzania jakością, formułowanie matematycznych relacji pomiędzy dobrą jakością a wynikającymi z niej benefitami należy do zagadnień skomplikowanych.

Niniejszy artykuł oparto na doświadczeniach autora we wdrożeniu procesu zarządzania kodem źródłowym w dużej organizacji, użytkującej kilkaset systemów informatycznych. Można jednak - lub nawet należy - stosować zawarte w nim zalecenia także na potrzeby zarządzania kodem źródłowym pojedynczych systemów.

Celem artykułu jest przedstawienie zarówno zagrożeń wynikających z nieprawidłowego zarządzania kodem źródłowym jak i sposobów uniknięcia negatywnych skutków zmaterializowania się tych zagrożeń.

2. Czym dostawcy mogą „szachować" zamawiającego oprogramowanie?

Zdarza się, że zamawiający spotyka się z żądaniem zapłaty za coś, co zdaniem dostawcy „nie było w zakresie umowy". Oczekiwana zapłata warunkuje przekazanie kompletu

kodów. Jeśli nawet przekazano komplet kodu źródłowego, często nie wiadomo, jak go skompilować lub uruchomić w środowiskach testowych i produkcyjnych - nie przekazano odbiorcy narzędzi lub licencji do nich, niezbędnego know-how oraz wiedzy w obszarze zarządzania konfiguracją. Już tylko to potrafi skutecznie ograniczyć konkurencję w obszarze dostaw zmian w systemach informatycznych i wzmacnia pozycję negocjacyjną dotychczasowego dostawcy. Podobnie negatywnie odbija się słaba dokumentacja kodu źródłowego (brak komentarzy), jego nazbyt skomplikowana struktura (np. złożoność cyklometryczna) i braki w dokumentacji projektowej.

3. Jak ochronić się przed utratą kontroli nad kodem źródłowym?

Intuicja podpowiada, że podstawowymi wartościami podlegającymi ochronie są **kompletność i aktualność** kodu źródłowego, umożliwiające w sensie technicznym swobodny wybór dostawcy zdolnego go rozwijać.

Właściwą ich ochronę zapewnić można czyniąc zadość poniżej przedstawionym postulatom niezależnie od tego, w jak dużym stopniu zadania informatyczne w danych przedsiębiorstwach zostały zlecone zewnętrznym dostawcom:

- zadbanie o aspekty prawne – prawo do samodzielnej modyfikacji kodu źródłowego, jeśli jest to racjonalnie uzasadnione.
- przechowywanie kodu źródłowego we własnym lub przynajmniej niezależnym od dostawców oprogramowania *repozytorium kodu źródłowego* zwanym także *systemem kontroli wersji.*
- nadzorowanie procesu wytwórczego gwarantujące, że zawartość repozytorium kodu źródłowego jest zgodna co najmniej ze środowiskami produkcyjnymi. Może to zostać zrealizowane albo poprzez opisaną poniżej hermetyzację procesu zarządzania kodem albo przez cykliczne weryfikacje zgodności repozytorium ze środowiskami produkcyjnymi.
- przeprowadzania audytu (inaczej – inspekcji, przeglądu) jakości kodu źródłowego.

Poniżej opisano narzędzia mające nie tylko uczynić powyższe czynności mało kłopotliwymi dla obu stron, ale także zapewnić wysoką sprawność procesu.

Chronić powinno się konsekwentnie kody źródłowe wszystkich systemów, nie tylko tych określanych jako krytyczne. Zdarza się bowiem, że zaginięcie kodu pozornie nieistotnego systemu, który jest zintegrowany (często w nieudokumentowany sposób) z innymi ważnymi systemami, zablokuje lub przynajmniej bardzo utrudni i przedłuży ważną zmianę. Takiemu przypadkowi zawdzięcza autor możliwość zgromadzenia materiału do tego artykułu.

4. Prawo do kodu źródłowego

Zagadnieniom prawnym związanym z tworzeniem kodu źródłowego na zamówienie poświęcono liczne artykuły. Zwrócić uwagę należy przede wszystkim na fakt, że umowne przeniesienie praw do kodu źródłowego nie jest tożsame z uprawnieniem zamawiającego do dokonywania w nim zmian czy zlecania zmian podmiotom trzecim. Jeśli więc zamiarem zamawiającego oprogramowanie jest pozostawienie sobie dowolności w wyborze dostawcy (w tym dostawcy w zakresie utrzymania tworzonego sys-

temu w przyszłości), powinien w umowie precyzyjnie wyszczególnić tzw. pola eksploatacji. W przypadku braku szczegółowych postanowień o przeniesieniu autorskich praw majątkowych uważać się będzie, że twórca kodu udzielił zamawiającemu jedynie licencji [3].

Niekiedy spór o własność kodu źródłowego powstaje wewnątrz przedsiębiorstw, gdy jego stroną jest pracownik etatowy. Kluczową wtedy kwestią jest, czy stworzony przez pracownika kod źródłowy powstał w ramach jego obowiązku pracowniczego i czy powstał w godzinach pracy. Konflikt z pracownikiem kontraktowym (ang. outsource) podlega tym samym regulacjom co relacja np. z podmiotem prawa handlowego.

5. Elementy kodu źródłowego

Warto pamiętać, że przez kod źródłowy rozumie się wszystkie pliki, w których umieszczono jedną lub więcej instrukcji wykonywanych w określonym porządku przez komputer i determinujących funkcjonalność systemu informatycznego niezależnie od języka programowania - C/C++, Java, PL/SQL, PHP, HTML, itp. Do kodów źródłowych zalicza się także skrypty SQL, np. te pozwalające na wycofanie zmian do poprzedniej, sprawnej wersji produkcyjnej.

W ramach kodu źródłowego zamawiający powinien otrzymać także:

- dokumentację opisującą proces (sekwencję) kompilacji oraz opisującą niezbędne ku temu komponenty i narzędzia,
- zasoby niebędące kodem a niezbędne w procesie kompilacji oraz uruchamiania systemu, np. pliki typu properties, skrypty Ant/Maven, biblioteki wymagane do kompilacji,
- skrypty powłoki do kompilacji systemu lub jego modułów, a także służące do uruchamiania, konfiguracji, zarządzania i monitorowania pracy systemu, jeśli są specyficzne dla tego systemu,
- w uzasadnionych przypadkach także pliki binarne, na zasadach jak wyżej.

W tym miejscu bardzo istotne jest zrozumienie podziału kodu źródłowego na kod kompilowany i interpretowany. Ten pierwszy, zanim zostanie uruchomiony, podlega kompilacji – przetworzeniu na język zrozumiały przez procesory czyli tzw. kod wynikowy (wersja wykonywalna, kompilat). Kod interpretowany realizuje swoje zadanie dzięki interpreterom zainstalowanym w środowiskach uruchomieniowych. Ten podział jest o tyle istotny, że niestety technicznie możliwe jest dostarczenie zamawiającemu funkcjonującego rozwiązania informatycznego bez dostarczenia mu kodu źródłowego w części kompilowalnej. Hermetyzacja procesu wytwórczego opisana w pkt. 9 może temu zapobiec.

6. W czym przejawia się jakość kodu źródłowego?

Punktem wyjścia w dyskusji o jakości dostarczanego kodu źródłowego jest jego **kompletność** w sensie opisanym w pkt 5, pokrycie kodu źródłowego testami jednostkowymi (ang. unit tests; patrz poniżej) oraz **aktualność** rozumiana jako zgodność ze środowiskami produkcyjnymi. Zapewnienie tych dwóch atrybutów jest fundamentem prawidłowej dbałości o to ważne aktywo przedsiębiorstwa. Tu także przynależy **kompilowalność** kodu źródłowego. Jeśli bowiem nie potrafimy skompilować kodu źródłowego,

nie możemy rozwijać systemu informatycznego. Jego kod źródłowy może mieć dla nas wtedy wartość wyłącznie sentymentalną. Autor spotkał się z żądaniami dostawców zapłaty sporych kwot za przekazanie wiedzy nt kompilacji pojedynczych systemów. Zdawać trzeba sobie sprawę także z negatywnych skutków monopolizowania tej wiedzy (i nie tylko tej) przez własnych, pojedynczych pracowników.

Wspomniana powyżej kompletność kodu źródłowego obejmuje dodatkowo dwa aspekty, odnoszące się głównie do jakości procesu zarządzania nim:

- **kompletność scalenia** (ang. merging, łączenie) wszystkich zmian w kodzie, które były planowane na potrzeby danego wydania - scalenie z określonych gałęzi do głównej gałęzi rozwojowej. By nie zapominać o scaleniu zmian, często wokół repozytorium kodu źródłowego nadbudowuje się narzędzia do zarządzania kodem źródłowym.
- **kompletność funkcjonalna** kodu źródłowego oddanego do testowania - czy przed rozpoczęciem testów odbiorczych zaimplementowano wszystkie oczekiwane funkcjonalności? Niestety praktyka konstruowania kodu dopiero na etapie testów jest powszechna i wynika m. in. ze słabości zarządzania wymaganiami oraz z problemów z dostępnością deweloperów i testerów po stronie dostawcy. Jest to zjawisko patologiczne, z którym trzeba bezwzględnie walczyć, przedłuża ono bowiem testy. Przy sztywnych datach wydań implikuje nadto uruchomienie nieprzetestowanych funkcjonalności. Z pomocą repozytorium kodu źródłowego można łatwo, szybko i tanio mierzyć, jaki procent kodu źródłowego powstał w czasie testów. Przy przekroczeniu założonego progu można analizować przyczyny tego faktu i ryzyka z niego wynikające. Oczywiście powyższe możliwe jest pod warunkiem, że zapewnimy także hermetyczność (patrz poniżej) procesu tworzenia środowisk testowych, nie tylko produkcyjnych. Jest to swego rodzaju quick-win wdrożenia postulowanego poniżej procesu zarządzania kodem źródłowym.

Często pojęcie jakości kodu źródłowego kojarzone jest z jego zgodnością ze **standardami kodowania** - dobrymi praktykami programistycznymi. Są one opisane w formie reguł, którymi rządzić powinien się dobry kod źródłowy. Poziom zgodności to tzw. wewnętrzna jakość kodu źródłowego (ISO-9126-3, patrz [6]) w odróżnieniu od jakości zewnętrznej, którą obserwować można na podstawie zachowania uruchomionego systemu. W przeciągu kilkunastu ostatnich lat wypracowano standardy kodowania dla większości języków programowania. Dostępne są także narzędzia, w których te standardy zaimplementowano i które służą do wyłapywania błędów programistów. Są to tzw. statyczne analizatory kodu źródłowego.

W ujęciu dynamicznym jakość kodu źródłowego może być definiowana jako **procent pozytywnie zakończonych testów jednostkowych**. Są to liczne, acz małe fragmenty kodu źródłowego, których zadaniem jest weryfikacja poprawności działania kodu źródłowego. Uruchamianiem testów jednostkowych zajmują się zazwyczaj narzędzia tzw. ciągłej integracji (patrz akapit „Narzędzia").

Dość świeżym pojęciem w kontekście jakości kodu źródłowego jest *dług IT* (ang. IT debt, technical debt). To próba oszacowania, jak duży jest niezbędny wysiłek programistów, by usunąć z kodu źródłowego wszelkie naruszenia dobrych praktyk. Wysiłek ów przelicza się na pieniądze, czyli rzeczony dług, który tkwi w kodzie. Metodyka wyliczenia tego wskaźnika jeszcze raczkuje.

7. „Jeśli czegoś nie potrafisz zmierzyć, nie jesteś w stanie tym zarządzać"

Na potrzeby zarządzania jakością kodu źródłowego opracowano wiele mierników, m.in. ISO/IEC 25040 [7] i SEI maintainability index [10]. Szczególnie interesujące mogą być te, które w sposób automatyczny wyliczane są przez statyczne analizatory kodu źródłowego (patrz poniżej „Standardy kodowania"). Równie istotne są także miary określające jakość i efektywność samego procesu zarządzania kodem źródłowym. Oto kilka przykładów:

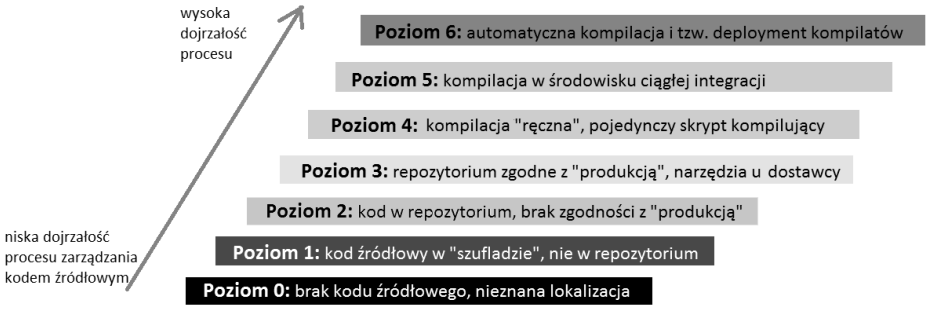
- procent kodu źródłowego wytworzonego podczas testów UAT (ang. user acceptance tests). Miernik określa **jakość testowania zmian** i można go oprzeć np. na porównaniu wielkości raportu zmian w repozytorium kodu źródłowego dla fazy konstrukcji i dla fazy testów. W przypadku narzędzia Subversion jest to instrukcja 'svn diff'.
- **aktualność repozytorium** kodów źródłowych – procent systemów, dla których znana jest lokalizacja kodu źródłowego aktualnej wersji produkcyjnej. Można go mierzyć porównując datę wersji wskazanej w bazie konfiguracji (ang. configuration management data base - CMDB) z datą ostatniego wydania, w którym wykonano zmianę w tym systemie. W CMDB zazwyczaj wskazuje się charakter zmiany - czy zmiana polegała na zmianie konfiguracji czy zmianie w kodzie źródłowym. Trzeba w tym miejscu podkreślić kluczową rolę prawidłowej inwentaryzacji systemów i zmian w nich dokonywanych dla zarządzania kodem źródłowym.
- **kompletność kodu źródłowego** – choćby wskazując procent systemów, w których zachowano hermetyczność procesu wytwórczego tj. zmiany w systemie oparte są na repozytorium kodów będącym pod kontrolą zamawiającego zmianę.

Autor na potrzeby transformacji procesu zarządzania kodem dużego zbioru systemów opracował miarę dojrzałości procesu zarządzania kodem źródłowym opisującą stopień jego hermetyzacji. W ten sposób możliwe było przejrzyste (np. przy pomocy sugestywnych kolorów), liczbowe raportowanie postępu upragnionych zmian organizacyjnych, podawanie wartości per departament i ich agregację. Polegała ona na przypisaniu liczby do atrybutów jakościowych opisujących osiągnięcie ważnego kamienia w projekcie, jak poniżej:

- poziom 0: brak kodu źródłowego, nieznana lokalizacja kodu źródłowego
- poziom 1: kod źródłowy w „szufladzie", na serwerze, nie w repozytorium
- poziom 2: kod źródłowy w repozytorium pod kontrolą wersji, znana lokalizacja odnotowana w CMDB, jednak bez możliwości skompilowania kodu lub bez pewności, że jest on zgodny z kodem wersji produkcyjnej (np. środowiska są stawiane z repozytorium dostawcy). Wymagane są testy regresji przed odtworzeniem środowiska produkcyjnego.
- poziom 3: system produkcyjny stawiany jest na podstawie repozytorium zamawiającego (proces jest nadzorowany) ale kompilacja nie jest wykonywana „w murach" zamawiającego lub narzędziami zamawiającego lub kompilacja jest wieloetapowa i jest wykonywana ręcznie. Weryfikowane są licencje bibliotek dostarczanych wraz z kodem źródłowym - patrz niżej „Hermetyzacja procesu wytwórczego".
- poziom 4: kompilacja (ang. build) systemu może odbywać się „ręcznie" lecz dokonuje się to w środowiskach zamawiającego a nie dostawcy, nadto istnieje

pojedynczy skrypt budujący system czyli wiedza o procesie kompilacji jest zaszyta w formie algorytmu, nie tylko w głowie developera. Tak wyprodukowane kompilaty są uruchamiane w środowiskach produkcyjnych.

- poziom 5: kompilacja jest wykonywana przez tzw. środowisko ciągłej integracji (patrz niżej „Podstawowe narzędzia"). To ważny etap zarówno ze względu na wyeliminowanie czynnika ludzkiego z procesu kompilacji jak i powtarzalność procesu.
- poziom 6: środowisko produkcyjne jest „stawiane automatycznie", wersja wykonywalna (kompilat) jest dystrybuowany przez automat (skrypt). To kolejny krok w pozyskaniu wiedzy zaszytej w głowach osób zaangażowany w proces wytwarzania oprogramowania służący także zwiększeniu efektywności operacyjnej.



Rysunek 1. Poziomy dojrzałości procesu zarządzania kodem źródłowym

8. Standardy kodowania

Dla większości języków programowania, w tym tych najpopularniejszych – Java, C/C++, PHP, .Net – opracowano zbiory reguł, których przestrzeganie ma pomóc uniknąć problemów wydajnościowych, uczynić kod czytelniejszym, łatwiejszym do modyfikacji, odporniejszym na ataki hackerów. Reguły te zaimplementowano w narzędziach, które w sposób automatyczny przeglądają kod źródłowy i wskazują naruszenia owych dobrych praktyk. Ich zastosowanie pozwala wykryć błędy na wczesnych etapach testów. Błędy wtedy wykryte można szybciej i taniej usunąć od razu nie czekając, aż wystąpi problem podczas produkcyjnej eksploatacji systemu. Pamiętać należy, że statyczna analiza nie wykryje błędów w logice oprogramowania i nie zastąpi tzw. testów end-to-end, czyli weryfikujących działanie procesów biznesowych.

Na rynku dostępne są narzędzia do statycznej analizy zarówno typu open-source, wśród których błyszczy Sonar [11], jak i komercyjne, których interesującym przykładem jest CAST [4]. Największą zaletą narzędzi dostępnych bezpłatnie jest to, że mogą być użytkowane przez każdego potencjalnego dostawcę na etapie konstrukcji kodu, zatem poprawianie błędów programistów nie odbywa się kosztem testów funkcjonalnych wykonywanych na środowiskach zintegrowanych u zamawiającego. Kilka cech narzędzi komercyjnych sprawia, że nie stoją one na pozycji przegranej:

- znacznie większe spectrum dostępnych technologii - PeopleSoft, WebMethods, PowerBuider, Oracle Forms, Business Objects, ...
- analiza zależności pomiędzy językami / technologiami (w tym martwy kod) –

np. odwoływanie się z poziomu Java do nieistniejącej tabeli z bazie danych
- obszerniejszy zestaw agregatów metryk – dług IT, backfired function points,
- rozbudowana funkcjonalność planów naprawczych (ang. action plan), minimalizująca pracochłonność ich obsługi
- kartografia systemu wspierająca zmiany architektoniczne lub wdrażanie nowych programistów
- możliwość oceny pracochłonności projektowanych zmian

Narzędzie CAST agreguje niskopoziomowe naruszenia i prezentuje je w postaci tzw. mierników zdrowia kodu źródłowego (ang. health factors) o wartości od 1 do 4, które opisują jego jakość w kontekście różnych biznesowych wymagań. Przedstawiono je w poniższej tabeli (za CAST Software SA).

Tabela 1. Czynniki zdrowia systemu informatycznego wg CAST Software (ang. health factors)

Miernik	Opis	Przykładowy benefit „biznesowy”
Transferability	Łatwość wdrożenia nowych programistów do kodowania danego systemu	- redukcja uzależnienia od jednego dostawcy - mniejsze koszty szkoleń programistów
Changeability	Łatwość i szybkość wykonania zmiany w systemie	- zmniejszenie wysiłku/kosztów niezbędnych do wykonania zmiany w systemie - większa „zwinność” procesu projektowego
Robustness	Stabilność systemu i prawdopodobieństwo wystąpienia defektu w konsekwencji wykonania zmiany	- zwiększenie dostępności systemu
Performance	Wydajność	- lepszy tzw. „user experience”
Security	Bezpieczeństwo	- ochrona wrażliwych informacji

W Polsce z narzędzia CAST korzysta Orange.

9. Hermetyzacja procesu wytwórczego

Jedną z technik stosowanych do zapewnienia aktualności i kompletności kodu źródłowego jest samodzielne stawianie przez zamawiającego środowisk produkcyjnych z własnego repozytorium, względnie realizowanie tego zadania przez zaufaną stronę trzecią. Pozwala to także uniknąć wdrożenia nieautoryzowanych lub nieprzetestowanych zmian w środowiskach produkcyjnych.

Zatem postuluje się, by:

1. Dostawca oprogramowania został zobowiązany umownie do dostarczenia kompletnego kodu źródłowego (vide definicja) wraz z jednoznaczną dokumentacją kompilacji.
2. Dostawca po fazie konstrukcji kodu źródłowego zasilił własnoręcznie lub

automatycznie zreplikował własne repozytorium do repozytorium zamawiającego np. via VPN. Zagadnienie, na jakich etapach projektów ma być dostarczany kod oraz czy środowiska testowe mają być także poddane reżimowi tego procesu, zależy od konkretnej organizacji i sytuacji.

3. Dostawca przekazuje lokalizację kodu źródłowego zamawiającemu (dla Subversion w formacie URL@REVISION). Lokalizację poszczególnych wersji kodu źródłowego należy przechowywać w odpowiedniej bazie danych CMDB. To detal o tyle istotny, że w repozytorium mogą znajdować się setki gałęzi kodu źródłowego, a np. wycofanie zmiany ze względu na wystąpienie nieprzewidzianych, negatywnych konsekwencji jej wdrożenia wymaga znajomości lokalizacji poprzedniej wersji kodu źródłowego.
4. Kod jest pobierany z repozytorium ze znanej lokalizacji pod nadzorem osoby lojalnej względem zamawiającego.
5. Kod jest kompilowany pod nadzorem zamawiającego, weryfikowana jest sekwencja kompilacji, tworzone są wersje wykonywalne.
6. Kod jest pod nadzorem przenoszony na środowiska testowe/produkcyjne. W przypadku często spotykanej automatyzacji procesu, zamawiający jedynie wpisuje do narzędzia ciągłej integracji lokalizację i punkty 3-5 przebiegają bez udziału człowieka. Powtórzmy - tylko wytworzony w pkt. 4 kompilat oraz kod interpretowany mogą być podstawą do tworzenia środowisk produkcyjnych, a tam gdzie to tylko możliwe i opłacalne – także środowisk testowych (patrz wcześniej miernik „jakość testowania zmian").
7. W dalszych etapach weryfikowana jest funkcjonalna zgodność systemu z wymaganiami i nanoszone są poprawki do kodu – proces iteracyjnie kontynuowany jest od pkt. 2.

Jeśli wyżej wspomniany proces realizujemy przynajmniej dla środowisk produkcyjnych, uzyskujemy pewność posiadania aktualnego i kompletnego kodu źródłowego. Z jednym wyjątkiem - proces ów nie gwarantuje dostarczenia kodu źródłowego do bibliotek innych niż open-source (z wyłączeniem komercyjnych bibliotek stron trzecich), dla których nie dostarczono kodu źródłowego, a w których zaszyta jest zamówiona funkcjonalność. Tu niezbędna jest ingerencja człowieka – każdorazowa weryfikacja zmian w bibliotekach niezbędnych na potrzeby kompilacji. Niestety czasem odkrywamy taki brak wiele lat po zmianie, gdy trzeba zmienić funkcjonalność zaszytą w danej bibliotece. Często wtedy na rynku nie ma już podmiotu – autora kodu do tej biblioteki lub umowa uwalnia go od zobowiązań. Taką bibliotekę trzeba często implementować od nowa. Nie zawsze spowodowane jest to złą wolą dostawcy oprogramowania, on także może mieć luki kompetencyjne lub zarządcze, podobnie jak jego poddostawcy.

10. Alternatywa dla hermetyzacji

Najbardziej popularną alternatywą dla wdrożenia wyżej opisanego podejścia do zarządzania kodem źródłowym (jako metody osiągnięcia kompletności i aktualności kodu) jest wykonywanie w określonych interwałach porównania zawartości archiwum kodu ze środowiskami produkcyjnymi. Jeszcze innym rozwiązaniem jest periodyczne odtwarzanie środowisk produkcyjnych z repozytorium np. pod koniec okresu trwania umowy wiążącej zamawiającego z dostawcą. Powyższe metody umożliwiają wyzbycie się utrzymywania nadmiernych kompetencji w murach zamawiającego.

Jednakże:

- takie ad-hoc akcje wykonywać lub nadzorować musi kompetentna, niezależna strona 3-cia, zazwyczaj wysoko opłacani konsultanci,
- ze względu na specyfikę technologii nie zawsze udaje się określić zgodność kodu wynikowego (kompilatów), nawet jeśli wytworzone są z tej samej wersji kodu źródłowego. Rozwiązanie w postaci odtworzenia środowiska produkcyjnego ad-hoc z repozytorium nie jest zazwyczaj akceptowane przez „biznes" ze względu na ryzyko przestoju i wymaga wykonania kosztownych testów regresji
- sprawnie, powtarzalnie przebiegający proces, poddany samo-optymalizacji, w którym ludzie działają w quazi-automatyczny, wyuczony sposób, jest zazwyczaj znacznie mniej kosztowny (a często wręcz niezauważalny) niż jednorazowe audyty, wymagające eskalacji i zakłócające przebieg innych procesów gospodarczych funkcjonujących w przedsiębiorstwie.
- postulowane techniki i narzędzia nie implikują tworzenia nowego bytu procesowego, lecz jedynie przesuwają już funkcjonujący proces w sensie nadzoru, ról i narzędzi w stronę zamawiającego. Wynikające z tego koszty są już ponoszone i zamawiający za nie płaci (dostawca zazwyczaj nie udostępnia filantropijnie zasobów dyskowych na repozytorium) a istotą postulowanej zmiany jest głównie kwestia lojalności osób nadzorujących proces.

11. Podstawowe narzędzia do zarządzania kodem źródłowym

Rozwój inżynierii oprogramowania w zakresie zarządzania kodem źródłowym zaowocował powstaniem wielu narzędzi, najczęściej typu open-source. Co istotne – absolwenci uczelni technicznych mają ich dobrą znajomość. To, uzupełnione o łatwość użytkowania i administrowania, czyni realizację postulatu hermetyzacji procesu zarządzania kodem źródłowym mało kosztownym i szybkim we wdrożeniu. Warto przypomnieć, że sposób wykorzystania w/w narzędzi stanowi istotne know-how i roztropnym jest, by strona zamawiająca kod źródłowy utrzymała nad nimi kontrolę.

Kluczowym narzędziem jest **repozytorium kodów źródłowych** (ang. version control, revision control), czyli archiwum kodu umożliwiające śledzenie zmian w nim zachodzących oraz zapobiegające nieautoryzowanej zmianie w historycznych wersjach kodu. Mamy tutaj do wyboru wiele darmowych i dojrzałych narzędzi - CVS, Git i chyba najbardziej popularny Subversion/SVN. Fakt ich bezpłatnej dostępności jest istotny także dlatego, że większość dostawców już z nich korzysta, umożliwiając np. automatyczną synchronizację z repozytorium zamawiającego.

Posiadanie **narzędzi deweloperskich,** umożliwiających wykonanie kompilacji kodu źródłowego nie tylko pozwala zamawiającemu na wykonanie drobnych poprawek produkcyjnych własnymi siłami. Samo wykonanie kompilacji powinno być elementem weryfikacji jakości kodu przed jego formalnym odebraniem. Jest to tak istotne, że powtórzone zostało w kilku akapitach poniżej. Z doświadczeń autora wynika bowiem, że około połowy wersji dostarczonych do repozytorium z jakiś względów nie kompiluje się. Fakt częstego występowania problemów z kompilacją był jednym z powodów wyodrębnienia się w inżynierii oprogramowania obszaru o nazwie **ciągła integracja** (ang. continous integration). Narzędzia takie jak Hudson/Jenkins (patrz [5] i [8]) i Artifactory [12], wskazane w umowie zamawiającego z dostawcą jako narzędzie wery-

fikacji i odbioru, pomagają uczynić dokumentację kompilacji kodu źródłowego jednoznaczną. Zamiast nieprecyzyjnego dokumentu Word operujemy plikami XML. Unikamy jakże częstego „u nas się kompiluje". A przecież ma się kompilować w murach zamawiającego, nawet jeśli większość prac przy konstrukcji zadania (ang. job) w Hudsonie wykonuje dostawca! Jak wspomniano wcześniej - te narzędzia często są już wykorzystywane przez dostawców, głównie by nadzorować prace ... poddostawców i własnych pracowników.

Ostatnim elementem w tej narzędziowej układance służącej hermetyzacji procesu jest automatyzacja przeniesienia kodu źródłowego i kompilatów do środowisk uruchomieniowych (ang. **deployment**). Najczęściej służą do tego dedykowane skrypty.

12. Business Case

Usprawnianie procesu wytwarzania oprogramowania jest zagadnieniem nie tylko inżynierskim. W wielu organizacjach gospodarczych jest ono realizowane w formie projektu i wymaga uzasadnienia ekonomicznego. W Internecie znaleźć można liczne prace na temat szacowanej stopy zwrotu z inwestycji w takich przedsięwzięciach, które wykorzystać można do zbudowania uzasadnienia biznesowego.

Badania empiryczne i doświadczenie autora wskazują, że:

- co najmniej 5% błędów podczas testów jest bezpośrednio związanych z naruszeniem dobrych praktyk programowania. W kontekście bardzo wysokich kosztów przeprowadzenia testów w środowiskach zintegrowanych i wysiłków zmierzających do skrócenia time-to-market procent ten uzasadnia poniesienie dodatkowych wydatków na audyt jakości kodu. Wydatki te są niższe niż koszty zaniechania obsługi błędów.
- analogicznie, około 5% czasu przestojów systemów produkcyjnych zawdzięczamy niechlujstwu programistów, które można wcześnie wykrywać z pomocą statycznej analizy kodu,
- w dużych organizacjach od czasu do czasu zapada decyzja, by stworzyć od zera przynajmniej jeden z kilkudziesięciu dużych systemów informatycznych - ze względu na problemy z kodem źródłowym: niekompletność kodu, złą jakość kodu lub architektury, niszową technologię wykonania (wybraną przez dostawcę by związać klienta technologią?), podnoszące się koszty rozwoju i utrzymania. Tych problemów można uniknąć poprzez prawidłowe zarządzanie kodem źródłowym. Wielu z czytelników zapewne przynajmniej słyszało o takim przypadku i ma wiedzę na temat kosztów takiego wdrożenia, które mogą stanowić liczbowy wkład do biznesowego uzasadnienia projektu.

Bardzo trudno wykazać jednoznacznie kwotowo, jakie oszczędności w stawkach wynagrodzeń wykonawców przynosi odzyskanie kontroli nad kodem źródłowym, w tym możliwość wymiarowania zmian. Zazwyczaj jednak działy zakupów dysponują własnym eksperckim doświadczeniem, na którym można oprzeć mniej lub bardziej wiarygodne szacunki.

Te liczby mogą posłużyć do zbudowania modelu oszczędności. Po stronie kosztów realizacji postulowanych przez autora zadań w ramach procesu zarządzania kodem źródłowym znajdują się przede wszystkim wynagrodzenia zaangażowanych specjalistów nadzorujących proces, gdyż w większości przypadków można skorzystać z bezpłatnych narzędzi, zaś wymagania sprzętowe są niewielkie.

13. Pominięte zagadnienia

Z zarządzaniem kodem źródłowym związanych jest wiele innych interesujących acz pominiętych tutaj aspektów, np.:

- organizacja pracy w repozytorium, włączając zagadnienia tak drobiazgowe jak dozwolone kierunki propagowania poprawek błędów (ang. bug-fix) w gałęziach projektów i samo nazewnictwo gałęzi,
- ponowne użycie kodu źródłowego pozwalające uniknąć duplikowania funkcjonalności i zmniejszyć koszty rozwoju systemów informatycznych,
- pomiar rozmiaru zmian w oprogramowaniu (ang. backfired function points), weryfikacja rzeczywistego kosztu zakontraktowanej zmiany,
- wpływ monitorowania zmian w kodzie źródłowym na wzrost efektywności pracy - efekt Hawthorne [2],
- wybór technologii / języka programowania na potrzeby wytworzenia oprogramowania,
- zarządzanie licencjami open-source, w tym statyczna analiza kodu (w Orange wykorzystywana jest Palamida [9]).

Także one mogą generować ryzyko dla poprawnego funkcjonowania przedsiębiorstwa, jak i stanowić źródło przewagi konkurencyjnej.

14. Podsumowanie

Zabezpieczenie kodu źródłowego przed utratą kontroli nad nim nie może polegać jedynie na dokonaniu odpowiednich zapisów w umowie pomiędzy dostawcą i zamawiającym. Bardzo ważne jest weryfikowanie jakości dostarczanych produktów, zabezpieczenie ich prawidłowej dokumentacji (np. sekwencji kompilacji) i organizacja całego procesu wytwórczego, w szczególności nadzór przez kompetentnych i lojalnych względem zamawiającego specjalistów. Jednakże zarządzanie kodem nie jest tylko zagadnieniem inżynierskim. Wynikają z niego niewspółmierne do kosztów korzyści finansowe – możliwość zachowania konkurencji w obszarze dostaw oprogramowania i wywarcia presji na wartość ofert dostawców.

Literatura

[1] DeMarco, T. (1982). *Controlling Software Projects: Management, Measurement, and Estimates.* Upper Saddle River, NJ, USA: Prentice Hall., s. 3
[2] Landsberger, H. A. (1958). *Hawthorne Revisited: Management and the Worker, Its.* Ithaca: Cornell University.
[3] Ustawa z dnia 4 lutego 1994 r o prawie autorskim i prawach pokrewnych, Dz. U. 1994, nr 24, poz. 83 z późn. zm.
[4] http://www.castsoftware.com, 2015-07-08
[5] http://www.hudson-ci.org, 2015-07-08
[6] http://www.iso.org/iso/catalogue_detail.htm?csnumber=22891, 2015-07-08
[7] http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=35765, 2015-07-08
[8] http://wiki.jenkins-ci.org, 2015-07-08
[9] http://www.palamida.com, 2015-07-08
[10] http://www.sei.cmu.edu/reports/97hb001.pdf, 2015-07-08
[11] http://www.sonarsource.org, 2015-07-08
[12] http://www.jfrog.com/artifactory, 2015-07-08

II. Zapewnienie jakości oprogramowania

Rozdział 5

Weryfikacja poprawności implementacji struktury wzorców projektowych w oparciu o model referencyjny

1. Wstęp

Współczesny człowiek otoczony jest przez oprogramowanie niemalże z każdej strony, nie tylko przez to zainstalowane w komputerze osobistym, ale też w wielu drobnych przedmiotach, o których większość ludzi nie ma zbyt dużej ilości informacji, np. breloczek do kluczy pełniący rolę pilota od alarmu. Wszechobecność oprogramowania jest wystarczającym argumentem, aby poważnie zająć się zagadnieniem dość szeroko rozumianej jakości kodu źródłowego. O sprawdzenie poprawności elementarnych rzeczy zadba środowisko programistyczne, często dostarczane jako zbiór zintegrowanych narzędzi, które sprawdzą poprawność syntaktyczną oraz zgodność kodu z założeniami wybranego paradygmatu programowania, np. obiektowego. Istnieją również inne zaawansowane narzędzia przeznaczone do statycznej analizy kodu, np. [9]. Niestety wspomniane narzędzia umożliwiają weryfikację bazowych elementów dostępnych w danym języku programowania, z których deweloper tworzy własne artefakty w tym logikę biznesową aplikacji, a to natomiast nie może być sprawdzone przez uniwersalny algorytm. Większość artefaktów wymaga indywidualnego podejścia i przygotowania niezbędnych danych oraz funkcji testujących.

Jednymi ze specyficznych składowych artefaktów są wzorce projektowe. Pomimo dużego uznania i rozpowszechnienia opisu wzorców przedstawionych w [4], implementacja większości z nich może być zróżnicowana i za każdym razem nieco inna, nawet jeśli dokonuje tego ten sam deweloper. Znaczące zróżnicowanie wynika już z samej idei wzorców projektowych, są to za [10] szkielety gotowych mechanizmów, które można wykorzystywać przy rozwiązywaniu typowych problemów projektowania i programowania. To oznacza, że za każdym razem rozwiązanie problemu wymaga odpowiedniej implementacji wzorca projektowego zgodnie z jego intencją, przeznaczeniem oraz kontekstem, w którym jest implementowany. Problemy związane z różnorodnością implementacji [12], [17] powodują, że pojawia się potrzeba opracowania modelu, który pozwoli na ocenę jakości implementacji wzorców projektowych.

Problem weryfikacji poprawności implementacji struktury wzorców projektowych oraz wybrane rozwiązania tego problemu zostały przedstawione w rozdziale 2. Rozdział 3 zawiera opis formalizmu oraz ideę wykorzystania proponowanego modelu referencyjnego. W rozdziale 4 przedstawiono przykładowe wyniki dla prostego przypadku wzorca Singleton. Rozdział 5 stanowi podsumowanie pracy.

2. Weryfikacja poprawności implementacji struktury wzorców projektowych

Weryfikacja poprawności implementacji struktury wzorców projektowych polega na przebadaniu pewnego artefaktu, inaczej fragmentu oprogramowania, w którym jest zaimplementowany wzorzec. Zasady i reguły poprawnej implementacji wzorców są często opisywane za pomocą opisu słownego i diagramów, np. w [4], jest to dobre rozwiązanie do nauki, lecz wykorzystanie tej reprezentacji do automatycznej weryfikacji wzorców jest bardzo problematyczne. Rozwiązaniem tego problemu jest wykorzystanie formalnej reprezentacji wzorców projektowych, która pozwoli na automatyzację procesu weryfikacji.

Proces weryfikacji wzorców projektowych jest często łączony z innymi bardziej złożonymi procesami, np. wyszukiwanie wystąpień wzorców czy ocena jakości implementacji. Dla uproszczenia w dalszych rozważaniach zostało to ujednolicone, tzn. przedmiotem zainteresowania jest proces weryfikacji wzorców projektowych niezależnie od celu wykorzystania.

Jednym z fundamentów procesu weryfikacji wzorców projektowych jest odpowiednia reprezentacja badanego oprogramowania. W wielu z podejść kod programu jest przekształcany do odpowiedniej reprezentacji, co wpływa na rodzaj stosowanej analizy. Jednym z takich podejść jest przekształcenie kodu programu do grafów, zostało to przedstawione w [16] oraz w [14]. W osobnych grafach należy opisać szablony wzorców projektowych, a proces weryfikacji wzorców (w przyjętym uproszczeniu, ponieważ wymienione podejścia dotyczą wyszukiwania) zrealizować jako wyszukiwanie podgrafu z szablonem wzorca w grafie reprezentującym oprogramowanie. Jak przyznają sami autorzy w takim podejściu nie możliwe jest odróżnienie np. wzorca strategii od stanu [16] oraz weryfikacja wzorca Singleton [14]. Inne wykorzystywane rozwiązanie to przekształcenie badanego oprogramowania do drzewa składniowego AST (od ang. Abstract Syntax Tree), zostało to wykorzystane w [2] oraz [7]. Za pomocą AST opisywane są elementy składowe klas oraz ich atrybuty. Powstają również autorskie systemy formalnej reprezentacji oprogramowania, są one bardzo rozbudowane i niezależne od języka [5].

Za drugi fundament procesu weryfikacji wzorców projektowych można uznać sposób w jaki wykonywana jest analiza, często jest on silnie związany z rodzajem opisu zasad i reguł definiujących wzorce projektowe. Dość popularnym podejściem jest wykorzystanie programowania w logice [1], [2], [3], [8], [15], a szczególnie języka Prolog lub podobnych. W takich podejściach definicja każdego wzorca projektowego opisana jest jako zbiór reguł. Reguły w większości podejść dotyczą strukturalnej budowy wzorca, tj. występujących relacji między klasami, wymaganych modyfikatorów dostępu, czy też typów danych. Definicje wariantów wzorców również muszą być opisywane dla każdego wariantu osobno. Jak zauważono w [6] opis regułowy powinien być wykorzystany wyłącznie do silnych relacji między klasami, utrudnione jest opisanie podobnych mikro-architektur, które różnią się drobnymi elementami. W [11] zauważono, że często utrudnione jest rozszerzanie definicji opisujących wzorce projektowe, a czasem jest to niemożliwe. Dodatkowo opis regułowy wymaga od osoby tworzącej definicję wzorców umiejętności programowania w językach typu Prolog, a w szczególnych przypadkach nauki nowego autorskiego języka. Ostatecznie zauważono w [2] oraz [13], że nie wszystkie wzorce projektowe można zweryfikować w tym podejściu.

3. Weryfikacja względem modelu referencyjnego

3.1. Model oceny jakości implementacji wzorców oraz model definicji

Opisywany model referencyjny jest składnikiem całościowego modelu oceny jakości implementacji wzorców projektowych. Zgodnie z aksjomatem twierdzącym, że jakość jest właściwością zbiorczą, można uznać, że jakość implementacji wzorców wyrażana jest przez wyniki odpowiednich miar różnych aspektów tej implementacji. W zależności od rozpatrywanego wzorca można wyznaczyć kilka aspektów, np.: poprawność strukturalną, nachodzenie na inne wzorce, zgodność z intencją wykorzystania. Każdy z aspektów należy scharakteryzować za pomocą zbioru odpowiednich gruboziarnistych cech, zbiór ten został nazwany modelem definicji. Model definicji w celu zweryfikowania opisywanych cech wskazuje na jeden z kilku mechanizmów weryfikacji drobnoziarnistych atrybutów. Jednym z mechanizmów jest weryfikacja względem modelu referencyjnego. Dzięki oddzieleniu mechanizmów weryfikacji od modelu definicji możliwe jest zwiększenie skuteczności tejże weryfikacji poprzez wybranie najbardziej odpowiedniego mechanizmu. Dodatkowo to podejście daje duże możliwości rozwoju na wypadek, gdyby dotychczasowe mechanizmy weryfikacji okazały się niewystarczające. Podobne rozwiązanie umożliwiające wybór jednego z kilku mechanizmów weryfikacji zostało zaproponowane w [11].

W uogólnionym ujęciu model oceny jakości implementacji wzorców projektowych jest modelem matematycznym realizującym funkcję oceny dla zbioru danych wejściowych, tj. dla badanego oprogramowania. Kod badanego oprogramowania jest przekształcany do struktury danych o postaci formalnej wywodzącej się z założeń paradygmatu programowania obiektowego. Wyjście modelu to wektor ocen uzyskanych dla każdego aspektu rozpatrywanego wzorca. Wyrażenie ocen w skali porządkowej pozwoli na porównywanie oraz wyznaczenie oczekiwanego poziomu jakości implementacji wzorców w badanym oprogramowaniu. Podstawowa funkcja oceny to średnia ważona, jednakże przewidywane jest zastosowanie innych funkcji.

Zasadnicze funkcje modelu definicji to sterowanie procesem analizy oprogramowania oraz kontrolowanie uzyskanych wyników. W celu spełnienia swoich funkcji model definicji wykorzystuje odpowiednie informacje:

- zależność od cechy - wskazanie na inną cechę i określenie rodzaju zależności od niej: i, lub, zawiera,
- ocena - złożona struktura danych wykorzystywana bezpośrednio przez funkcję oceny. Zawiera: współczynnik wagi, próg aktywacji oraz zalecaną wartość, a po wykonaniu weryfikacji dodatkowo uzyskany wynik,
- krotność - złożona struktura danych, która zawiera informacje o oczekiwanej ilości wystąpienia danej cechy w badanym oprogramowaniu. Określa oczekiwaną wartość w postaci przedziałów: od do, równe lub większe, mniej niż, dokładnie.

Dodatkowo model definicji przewiduje opisanie wymaganych zasad wykrywania i reagowania na błędy oraz wystąpienia zerowego poziomu dopasowania, jednakże wymienione zasady oraz inne szczegółowe informacje o modelu definicji nie są przedmiotem niniejszej pracy.

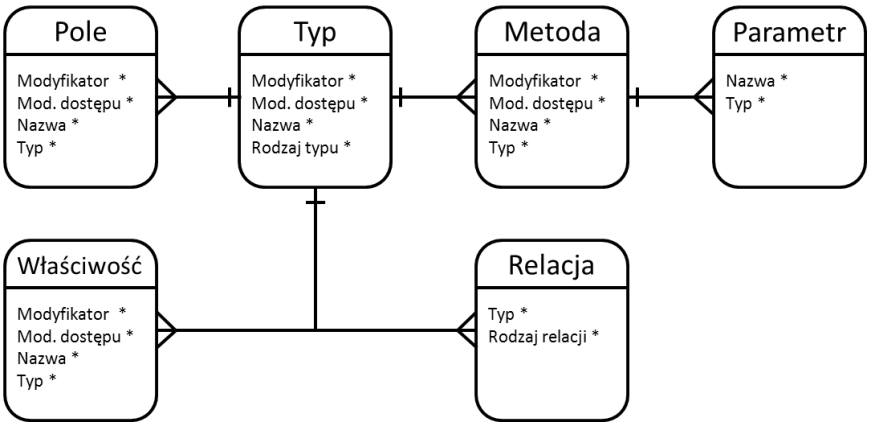
Motywacją do opracowania modelu referencyjnego było jednoczesne osiągnięcie następujących korzyści wyróżniających go na tle konkurencyjnych podejść:

- możliwość opisania różnych wariantów wzorców oraz atrybutów bez potrzeby powielania definicji wspólnych elementów,
- półautomatyczne uczenie się modelu poprzez dodawanie definicji wzorców oraz ich wariantów na podstawie badanego oprogramowania,
- wykorzystanie struktury danych wynikającej z paradygmatu programowania obiektowego jako formalnej reprezentacji szablonów wzorców projektowych, w wielu przypadkach jest to intuicyjne dla programistów języków C# oraz Java, podejście to nie wymaga uczenia się innych języków,
- powiązanie elementów modelu referencyjnego z elementami badanego artefaktu w celu wskazania różnic oraz zasugerowania zmian.

Całokształt własności modelu referencyjnego sprawia, że jest to bardzo ważny składnik omówionego wcześniej modelu oceny jakości implementacji wzorców projektowych. Szczególnie wynika to z faktu, że każdy wzorzec projektowy to odmienna struktura implementacji, której poprawność można zweryfikować. Upodobnienie modelu referencyjnego do założeń wywodzących się z paradygmatu programowania obiektowego, oraz częściowe zautomatyzowanie procesu tworzenia nowych definicji wzorców, powoduje zmniejszenie nakładu pracy potrzebnej do uzupełnienia modelu, co jest znaczącą korzyścią dla użytkownika.

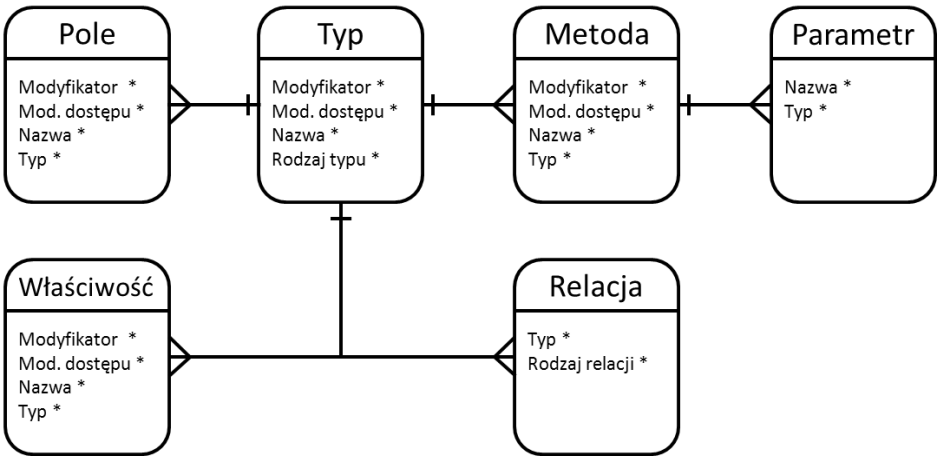
3.2. Model referencyjny

Model referencyjny, inaczej odniesienia, jest złożoną strukturą danych, która przechowuje abstrakcyjne definicje wzorców projektowych. W odróżnieniu od innych podejść, model ten nie przechowuje badanego oprogramowania. Dostarcza szablonowy opis wzorców oraz możliwych wariantów. Dla każdego rozpatrywanego wzorca projektowego należy przygotować osobny opis. Metamodel w notacji ERD tej struktury przedstawia rysunek 1, natomiast tabela 1 opisuje występujące atrybuty. Warto zaznaczyć, że schemat z rysunku 1 nie jest bezpośrednim odpowiednikiem struktury bazy danych. Symbol * przy atrybutach oznacza możliwość wystąpienia wielu atrybutów danego rodzaju przynależnych do konkretnego elementu, jest to konieczne aby możliwe było opisanie różnych wariantów. Typ pełni podwójną rolę, jest jednocześnie nadrzędnym elementem oraz występuje jako atrybut w niektórych elementach. W encji relacji jest wykorzystywany do wskazania zależnego typu wraz z określeniem rodzaju zależności.



Rysunek 1. Metamodel struktury modelu referencyjnego w notacji ERD.

Rysunek 1 oraz uzupełniającą go tabelę 1 należy interpretować następująco: nadrzędnym elementem każdego artefaktu jest przynajmniej jeden typ, np. klasa, interfejs; typ może być w relacji z wieloma innymi typami, np. poprzez dziedziczenie; każdy typ może zawierać wiele pól, właściwości oraz metod; metody mogą przyjmować wiele parametrów. Każdy z wymienionych elementów może charakteryzować się odpowiednimi drobnoziarnistymi atrybutami w wielu wariantach dopasowania (zob. tabela 1). Każdy atrybut zawiera skonkretyzowaną wartość oraz jednoznaczny poziom dopasowania w skali interwałowej. Przykład: właściwość, która udostępnia prywatną instancję wzorca Singleton powinna charakteryzować się publicznym modyfikatorem dostępu - jest to najbardziej dopasowany atrybut, gdy w badanym artefakcie wystąpi modyfikator wewnętrzny ograniczający zasięg singletonu to w ogólnym przypadku również będzie poprawnym rozwiązaniem, jednakże będzie to spełniało niższy poziom dopasowania niż modyfikator publiczny. Rodzaj udostępniania instancji singletonu (poprzez właściwość, metodę lub bezpośrednio pole statyczne) to cecha gruboziarnista, której weryfikacja wskazywana jest przez wspomniany wcześniej model definicji. Dodatkowo model definicji pozwala na zanegowanie opisu w modelu referencyjnym, wtedy prawidłowe jest wystąpienie dowolnych elementów pod warunkiem, że są różne od szablonu zanegowanego modelu referencyjnego. Może to być wykorzystane do opisania dowolnych pól oraz metod, które są zawarte w klasie wzorca Singleton i jednocześnie nie należą do struktury tworzącej wzorzec. Wymienione elementy mogą mieć dowolne atrybuty pod warunkiem, że nie będzie występował modyfikator statyczny.

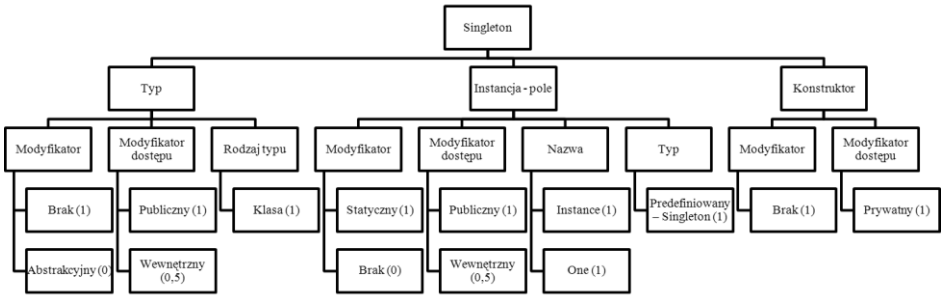


Rysunek 1. Metamodel struktury modelu referencyjnego w notacji ERD.

Tabela 1. Atrybuty występujące w elementach modelu referencyjnego

Atrybut	Opis	Wartości	Poziom dopasowania
Modyfikator	Określa modyfikator dla danego elementu.	Brak, abstrakcyjny, stały, statyczny i inne.	W przypadku każdego atrybutu poziom dopasowania zależny jest od kontekstu występowania. Poziom dopasowania to wartość liczbowa mieszcząca się w przedziale od 0 do 1, gdzie 1 to najwyższy poziom dopasowania. Możliwe jest opisanie wielu atrybutów o identycznym poziomie dopasowania.
Modyfikator dostępu	Określa modyfikator dostępu, który ogranicza zasięg elementów.	Brak, publiczny, prywatny, wewnętrzny.	
Nazwa	Wskazuje sugerowaną nazwę elementu wraz z doprecyzowaniem, że sugerowana nazwa to: przedrostek, przyrostek, dokładnie lub powinno zawierać	Zależne od kontekstu.	
Typ	Wskazuje na predefiniowany lub wyspecyfikowany typ, w przypadku pól i właściwości oznacza typ danych przechowywany przez te elementy, w metodach oznacza zwracany, a w parametrach przyjmowany typ danych, w przypadku relacji wskazuje typ, od którego zależny jest aktualnie rozważany typ.	Zależne od kontekstu.	
Rodzaj typu	Określa rodzaj typu.	Klasa, interfejs, typ wyliczeniowy, wartościowy i inne.	
Rodzaj relacji	Określa rodzaj relacji występującej pomiędzy typami.	Asocjacja, generalizacja, realizacji, agregacja, kompozycja.	

Możliwe wartości atrybutów są predefiniowane w zależności od kontekstu wykorzystania, w tabeli 1 oraz w dalszych rozważaniach użyty został kontekst zgodny z syntaktyką języka C#. Oprócz tego możliwe jest predefiniowanie specjalnych elementów (tj. typów, pól, metod i innych) o specjalnej semantyce, np. typy reprezentujące proste typy danych lub wartość atrybutu „brak".
Zaproponowany poziom dopasowania w przedstawionym przykładzie nie jest ostateczny. Przedstawione wartości 1 dla najwyższego poziomu oraz 0,5 dla niższego poziomu dopasowania mają charakter symboliczny i służą przybliżeniu idei. Docelowo wartości te należy dobrać na podstawie wiedzy eksperckiej.



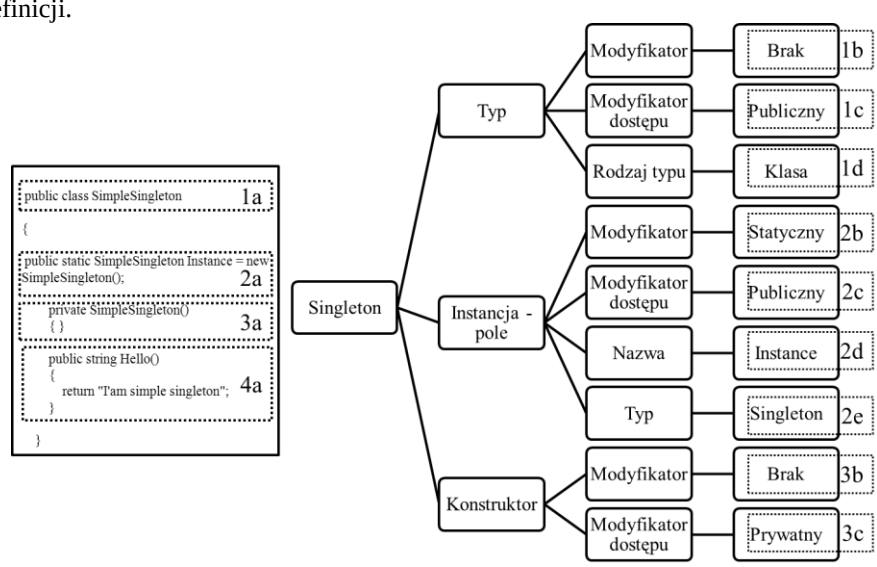
Rysunek 2. Przykładowa graficzna reprezentacja wzorca Singleton zawarta w modelu referencyjnym (w nawiasie podany poziom dopasowania).

3.3. Weryfikacja

Przed rozpoczęciem weryfikacji należy wskazać jeden lub kilka elementów będących rdzeniem wzorca, inaczej nadrzędnym elementem, np. dla wzorca strategia jest to implementowany interfejs. Rysunek 2 przedstawia opis uproszczonego wzorca Singleton zrealizowany za pomocą modelu referencyjnego, liczby podane w nawiasach oznaczają poziom dopasowania, na rysunku pominięte zostały mniej znaczące atrybuty. Proces weryfikacji polega na pobieraniu kolejnych elementów z badanego artefaktu, a następnie porównywaniu każdego z nich z odpowiednikiem w modelu referencyjnym oraz dookreśleniem, który z poszczególnych atrybutów wystąpił. Dokładniej przedstawia to rysunek 3, na przykładzie uproszczonego wzorca Singleton. Kolejne etapy weryfikacji można opisać następująco:

- pobierany jest typ zawierający badany artefakt: 1a, następnie porównywane są kolejne atrybuty: 1b – brak modyfikatorów, 1c – publiczny modyfikator dostępu, 1d – klasa jako rodzaj typu,
- pobierany jest kolejny element z badanego artefaktu: 2a to instancja singletonu, weryfikowane są atrybuty: 2b – modyfikator statyczny, 2c – publiczny modyfikator dostępu, 2d – sugerowana nazwa pola będącego instancją, 2e – typ zwracany przez instancję tj. obiekt singletonu,
- 3a – konstruktor, dokładniej jest to szczególny przypadek metody, dla którego weryfikowane są atrybuty: 3b – brak modyfikatorów, 3c – modyfikator dostępu prywatny,
- ostatecznie weryfikowane są pozostałe elementy (nie widoczne na rysunku 3): 4a – dowolna metoda, która nie tworzy struktury wzorca Singleton.

Wystąpienie każdego z atrybutów wiąże się z określonym poziomem dopasowania elementów badanego artefaktu, co ostatecznie przekłada się na poziom dopasowania całego artefaktu względem modelu referencyjnego. Uzyskany poziom dopasowania jest cząstkową informacją potrzebną do określenia jakości implementacji badanego wzorca, za całość procesu oceny jakości odpowiedzialny jest wymieniony wcześniej model definicji.



Rysunek 3. Etapy weryfikacji fragmentu oprogramowania względem modelu referencyjnego.

Opisana wyżej zasada weryfikacji jest taka sama dla wszystkich rozważanych wzorców projektowych. Jednakże dla bardziej rozbudowanych wzorców należy zastosować odpowiednią dekompozycję. W przypadku wzorca strategia, należy wyznaczyć osobno interfejs deklarujący żądane operacje oraz klasy, które go implementują. Klasy implementujące strategię mogą być reprezentowane jako jeden wspólny typ, a poprzez krotność modelu definicji należy wskazać oczekiwaną ilość wystąpień. Aby doprecyzować zależności występujące w tym wzorcu należy docelowo wykorzystać elementy rozszerzeń opisane w dalszej części artykułu.

3.4. Dalsza rozbudowa modelu

Przewidywane dalsze prace nad modelem referencyjnym ukierunkowane są głównie na rozszerzenie możliwości opisania złożonych wzorców, a dokładniej zakładają one:

- rozbudowę struktury o abstrakcyjną definicję treści metod w celu opisania zachowań wzorców, inaczej cech behawioralnych,
- rozbudowę wewnętrznych relacji pomiędzy elementami, w celu silniejszego wskazania zależności,
- wprowadzenie specjalnych wskaźników umożliwiających tymczasowe podstawienie wartości atrybutu z badanego oprogramowania do modelu referencyjnego, w celu skuteczniejszego weryfikowania występujących nazw elementów,
- wprowadzenie rekurencyjnych odniesień do pośrednich produktów powstających w procesie weryfikacji, aby ograniczyć przestrzeń weryfikacji,
- rozbudowa syntaktyki elementów predefiniowanych,
- opracowanie powtarzających się elementów predefiniowanych tzw. mini wzorców lub idiomów, oraz opisanie ich w modelu referencyjnym.

4. Przykładowe wyniki

Uzyskanie wyników będących znamionami jakości możliwe jest przy wykorzystaniu całego modelu oceny jakości implementacji wzorców projektowych oraz zawartej w nim funkcji oceny. Natomiast weryfikacja wyłącznie względem modelu referencyjnego pozwala na uzyskanie wyników cząstkowych. Omówiony wcześniej formalizm został zrealizowany jako prototypowe narzędzie wspierające ocenę jakości implementacji wzorców, niestety aktualne zaawansowanie prac pozwala na weryfikację wyłącznie niektórych cech wzorców. Zwiększenie zakresu weryfikacji wymaga zrealizowania wymienionych wcześniej rozszerzeń.

W tabeli 2 zostały przedstawione wyniki procentowe uzyskane na podstawie poziomu dopasowania dla dwóch prostych przypadków wzorca Singleton z zaznaczonymi różnicami wynikającymi z informacji zawartych w modelu referencyjnym. Artefakty poddane weryfikacji zostały przygotowane na potrzeby eksperymentu i mają na celu praktyczne przybliżenie idei. W przyszłości możliwe będzie przeprowadzenie praktycznego eksperymentu również dla bardziej rozbudowanych przypadków wzorca Singleton, w tym wariantów z różnymi metodami synchronizacji w środowisku wielowątkowym. Listingi od 1 do 2 zawierają kod źródłowy zweryfikowanego oprogramowania.

Tabela 2. Przykłady wyników

	Przypadek 1	**Przypadek 2**
Ogólna charakterystyka	Najprostsza implementacja wzorca Singleton, w którym instancja udostępniania jest poprzez pole statyczne.	Instancja singletonu udostępniania poprzez właściwość lub metodę.
Poziom dopasowania	90%	75%
Różnice względem najwyższego poziomu dopasowania	Modyfikator dostępu pola instancji oraz klasy o niższym poziomie dopasowania (wewnętrzny).	Wystąpił zerowy poziom dopasowania modyfikatora dla właściwości/metody udostępniającej instancję (niestatyczny) oraz dla modyfikatora dostępu konstruktora (nieprywatny).
Sugestie zmian	Osiągnięcie możliwie najwyższego poziomu dopasowania zgodnie z wyznaczonymi różnicami.	
Konsekwencje	Występujące różnice są poprawną implementacją. Konsekwencją ich wystąpienia jest ograniczenie zasięgu dostępności instancji wzorca, co może być niekorzystne w rozbudowanie aplikacji poprzez dodanie dodatkowych bibliotek, ponieważ będzie wymagało ingerencji w istniejący kod.	Nie możliwe jest odniesienie się do instancji bez inicjalizacji obiektu. Po inicjalizacji obiektu możliwe jest odniesienie się jednocześnie do instancji jak też bezpośrednio do elementów składowych, które powinny być dostępne wyłącznie poprzez odniesienie się do instancji. Oba błędy mogą prowadzić do sytuacji, że wystąpią różne instancje Singletonu, a przez to cel wzorca nie zostanie spełniony.

Uzyskanie przez przypadek nr 2, który zawiera poważne błędy, nieznacznie niższego wyniku niż przypadek nr 1, o poprawnej implementacji, wynika z nieuwzględnienia współczynnika wagi dla weryfikowanych cech. Konsekwencji nie można porównywać bezpośrednio ze sobą, ponieważ mogą objawić się w różnych fazach życia oprogramowania. W każdym przypadku warto ich uniknąć. Warto również zauważyć, że poważne błędy (przypadek 2) będą utrudniać proces wytwórczy już w czasie implementacji. Natomiast błędy z pozoru mniej poważne (przypadek 1) będą miały duże znaczenie po zakończeniu procesu wytwórczego, tj. w czasie rozwoju i konserwacji oprogramowania.

```
internal class Singleton1
{
    internal static Singleton1 Instance = new Singleton1();
    private Singleton1()
    { }
    public string Hello()
    {
        return "I'am Singleton";
    }
}
```

Listing 1. Kod źródłowy przypadku nr 1.

```
public class Singleton2
 {
     private static Singleton2 instance = new Singleton2();
     public Singleton2()
     { }
     public Singleton2 Instance
     {
         get { return instance; }
     }
     public string Hello()
     {
         return "I'am Singleton";
     }
 }
```

Listing 2. Kod źródłowy przypadku nr 2.

5. Podsumowanie

W pracy krótko uzasadniono, dlaczego ważne jest weryfikowanie poprawności implementacji wzorców projektowych oraz przedstawiono najważniejsze fundamenty badań powiązanych z weryfikacją wzorców projektowych.

Omówiono proces weryfikacji struktury wzorców projektowych, który polega na porównywaniu elementów badanego oprogramowania z szablonowymi elementami w modelu referencyjnym. Struktura abstrakcyjnego opisu wzorców projektowych w modelu referencyjnym oparta jest o założenia paradygmatu programowania obiektowego, dodatkowo umożliwia opisanie różnych wariantów wzorców. Weryfikacja względem modelu referencyjnego dostarcza cząstkowe wyniki, ponieważ ów model jest częścią większego modelu, który to umożliwia całościową ocenę jakości implementacji wzorców projektowych.

Dalsze prace przewidują przede wszystkim rozbudowę modelu referencyjnego, aby możliwe było opisanie złożonych wzorców oraz zwiększenie szczegółowości ich definicji.

Przeprowadzone prace nad opracowaniem i wykorzystaniem modelu referencyjnego do weryfikacji wzorców projektowych wykazały, że jest to odpowiednie rozwiązanie do postawionego celu. Należy pamiętać, że ostatecznie to model oceny jakości implementacji wzorców projektowych powinien zostać oceniony pod względem poprawności w ocenie wzorców. Aktualne niedostatki proponowanego modelu referencyjnego ograniczają jego wykorzystanie do weryfikacji wzorców o prostej strukturze, np. Singleton lub Strategia. Rozbudowa modelu o wymienione wcześniej rozszerzenia oraz równoległe wykorzystanie alternatywnych mechanizmów weryfikacji powinno być wystarczające do opisania szablonów oraz weryfikacji większości z wzorców przedstawionych w [4].

Literatura

[1] Binun A.: *High Accuracy Design Pattern Detection*. Dysertacja doktorska, Rheinischen Friedrich Wilhelms Universitat Bonn, 2012.

[2] Blewitt A.*: HEDGEHOG: Automatic Verification of Design Patterns in Java*. Dysertacja doktorska, University of Edinburgh, 2006.

[3] Fabry J., Mens T., Language-Independent Detection of Object-Oriented Design Patterns, *Journal Computer Languages: Systems and Structures*, Vol. 30, Issue 1-2, 2004, s. 21-33.
[4] Gamma E. i inni: *Wzorce projektowe. Elementy oprogramowania wielokrotnego użytku*. Helion, Gliwice, 2010.
[5] Grzanek K., Realizacja systemu wyszukiwania wystąpień wzorców projektowych w oprogramowaniu przy zastosowaniu metod analizy statycznej kodu źródłowego, Dysertacja doktorska, Politechnika Częstochowska, Łódź, 2008
[6] Gueheneuc Y.G., Jussien N., Using explanations for design patterns identification, *Proc. First IJCAI Workshop Modeling and Solving Problems with Constraints*, C. Bessière, ed., pp. 57-64, 2001.
[7] Kirasić D., Basch D., Ontology-Based Design Pattern Recognition, *Knowledge-Based Intelligent Information and Engineering Systems*, Zagreb, Croatia, 2008
[8] Krishnaswami N. R., Design Patterns in Separation Logic, *TLDI'09*, ACM, New York, 2009, s. 105-116.
[9] Strona internetowa: http://www.ndepend.com/, dostęp: 18.06.2015.
[10] McConnell S., *Kod Doskonały*, Helion, Gliwice, 2010
[11] Rasool G.: Customizable *Feature based Design Pattern Recognition Integrating Multiple Techniques*. Dysertacja Doktorska, Technische Universitat Ilmenau, Ilmenau 2010
[12] Rasool G. i inni, Evaluation of design pattern recovery tools, *Elsevier*, Procedia Computer Science, Vol 3, 2011.
[13] Shi N., Ollson R.A., Reverse Engineering of Design Patterns from Java Source Code, *ASE '06: Proceedings of the 21st IEEE/ACM*, International Conference on Automated Software Engineering, pages 123–134, Washington, 2006.
[14] Singh Rao R., Gupta M.: Design Pattern Detection by Greedy Algorithm Using Inexact Graph Matching, *International Journal Of Engineering And Computer Science*, Volume 2 Issue 10, 2013, s. 3658-3664.
[15] Smith J., *SPQR: Formal Foundations and Practical Support for the Automated Detection of Design Patterns from Source Code*, Dysertacja doktorska, University of Noth Carolina, 2005.
[16] Tsantalis N. i inni: Design Pattern Detection Using Similarity Scoring, *IEEE Transactions on Software Engineering,Volume: 32*, Issue: 11, 2006, s. 896-908.
[17] Wojszczyk R.: Koncepcja hybrydowej metody do oceny jakości zaimplementowanych wzorców projektowych, *Zeszyty Naukowe Wydziału Elektroniki i Informatyki nr 7*, strony od 17 do 26, Wydawnictwo Uczelniane Politechniki Koszalińskiej, ISSN 1897-7421, Koszalin 2015.

Rozdział 6

Model jakości danych: definicja i pomiary

1. Wprowadzenie

Dane w dzisiejszych czasach mają wartość podniesioną do rangi dobra gospodarczego, takiego jak waluta czy złoto. Mogą być reprezentowane w postaci napisów, liczb, znaków lub sygnałów, jak również w formie graficznej i audiowizualnej. Dane są zbierane i gromadzone w trakcie realizacji procesów o charakterze badawczym i biznesowym. W związku z tym, że pełnią kluczową rolę w procesach decyzyjnych, ich jakość jest niezwykle istotna. Informacja jest efektem interpretacji i przetworzenia danych, dlatego gromadząc i organizując dane, należy mieć na uwadze również wymagania dotyczące pozyskiwania informacji. Informacja jest wnioskowana z danych, a jakość informacji jest ściśle związana z jakością danych i oznacza konsekwentne spełnienie oczekiwań klienta. Z informacją oraz danymi powiązana jest także wiedza. Wiedzę można zdobyć poprzez uczenie się. Można ją podzielić na teoretyczną i praktyczną. Jest dynamiczna, żyje w nas. Biorąc pod uwagę jakościową informację, kontekst i doświadczenie tworzy się nową, jakościową wiedzę.

Istnieje wiele różnych definicji jakości. Już w czasach starożytnych pojawiła się definicja, której autorem był Platon. Określił on jakość jako pewien stopień doskonałości [4]. Obecnie jest wiele definicji tego pojęcia. Europejska Organizacja Jakości (EOQ) definiuje jakość jako stopień spełnienia wymagań, zgodność parametrów założonych z uzyskanymi [12]. Z kolei Olson postrzega jakość poprzez spełnienie wymagań dotyczących przeznaczenia i przy jej ocenie proponuje wziąć pod uwagę aktualność, istotność i zrozumiałość danych [9]. W pracy „Information Quality" zaproponowano dodatkowe cechy, ważne w kontekście oceny jakości, a mianowicie dokładność, kompletność, wiarygodność i współzależność [3]. Należy jednak pamiętać, że ocena jakości danych zależna jest od sytuacji, kontekstu czy preferencji. Dlatego zalecane jest podejście rozpatrujące różne perspektywy oceny, z uwzględnieniem opinii grup użytkowników przy jednoczesnej bezstronnej analizie danych [2].

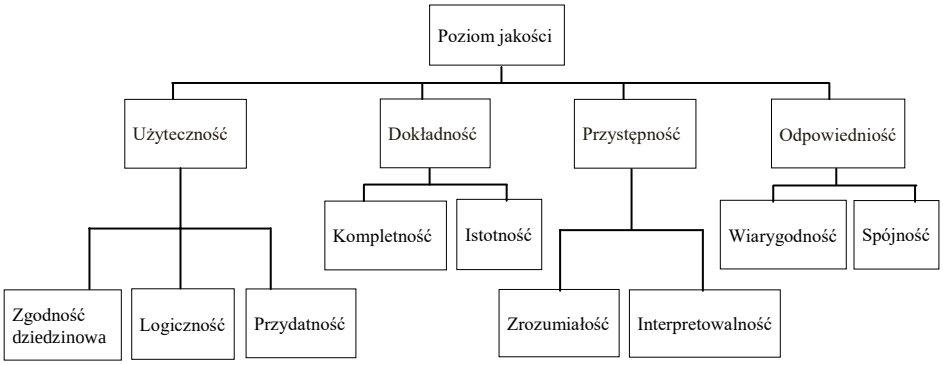
2. Model oceny jakości danych

Na podstawie wniosków wynikających z analizy różnych definicji pojęć jakości oraz propozycji modeli jakości przeznaczonych do oceny artefaktów zarówno w procesie ich wytwarzania, jak i użytkowania [6], został zaproponowany model oceny jakości danych. Model ten uwzględnia zarówno techniczne, jak i biznesowe aspekty, które powinny spełniać dane, by można było uznać je za jakościowo akceptowalne. Założono,

że dane przeznaczone do oceny będą reprezentowane w postaci napisów, liczb lub znaków oraz będą dostępne w postaci tabelarycznej.

Podstawą opracowania modelu był zbiór charakterystyk zdefiniowany w normach ISO/IEC 25010 [6] i ISO/IEC 25012 [5] oraz w „Information Quality" [3]. Punktem startowym do opracowania modelu jakości danych było założenie, by zbiór charakterystyk określających perspektywy oceny umożliwiał szeroki kontekst oceny danych, między innymi typ, rodzaj, przeznaczenie danych oraz aspekt oceny obiektywnej i subiektywnej. Programista skupia się przede wszystkim na formacie danych, spójności, kompletności, natomiast biznes na użyteczności. Dlatego też, oceniając jakość danych należy uwzględnić różne perspektywy oceny, by nie dopuścić do sytuacji, w której dane uznane za wysokiej jakości przez jedną osobę okażą się całkowicie nieprzydatne przez inną. Ocena obiektywna jest bezkontekstowa i nie zależy od wiedzy, doświadczenia, jak również od stanu psychicznego i preferencji osoby oceniającej. Natomiast ocena subiektywna może być obarczona ich wpływem. Osobą oceniającą może być zarówno programista jak i biznes. W modelu uwzględniono cztery najważniejsze cechy danych, wartych gromadzenia w bazach danych (zarówno transakcyjnych, jak i analitycznych) oraz użytecznych, w kontekście generowania istotnych informacji wykorzystywanych w procesach wspomagania decyzji. Dane, które są niewłaściwie reprezentowane, niekompletne, przekłamane lub niewiarygodne, nie mogą być źródłem dla uzyskiwania wiedzy o otaczającym nas świecie.

Proponowany model ma strukturę hierarchiczną i składa się z czterech charakterystyk oraz zbioru podcharakterystyk. Dla podcharakterystyk zostały zdefiniowane atrybuty i miary, które są wykorzystywane w procesie pomiarów własności ocenianych artefaktów. Model oceny jakości danych został przedstawiony na rysunku 1.



Rysunek 1. Model oceny jakości danych.

Definicje poszczególnych składowych modelu jakości zostały określone w następujący sposób:

- **Użyteczność** – stopień, w jakim dane mogą być wykorzystane i są potrzebne w kontekście realizowanych zadań [1]
    - **Zgodność dziedzinowa** - podcharakterystyka obiektywna. Stopień zgodności danych z reprezentowaną dziedziną. W przypadku niezgodności dziedzinowej i braku wymaganej struktury danych, wykorzystanie danych może być ograniczone lub niemożliwe. Poniżej zde-

finiowano atrybuty umożliwiające ocenę zgodności dziedzinowej (ograniczenia dziedzinowe):

- **Zgodność formatów** – badanie: *Czy w kolumnach występują dane zgodne z ustalonymi formatami?*
  - funkcja miary: X/Y (1)
  - znaczenie składowych:
    - X – liczba wartości zgodnych z obowiązującymi formatami
    - Y – liczba wszystkich wartości
- **Brak białych znaków** – badanie: *Czy występują białe znaki?*
  - funkcja miary: X/Y (2)
  - znaczenie składowych:
    - X – liczba wartości bez białych znaków na początku, końcu lub podwójnych białych znaków w środku wartości ocenianej danej
    - Y – liczba wszystkich wartości
- **Brak fluktuacji** – badanie: *Czy występuje fluktuacja danych?*
  - funkcja miary: X/Y (3)
  - znaczenie składowych:
    - X – liczba wartości danych, które nie są przypadkowymi odchyleniami
    - Y – liczba wszystkich wartości
- **Zgodność typu** – badanie: *Czy w zbiorach danych znajdują się wyłącznie wartości zgodne ze zdefiniowanymi typami danych?*
  - funkcja miary: X/Y (4)
  - znaczenie składowych:
    - X – liczba wartości danych zgodnych z obowiązującymi typem danych
    - Y – liczba wszystkich wartości

- **Przydatność** - podcharakterystyka subiektywna. Stopień, w jakim dane mogą zostać wykorzystane do określonych celów. Pomiar wartości na podstawie odpowiedzi na pytanie pomocnicze: *Czy dane są przydatne (T/N)?*
- **Logiczność** - podcharakterystyka subiektywna. Ocena, czy dane mają strukturę odpowiadającą rozpatrywanej dziedzinie (umożliwiającą ich interpretację i poprawne zrozumienie). Pomiar wartości na podstawie odpowiedzi na pytanie pomocnicze: *Czy dane są logiczne (T/N)?*

- **Dokładność** – stopień, w jakim dane zostały zebrane z należytą starannością, dbałością o szczegóły i precyzją [1]
  - **Kompletność** - podcharakterystyka obiektywna. Stopień, w jakim dane nie mają braków. Braki w danych oznaczają, że nie zostały one zebrane z dużą starannością i precyzją. Poniżej zdefiniowano atrybuty umożliwiające ocenę kompletności:
    - **Uzupełnione wartości** – badanie: *Czy nie ma brakujących wartości?*

- funkcja miary: X/Y (5)
- znaczenie składowych:
  X – liczba podanych wartości
  Y – liczba wymaganych, obligatoryjnych wartości

- **Współzależność** – badanie: *Czy dane referencyjne są poprawne?*
  - funkcja miary: X/Y (6)
  - znaczenie składowych:
    X - liczba poprawnych wartości referencyjnych (wskazują na istniejący obiekt)
    Y - liczba istniejących wartości referencyjnych w zbiorze danych
- **Łączność historyczna** – badanie: *Czy są dane z wszystkich wymaganych przedziałów czasowych?*
  - funkcja miary: X/Y (7)
  - znaczenie składowych:
    X - liczba okresów, dla których określono wartości danych
    Y - liczba okresów, dla których wymagane są wartości danych
- **Unikatowość** – badanie: *Czy nie występują duplikaty wartości danych cech unikatowych?*
  - funkcja miary: X/Y (8)
  - znaczenie składowych:
    X - liczba wartości unikatowych
    Y - liczba wszystkich wartości danych cech unikatowych

- **Istotność** - podcharakterystyka subiektywna. Stopień, w jakim dane zostały zebrane w sposób poprawny, czyli taki, że dane są ważne i znaczące w kontekście rozpatrywanej dziedziny. Pomiar wartości na podstawie odpowiedzi na pytanie pomocnicze: *Czy dane dotyczą rozpatrywanej dziedziny (T/N)?*

- **Przystępność** – stopień interpretowalności danych, możliwość zrozumienia ich znaczenia [1]
  - **Poprawność ortograficzna** - podcharakterystyka obiektywna. Brak błędów ortograficznych w napisach (dane łańcuchowe). Poniżej zdefiniowano atrybuty umożliwiające ocenę:
    - **Brak błędów ortograficznych/Literówek** – badanie: *Czy występują błędy ortograficzne?*
      - funkcja miary: X/Y (9)
      - znaczenie składowych:
        X - liczba słów poprawnych ortograficznie (z uwzględnieniem błędów literowych)
        Y - liczba wszystkich słów
  - **Interpretowalność** - podcharakterystyka subiektywna. Stopień, w jakim dane mogą zostać poprawnie zrozumiałe i pasujące do rozpatrywanej dziedziny, a osoba oceniająca może wyciągnąć wnioski na ich podstawie. Pomiar wartości na podstawie odpowiedzi na pytanie

pomocnicze: *Czy interpretacja danych jest jednoznaczna? Czy znana jest dziedzina źródła danych (T/N)?*

- **Odpowiedniość** – zgodność danych z określonymi wymaganiami [11]. Stopień pewności i stabilności, gwarantujący, że posiadane dane nie są sfałszowane.
    - **Spójność** - podcharakterystyka obiektywna. Stopień w jakim dane są zgodne z założeniami, spełniają ustalone wymagania. Poniżej zdefiniowano atrybuty umożliwiające ocenę spójności:
        - **Akceptowalność klucza** – badanie: *Czy dla kolumn kandydujących do klucza głównego wartości są zdefiniowane i unikalne?*
          - funkcja miary: X/Y (10)
          - znaczenie składowych:
            X - liczba kolumn, w których podane są unikatowe i wszystkie wartości
            Y – liczba kolumn kandydujących do klucza głównego
        - **Zgodność z ograniczeniami** – **badanie:** *Czy dane spełniają nałożony na nie zbiór ograniczeń?*
          - funkcja miary: X/Y (11)
          - znaczenie składowych:
            X - liczba wartości, które spełniają określone w wymaganiach ograniczenia
            Y – liczba wszystkich wartości
    - **Wiarygodność** - podcharakterystyka subiektywna. Stopień, w jakim dane są godne zaufania. Ocena możliwości sfałszowania lub przekłamania danych. Pomiar wartości na podstawie odpowiedzi na pytanie pomocnicze: *Czy możemy ufać, zawierzać danym (T/N)?*

Poziom jakości danych jest wyznaczany na podstawie wartości miar zdefiniowanych dla poszczególnych podcharakterystyk.

Dla podcharakterystyki **zgodność dziedzinowa** została zdefiniowana miara wywiedziona z zależności (1), (2), (3) i (4) w następujący sposób:

$$P_7 = \left(\frac{\frac{X1}{Y1}+\frac{X2}{Y1}+\frac{X3}{Y1}+\frac{X4}{Y1}}{4}\right) * 100\% , \quad (12)$$

gdzie P_7 jest miarą zgodności dziedzinowej, x_1 jest liczbą wartości zgodnych z obowiązującymi formatami, x_2 jest liczbą wartości bez białych znaków na początku, końcu lub podwójnych w środku wartości, x_3 jest liczbą wartości, które nie są przypadkowymi odchyleniami, x_4 jest liczbą wartości zgodnych z obowiązującymi typem danych, y_1 jest liczbą kardynalną rozpatrywanego zbioru wartości danych.

Dla podcharakterystyki **spójność** została zdefiniowana miara wywiedziona z zależności (10) i (11) w następujący sposób:

$$P_8 = \left(\frac{\frac{X5}{Y2}+\frac{X6}{Y1}}{2}\right) * 100\%, \quad (13)$$

gdzie P_8 jest miarą spójności, x_5 jest liczbą kolumn, w których podane są unikatowe i wszystkie wartości, x_6 jest liczbą wartości, które spełniają określone w wymaganiach ograniczenia, y_1 jest liczbą kardynalną rozpatrywanego zbioru wartości danych, y_2 jest liczbą kolumn kandydujących do klucza głównego.

Dla podcharakterystyki **kompletności** została zdefiniowana miara wywiedziona z zależności (5), (6), (7) i (8) w następujący sposób:

$$P_9 = \left(\frac{\frac{X7}{Y3}+\frac{X8}{Y4}+\frac{X9}{Y5}+\frac{X10}{Y6}}{4}\right) * 100\% , \quad (14)$$

gdzie P_9 jest miarą kompletności, x_7 jest liczbą podanych wartości, x_8 jest liczbą poprawnych wartości referencyjnych, x_9 jest liczbą okresów, dla których określono wartości danych, x_{10} jest liczbą wartości unikatowych, y_3 jest liczbą wymaganych, obligatoryjnych wartości, y_4 jest liczbą istniejących wartości referencyjnych w zbiorze danych, y_5 jest liczbą okresów, dla których wymagane są wartości danych, y_6 jest liczbą wszystkich wartości danych cech unikatowych.

Dla podcharakterystyki **zrozumiałość** została zdefiniowana miara wywiedziona z zależności (9) w następujący sposób:

$$P_{10} = \left(\frac{X11}{Y7}\right) * 100\% , \quad (15)$$

gdzie P_{10} jest miarą zrozumiałości, x_{11} jest liczbą wartości z błędami ortograficznym lub literówkami, y_7 jest liczbą wszystkich słów.

Miarami podcharakterystyk subiektywnych, tj. **przydatność**, **logiczność**, **istotność**, **interpretowalność** i **wiarygodność**, są odpowiedzi na zaproponowane w modelu pytania przez osobę oceniającą (pomiar nie podlega automatyzacji). Jakość danych oceniana jest w skali od 0% do 100% (0% oznacza niespełnienie wymagań) lub T/N (T - ocena pozytywna 100%, N – negatywna 0%).

Miary podcharakterystyk stanowią podstawę wyznaczenia wartości miar charakterystyk, znajdujących się na wyższym poziomie modelu jakości danych. W celu umożliwienia określenia istotności perspektywy oceny, zostały wprowadzone wagi. Tak samo jak w przypadku miar podcharakterystyk, tak i tutaj, wzrost miary wpływa pozytywnie na poziom jakości ocenianych danych.

Dla charakterystyki **użyteczności** została zdefiniowana miara określona zależnością:

$$C_{użyt} = \frac{\frac{P1+P2}{2} * WS + P7 * WO}{2} , \quad (16)$$

gdzie $C_{użyt}$ jest miarą użyteczności, P_1 jest miarą przydatności, P_2 jest miarą logiczności, P_7 jest miarą zgodności dziedzinowej, W_O jest wagą oceny obiektywnej, W_S jest wagą oceny subiektywnej.

Dla charakterystyki **dokładności** została zdefiniowana miara określona zależnością:

$$C_{dokł} = \frac{P6 * WS + P9 * WO}{2} , \quad (17)$$

gdzie $C_{dokł}$ jest miarą dokładności, P_6 jest miarą istotności, P_9 jest miarą kompletności, W_O jest wagą oceny obiektywnej, W_S jest wagą oceny subiektywnej.

Dla charakterystyki **przystępności** została zdefiniowana miara określona zależnością:

$$C_{przyst} = \frac{\frac{P4+P5}{2} * WS + P10 * WO}{2} , \quad (18)$$

gdzie C_{przyst} jest miarą przystępności, P_4 i P_5 jest miarą interpretowalności, P_{10} jest miarą zrozumiałości, W_O jest wagą oceny obiektywnej, W_S jest wagą oceny subiektywnej.

Dla charakterystyki **odpowiedniość** została zdefiniowana miara określona zależnością:

$$C_{odp} = \frac{P3*WS+P8*WO}{2}, \quad (19)$$

gdzie C_{odp} jest miarą odpowiedniości, P_3 jest miarą wiarygodności, P_8 jest miarą spójności, W_O jest wagą oceny obiektywnej, W_S jest wagą oceny subiektywnej.

Znając miary charakterystyk, można obliczyć poziom jakości danych, na podstawie zależności:

$$PJ = \frac{Cużyt + Cdokł + Cprzyst + Codp}{4}, \quad (20)$$

gdzie PJ jest poziomem jakości danych, $C_{użyt}$ jest miarą użyteczności, $C_{dokł}$ jest miarą dokładności, C_{przyst} jest miarą przystępności, a C_{odp} jest miarą odpowiedniości.

Po wyznaczeniu poziomu jakości danych można dokonać oceny jakości i stwierdzić, czy dane spełniają oczekiwania potencjalnego użytkownika. W tym celu należy ustalić próg akceptacji. Na podstawie wyników uzyskanych w procesie weryfikacji modelu, przyjęto, że minimalny, akceptowalny poziom jakości nie może być niższy niż 50% wartości funkcji oceny. Wartości z przedziału pomiędzy progiem akceptowalności a 100% mogą być traktowane, jako wskaźnik opłacalności realizacji procesu korekty rozpatrywanych danych.

3. Ocena jakości danych

Ocena jakości danych przebiega w kilku etapach. Poniżej zostały opisane czynności, które zalecane są podczas realizacji każdego z etapów.

3.1. Charakterystyka danych źródłowych podlegającego ocenie

Oceniając jakość danych źródłowych należy określić dziedzinę, z której pochodzą dane, następnie ustalić sposób w jaki zostały one pozyskane, dla kogo są przeznaczone oraz jak często podlegają zmianom. Istotne znaczenie dla procesu oceny jakości danych ma wartość poziomu akceptacji danych, co jest zależne i wynika z przeznaczenia danych. Inaczej traktuje się dane wykorzystywane do oceny np. zagrożenia zdrowia lub życia (w tym przypadku oczekuje się 100% jakości danych), a inaczej w badaniach ankietowych. gdzie zakłada się, że dane nie zawsze będą wiarygodne.

3.2. Analiza danych źródłowych

Podczas analizy zbioru danych źródłowych należy zapoznać się z jego strukturą. Przydatna staje się dokumentacja, model fizyczny lub logiczny danych, metadane oraz wiedza użytkownika na temat dziedziny, której dotyczą dane. Taka analiza ma na celu określenie wymaganych typów i formatów, ustalenie obowiązujących ograniczeń, które powinny być spełnione przez dane.

3.3. Profilowanie, uzyskanie wyników badań

Proces profilowania danych może być wykonany za pomocą dostępnych narzędzi programowych, między innymi, takich jak Oracle Enterprise Data Quality, Microsoft Server Data Quality Client i innych [8,10]. W trakcie profilowania określa się zestaw danych i zakres analizy. W tym procesie uzyskuje się, między innymi, następujące własności analizowanego zbioru danych:

- maksymalną i minimalną liczbę znaków dla danych łańcuchowych
- minimum, maksimum, średnią, odchylenie standardowe dla danych liczbowych
- liczbę znaków w zbiorze danych
- typy danych
- nieprawidłowości w danych (np. niepoprawne daty)
- wzorce i szablony danych
- liczbę brakujących i/lub nieokreślonych wartości
- liczbę duplikatów w zbiorze danych
- ocenę zależności pomiędzy danymi
- liczbę niepoprawnych referencji
- niezgodność z regułami biznesowymi
- różnice pomiędzy porównywanymi zbiorami danych.

Zaleca się użycie predefiniowanych komponentów dostępnych w narzędziach do profilowania oraz przeanalizowanie uzyskanych wyników, bowiem informacje z procesu profilowania są niezbędne do oceny jakości danych, a wykorzystanie dedykowanych narzędzi znacznie skraca czas wymagany na analizę danych.

3.4. Ocena jakości danych

Wykorzystując zaproponowany model oceny jakości danych można ustalić perspektywę oceny, następnie dokonać pomiarów w celu określenia wartości miar dla wybranych charakterystyk i podcharakterystyk i ocenić jakość na podstawie uzyskanej wartości funkcji oceny. W przypadku, gdy ta wartość nie jest satysfakcjonująca, to należy oszacować opłacalność akcji naprawczej w kontekście znaczenia rozpatrywanych danych dla realizowanych procesów przetwarzania danych.

4. Przypadek studialny

W celu weryfikacji zaproponowanego modelu jakości danych, wykonano serię badań oceny jakości danych pochodzących z różnych źródeł. Przykład realizacji takiego procesu przedstawiono w tym rozdziale.

4.1. Charakterystyka danych źródłowych

Przykładowe badanie jakości danych przeprowadzono na zbiorze danych „Badanie postaw społeczeństwa względem bezpieczeństwa ruchu drogowego” pobranych ze strony Krajowej Rady Bezpieczeństwa Ruchu Drogowego (KRBRD) [7]. Jest to organ powołany w celu wspomagania Rady Ministrów w sprawach bezpieczeństwa ruchu drogowego. Członkowie RBRD zbierają dane i zlecają badania na temat sytuacji na

polskich drogach. Na podstawie zgromadzonych danych tworzone są statystyki o liczbie nietrzeźwych kierowców, wypadkach i ich przyczyn, zgonów itp. Analiza uzyskanych wyników ma służyć poprawie bezpieczeństwa na drogach. Stąd też, wykorzystanie tych danych powinno być poprzedzone oceną jakości, by wnioski wynikające z analizy były przekonywujące.

Krajowa Rada Bezpieczeństwa Ruchu Drogowego posiada bazę danych, której źródłem są dane zbierane w różny sposób, między innymi za pomocą:

- monitoringów na drogach,
- raportów policji,
- ankiet.

Dane, które podlegały ocenie jakościowej w tym przykładzie, pochodzą z ankiet przeprowadzonych w ramach projektu Omni w listopadzie 2014 roku i dotyczyły opinii Polaków na temat bezpieczeństwa ruchu drogowego. Celem tego przedsięwzięcia było ustalenie poziomu bezpieczeństwa na drogach, ocena znajomości technik udzielania pierwszej pomocy oraz stan i uzasadnienie wykorzystania urządzeń i mechanizmów służących poprawie bezpieczeństwa na drogach (pomiar prędkości, sygnalizacja świetlna, ograniczenia w ruchu, itp.). W raporcie nie przedstawiono stopienia wiarygodności źródeł danych, ani nie została zbadana ich jakość (brak takiej informacji).

4.2. Analiza danych źródłowych

W plikach źródłowych znajduje się tabela główna składająca się z 209 kolumn. Oprócz kolumny, która jest jednoznacznym identyfikatorem wywiadu przeprowadzonego przez ankietera, znajdują się kolumny zawierające klucze obce do odpowiedzi respondenta. Nazwy kolumn to kody, którym odpowiada pytanie zadawane przez ankietera. Dodatkowo, w źródle znajduje się 38 tabel podzielonych ze względu na kod pytania wraz z odpowiedziami respondenta. Informacje opisujące dane źródłowe znajdują się w plikach z metadanymi i udostępnionym raporcie.

4.3. Profilowanie danych

Profilowanie danych wykonano dwukrotnie: raz przy użyciu oprogramowania Oracle Enterprise Data Quality, a następnie za pomocą Microsoft Server Data Quality Client i SQL Server Integration Services.

Podstawowe funkcjonalności, które były rozpatrywane i możliwe do wykorzystania przez Autorów, w przeprowadzonych eksperymentach za pomocą wspomnianych narzędzi, przedstawia tabela 1.

Tabela 1. Porównanie narzędzi do profilowania i czyszczenia danych

Funkcjonalność	Oracle EDQ	MS Data Quality	MS SSIS
Sprawdzanie typów danych (bez zdefiniowania)	Tak	Nie	Nie
Sprawdzanie minimalnej, maksymalnej wartości	Tak	Tak	Nie
Sprawdzanie liczby znaków	Tak	Tak	Nie
Sprawdzenie częstotliwości występowania	Tak	Tak	Tak
Sprawdzenie unikalnych wartości	Tak	Tak	Tak
Sprawdzenie liczby duplikatów	Tak	Tak	Tak
Sprawdzenie wzorów i masek	Tak	Tak	Tak
Sprawdzenie, czy badane wartości występują w innej tabe-	Tak	Tak	Tak

li/słowniku			
Sprawdzenie ciągów białych znaków	Tak	Tak	Tak
Tworzenie zapytań SQL	Nie	Tak	Nie
Tworzenie skryptów w jednym z języków programowania (C#, Java, C++ itp.)	Tak	Tak	Nie
Tworzenie ograniczeń (typu większe od, mniejsze od, równe) odwołując się do badanej wartości	Tak	Tak	Tak
Tworzenie ograniczeń (typu większe od, mniejsze od, równe) odwołując się do innej wartości w źródle	Nie	Nie	Tak
Sprawdzanie zależności pomiędzy kolumnami	Tak	Tak	Tak
Wykonywanie operacji zamiany, wycinania, konkatenacji znaków	Tak	Tak	Nie
Wykonywanie operacji matematycznych	Tak	Tak	Nie
Intuicyjność narzędzia	wysoka	średnia	niska
Elastyczność narzędzia	średnia	wysoka	niska

Przykładowe wyniki uzyskane podczas profilowania danych przedstawiono w tabelach 2. oraz 3.

Tabela 2. Fragment wyników uzyskanych podczas profilowania podcharakterystyk **Zgodność dziedzinowa** i **Spójność**

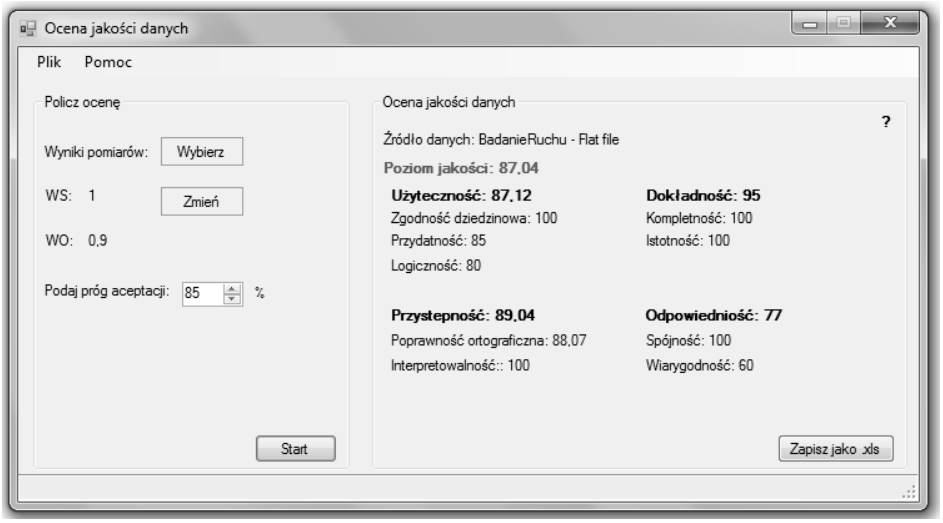
Nazwa tabeli	**Format**		**Białe znaki**		**Fluktuacja**		**Zgodność typu**		**Akceptacja klucza**		**Ograniczenia**	
	P	**N**	**P**	**N**	**P**	**N**	**P**	**N**	**P**	**N**	**P**	**N**
DOCHOD_GS_DIM	2	0	14	0	0	0	28	0	14	0	0	0
DOCHOD_HHPP_DIM	2	0	5	0	0	0	12	0	6	0	0	0
DOCHOD_OS_DIM	2	0	14	0	0	0	28	0	14	0	0	0
…												
WYNIKI_ANKIETY_FCT	209	0	2000	0	194	5017	209000	0	1	0	3509	0
Suma	**494**	**0**	**3880**	**8**	**194**	**5017**	**253117**	**0**	**219**	**0**	**3927**	**0**

Tabela 3. Fragment wyników uzyskanych podczas profilowania podcharakterystyk **Poprawność ortograficzna i Kompletność**

Nazwa tabeli	**Łączność historyczna**		**Unikatowość**		**Błędy ort.**		**Brakujące wartości**		**Współzależność**	
	P	**N**	**P**	**N**	**P**	**N**	**P**	**N**	**P**	**N**
DOCHOD_GS_DIM	0	0	14	0	30	0	28	0	0	0
DOCHOD_HHPP_DIM	0	0	6	0	15	0	12	0	0	0
DOCHOD_OS_DIM	0	0	14	0	30	0	28	0	0	0
…										
WYNIKI_ANKIETY_FCT	0	0	0	0	10	5	209000	0	198000	0
Suma	**0**	**0**	**636**	**0**	**1299**	**176**	**210481**	**0**	198000	**0**

4.4. Ocena jakości danych

Ocena jakości danych jest procesem żmudnym i czasochłonnym. W celu usprawnienia tego procesu została stworzona aplikacja „Ocena jakości danych" (rys. 2.), wspierająca ten proces z wykorzystaniem zaproponowanego w pracy modelu jakości danych.

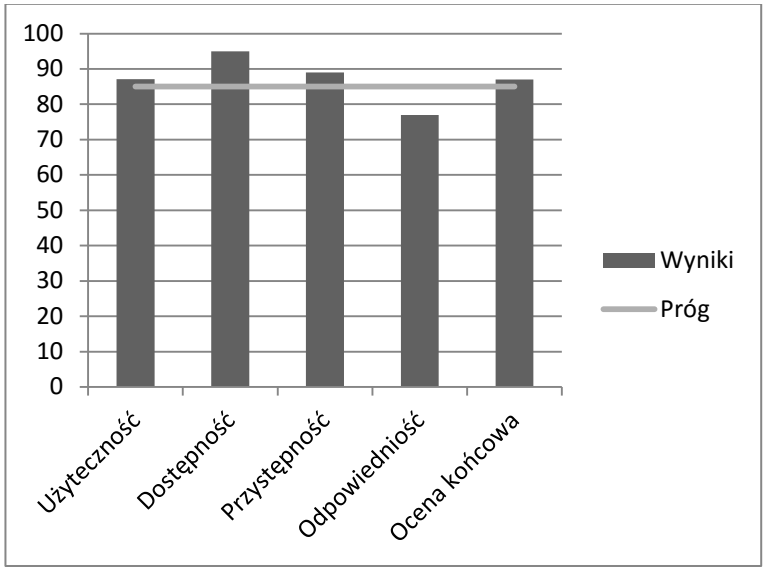


Rysunek 2. Ocena jakości danych źródłowych [7]

Poniżej przedstawiono wyniki oceny jakości danych ankietowych KRBRD [7], uzyskane za pomocą opracowanej aplikacji:

- Czy dane są przydatne? – wartość miary **Przydatność** = 85%
- Czy dane są logiczne? – wartość miary **Logiczność** = 80%
- Czy możemy ufać, zawierzać danym? – wartość miary **Wiarygodność** = 60%
- Czy interpretacja danych jest jednoznaczna? – wartość miary **Interpretowalność** = 100%
- Czy znana jest dziedzina źródła danych? – wartość miary **Interpretowalność** =100%
- Czy dane dotyczą rozpatrywanej dziedziny? – wartość miary **Istotność** =100%

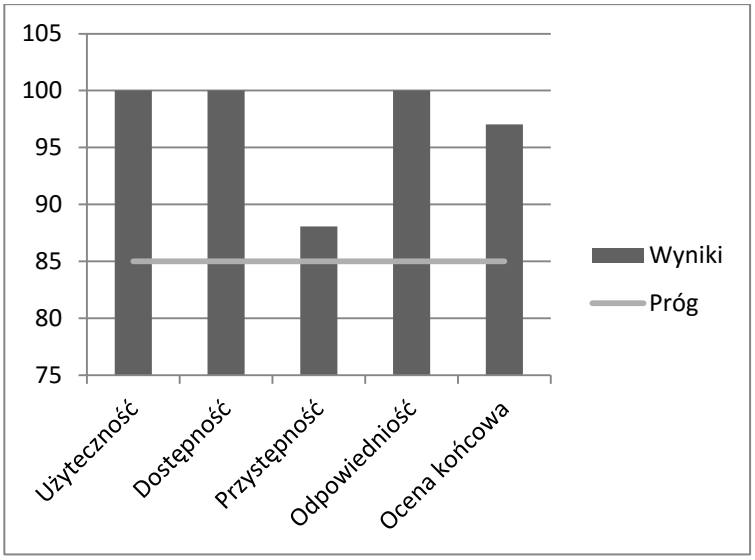
Uzyskane wartości miar subiektywnych mogą być obarczone znacznym błędem ze względu na brak informacji o sposobie przeprowadzenia procesu ankietyzacji i wiedzy na temat badanych grup społecznych. Dlatego też, przy wyznaczaniu wartości miar, przyjęto mniejszą wartość wagi dla oceny subiektywnej w stosunku do wartości wagi oceny obiektywnej. Próg akceptacji ustalono na poziomie 85%. Wynik oceny rozpatrywanego zbioru danych został przedstawiony na rysunku 3. Poziom jakości danych przekracza, przyjęty w eksperymencie, próg akceptacji. Analizując wartości miar charakterystyk, należy zwrócić uwagę na negatywny wpływ miar charakterystyki **Odpowiedniość**, które obniżają wartość oceny końcowej. Jest to spowodowane niską oceną wiarygodności i spójności danych (wartości miar podcharakterystyk **Wiarygodność** i **Spójność**).



Rysunek 3. Wykres przedstawiający wyniki oceny jakości danych źródłowych

Mimo wszystko uznano, że dane są wartościowe i mogą być wykorzystane do analiz biznesowych. Potwierdzeniem tego są wnioski uzyskane z raportów dotyczących poziomu bezpieczeństwa na drogach krajowych w Polsce [7].

W celu weryfikacji wpływu czynników subiektywnych na ocenę jakości danych, przeprowadzono badanie, w którym stworzono instancję modelu bez podcharakterystyk subiektywnych: **Przydatność, Logiczność, Wiarygodność, Istotność i Interpretowalność**. Następnie korzystając z aplikacji obliczono poziom jakości, otrzymując wynik 97.02% (rysunek 4.).



Rysunek 4. Wykres przedstawiający wyniki oceny jakości z wyłączeniem podcharakterystyk subiektywnych

Pomijając podcharakterystyki subiektywne podczas oceny jakości danych, otrzymano dużo wyższy poziom jakości, niż we wcześniejszym badaniu. Różnica pomiędzy poziomami jakości wynosi prawie 10%. Taki wynik oznacza, że czynnik ludzki oraz osobiste preferencje mają duży wpływ na ocenę jakości danych. Jednak w obu przypadkach, poziom jakości przekroczył próg akceptacji dowodząc, iż jakość badanych danych jest wysoka, zgodna z wymaganiami, niezależnie od perspektywy.

5. Podsumowanie

W pracy zaproponowano model oceny jakości danych opracowany na podstawie normy jakości ISO/IEC 25012. Model składa się ze zbioru czterech charakterystyk **Użyteczność**, **Dokładność**, **Przystępność**, **Odpowiedniość**. Charakterystyki umożliwiają ocenę jakości danych z różnych perspektyw z uwzględnieniem możliwości oceny obiektywnej (bez wpływu opinii osoby uczestniczącej w procesie oceny) oraz oceny subiektywnej wyrażającej opinię osoby oceniającej. W zależności od potrzeb i celu oceny jakościowej danych można tworzyć instancję modelu jakości, która uwzględnia wymagany zakres oceny danych, poprzez wybór odpowiednich charakterystyk, ich podcharakterystyk i miar oceny. Model jest otwarty, to znaczy, że można rozszerzyć zakres oceny danych uzupełniając zbiór zaproponowanych elementów składowych modelu.

Wstępne wyniki uzyskane z przeprowadzonych eksperymentów badania jakości danych pobranych z różnych źródeł potwierdzają użyteczność modelu jakości danych. Zdajemy sobie jednak sprawę, że ostateczna ocena użyteczności modelu i bazujące na nim zaproponowane podejście do oceny jakości danych będzie możliwe po wykonaniu kolejnej serii ocen oraz konfrontacji uzyskanych wyników z opiniami użytkowników tych danych.

Literatura

[1] Drabik L., Sobol E. (2014), *Słownik Języka Polskiego PWN*, Wydawnictwo Naukowe PWN, Warszawa
[2] Even A., Shankaranarayanan G., *Understanding impartial versus utility-driven quality*, Information Systems Department, 2007
[3] Fisher C. W., Madnick S. E., Pierce E. M., *Information Quality*, AMIS, New York, 2005
[4] Hamrol A., *Zarządzanie jakością. Teoria i praktyka*. , Państwowe Wydawnictwo Naukowe, Warszawa, 1998
[5] ISO/IEC 25012:2008 *Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Data quality model*, 2008
[6] ISO/IEC CD 25011 *Information technology -- Service Quality Requirement and Evaluation(SQuaRE) -- IT Service Quality Model*, 2008
[7] Krajowa Rada Bezpieczeństwa Ruchu Drogowego, *Badanie postaw społeczeństwa względem bezpieczeństwa ruchu drogowego,* http://www.krbrd.gov.pl/pl/72-badania.html, 2015
[8] Loshin D., *The Practitioner's Guide to Data Quality Improvement*, Morgan Kaufmann, 2011
[9] Olson J. E., *Data quality: the Accuracy Dimension*, Morgan Kaufmann, San Francisco, 2003
[10] Svolba G., *Data Quality for Analytics Using SAS*, SAS Institute Inc., 2012
[11] Śpiewla T., *Encyklopedia zarządzania*, http://mfiles.pl/pl/index.php/Niezawodno%C5%9B%C4%87, 2014
[12] Zymonik J., *Zarządzanie jakością*, http://www.ioz.pwr.wroc.pl/Pracownicy/j_zymonik/Pliki/TEKST %20Filozofia%20jako%9Cci..pdf

Rozdział 7

Metoda oceny użyteczności i funkcjonalności hurtowni danych

1. Wprowadzenie

Użyteczność systemu informatycznego zwana także jego jakością użytkową oznacza stopień jego dopasowania do potrzeb użytkowników. Każdy system informatyczny realizuje określone funkcje oraz daje określone możliwości użytkowe, które definiują jego funkcjonalność. Norma ISO 9126 definiuje pięć charakterystyk funkcjonalności, do których należą: odpowiedniość (*suitability*), dokładność (*accuracy*), współdziałanie (*interoperability*), bezpieczeństwo (*security*), zgodność (*functionality compliance*). Pojęcia „użyteczność" oraz „funkcjonalność" nie są więc tożsame. Zwiększenie funkcjonalność systemu oprogramowania może jednak oznaczać zwiększenie jego użyteczności. Użyteczność systemu informatycznego zależy od tego, jakie funkcje on realizuje, jak łatwo się go używa, jak system wspomaga użytkownika czy w jaki sposób system ten spełnia oczekiwania jego użytkowników. Dlatego miarami użyteczności systemu informatycznego jest łatwość wyuczenia się i obsługi tego systemu, skuteczność, efektywność i możliwość wykorzystania uzyskanych z tego systemu wyników pracy oraz poziom satysfakcji z jego użycia a także funkcjonalność w sensie definicyjnym. Zapewnienie użyteczności i funkcjonalności systemu informatycznego wydłuża okres jego życia. Wydłużenie tego okresu jest w pełni uzasadnione zwłaszcza, gdy jego wymiana na nowy system jest kosztowna, wiąże się ze stosunkowo dłuższym okresem jego wdrażania, podczas gdy organizacja (firma) musi funkcjonować i realizować swoje zadania biznesowe.

Jednym z rodzajów systemów informatycznych są hurtownie danych (ang. Data Warehouse system). Wg Inmona [1] hurtownia danych jest „…tematycznie zorientowaną, spójną, chronologiczną i niezmienną kolekcję danych, stanowiącą podstawę procesu podejmowania decyzji…", zwłaszcza na szczeblu strategicznym i taktycznym.

Słaba użyteczność hurtowni danych obniża produktywność, efektywność i zasadność realizacji działań organizacji w jej biznesowym otoczeniu. Hurtownia danych jest użyteczna, gdy model danych w hurtowni jest użyteczny i gdy użyteczne są funkcje oraz użyte w niej metody, które wspomagają użytkownika w procesach podejmowaniu decyzji zwłaszcza strategicznych czy taktycznych. Model danych w hurtowni może tracić na użyteczności ze względu na zmienne w czasie potrzeb jej użytkowników, procesy starzenia się danych w niej zgromadzonych, zmienne z upływem czasu uwarunkowania organizacji w jej biznesowym otoczeniu a także zmienne cele strategiczne organizacji w stosunku do chwili, gdy hurtownię danych utworzono. Dlatego ważnym zadaniem inżynierii systemów informatycznych klasy hurtownia danych jest pomiar i ocena użyteczności tej hurtowni by procesy wspomagane przez ten system były realizowane i zapewniane w dowolnej chwili eksploatacji hurtowni.

W literaturze [2],[5] i [6] podejmowane są próby budowy hurtowni danych, w której ocenia się użyteczność danych w niej zgromadzonych przy pomocy takich wskaźników jak świeżość danych, dostępność czy częstotliwość ich użycia. Jednak w pracach tych mniej dotychczas uwagi poświęcono wieloaspektowej i zarazem globalnej ocenie użyteczności hurtowni w dowolnej chwili czasu oraz sposobom podwyższenia tej użyteczności, gdy zajdzie taka potrzeba.

Uzasadnionym więc są pytania jest jak ocenić użyteczność i funkcjonalność hurtowni danych, kiedy i jaką metodą to realizować w całym cyklu jej życia? Temu zagadnieniu poświęcony jest kolejny rozdział artykułu.

2. Możliwości oceny użyteczność i funkcjonalność hurtowni danych?

O użyteczności hurtowni danych należy myśleć na etapie jej projektowanej oraz wówczas, gdy jest ona eksploatowana. Użyteczność hurtowni danych jest zmienna w czasie z różnych powodów, do których należą m.in. przyjęte metody wytwarzania takich systemów, zaangażowanie użytkownika w proces jej budowy, rozwój technologii informatycznych, które mają wpływ na utratę użyteczności czy tzw. „starzenie się" systemu, podatność na zmiany w systemie. Skoro użyteczność hurtowni danych nie jest stała to należy dokonywać jej pomiaru i oceny po to by dążyć do zapewnienia tej użyteczności w jak najpełniejszy sposób. Oceny użyteczności i funkcjonalności hurtowni danych można i trzeba dokonać z następujących punktów widzenia:

- Projektowania, gdy powstaje pierwsza wersja hurtowni, z punktu widzenia potrzeb,
- Satysfakcji i zainteresowania użytkownika, gdy korzysta on z hurtowni,
- Ekspertów, którzy mogą ocenić użyteczność systemu informatycznego, jakim jest system z hurtownią danych
- Efektem procesów podejmowania decyzji biznesowych jest również ocena stopnia realizowalności strategii organizacji w oparciu o hurtownię danych.

Oceny użyteczności hurtowni danych w początkowych fazach projektowych polega na zbieraniu informacji na podstawie, których zostaną określone wymagania użytkownika względem systemu. Zaimplementowanie nawet w późniejszych etapach wytwarzania hurtowni danych, funkcjonalności spełniających takie wymagania z pewnością korzystnie wpłynie na użyteczność finalnego produktu. Poziom użyteczność i funkcjonalności hurtowni danych wynika także z przebiegu procesu jej projektowania, programowania i wdrażania. Gdy tworzy się hurtownię danych po raz pierwszy to można posłużyć się tu znanymi metodami projektowania funkcjonalności i użyteczności systemu oprogramowania uwarunkowanego czasem, metodami tworzenia analitycznego oprogramowania czy metodami projektowania systemu z bazą danych. Praktyka a także literatura (choćby [5]) wskazują, że projektowanie systemu z hurtownią danych różni się jednak od projektowania dowolnego systemu informatycznego i zaproponowano tzw. adaptacyjną metodę projektowania i wdrażania hurtowni danych. Istotnym elementem tej metody jest określenie chwili, w której hurtownia staje się nieużyteczna, nie spełnia pokładanych w niej wymagań. Ponadto w literaturze (choćby [6]) podejmowane są próby budowy takiej hurtowni danych, w której ocenia się użyteczność danych w niej zgromadzonych przy pomocy takich wskaźników jak świeżość

danych, dostępność czy częstotliwość ich użycia. Jednak w pracach tych mniej uwagi poświęcono odpowiedzi na pytanie jak na bieżąco w trakcie użytkowania i eksploatacji hurtowni te oceny wykorzystać, jak określić chwile, w których hurtownia staje się nieużyteczna czy niefunkcjonalna oraz jak wpłynąć na podniesienie jej poziomu użyteczności i funkcjonalności.

Użyteczność hurtowni danych może być również oceniana przez niezależnych i kompetentnych ekspertów. Opinie ekspertów mogą być wykonywane zarówno na etapie projektowania hurtowni jak również na etapie jej eksploatacji. Hurtownia danych może być oceniana z punktu widzenia funkcji, jakie realizuje, czyli z punktu widzenia procesów wspomagania decyzji zwłaszcza taktycznych i strategicznych, tj. wyników działań biznesowych zrealizowanych w oparciu o wspomagane hurtownią danych decyzje, bowiem efektem procesów podejmowania decyzji biznesowych jest również ocena stopnia realizowalności strategii organizacji w oparciu o hurtownię danych. Im lepsze jest wspomaganie informacyjne procesów decyzyjnych przez hurtownię danych tym wyższa jest użyteczność i funkcjonalność hurtowni.

Na każdym z wymienionych poziomów oceny użyteczności i funkcjonalności hurtowni danych ocena ta jest dokonywana wielokryterialnie. Można jej dokonać niezależnie od zaprezentowanych punktów widzenia stosując następujące metody:

- Wyboru najlepszego rozwiązania,
- Klasyfikacji wariantów w obrębie wyróżnionych rozwiązań,
- Ustalania rankingu (preferencji, wag) najlepszego wyboru rozwiązania.

Metody oceny oparte na poszukiwaniu najlepszego modelu użyteczności i funkcjonalności hurtowni danych polegają na określeniu funkcji najlepszego wyboru przy ustalonych ograniczeniach. Są to metody wymagające opisu problemu decyzyjnego w kategoriach modelu optymalizacji matematycznej. Stosowalność tych metod jest niewielka choćby, dlatego, że tylko dla nielicznych przypadków można uzyskać tą drogą formalne rozwiązanie problemu najlepszego wyboru.

Metody klasyfikacji wariantów rozwiązań stosuje się w wyborze koncepcji realizowalnej użyteczności i funkcjonalności hurtowni na etapie projektowania zwłaszcza jej pierwszej wersji.

Najczęstszą klasą metod stosowanych do oceny użyteczności i funkcjonalności systemu z hurtownią danych są metody oparte na ustalaniu rankingu i wybór tego rozwiązania, które jest najlepsze z punktu widzenia wielu kryteriów w kontekście tego rankingu. Z formalnego punktu widzenia metody te koncentrują się na tworzeniu charakterystycznej relacji między wariantami wyboru, które reprezentują określone preferencje decydenta i są nazywane „relacją przewyższania". Relacją przewyższania alternatywy (wariantu) *A* nad alternatywą (wariantem) *B* opisuje się stwierdzeniem: alternatywa *A* jest preferowana względem alternatywy *B*. Test na prawdziwość relacji przewyższania można przeprowadzać z punktu widzenia jednego lub wielu kryteriów tej relacji. Relacja przewyższania dwóch alternatyw *A* i *B* definiuje stany wspomagające podejmowanie decyzji wyboru alternatywy *A* nad alternatywą *B*, *B* nad alternatywą *A* lub stan, w którym te alternatywy nie są rozróżnialne. Relacja przewyższania charakteryzuje się nieprzechodniością.

Niezależnie od wybranych metod ocena użyteczności i funkcjonalności hurtowni dotyczy heterogenicznych ich środowisk, na które składają się między innymi różne wskaźniki (kryteria, parametry) ocen, różne kategorie wskaźników oceny (np. ilościowe i jakościowe), różne poziomy odniesienia wyników ocen (np. na etapie wdrażania,

na etapie eksploatacji hurtowni), różne typy danych na podstawie, których dokonuje się oceny(np. satysfakcja, świeżość, częstotliwość użycia) itp.

Oceny użyteczności i funkcjonalności hurtowni wykonywana na różnych poziomach z różnego punktu widzenia jest każdorazowo wielokryterialnym wyborem czynników wpływających na użyteczność i funkcjonalność hurtowni danych – jest wielokryterialnym podejmowaniem decyzji o użyteczności i funkcjonalności hurtowni w okresie od chwili T_0. Decyzja taka ma odpowiednie skutki w przyszłości po chwili t_0 w obszarze użyteczności i funkcjonalności hurtowni. O ile skutki błędnej decyzji na etapie projektowania pierwszej wersji hurtowni danych są naprawialne w okresie jej eksploatacji o tyle skutki złej oceny użyteczności i funkcjonowania hurtowni w okresie jej eksploatacji mogą mieć większe znaczenie szczególnie z punktu widzenia zmiennej w czasie strategii biznesowej organizacji – firmy, dla której hurtownia funkcjonuje.
W istniejących koncepcjach budowy hurtowni danych z punktu widzenia oceny użyteczności i funkcjonalności na etapie projektowania wykorzystywana jest tzw. adaptacyjna metoda projektowania.

Cechą szczególną metody adaptacyjnej projektowania hurtowni opisanej choćby w [6] jest iteracyjny cykl badania oraz oceny użyteczności, funkcjonalności i efektywności hurtowni danych z punktu widzenia nie tylko zmiennych w czasie preferencji użytkowników czy zmiennych potrzeb biznesowych organizacji, ale również ze względu na zmienny w czasie źródła dostępu do danych czy inne możliwe do wykorzystania metody wspomagania procesów decyzyjnych z użyciem hurtowni. Dzięki tej metodzie możliwe jest usprawnianie systemu z hurtownią danych pod względem jej użyteczności i funkcjonalności.

3. Wielokryterialna oceny użyteczności oraz funkcjonalności hurtowni danych na etapie jej eksploatacji

Ocena użyteczności i funkcjonalności na etapie eksploatacji hurtowni danych stwarza możliwości do ich zwiększenia oraz dostosowania do zmiennych potrzeb informacyjnych organizacji, jakie się pojawiły od czasu jej wdrożenia. Technologiami i koncepcjami, które mogłyby wspomagać ocenę użyteczności i funkcjonalności hurtowni danych w okresie jej eksploatacji może być budowa hurtowni danych, w której wdrożono następujące rozwiązania:

- Bieżącą ewidencję (w trybie on_line) (pomiar) wartości parametrów (miar) charakteryzujących użyteczność i funkcjonalność w warstwie tzw. metadadanych hurtowni zgodnie z koncepcją DWQ (ang. Datawarehouses Ouality),
- Obliczanie i przechowywanie w hurtowni danych wskaźników dopasowania organizacji do jej biznesowego otoczenia, wskaźników dopasowania danych przechowywanych w hurtowni do potrzeb użytkowników oraz możliwości zasilania hurtowni w nowe dane w metodzie dopasowania hurtowni danych do zmiennych potrzeb informacyjnych przedsiębiorstwa (organizacji) zgodnie z metodą adaptacyjnego dopasowania hurtowni danych do zmiennych potrzeb przedsiębiorstwa.
- Wielowersyjny model hurtowni danych wraz z systemem zarządzania wielowersyjną hurtownią danych.

Koncepcję DWQ zaprezentowana choćby w pracy [2]. U jej podstaw leżą bieżący pomiar i kontrola (najczęściej za pośrednictwem administratora) wskaźników jakości danych takich jak użyteczność, spójność, niezawodność, świeżość itp. Wskaźniki te mogą być ustalane administracyjnie i nie wyrażają wszystkich aspektów oceny funkcjonalności i użyteczności hurtowni w ujęciu systemowym.

Takie systemowe podejście do użyteczności i funkcjonalności hurtowni danych posiada koncepcja adaptacyjnego dopasowania hurtowni danych do zmiennych potrzeb informacyjnych zwana w skrócie AMDH. Metodę tę zaprezentowano w pracy [6]. U jej podstaw leży możliwość przechowywania w hurtowni danych następujących miar ilościowych, tzw. wskaźników dopasowania, do których należą:

- Wskaźniki dopasowania organizacji do jej biznesowego otoczenia zgodnie ze strategią tej organizacji,
- Wskaźniki dopasowania danych przechowywanych w hurtowni do potrzeb użytkowników,
- Współczynnik pokrycia potrzeb informacyjnych w hurtowni danych,
- Współczynnik wykorzystania możliwości hurtowni,
- Współczynnik wykorzystania potencjału przedsiębiorstwa,
- Współczynnik pokrycia potrzeb otoczenia.

Do określenia tych wskaźników wykorzystano:

- Podstawy inżynierii systemów działania organizacji,
- Tabele identyfikacji matematycznej zbudowane dla organizacji oraz hurtowni danych w ich przedziale systemowym,
- Równania potencjału, użyteczności, potrzeb i możliwości dla organizacji i hurtowni danych w formie wzorów oraz algorytmów ich rozwiązywania,

Ze względu na ograniczony charakter artykułu wszystkie wskaźniki, równania i podstawy merytoryczne pominięto zwarzywszy na fakt, ze są one zaprezentowane szczegółowo w literaturze [6].

Dzięki koncepcji systemu z wielowersyjną hurtownią danych możliwe będzie jej przeprojektowywanie, tak by osiągnąć lepszą wielokryterialną ocenę użyteczności i funkcjonalności hurtowni na podstawie oceny zgromadzonych w hurtowni miar jej użyteczności i funkcjonalności zgodnie z koncepcjami AMDH oraz DWQ. Model wielowersyjnej hurtowni danych zaprezentowano choćby prace [3] oraz [7].

W dalszej części artykułu zaprezentowano metodę oceny użyteczności i funkcjonalności hurtowni danych na etapie jej eksploatacji z wykorzystaniem koncepcji AMDH, DWQ oraz wielowersyjności modelu danych. Metoda ta jest również zgodna z adaptacyjną metodą projektowania hurtowni danych, która w kolejnych etapach realizowanych podczas eksploatacji hurtowni pozwoli dopasować tę hurtownię do nowych potrzeb. Dodatkowo należy zaznaczyć, że metoda AMDH umożliwia ocenę hurtowni danych z punktu widzenia jakości decyzji wypracowanych w oparciu o dane zgromadzone w hurtowni. Miarą tej jakości jest choćby współczynnik dopasowania organizacji do jej biznesowego otoczenia. Jest to również ocena użyteczności i funkcjonalności hurtowni zgodnie z jej przeznaczeniem.

Załóżmy, ze zgodnie z koncepcjami DWQ i MDHD w hurtowni danych gromadzone są miary odwzorowujące użyteczność i funkcjonalność hurtowni w kolejnych chwilach osi czasu po czasie T_0, w którym rozpoczęła się eksploatacja hurtowni i oznaczmy te miary zgodnie ze wzorem (1)

$$M_1(T), M_2(T), .., M_n(T) \quad (1)$$

gdzie n – liczba miar (ocen),
$M_i(T)$ – wartość i-ta miara ocena użyteczności i funkcjonalności hurtowni danych w chwili T,
$T > T_0$.

Każdej ocenie miary *Mi* ze wzoru (1) można przypisać kierunek poszukiwania pożądanej wartości (ilościowej lub wartościowej) tej miary oraz rangę (wagę tej miary w ocenie użyteczności i funkcjonalności hurtowni. Oznaczmy je odpowiednio symbolem *Ki* oraz *Ri*.

Ponieważ mechanizmy hurtowni danych umożliwiają pomiar miar (1) w kolejnych chwilach eksploatacji hurtowni to dostępne są stany tych miar w kolejnych chwilach, osi czasu. Oznaczmy te stany zgodnie ze wzorem (2) symbolem *M*

$$M(T_j) = (M_1(T_j), M_2(T_j), \ldots, M_n(T_j),) \quad (2)$$

gdzie:
j- indeks (znacznik) kolejnej chwili na osi czasu $j=0,1,2,\ldots$
Tj – chwila na osi czasu $(tj>t_0)$,
T_o – chwila, w której rozpoczęto eksploatację hurtowni danych,
$Mi(T_j)$ – miara oceny M_i ze wzoru (1) w chwili T_j.

Odpowiednio do stanu *M* w kolejnych chwilach osi czasu znane są kierunki poszukiwania pożądanych wartości tych miar oraz ich rangi. Oznaczono je odpowiednio wzorami (3).

$$K(T_j) = (K_1(T_j), K_2(T_j), \ldots, K_n(T_j),) \quad (3)$$

$$R(T_j) = (R_1(T_j), R_2(T_j), \ldots, R_n(T_j),)$$

gdzie:
j- indeks (znacznik) kolejnej chwila osi czasu $j=0,1,2,\ldots$
n – liczba miar (ocen),
Tj – chwila czasu $(Tj>T_0)$,
T_o – chwila, w której rozpoczęto eksploatację hurtowni danych,
$Ki(T_j)$ – kierunek poszukiwania miary oceny M_i ze wzoru (1) w chwili T_j.
$Ri(T_j)$ – ranga miary oceny M_i ze wzoru (1) w chwili T_j.

Na podobnych zasadach mogą być prowadzone niezależnie od eksploatacji hurtowni danych pomiary użyteczności i funkcjonalności hurtowni w grupie ekspertów. Pomiary takie powinny być ukierunkowane na oceny użyteczności danych zgromadzonych w hurtowni oraz funkcje realizowane w hurtowni. Jedną z miar oceny eksperckiej może być tzw. próg dopuszczalnego spadku tej użyteczności i funkcjonalności.

Dzięki temu w wyniku badania opinii ekspertów możemy zgromadzić następujące miary zgodne ze wzorem (4) ze wskazaniem dla nich kierunków zmian oraz rang.

$$M''_1(T), M''_2(T), \ldots, M''_m(T),) \quad (4)$$

$K''(T_j) = (K''_{1''}(T_j), K''_2(T_j), \ldots, K''_m(T_j))$

$R''(T_j) = (R''_1(T_j), R''_2(T_j), \ldots, R''_m(T_j),)$

gdzie:
j- indeks (znacznik) kolejnej chwila osi czasu $j=0,1,2,\ldots$
m – liczba miar (ocen),
Tj – chwila $(Tj>T_0)$,
T_o – chwila, w której rozpoczęto eksploatację hurtowni danych,
$K''p(T_j)$ – kierunek poszukiwania miary oceny M''_p ze wzoru (4) w chwili $T_{j.}$
$R''p(T_j)$ – ranga miary oceny M''_p ze wzoru (4) w chwili $T_{j.}$

Problem oceny użyteczności i funkcjonalności hurtowni danych sprowadza się do oceny stanów zdefiniowanych jako uporządkowana szóstka elementów zgodnie ze wzorem (5).

$$S(T_j)=\{M(T_j),K(T_j),R(T_j),M''(T_j),K''(T_j),R''(T_j)\} \quad (5)$$

gdzie:
j- indeks (znacznik) kolejnej chwili na osi czasu $j=0,1,2,\ldots$
Tj – chwila na osi czasu $(tj>t_0)$,
T_o – chwila, w której rozpoczęto eksploatację hurtowni danych,
$M(T_j)$ –oceny miar zgodne ze wzoru (2) w chwili $T_{j.}$
$K(T_j)$ – kierunki zmian ocen zgodne ze wzorem (3) w chwili $T_{j.}$
$R(T_j)$ – rangi (wagi) oceny zgodne wzoru (3) w chwili $T_{j,}$
$M''(T_j)$ –oceny eksperckie miar zgodne ze wzoru (4) w chwili $T_{j.}$
$K''(T_j)$ – kierunki zmian ocen eksperckich zgodne ze wzorem (4) w chwili $T_{j.}$
$R''(T_j)$ – rangi (wagi) oceny eksperckich zgodne wzoru (4) w chwili $T_{j.}$

U podstaw oceny pojedynczych stanów $S(T_j)$ (ocena użyteczności i funkcjonalności hurtowni danych w chwili T_j (gdzie $T_j > T_0$, T_0 – chwila rozpoczęcia eksploatacji hurtowni danych) mogą być być zastosowane metody oparte na regułach zgodności i niezgodności stanów opisane choćby w [4]. Ideę taka umożliwia ocenę stanu poprzez wzajemne porównywanie miar $M_i(Tj)$ dokonywane przez decydenta a nawet wówczas, gdy miary ocen użyteczności i funkcjonalności hurtowni ze wzoru (1), (2) oraz odpowiednio (3) są heterogeniczne (ilościowe i jakościowe, porównywalne ze sobą lub nie), ale zawsze w obrębie jednej miary alternatywy wyboru charakteryzują się zbliżonymi wartościami tych miar. Jedną z grupy metod, które oparte są na ocenie stanu $S(T_j)$ (ocena użyteczności i funkcjonalności hurtowni danych w chwili T_j (gdzie $T_j > T_0$, T_0 – chwila rozpoczęcia eksploatacji hurtowni danych) ze wzoru (5) są metody *ELECTRE*, *PROMETEE* oraz *MELCHIOR*. W tabeli 1 zestawiono krótką charakterystykę tych metod.

Gdy w analizie miar oceny użyteczności i funkcjonowania nieznane są kierunki poszukiwania pożądanych wartości tych miar, tzn. nie są znane $K(T_j)$ lub $K''(T_j)$ a znane są odpowiednie rangi miar oceny $R(Tj)$ lub $R''(Tj)$ można wykorzystać w ocenie użyteczności i funkcjonalności hurtowni metodę AHP (ang. *Analitic Hierarchy Proces*s), która oparta jest na konwersji subiektywnych ocen (ocen ekspertów) i wykorzystuje porównania parami, jako podstawę oceny alternatywnych rozwiązań.

Tabela 1. Charakterystyka metod oceny wielokryterialnej z rankingiem alternatyw i relacją przewyższania alternatyw

ELECTRE IS	Rozwiązuje problem wyboru przy istnieniu wielu kryteriów, z progami, relacją równoważności i preferencji. Główna część tej metody oparta na teorii grafów.
ELECTRE II	Wykorzystuje dwie relacje przewyższania: słabą i silną i ocenę opartą na teorii grafów
ELECTRE III	Przewyższanie wyrażane jest za pomocą indeksów wiarygodności.
ELECTRE IV	Jak ELECTRE II z tym, że nie wykorzystuje wag.
ELECTRE Iv	Procedura polega na redukowaniu rozmiaru zbioru niezdominowanych alternatyw.
ELECTRE TRI	Jak Electre II tylko zajmuje się ona zagadnieniem sortowania, wykorzystuje pseudo-kryteria.
PROMETHEE I	Bazuje na podobnych zasadach jak ELECTRE. Wprowadza jednak sześć funkcji, które określają preferencje dla kryterium oceny oraz metodę określania częściowego rankingu alternatyw
PROMETHEE II	Rozszerza PROMETHEE I, tworzenie pełnego rankingu alternatyw
MELCHIOR	Jest rozszerzeniem metody ELECTRE IV

W przypadku oceny miar jakościowych a nie ilościowych stosuje się skale porządkowe lub przedziałowe. Dlatego metody wykorzystujące wzajemne porównania dokonywane przez decydenta wykonującego ocenę *S(Tj)* stosowane są w przypadkach, gdy alternatywy wyboru charakteryzują się zbliżonymi wartościami atrybutów oraz gdy porównywanie dokonywane jest pomiędzy wartościami leżącymi w zgodnych (tych samych) skalach.

Stosują jedną z podanych metod można ocenić użyteczność i funkcjonalność ϕ hurtowni danych w chwili T_j. określając dla każdej miary $M_i(Tj)$ jej wagę ω_i oraz ocenę $\gamma(M(T_i))$ a także określając dla każdej miary $M''_p(Tj)$ jej wagę α_p oraz ocenę $\tau(M''_p(T_j))$. Jest ona zgodna ze wzorem (5).

$$\phi(S(T_j)) = \frac{\sum_{i=1}^{n} \omega_i * \gamma(M_i(T_j)) + \sum_{p=1}^{m} \alpha_p * \tau(M''_p(T_j))}{\sum_{p=1}^{m} \alpha_p + \sum_{i=1}^{n} \omega_i} \quad (6)$$

Iteracyjnie realizowana procedura oceny użyteczności i funkcjonalności hurtowni danych pozwala określić te oceny w kolejnych chwilach osi czasu

$S(T_0), S(T_1), \ldots, S(T_j)\ldots$ (7)

Do porównania stanów ze wzoru (4) oraz (7) można również zastosować metodę oceny wariantów (tu każda ocena $S(T_j)$ stanowi wariant) z analizą zgodności i niezgodności tych wariantów.

Wówczas funkcję zgodności σ ocen użyteczności i funkcjonalności hurtowni można zdefiniować wzorem (8)

$$\sigma(S(T_r), S(T_k)) = 1 \text{ gdy } \phi(S(T_r)) - \phi(S(T_k)) >= 0 \quad (8)$$

$$\sigma(S(T_r), S(T_k)) = 0 \text{ gdy } \phi(S(T_r)) - \phi(S(T_k)) < 0$$

gdzie:
r,k- indeksy (znaczniki) kolejnych chwil na osi czasu takie, że $r,k \in \{0,1,2,...\}$,
T_r, T_k – chwile osi czasu $(Tr > T_0 \text{ oraz } T_k > T_0)$,
T_o – chwila, w której rozpoczęto eksploatację hurtowni danych,
ϕ- syntetyczny wskaźnik oceny stan użyteczności i funkcjonalności hurtowni.

4. Podsumowanie

W artykule zaprezentowano możliwość oceny użyteczności i funkcjonalności hurtowni danych zwłaszcza na etapie jej eksploatacji. Podstawą zaprezentowanej metody są koncepcje DWQ oraz AMDH, gromadzenie na bieżąco w kolejnych chwilach osi czasu w metadanych tej hurtowni oceny jej użyteczności i funkcjonalności a także mechanizmy wielowersyjnego modelu danych.

Dzięki tej metodzie możliwa będzie adaptacja hurtowni do nowej jej wersji na podstawie przeprowadzonych w kolejnych chwilach ocen użyteczności i funkcjonalności hurtowni danych i dostosowanie do nowych potrzeb informacyjnych jej użytkowników poprzez utworzenie kolejnej wersji tego systemu. Pozwoli to dodatkowo odpowiedzieć na następujące pytania

- Czy w kolejnych chwilach osi czasu użyteczność hurtowni maleje (funkcja zgodności $\sigma()=0$) czy rośnie (funkcja zgodności $\sigma()=1$) ?
- Czy jest od pewnego czasu satysfakcjonująca czy nie a także, w jakim przedziale czasu ΔT funkcja zgodności jest malejąca ($\sigma()=0$))?
- Czy należy rozpocząć proces restrukturyzacji i przebudowy hurtowni danych ze względu na próg spadku użyteczności i funkcjonalności hurtowni?.
- Jakie czynniki (miary) decydują o jej użyteczności i funkcjonalności?
- Jaki jest ich wpływ czynników (miar) na użyteczność i funkcjonalność hurtowni (badanie wrażliwości spadku użyteczności i funkcjonalności hurtowni)?.

Zaprezentowana metoda jest specyficzna dla hurtowni danych. Nie może być zastosowana w innych systemach informatycznych, zwłaszcza tych, które nie są systemami uwarunkowanymi czasem. Jest to metoda o dużym potencjale zastosowań, bowiem hurtownia danych wspomaga procesy podejmowania decyzji zwłaszcza taktycznych i strategicznych a organizacje ponoszą duże koszty związane z budową hurtowni danych, która powinna uwzględniać zmienne potrzeby jej użytkowników.

Literatura

[1] Inmon W. H., Building the Data Warehouse, Second Edition, Wiley & Sons, New York, 1996.
[2] Jarke M., Lenzerini M., Vassiliou Y., Vassiliadis P., Hurtownie danych. Podstawy organizacji i funkcjonowania. Wydawnictwo szkolne i pedagogiczne, Warszawa, 2003.
[3] Morzy T., Wrembel R., Managing and Querying Versions of Multi-version Data Warehouse. In Proceeding of International Conference on Extending Database Technology, EDBT, 2006.
[4] Roy B., Multicriteria Methodology for Decision Aiding, volume 12 of Nonconvex Optimization and its Applications. Kluwer Academic Publishers, Dordrecht, 1996.
[5] Smiałkowska B., Metoda dopasowania hurtowni danych do zmiennych potrzeb informacyjnych przedsiębiorstwa. Wydawnictwo Zachodniopomorskiego Uniwersytetu Technologicznego w Szczecinie, Szczecin, 2009.
[6] Śmiałkowska B., Metoda projektowania hurtowni danych dla potrzeb adaptacyjnego wspomagania zarządzania strategią firmy. Wydawnictwo Katedry Informatyki w Zarządzaniu, Akademii Rolniczo-Technicznej, Bydgoszcz, 2003.
[7] Wrembel R., Management of schema and data evaluation in multiversion data warehouse. Wydawnictwo Politechniki Poznańskiej, Seria:Rozprawy, nr 411, Poznań, 2007.

III. Praktyczne zastosowanie technologii dla oprogramowania

Rozdział 8

Standardy architektury modelu systemu B2B wspomagającego zarządzanie procesami budowlanymi

1. Wprowadzenie

Złożoność procesów budowlanych, które przedsiębiorstwa budowlane realizują we współpracy z wieloma uczestnikami oraz wysokie wymagania stawiane przez inwestorów spowodowały, że istnieje rynkowe zapotrzebowanie na zaawansowane systemy B2B, które są w stanie wspomóc efektywną realizację inwestycji w kooperacji z partnerami, integrować działania wykonywane w siedzibie firmy z tymi na placu budowy. Firmy budowlane korzystają z różnych systemów informatycznych oraz aplikacji wyposażonych w różne funkcjonalności, zbudowane w oparciu o różne technologie informatyczne. Badania przeprowadzone przez OPTeam SA wykazały, że system B2B wspierający procesy budowlane powinien być zatem rozwiązaniem, które będzie działało w warunkach niezależnych od rodzaju systemu zarządzania przedsiębiorstwem klasy ERP oraz będzie łatwo integrować się ze specjalnymi aplikacjami, np. GPS. Ponadto, z badań OPTeam SA wynika, że w systemach informatycznych dla branży budowlanej należy zwrócić szczególną uwagę na kilka grup dostępnych rozwiązań:

- programy do kosztorysowania projektów, np. Pro Norma, Winbud, Zuzia;
- systemy do zarządzania firmą, w tym klasy ERP, np. Sap Business One, Comarch CDN XL, Comarch Optima, Enova, Simple ERP, MS Dynamics, Capital;
- klasyczne narzędzia do różnego rodzaju wyliczeń i raportowania (Excel) oraz zaawansowane narzędzia do zarządzania projektami (MS Project, Planista).

Ze względu na wysoką zmienność danych podczas realizacji procesów budowlanych firmy budowlane oczekują multifunkcjonalnego systemu, który umożliwiałby sprawną wymianę danych pomiędzy systemem ERP postrzeganym jako warstwa operacyjna i rozwiązaniami służącymi do projektowania, kosztorysowania, monitorowania i kontroli procesów budowlanych.

Biorąc pod uwagę powyższe, celem artykułu jest prezentacja wyników badań jednego z etapów zrealizowanych w ramach projektu badawczo – rozwojowego pn. „Prototyp innowacyjnej i zaawansowanej technologicznie platformy B2B OPTIbud, wspomagającej zarządzanie procesami budowlanymi, poprzez integrację danych i informacji z wielu źródeł", dotyczących standardów architektury modelu systemu B2B OPTIbud.

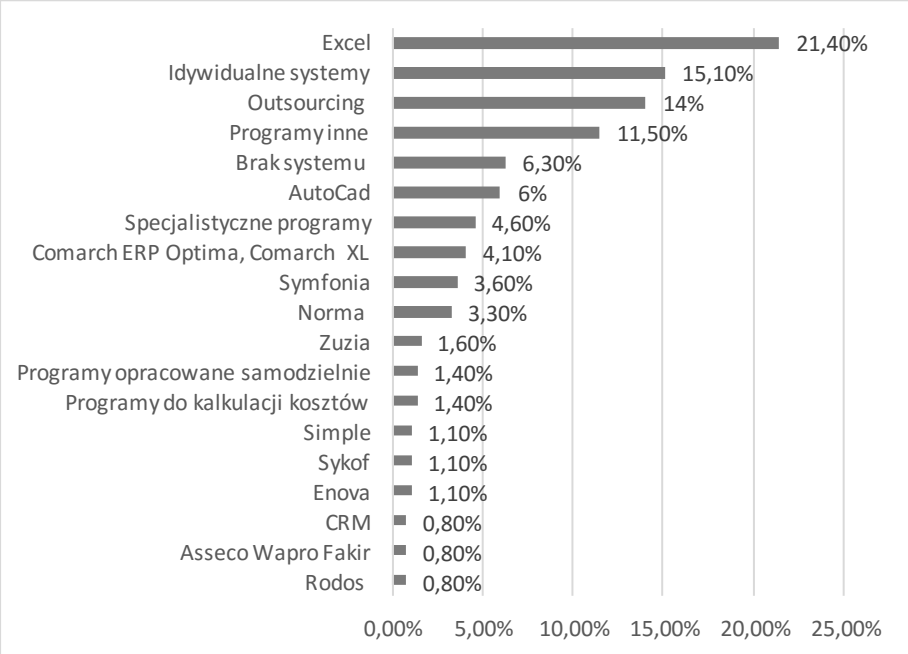
2. Metodologia badań

Badania dotyczące wyboru standardów architektonicznych, które będą zaimplementowane w systemie B2B OPTIbud zostały poprzedzone następującymi zadaniami badawczymi:

1. Badanie funkcjonalności systemów informatycznych dedykowanych firmom budowlanym, dostępnych na rynku krajowym w celu opracowania koncepcji architektury standardowych i specjalistycznych funkcjonalności systemu B2B OPTIbud.
2. Analiza możliwości integracji projektowanego systemu B2B OPTIbud z istniejącymi i najczęściej stosowanymi przez przedsiębiorstwa budowlane systemami klasy ERP.

W ramach pierwszego zadania badawczego zostały przeprowadzone dwie ankiety:

- ogólna - w celu zbadania jakie systemy i narzędzia informatyczne są najczęściej używane w polskich firmach budowlanych (Wykres 1). W badaniach wzięło udział 611 firm z terenu całego kraju. W sumie odpowiedzi pełnych lub częściowych udzieliło 405 firm budowlanych.
- szczegółowa – ankieta zawierała 60 pytań skierowanych do grupy 10 średniej wielkości firm budowlanych. Na postawie badań zdefiniowano moduły, które będą obsługiwały poszczególne działania i podprocesy w ramach procesu budowalnego. Należą do nich: „Administrator", „Przetargi i ofertowanie", „Budżetowanie i harmonogramowanie", „Panel Kierownika Budowy", „Baza sprzętu i transportu" z uwzględnieniem geolokalizacji, „System obiegu dokumentów", „Księga raportów", „Spedycja".



Rysunek 1. Systemy i narzędzia informatyczne wykorzystywane podczas realizacji budów przez firmy budowlane. Źródło: OPTeam S.A. [11]

Etap drugi pozwolił na określenie mechanizmów integracji systemu B2B OPTIbud ze znanymi systemami klasy ERP, na których pracują firmy branży budowlanej.

Realizacja badań w etapie trzecim pozwoliła na określenie standardów modelu systemu B2B OPTIbud. Analiza różnych modeli, dostępnych narzędzi i technologii informatycznych doprowadziła do przyjęcia następujących założeń dla systemu B2B OPTIbud:

- 3-warstwowa architektura systemu oparta na koncepcji SOA (Service Oriented Architecture),
- system modularny i niezależny od oprogramowania klasy ERP,
- system elastyczny, zapewniający łatwą rozbudowę funkcjonalności przez dedykowane rozszerzenia,
- przepływ informacji między modułami realizowany w trybie online,
- system będzie posiadał jedną bazę danych integrującą dane z różnych źródeł dzięki czemu użytkownik nie będzie wprowadzał wielokrotnie tych samych danych w różnych modułach funkcjonalnych,
- system mający jednolity interfejs użytkownika.

Ten etap prac badawczych pozwolił na równoległe prowadzenie badań nad poszczególnymi warstwami systemu B2B. Istotą przyjętych założeń jest uniezależnienie silnika systemu od wykorzystywanej bazy danych oraz łatwość rozszerzania funkcjonalności interfejsu użytkownika. W ramach badań określone zostały procedury zasilania systemu B2B OPTIbud z różnych źródeł danych (m.in. web service, zewnętrzne bazy danych, pliki zewnętrzne), określone zostały standardy przesyłania do systemu danych zewnętrznych zapisanych w plikach tekstowych, xml, csv oraz standardy pobierania z systemu danych o dokumentach, danych podwykonawców w najczęściej wykorzystywanych formatach. Istotnym warunkiem mającym duży wpływ na wybór sposobów realizacji tych założeń miały wyniki badań z dwóch pierwszych etapów projektu. Ponadto, w ramach prac badawczych nad warstwą użytkownika zostały zaprojektowane formaty elektronicznych dokumentów, formularzy wymiany danych i informacji, sposób przepływu informacji.

W rezultacie opracowano standardy architektury modelu systemu B2B wspierającego realizację procesów budowlanych, na którą składają się:

- warstwa bazy danych – określa metody aktualizacji bazy danych, formaty skryptów, metody wersjonowania oraz narzędzia do automatycznej aktualizacji bazy danych;
- warstwa usług – posiada standardy tworzenia serwisów, dzięki którym warstwa ta może zostać wydzielona na serwer aplikacji. Ponadto daje możliwość tworzenia dedykowanych serwisów dla różnych systemów ERP;
- warstwa klienta – opracowany został dokument standaryzacji interfejsu użytkownika.

Dodatkowo wskazane zostały technologie pozwalające na łatwe tworzenie w przyszłości rozszerzeń systemu i jego konfiguracji. System B2B OPTIbud został oparty na koncepcji otwartej architektury systemu, pozwalającej na integrację z innymi systemami na poziomie procesów biznesowych.

3. Specyfika procesów budowlanych

Na specyfikę procesów budowlanych mają wpływ takie czynniki jak: wymagania prawne oparte między innymi na prawie budowlanym, prawie zamówień publicznych,

dokumentacja projektowa, wymagania BHP, czynnik ludzki, oddziaływanie otoczenia na przebieg prac budowlanych, zaangażowanie podwykonawców, zróżnicowanie geograficzne pomiędzy siedzibą firmy a placem budowy, zaangażowanie różnego rodzaju sprzętu budowlanego i środków transportu, materiałów budowlanych, system dostaw, czynniki finansowe. W rezultacie procesy budowlane charakteryzują się wysokim stopniem złożoności organizacyjnej i wykonawczej, koniecznością harmonogramowania robót, budżetowania, nadzoru nad podwykonawcami, prowadzenia dużej ilości dokumentacji budowlanej, nadzoru nad jakością realizacji prac i dostaw. Tempo prac budowlanych uzależnione jest od współpracy z podwykonawcami, dostawcami, inwestorami, warunkami atmosferycznymi, decyzjami organów państwowych. Ponadto, procesy budowlane są narażone na wysokie ryzyko zakłóceń wynikających z siły oddziaływania wskazanych wyżej czynników.

W celu optymalizacji realizacji procesów budowlanych i poprawy efektywności zarządzania nimi pomocne są rozwiązania informatyczne, których funkcjonalność odzwierciedla poszczególne etapy ich wykonania, zapewnia płynną komunikację pomiędzy siedzibą firmy a miejscem budowy oraz podmiotami współpracującymi. Badania przeprowadzone przez OPTeam SA oraz doświadczenie zdobyte poprzez obsługę informatyczną firm budowlanych prowadzą do wniosku, że system B2B powinien być wyposażony w moduły, których zsynchronizowane działania odzwierciedlają specyfikę procesów budowlanych, których etapy realizowane są zarówno wewnątrz jak i na zewnętrz przedsiębiorstwa (Tabela 1).

Tabela 1. Koncepcja funkcjonalności modułów systemu B2B OPTIbud. Źródło: OPTeam S.A. [11]

Nazwa modułu	Charakterystyka funkcjonalności
PRZETARGI I OFERTOWANIE	Ewidencja przetargów oraz etapów przygotowania oferty przetargowej; automatyczne zaczytywanie przetargów opublikowanych na portalach zamówień publicznych z funkcjonalnością filtrowania danych; etapowanie, dzięki czemu możliwe będzie na bieżąco śledzenie postępów prac; wprowadzanie oraz analiza protokołów z otwarcia ofert, gromadzenie informacji o wygranych/przegranych przetargach oraz wykonawcach; baza wiedzy na temat zrealizowanych projektów oraz doświadczenie pracowników. Moduł powiązany będzie z pozostałymi obszarami systemu dzięki czemu możliwe będzie założenie karty budowy z wygranego przetargu.
BUDŻETOWANIE I HARMONOGRAMOWANIE	Tworzenie budżetów dla prowadzonych projektów w oparciu dane z wielu źródeł; przekształcanie budżetów do harmonogramów oraz odzwierciedlenie na osi czasu realizacji projektu. Do modułu wbudowane zostaną mechanizmy grupowania, scalania, rozbijania pozycji budżetowych oraz możliwość tworzenia różnego rodzaju budżetów: uproszczonych lub pełnych; ewidencja prac podwykonawców oraz kopiowanie pozycji z innych budżetów; śledzenie stopnia realizacji projektu od strony kosztowej jak również czasowej.
PANEL KIEROWNIKA PROJEKTU	Dla kierowników projektów gromadzenie informacji dotyczących prowadzonych projektów; nadzór nad dokumentacją projektową oraz możliwość przypisywania kosztów do pozycji w budżecie; możliwość założenia karty budowy, w której zgromadzone zostaną wszystkie najważniejsze informacje dotyczące projektu oraz postawienia z umowy z inwestorem; komunikacja oraz rozliczanie z podwykonawcami;

	ewidencja czasu pracy podległych pracowników w rozbiciu na projekty; zgłaszanie ofert, zapotrzebowania, zamówień materiałowych oraz na usługi (w tym sprzęt i transport). W ramach tego modułu zostanie przygotowany webowy interfejs umożliwiający przeniesienie najważniejszych funkcji do przeglądarki internetowej co podniesie mobilność rozwiązania (obsługa za pomocą smartfonów).
BAZA SPRZĘTU I TRANSPORTU	Ewidencja i rejestracja procesu wynajmu/wypożyczenia sprzętu lekkiego, ciężkiego, transportu oraz elektronarzędzi wraz z ewidencją kosztów; gromadzenie informacji o osobach odpowiedzialnych, przeglądach i ubezpieczeniach, historia sprzętu i napraw; rozliczanie sprzętu oraz transportu w stosunku do budżetu oraz w oparciu o stawki (np. tonokilometry); wyliczenie podatku środowiskowego na podstawie zużycia materiałów pędnych (ewidencja tankowań); magazynu eksploatacyjny umożliwiający zarządzanie materiałami zużywanymi do obsługi bazy sprzętowo-transportowej; śledzenie pracy sprzętu w kontekście realizowanych projektów (np. na jakiej budowie znajduje się dany sprzęt, lub jaki sprzęt jest na wybranej budowie). Moduł ten podobnie jak w SPEDYCJI – wykorzystywał będzie technologię RFID oraz GPS do śledzenia w czasie rzeczywistym pracy sprzętu ewidencjonowanego.
SPEDYCJA	Ewidencja zleceń (spedycyjnych, transportowych, dostawy, przeładunku) oraz organizowania przewozu towarów; tworzenie tras oraz generowanie zleceń z wszystkimi najważniejszymi informacjami (typu: data wystawienia, załadunku, status, zleceniodawca, ładunek, ilość, trasa, miejsce załadunku i wyładunku). Moduł wyposażony będzie w technologię RFID oraz system GPS, dzięki czemu na bieżąco, w czasie rzeczywistym możliwa będzie kontrola lokalizacji sprzętu oraz jego pracy (trasa, rozbieżności, postój, praca silnika, tankowania)
OBIEG DOKUMENTÓW	Rejestracja pism i całej dokumentacji przychodzącej i wychodzącej; archiwizacja dokumentów; zapotrzebowania, oferty oraz zamówienia materiałowe (obsługa ścieżki akceptacji obiegu dokumentu); obieg faktur kosztowych (opisywanie dokumentów odpowiednimi projektami; przygotowanie przetargów; pozyskanie zleceń/projektów (rejestracja spraw, zadań, teczek; przydział zadań do realizacji); kontrola terminowości spraw/zleceń/zadań; modelowanie procesów biznesowych.
KSIĘGA RAPORTÓW	Zgromadzenie w jednym miejscu wszystkich dostępnych w całym systemie statycznych raportów potrzebnych do prowadzenia bieżącej działalności; budowanie raportów i analiz na bazie informacji wprowadzonych do systemu; analizy podzielone na obszary/moduły; graficzna prezentacja danych.
ADMINISTRATOR	Administrowanie systemem z poziomu aplikacji desktopowej (parametryzacja systemu, interfejsu, nadawanie uprawnień, itp.); nadawanie uprawnień oraz podpinanie akcji do zdarzeń, czyli np. nowych wydruków, uruchamianie określonych zdarzeń w procesie, czy też tworzenie alertów w systemie.

4. Architektura warstwowa w projektowaniu systemów informatycznych B2B

Z perspektywy logicznej architektury aplikacji każdy system informatyczny jest traktowany jako zbiór współpracujących warstw, z których każda prezentuje określony charakter usług [9]. Podczas implementacji każdej warstwy, segmentacja umożliwia wybór konkretnych składników architektury i składników projektowych oraz stworzenie aplikacji łatwiejszej do jej obsługi i serwisowania podczas użytkowania. Istnieje kilka koncepcji podziału logicznego systemu na warstwy, przy czym podstawową jest architektura zorientowana na usługi (SOA), która bazuje na założeniu, że logika biznesowa nie stanowi monolitycznego programu, lecz składa się z wielu rozproszonych komponentów usługowych, koordynowanych przez centralną aplikację zarządczą ([7], s.1-9). Według Garthner'a SOA jest architekturą oprogramowania bazującą na definicji interfejsów [3].Tworzy ona topologie aplikacji na podstawie usług, ich implementacji oraz wywołań. Architektura SOA [2] kategoryzuje zależności pomiędzy dostawcami usług, a ich odbiorcami reprezentowanymi przez komponenty oprogramowania realizujące złożone procesy biznesowe, do których również należą procesy budowlane. Zapewnia ponowne użycie składników oprogramowania, enkapsulację funkcjonalności, precyzyjną definicję interfejsów oraz elastyczność aplikacji tworzonych na drodze kompozycji. Komponenty SOA są luźno powiązane i współpracują ze sobą realizując proces biznesowy [8]. W SOA, jako wzorcu projektowania heterogenicznych systemów rozproszonych zorientowanych biznesowo takich jak B2B, rozróżnia się dwa modele rozwiązań:

- Web Services - usługi sieciowe, które są opublikowane w rejestrze usług. Można je wywołać zdalnie przez zdefiniowany interfejs. Wywoływanie udostępnionych usług może być koordynowane za pomocą mechanizmu kompozycji usług, który zarządza wykonywaniem procesów biznesowych opartych na usługach z różnych systemów, a całość jest udostępniona jako nowa usługa. Budowa nowych, złożonych usług sieciowych odbywa się za pomocą orkiestracji rozumianej jako proces przepływu informacji między usługami w ramach pojedynczego procesu biznesowego [1], oraz choreografii, która umożliwia wyeksponowanie fragmentu procesu biznesowego opisującego interakcje z daną usługą na zewnątrz organizacji [4];
- Representational State Transfer (REST) jest technologią tworzenia protokołu i umożliwia transmisje danych przy użyciu cech protokołu HTTP. Nie stosuje on dodatkowej warstwy do przesyłania wiadomości, takiej jak na przykład SOAP.

Badania przeprowadzone przez OPTeam wykazują, że optymalnym rozwiązaniem będzie oparcie architektury systemu B2B OPTIbud na koncepcji SOA ponieważ ma on współpracować z różnymi systemami do zarządzania przedsiębiorstwem oraz specjalistycznymi narzędziami projektowania w budownictwie.

5. Środowiska programowania systemów B2B

Jak wcześniej zauważono, współczesne systemy oprogramowania odpowiadają strukturze wielowarstwowej złożonej z co najmniej trzech głównych warstw: prezentacji, logiki biznesowej, źródeł danych. Każda z tych warstw wykształciła odpowiednie do pełnionej funkcji środowisko systemowe. W odniesieniu do systemów B2B, które mają

charakter rozproszony, technologie, które należy wziąć pod uwagę ponieważ ułatwiają tworzenie aplikacji w takim środowisku programistycznym, to:

- CORBA (Common Object Request Broker Architecture) - standard technologiczny, który umożliwia współpracę między aplikacjami pracującymi na różnych systemach operacyjnych, napisanych w różnych językach programowania oraz niezależnych od platformy sprzętowej. Istotnym elementem CORBA jest Object Request Broker (ORB). Odpowiada on za komunikację między serwerami, która polega na tym, że klient wysyła zapytanie do „brokera", a ten szuka najbardziej odpowiedni serwer realizujący usługi i oczekiwania serwera klienta ([5], s. 1155-1156). Architektura CORBA zawiera trzy główne elementy: serwer aplikacji, kontener, komponenty i klienta. CORBA jest implementacją warstwy pośredniej (logiki biznesowej) w trójwarstwowej architekturze systemu informatycznego [13];
- EJB (Enterprise JavaBeans, Platforma J2EE) - jest platformą programistyczną języka Java, która zapewnia środowisko API oraz środowisko uruchomieniowe dla działania oprogramowania, między innymi usług sieciowych, wielowarstwowych, skalowalnych i wydajnych aplikacji. EJB definiuje standard tworzenia aplikacji szczególny nacisk kładąc na architekturę komponentową. Obsługą osadzonych komponentów zajmuje się odpowiedni serwer aplikacyjny, w którym działają aplikacje stworzone w JEE (Java Enterprise Edition). Charakteryzuje się logicznym podziałem na warstwy, z których każda odpowiada za określony zakres funkcjonalności i może się także składać z wielu komponentów;
- .NET (dotNet)- jest programistyczną technologią przeznaczoną do tworzenia aplikacji działających na platformie Windows. Framework .NET dostarcza środowisko uruchomieniowe oraz biblioteki klas z narzędziami i funkcjami dla programistów, umożliwia obsługę takich języków jak: Visual Basic .NET, C#, C++, J#. Technologia .NET z racji swej popularności wykorzystywana jest do budowy wielu aplikacji. Architektura aplikacji wielowarstwowych w technologii .NET bazuje na modelu DNA, zdefiniowanym wraz z pojawieniem się modelu COM+ oraz Windows 2000. Architektura DNA to klasyczny trójwarstwowy model budowy oprogramowania. W zakresie organizacji dostępu do baz danych technologia .NET wykorzystuje ADO.NET.

6. Modele zasilania systemu B2B z różnych źródeł danych

Badania przeprowadzone w ramach projektu wykazują, że zasilanie systemu B2B wiąże się z takimi zagadnieniami jak: język opisu danych przesyłanych do systemu, elektroniczna wymiana danych (EDI), web service, pliki zewnętrzne.

SGML (Standard Generalized Markup Language) jest standardem uogólniającym język znaczników dokumentu. Specyfikacja SGML umożliwia zapisanie danych w formie dokumentu tekstowego, który jest uniwersalny do późniejszych zastosowań (drukowanie, przesyłanie, odczyt). Opiera się na założeniu, że dokumenty mają strukturalną budowę możliwą do wyodrębnienia i opisania za pomocą zestawu znaczników. SGML nie definiuje zbioru znaczników (jak np. HTML), tylko opisuje i standaryzuje sposób ich definiowania.

XML (Extensible Markup Language) jest prostym językiem znaczników, za pomocą którego możliwe jest reprezentowanie danych w sposób elastyczny, a z drugiej

strony ustrukturalizowany. Jako język niezależny od platformy umożliwia wymianę danych między różnymi platformami i systemami.W obecnej chwili najbardziej znaną pochodną języka XML jest hipertekstowy język HTML (HyperText Markup Language) oraz jego następca XHTML (Extensible HyperText Markup Language). Podobnie jak w języku SGML struktura danych w XML jest niezależna od formatowania, natomiast w HTML formatowanie jest wbudowane w polecenia języka.

OpenDDL (Open Data Description Language) jest tekstowym językiem przeznaczonym do przechowywania dowolnych danych w zwięzłej i czytelnej formie. Może być stosowany jako sposób wymiany danych między programami lub jako sposób przechowywania danych w zdefiniowanym formacie. Od innych tego typu języków wyróżnia się tym, że każda jednostka danych w pliku OpenDDL ma określony typ. Eliminuje to możliwości pomyłek lub złej interpretacji, co wpływa na integralność i poprawność danych. Struktura danych w pliku OpenDDL jest zorganizowana jako zbiór drzewek. Język posiada wbudowany mechanizm do tworzenia referencji z jednej struktury danych do innej co pozwala takiemu plikowi przyjąć postać kierunkowego grafu.

Technologia EDI (Electronic Data Interchange) zakłada integrację wewnętrznego oprogramowania firmy z systemem wymiany na poziomie wymiany plików. W przypadku dokumentów przychodzących elektronicznie łącznik komunikacji przekazuje dokument do translatora EDI, który transformuje format EDI dokumentu do postaci firmowych aplikacji informatycznych [14]. Interfejs aplikacji akceptuje wejście dokumentu z translatora EDI i udostępnia mu właściwą aplikację firmy. Po weryfikacji kompletności i poprawności formatu danych zawartych w dokumencie zasila się nimi odpowiedni system informatyczny przedsiębiorstwa. W przypadku wychodzących dokumentów proces przebiega w odwrotnej kolejności. Kluczową funkcją EDI jest transmisja danych pomiędzy współpracującymi partnerami, kooperantami czy osobami we właściwym, ustalonym formacie przez protokół transmisji danych.

Web Service to technologia projektowania rozproszonych elementów usługowych, służących do implementacji aplikacji biznesowych w architekturze zorientowanej na usługi. Usługi sieciowe są dostępnymi poprzez sieć komponentami przeznaczonym do wykorzystania przez inne aplikacje. Technologia usług sieciowych bazuje na zestawie skorelowanych rozwiązań informatycznych, spośród których najważniejsze to: protokół komunikacyjny SOAP – służący do przekazywania zdalnych wywołań, język opisu interfejsu usługi WSDL (Web Services Description Language) – służący do dystrybucji parametrów połączeń sieciowych, specyfikacja bazy danych UDDI (Universal Description, Discovery and Integration)– służąca do rejestracji udostępnianych komponentów usługowych [10]. Siłą Web Service jest wykorzystanie rozpowszechnionych rozwiązań: protokołu HTTP i języka XML HTTP. XML dostarcza metajęzyk za pomocą, którego porozumiewają się klienci z usługami oraz poszczególne komponenty.

Zastosowanie odseparowania źródła danych od interfejsu użytkownika w postaci warstwy Web Service powoduje, że można zasilić system B2B dowolnymi danymi zewnętrznymi o ustalonym formacie. Web Service zapewnia dużą uniwersalność użycia, nie tylko w przypadku systemu B2B. Pozwala na integrację systemu z zewnętrznymi systemami. Mogą to być aplikacje zarówno www jak i programy desktopowe, ale również coraz bardziej powszechne programy mobilne. Wykorzystanie technologii Web Service w rozwiązaniu typu B2B, w tym dla B2B OPTIbud wydaje się bardzo korzystne ponieważ zapewnia dużą elastyczność przy późniejszym rozwoju systemu.

7. Podsumowanie

Projektowany system typu B2B OPTIbud powinien spełniać założenia dedykowane do specyfiki pracy nowoczesnego systemu B2B w środowisku rozproszonym. Ze względu na przyjętą w OPTeam S.A. technologię programowania na platformie .NET sugeruje się najnowsze w tym zakresie rozwiązania firmy Microsoft. Zapewnią one wsparcie dla wzorców projektowych, współpracują z przyjętymi standardami przekazywania danych. Rekomenduje się wydzielenie 3 warstw głównych: prezentacji, logiki biznesowej, danych. Z kolei, zastosowanie Web Service zapewni możliwość pobierania danych poprzez różne źródła (system ERP, mobile, www).

Wydzielenie warstw otwiera system na współpracę, umożliwia publikowanie na zewnątrz usług oraz wymianę informacji. Zapewnia również odpowiednie rozwiązania architektoniczne i wydajnościowe, które mają znaczenie w realizacji procesów budowlanych. W szczególności jest to istotne na poziomie warstwy dostępu do danych, która powinna zapewnić niezbędną szybkość reakcji na zapytania od warstwy logiki. Należy pamiętać, że firmy używają wielu innych rozwiązań, które obciążają bazę danych, a więc pośrednio również system B2B. Zdiagnozowanie „wąskich gardeł" systemu informatycznego i możliwości wsparcia ich wydajniejszym sprzętem przemawia za rozwiązaniem wielowarstwowym. Powinny w tym pomóc testy wydajnościowe przeprowadzone w dalszej części projektu.

Badania przeprowadzone w ramach projektu implikują istotnym spostrzeżeniem, że największą trudnością w tworzeniu systemu B2B dedykowanemu branży budowlanej jest prawidłowe zdefiniowanie jego funkcjonalności, zaś technologie i narzędzia informatyczne są tylko środkiem do ich zaprojektowania. Są one wystandaryzowane, natomiast procesy budowalne i zarządzanie nimi wymagają zdefiniowania w kontekście zaspokojenia potrzeb wielu interesariuszy, z których kluczowymi są wykonawca (firma budowlana) oraz inwestor

Literatura

[1] Bluemke I., Kiermasz W., Kompozycja i integracja usług w architekturze SOA, [w:] J. Górski, C. Orłowski (red.), Inżynieria oprogramowania w procesach integracji systemów informatycznych, PWNT Gdańsk, 2011.
[2] Erl T., What is SOA: an Introduction to Service Oriented Computing, http://www.whatissoa.com, SOA System, 2012.
[3] Erl T., Gee C., Kress J., Maier B., Normann H., Raj P., Shuster L., Trops B., Utschig C, Wik P., Winterberg T., Next Generation SOA, A Concise Introduction to Service Technology & Service-Orientation, Prentice Hall/Pearson PTR, 2012.
[4] Fronckowiak J., SOABestPractices andDesignPatterns Keys to Successful Service-Oriented Architecture Implementation, White Paper Published by Oracle Corp, 2009.
[5] Giachetti R. E., A framework to review the information integration of the enterprise, International Journal of Production Research, vol. 42, no. 6, 2004.
[6] Haas H., Brown A., Web Services Glossary: http://www.w3.org/ TR/2004/NOTE-ws-gloss-20040211/, 2015.
[7] Jorgensen R., Philpott I., Architectural abstractions, in INCOSE Symposium Proceedings, 2002.
[8] Łagowski J., SOA – ideologia nie technologia, XV Konferencja PLOUG, Kościelisko, 2009.
[9] Meier J.D., Mackman A., Dunner M., Vasireddy S., Bulding Secure Microsoft ASP.NET Applications, 2002.
[10] Newcomer E., Understanding Web Services- XML, WSDL, SOAP and UDDI, Finding Web Services : UDDI Registry, chapter 5,Addison Wesley Professional, 2004.

[11] OPTeam SA, Raport 1 z badań „Prototyp innowacyjnej i zaawansowanej technologicznie platformy B2B OPTIbud, wspomagającej zarządzanie procesami budowlanymi, poprzez integrację danych i informacji z wielu źródeł", 2014.
[12] Oracle, http://www.oracle.com/technetwork/java/dataaccessobject-138824.html, 2015.
[13] Orfali R., Harkey R., Edwards R., The Essential Distributed Objects Survival Guide, John Willey, 1996.
[14] Pfeiffer, H. K. C., The diffusion of Electronic Data Interchange, Heidelberg: Physica-Verlag, 1992.
[15] SOA Definition, Service-oriented architecture (SOA) definition, http://www.servicearchitecture.com/web-services/articles/serviceoriented_architecture _soa_ definition.html, Barry & Associates, 2012.

Rozdział 9

Praktyczne wykorzystanie parametryzowanych rozmytych sieci Petriego

1. Wprowadzenie

Istnieje bardzo wiele typów sieci Petriego, dla których ugruntowały się dziedziny zastosowań [4]. Stały się one nawet fundamentem dla graficznego języka programowania sterowników przemysłowych [1]. Wśród sieci Petriego ważne miejsce zajmuje ich wersja rozmyta, pozwalająca na wyrażenie w formie sieci reguł eksperta, opisanych językiem naturalnym. Prace nad kolejnymi typami sieci trwają nadal. Pojawienie się kolejnej wersji sieci Petriego pobudza do pytania, czy prace nad tą siecią mają wymiar nauk podstawowych, czy też może jest dla nich miejsce w zastosowaniach praktycznych. Sensem prac przedstawionych w niniejszym rozdziale jest właśnie próba zbadania, czy nowy typ sieci Petriego, a mianowicie parametryzowane rozmyte sieci Petriego mogą mieć zastosowanie praktyczne, a jeśli tak, to czy ich stosowanie daje jakieś korzyści w stosunku do ugruntowanych już rozwiązań.

2. Parametryzowana rozmyta sieć Petriego

Parametryzowane rozmyte sieci Petriego są rozwinięciem uogólnionych rozmytych sieci Petriego. Upraszczając problem można powiedzieć, że zasadnicze rozwinięcie polega na wprowadzeniu w miejsce s- i t-norm odpowiednich norm parametryzowanych. Normy parametryzowane obejmują pewną rodzinę norm nieparametryzowanych. Na przykład parametryzowana s-norma Hamachera obejmuje m.in. następujące nieparametryzowane s-normy:

$S_D(a,b) = max(a, b)$, $S_L(a,b) = \min(a + b, 1)$ oraz $S_P(a,b) = a + b - a * b$.

W efekcie zmieniając wartość parametru v danej normy parametryzowanej można uzyskać efekt, jak przy stosowaniu kolejnych norm nieparametryzowanych, należących do danej rodziny [5,6,7].

2.1. Definicja sieci

Parametryzowana rozmyta sieć Petriego jest krotką:

$N_P = (P, T, S, I, O, \alpha, \beta, \gamma, Op, \delta, M_0)$

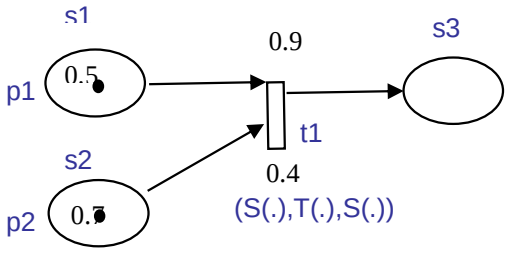
gdzie

- P, T, S – zbiory parami rozłączne *i* $card(P) = card(S)$
- $P = \{p_1, p_2, \ldots, p_n\}$ – skończony zbiór miejsc, $n > 0$
- $T = \{t_1, t_2, \ldots, t_m\}$ – skończony zbiór tranzycji, $m > 0$
- $S = \{s_1, s_2, \ldots, s_n\}$ – skończony zbiór wyrażeń
- $I: T \rightarrow 2^P$ – funkcja wejściowa
- $O: T \rightarrow 2^P$ – funkcja wyjściowa
- $\alpha: P \rightarrow S$ – funkcja wiązania wyrażeń
- $\beta: T \rightarrow [0,1]$ – funkcja stopnia prawdziwości
- $\gamma: T \rightarrow [0,1]$ – funkcja progowa
- Op – skończony zbiór parametryzowanych rodzin sum i iloczynów (w których występuje parametr v)
- $\delta: T \rightarrow Op^3$ – funkcja mapująca tranzycje w Op^3, np. zbiór wszystkich trójek operatorów (In, Out_1, Out_2)
- $M_0: P \rightarrow [0,1]$ – znakowanie początkowe

Operator *In* może być parametryzowaną s- lub t-normą, Out_1 musi być parametryzowaną t-normą, a Out_2 jest parametryzowaną s-normą.

Zgodnie z przytoczoną definicją graf pokazany na rys. 1 przedstawia elementarną parametryzowaną rozmytą sieć Petriego, gdzie:

- $P = \{p_1, p_2, p_3\}$
- $T = \{t_1\}$
- $S = \{s_1, s_2, s_3\}$
- $I(t_1) = \{p_1, p_2\}$
- $O(t_1) = \{p_3\}$
- $\alpha(p_1) = s_1$, $\alpha(p_2) = s_2$, $\alpha(p_3) = s_3$,
- $\beta(t_1) = 0{,}9$
- $\gamma(t_1) = 0{,}4$
- Op = {S(.), T(.)}
- $\delta(t_1)$ = {S(.),T(.),S(.)}
- M_0 = (0,5, 0,7, 0)



Rysunek 1. Graf przykładowej, elementarnej parametryzowanej rozmytej sieci Petriego

2.2. Parametryzowane normy

Z praktycznego punktu widzenia szczególnie interesująca jest możliwość *płynnego* wpływania na sposób wnioskowania sieci, poprzez zmianę wartości parametru *v*. Jak widać z definicji sieci w rozdz. 2.1 oraz dalej w tab. 1, parametryzowana jest funkcja wejściowa, funkcja wiążąca stopień pewności reguły z wnioskiem generowanym w wyniku odpalenia tranzycji oraz parametryzowana jest funkcja agregacji, określająca sposób łączenia w jednym miejscu wniosków płynących z różnych tranzycji. Ponadto mamy wpływ na wybór rodziny s-norm i t-norm. Warto podkreślić, że definicja sieci nie wymaga, aby dla poszczególnych tranzycji były to te same rodziny norm. Niezbędne jest zatem wskazanie pewnych sugestii, co do wyboru rodziny norm oraz sposobu korygowania wartości parametrów tych norm.

W pracach [5,6] można znaleźć informację o wzajemnych zawieraniu się lub częściach wspólnych różnych norm. Zdefiniowano także relację częściowego porządku tych norm, także w odniesieniu do norm nieparametryzowanych. Na rys.2 podano poglądowo relację częściowego porządku dla parametryzowanych s-norm Yagera, Hamachera i Dombiego oraz t-norm Yagera, Hamachera i Dombiego.

Rysunek 2. Poglądowy widok wzajemnego położonia wartości parametryzowanych s- i t norm: Yagera, Hamachera i Dombiego

Precyzyjniej wspomniane uporządkowanie dla s-norm można opisać jak niżej:
parametryzowana s-norma Hamachera
$S_H^1 = S_P \leq S_H^v \leq S_D = S_H^\infty$
parametryzowana s-norma Yagera
$S_Y^0 = S_D \geq S_Y^1 = S_L \geq S_Y^v \geq S_M = S_Y^\infty$
parametryzowana s-norma Dombiego
$S_D^0 = S_D \geq S_D^v \geq S_M = S_D^\infty$
gdzie $S_D(a,b) = max(a, b)$, $S_L(a,b) = \min(a + b, 1)$ oraz $S_P(a,b) = a + b - a * b$.

Ze względu na najszerszy zakres rodzin norm lub ich wzajemne uzupełnianie w niniejszej pracy rozważano właśnie normy Yagera, Hamachera oraz Dombiego. Z punktu widzenia praktycznego zastosowania nie bez znaczenia jest złożoność formuł. Na ogół wybieramy formuły prostsze, kierując się np. łatwością ich implementacji, mniejszą złożonością obliczeniową, a zatem i mniejszym zapotrzebowaniem na moc procesora i.in. W tabeli 1 przytoczono wzory opisujące wymienione rodziny norm.

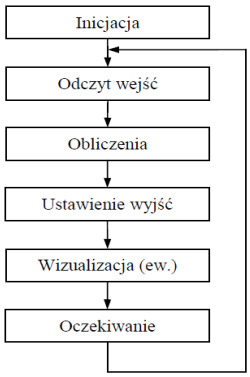
Tabela 1. Wzory opisujące parametryzowane s- i t-normy Hamachera – H, Yagera – Y i Dombiego – D wraz z podaniem dopuszczalnego (matematycznie) zakresu zmian wartości parametr v

$S_i(a,b,v)$	$T_i(a,b,v)$	i	Zakres
$\frac{a+b-(2-v)ab}{1-(1-v)ab}$	$\frac{ab}{v+(1-v)(a+b-ab)}$	H	$(0,\infty)$
$\min[1,(a^v+b^v)^{1/v}]$	$1\text{-}\min[1,((1-a)^v+(1-b)^v)^{1/v}]$	Y	$(0,\infty)$
$1-\frac{1}{1+\left[\left(\frac{1}{a}-1\right)^v+\left(\frac{1}{b}-1\right)^v\right]^{1/v}}$	$\frac{1}{1+\left[\left(\frac{1}{a}-1\right)^v+\left(\frac{1}{b}-1\right)^v\right]^{1/v}}$	D	$(0,\infty)$

3. Implementacja sieci w sterowniku PLC

Sterownik PLC jest programowalnym urządzeniem przeznaczonym do prowadzenia terowania i regulacji (o ile jest wyposażony w moduły ciągłe). Producenci sterowników oferują, na ogół, specjalizowane oprogramowanie narzędziowe do konfigurowania sprzętu sterownika, programowania algorytmów sterowania, prostej wizualizacji oraz symulacji pracy sterownika z działającym programem użytkownika (inżyniera).

Przebieg pracy sterownika z uruchomionym programem sterowania, przygotowanym i zaprogramowanym przez użytkownika (inżyniera) pokazano na rys. 3. Program użytkownika jest wykonywany w trakcie realizowania bloku obliczenia.



Rysunek 3. Schemat cyklu pracy sterowników PLC

3.1. Języki programowania PLC - Norma IEC 61131

Kwestie dotyczące funkcjonowania sterowników i ich programowania określa norma IEC 61131 (w Europie EN 61131), składająca się z 5 części [1,2]:

1. Postanowienia ogólne.
2. Wymagania i badania dotyczące sprzętu.
3. Języki programowania.

4. Wytyczne dla użytkownika.
5. Wymiana informacji.

W ramach części 3 normy zdefiniowano języki programowania sterowników PLC.

Języki tekstowe

IL (ang. *Instruction List – Lista rozkazów*) – odpowiednik języka asemblerowego;

ST (ang. *Structured Text – Tekst strukturalny*) – język strukturalny wysokiego poziomu;

Języki graficzne

LD (ang. *Ladder Diagram – Schemat drabinkowy*);

FBD (ang. *Function Block Diagram – Funkcjonalny schemat blokowy*);

SFC (ang. *Sequential Function Chart*), pozwala na opisywanie zadań sterowania sekwencyjne-go za pomocą grafów;

Wykorzystując język ST przygotowano m.in. funkcje realizujące s- t-normy zgodnie z podanymi wcześniej, w tab. 1, wzorami. Kodowanie konkretnej sieci polega zatem na wywoływaniu przygotowanych funkcji i budowaniu nowych funkcji reprezentujących tranzycje. W konsekwencji obliczenie wartości danego miejsca sieci jest wynikiem wywołania funkcji, związanej z odpowiednią tranzycją.

Współczesne sterowniki PLC realizują operacje wielowątkowe, gdzie wątek można rozumieć klasycznie jako elementarny byt, któremu jest przydzielany procesor. Programując sterownik operujemy tzw. jednostkami programowymi, czyli pojęciami jak niżej, gdzie tzw. Task odpowiada za przypisanie programu do wątku poprzez jego nazwę:

- Program – uruchamiany bezpośrednio przez Task lub przez inny Program. Może wywoływać funkcje, bloki funkcyjne lub inny „Program". Nie wymaga deklaracji.
- Blok Funkcyjny (ang. Function Block) – wywoływany bezpośrednio w Programie lub w innym bloku funkcyjnym. Może sam wywoływać funkcje lub inny blok funkcyjny. Może posiadać dowolną liczbę wejść i wyjść.
- Funkcja (ang. Function) – wywoływana z Programu lub Bloku Funkcyjnego. Z poziomu funkcji można wywołać tylko inne funkcje. Posiada jedno zwracane wyjście oraz dowolną liczbę wejść. Nie wymaga deklaracji.

Jak powiedziano wcześniej, sterowniki mogą oferować funkcjonalność wizualizacji (por. rys. 1). Wykorzystano ją w ramach opracowanych przykładów zastosowań sieci do prezentacji zachowania się sieci. Możliwość prowadzenia tzw. sterowania operatorskiego pozwoliła na ingerencję on-line w wartość parametrów sieci.

Przy pracach prezentowanych w niniejszym rozdziale wykorzystano oprogramowanie narzędziowe firmy Beckhoff. Warto zaznaczyć, że w ramach swojego pakietu firma udostępnia symulator sterownika, co pozwala na wstępną weryfikację dynamiczną oprogramowania na lokalnym komputerze.

3.2. Przykład ilustrujący implementację sieci w PLC

Do zilustrowania implementacji parametryzowanej rozmytej sieci Petriego w sterowniku PLC posłuży dość znany przykład sterowania ruchem pociągów. W przykładzie chodzi o podjęcie decyzji co do podstawienia dodatkowego pociągu lub wyrażenie zgody na opóźnienie pociągów w zależności od liczby przesiadających się pasażerów, wartości aktualnego opóźnienia i sytuacji na torach [7].

Przykład ten wybrano również dlatego, iż możliwa jest sytuacja braku rozstrzygnięcia, tzn. sugestia, aby pociąg zaczekał lub nie czekał może być

równoczesna (takie same wartości odpowiednich miejsc sieci). W takiej sytuacji modyfikacja wartości parametru *v* omawianego typu sieci, pozwala przesądzić o trafniejszej decyzji.

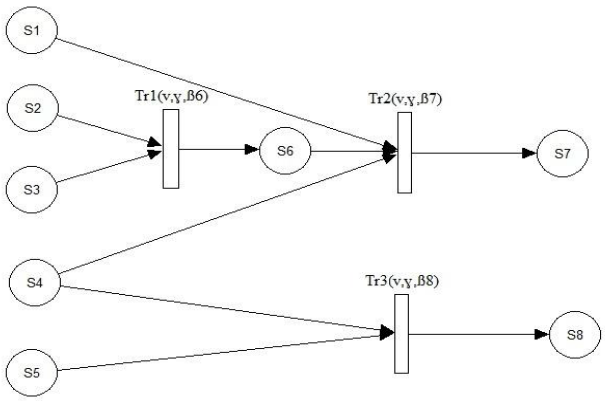
Opisywany problem rozwiązuje realizacja sieci o 8 miejscach i 3 tranzycjach wywiedzionych na bazie odpowiednich reguł. Znaczenie miejsc jest następujące:

- S1 – B jest ostatnim pociągiem w danym dniu;
- S2 – opóźnienie pociągu A jest duże;
- S3 – należy pilnie odprawić pociąg B;
- S4 – jest wielu przesiadających się z A do B;
- S5 – opóźnienie pociągu A jest małe;
- S6 – pociąg B odjeżdża wg rozkładu;
- S7 – podstawia się dodatkowy pociąg C;
- S8 – pociąg B czeka na pociąg A.

Wykorzystując wprowadzone symbole, odpowiednie reguły wnioskowania dla ruchu pociągów przyjmują postać:

- IF S2 OR S3 THEN S6;
- IF S1 AND S4 AND S6 THEN S7;
- IF S4 AND S5 THEN S8;

Na rysunku 4 pokazano graf parametryzowanej rozmytej sieci Petriego odpowiadającej podanym regułom wnioskowania. Przy graficznych symbolach tranzycji wprowadzono nazwy funkcji, zdefiniowanych w języku ST.



Rysunek 4. Sieć dla przykładu sterowania ruchem pociągów z opisem tranzycji funkcjami implementowanymi w j. ST w ramach programu sterującego sterownika PLC

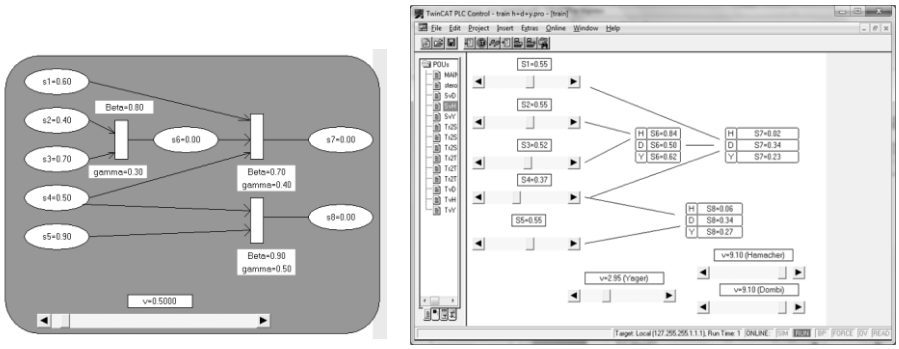
W zależności od aktualnie wykorzystywanych norm nazwy funkcji przypisanych do tranzycji na rys. 2 ulegną modyfikacji. Na przykład, w sytuacji użycia norm Hamachera kod programu w języku ST przyjmuje postać:

```
s6:=Tr2SvH(s2,s3,v,g,Bt);
s7:=Tr3TvH(s1,s6,s4,v,g3,Bt3);
s8:=Tr2TvH(s4,s5,v,g2,Bt2);
```

Jak już powiedziano, dla wygody analizy i parametryzacji programu wykorzystano możliwość prostej wizualizacji. Rysunek 5 przedstawia dwie realizacje wizualizacji: (a) dla przypadku użycia jedynie norm Hamachera (jak w kodzie w ST powyżej) i (b)

dla sytuacji ogólnej – zestawienie trzech rodzin norm, omawianych w ramach niniejszego rozdziału.

a) b)



Rysunek 5. Wizualizacja sieci zaimplementowanej w sterowniku PLC: a) tylko dla norm Hamachera, b) dla norm Hamachera, Yagera, Dombiego parametryzowanych oddzielnie

4. Monitorowanie pracy silnika

Przy monitorowaniu pracy urządzeń relatywnie często wykorzystuje się podejście regułowe. Ekspert bazując na swojej wiedzy, definiuje zestaw reguł, które dotyczą poprawności lub niepoprawności pracy danego urządzenia. Na podstawie wniosków wynikających z tych reguł mogą być uruchamiane dodatkowe systemy zabezpieczające np. trwałość urządzenia [3,9].

4.1. Baza reguł i logiczna sieć rozmyta

Dla diagnozowania jakości pracy silnika spalinowego ekspert zaproponował 9 reguł o treści jak niżej [9]:

1. Jeśli pierścienie tłokowe są zużyte lub tłoki zużyte to zużycie oleju wzrasta.
2. Jeśli zużycie oleju wzrasta to ilość oleju w silniku maleje.
3. Jeśli coraz mniej oleju to temperatura silnika wzrasta.
4. Jeśli jest za mało oleju to kontrolka oleju świeci.
5. Jeśli temperatura silnika zbyt wysoka to słabsze przyspieszenie.
6. Jeśli są złe warunki drogowe i niski prześwit samochodu to miska olejowa może się uszkodzić.
7. Jeśli miska olejowa jest uszkodzona to oleju w silniku jest coraz mniej.
8. Jeśli świece są stare (duży przebieg) to zapłon jest nieregularny.
9. Jeśli zapłon jest nieregularny to gorsze przyspieszenie.

Aby w sposób formalny opisać podane reguły, należy wprowadzić następujące oznaczenia:

- P1 – stan pierścieni tłokowych (zużyte).
- P2 – stan tłoków (zużyte).
- P3 – zużycie oleju (wzrost zużycia).
- P4 – ilość oleju w silniku (spadek).

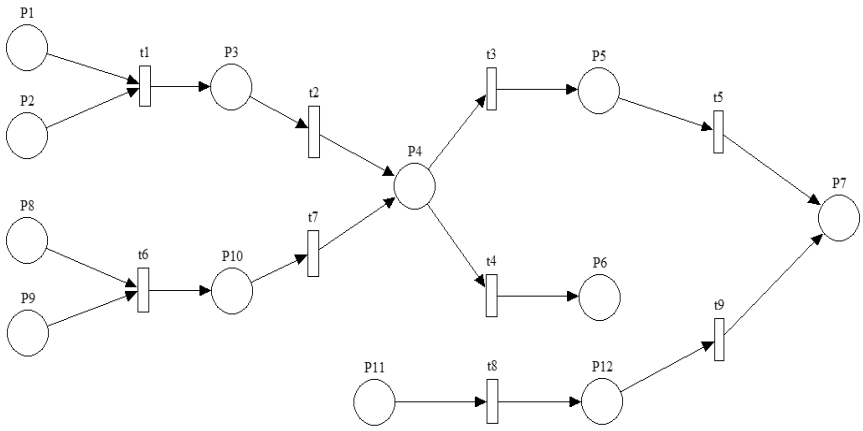
- P5 – temperatura silnika (wzrost).
- P6 – kontrolka oleju (czerwona).
- P7 – odpowiedź na przyspieszenie (opóźnione).
- P8 – warunki drogowe (złe).
- P9 – prześwit podwozia (niski).
- P10 – miska olejowa (dziurawa).
- P11 – przebieg świec zapłonowych (wysoki).
- P12 – zapłon (nieregularny).

Powyższe oznaczenia stanowią zarazem miejsca przyszłej sieci rozmytej.

Jeśli teraz w definicji reguł eksperta uwzględnić, wprowadzone powyżej, oznaczenia miejsc, to reguły przyjmują postać jak niżej, przy zachowaniu numeracji:

1. IF P1 OR P2 THEN P3.
2. IF P3 THEN P4.
3. IF P4 THEN P5.
4. IF P4 THEN P6.
5. IF P5 THEN P7.
6. IF P8 AND P9 THEN P10.
7. IF P10 THEN P4.
8. IF P11 THEN P12.
9. IF P12 THEN P7

Tak opisane reguły wnioskowania można przedstawić w formie logicznej rozmytej sieci Petriego o grafie jak na rys. 6.



Rysunek 6. Graf logicznej rozmytej sieci Petriego: Pi – miejsca sieci, ti - tranzycje

Ekspert określił także dla każdej tranzycji wartość funkcji progowej γ oraz współczynnik pewności reguły β. Zebrano je odpowiednio w tabelach 2 i 3.

Tabela 2. Wartości współczynnika γ podane przez eksperta dla kolejnych tranzycji

t1	t2	t3	t4	t5	t6	t7	t8	t9
0,34	0,51	0,42	0,39	0,4	0,55	0,45	0,29	0,56

Tabela 3. Wartości współczynnika β podane przez eksperta dla kolejnych tranzycji

t1	t2	t3	t4	t5	t6	t7	t8	t9
0,34	0,51	0,42	0,39	0,4	0,55	0,45	0,29	0,56

Obliczanie wartości miejsc dla prezentowanej logicznej rozmytej sieci Petriego odbywa się wg następujących wzorów:

$$\mu_{P3}=max(\mu_{P1},\mu_{P2})\beta_{t1} \tag{1}$$

$$\mu_{P4}=max(\mu_{P10}\beta_{t7},\mu_{P3}\beta_{t2}) \tag{2}$$

$$\mu_{P5}=\mu_{P4}\beta_{t3} \tag{3}$$

$$\mu_{P6}=\mu_{P4}\beta_{t4} \tag{4}$$

$$\mu_{P7}=max(\mu_{P12}\beta_{t9},\mu_{P5}\beta_{t5}) \tag{5}$$

$$\mu_{P10}=min(\mu_{P8},\mu_{P9})\beta_{t6} \tag{6}$$

$$\mu_{P12}=\mu_{P11}\beta_{t81} \tag{7}$$

Niech teraz, dla ustalenia uwagi, wartości miejsc wejściowych sieci, czyli wartości przesłanek, przyjmują wartości zgodne z zawartością tabeli 4.

Tabela 4. Przyjęte wartości miejsc wejściowych sieci

P1	P2	P8	P9	P11
0,92	0,85	0,88	0,79	0,82

Wówczas, na mocy wzorów od (2) do (7), wartości pozostałych miejsc rozważanej sieci będą miały wartości jak w tab. 5.

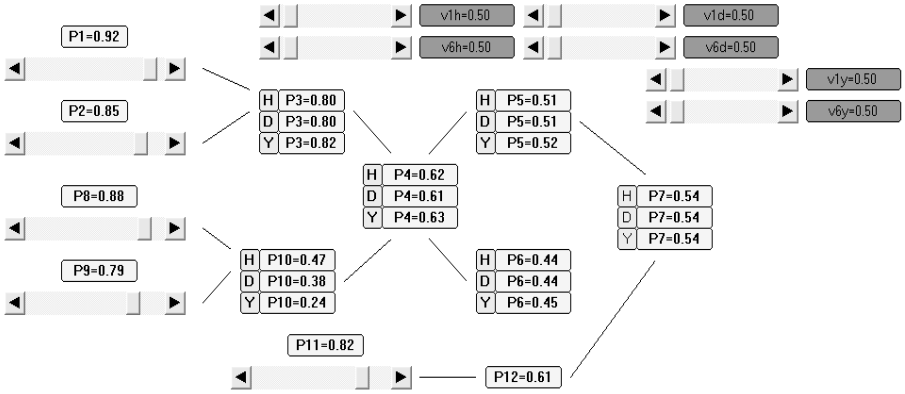
Tabela 5. Obliczone wartości miejsc (wniosków) sieci

P3	P4	P5	P6	P7	P10	P12
0,75	0,58	0,48	0,41	0,55	0,53	0,62

4.2. Monitorowanie pracy silnika jako parametryzowana rozmyta sieć Petriego

Obecnie problem opisany w rozdz. 4.1 zostanie rozwiązany na bazie parametryzowanej rozmytej sieci Petriego, której definicję przytoczono w rozdz. 1. Implementacja sieci nastąpiła dla urządzenia czasu rzeczywistego, sterownika PLC, por. rozdz. 3. Tak jak w poprzednim rozdziale operujemy 9 regułami, na bazie których można wskazać 12 miejsc sieci oraz 9 tranzycji. Graf dla parametryzowanej rozmytej sieci Petriego jest analogiczny jak dla logicznej rozmytej sieci Petriego, por. rys. 5. W rzeczywistości rozważono trzy sieci rozmyte odpowiednio dla norm Dombiego, Hamachera oraz Yagera, tak jak to sugerowano w rozdz. 2.2.

Na rysunku 7 przedstawiono widok ekranu z wizualizacją sieci zrealizowaną w pakiecie TwinCat. Jak już powiedziano w rozdz. 3 pakiet ten jest przeznaczony do programowania sterowników przemysłowych PLC i PAC, a jego moduł wizualizacji jest dedykowany do realizacji zadań analogicznych jak dla panelu operatorskiego. Zadania te obejmują przede wszystkim schematyczną wizualizację, alarmowanie oraz proste sterowanie operatorskie. Z tego względu ekspresja wizualna modułu jest ograniczona.



Rysunek 7. Wizualizacja rozmytej parametryzowanej siecie Petriego zrealizowana w pakiecie TwinCat

Porównując wartości miejsc dla poszczególnych norm w tab. 6 z wartościami uzyskanymi dla logicznej rozmytej sieci Petriego, przedstawionymi wcześniej w tab. 5 widać znaczące różnice.

Tabela 6. Obliczone wartości miejsc (wnioski) dla parametryzowanej rozmytej sieci Petriego, gdy v=0,5

Column1	P3	P4	P5	P6	P7	P10	P12
Hamacher	0,8	0,62	0,51	0,44	0,54	0,47	0,61
Yager	0,82	0,63	0,52	0,45	0,54	0,24	0,61
Dombi	0,8	0,61	0,51	0,44	0,54	0,38	0,61

Pierwszym problemem jest zatem takie dobranie wartości parametru *v*, aby rozważane parametryzowane rozmyte sieci Petriego przyjmowały zbliżone wartości miejsc do uzyskanych w logicznej rozmytej sieci Petriego. W rozważanym przypadku strojenie parametru *v* jest proste i właściwe wartości udaje się uzyskać przy jednej płynnej zmianie położenia suwaków pokazanych na rys. 7. Satysfakcjonujący rezultat można uzyskać modyfikując dla każdej z norm współczynnik dla tranzycji t1 i t6. Dla normy Hamachera współczynniki przyjmują odpowiednio wartości: v1h=0,5, v6h=0,5. W przypadku norm Dombiego: v1d=0,38, v6d=0,38. Dla norm Yagera należało ustawić: v1y=6,95, v6y=1,17. W tabeli 7 pokazano uzyskane wartości miejsc dla tak ustawionych parametrów. Dla wszystkich przypadków rozważanej sieci uzyskano identyczne wyniki. Zatem rozważane parametryzowane sieci Petriego mogą zastąpić logiczną rozmytą sieć Petriego w przypadku monitorowania pracy silnika.

Tabela 7. Obliczone wartości miejsc (wnioski) dla parametryzowanej rozmytej sieci Petriego, gdy v1h=0,5, v1d=0,38, v1y=6,95, v6h=0,5, v6d=0,98, v6y=1,17

Column1	P3	P4	P5	P6	P7	P10	P12
Hamacher	0,8	0,62	0,51	0,44	0,54	0,47	0,62
Yager	0,8	0,62	0,51	0,44	0,54	0,47	0,62
Dombi	0,8	0,62	0,51	0,44	0,54	0,47	0,62

Słowna interpretacja uzyskanego wyniku może być następująca:

- Z powodu dość mocno zużytych pierścieni tłokowych zużycie oleju jest na bardzo wysokim poziomie - zmalała jego ilość, to z kolei wpłynęło na wzrost temperatury silnika.
- Natomiast konsekwencją złego stanu świec zapłonowych - ich dużego przebiegu, jest wzrost czasu reakcji silnika na przyspieszenie.

4.3. Potencjalna możliwość korygowania przez eksperta wniosków

Nadal rozważamy parametryzowaną rozmytą sieć Petriego taką, jak w poprzednim rozdziale (por. rys. 6), zmianie ulegną wartości przesłanek oraz wnioski proponowane przez eksperta (przy niezmienionych regułach wnioskowania).

Przyjmijmy zmianę wcześniejszych przesłanek na następujące: niewielkie zużycie pierścieni tłokowych oraz tłoków, bardzo dobre warunki drogowe oraz dobry stan świec. Dla takich przesłanek ekspert podał następujący opis wniosku: przy bardzo dobrych warunkach drogowych, relatywnie nowych świecach zapłonowych i niewielkim zużyciu pierścieni tłokowych następuje wzrost zużycia oleju silnikowego, jednak nie w takim stopniu, aby drastycznie wpłynąć na pracę i osiągi silnika.

Wartości przesłanek, czyli w omawianej sieci wartości miejsc wejściowych, przyjmują wartości jak podano w tab. 8.

Tabela 8. Przyjęte wartości miejsc wejściowych sieci

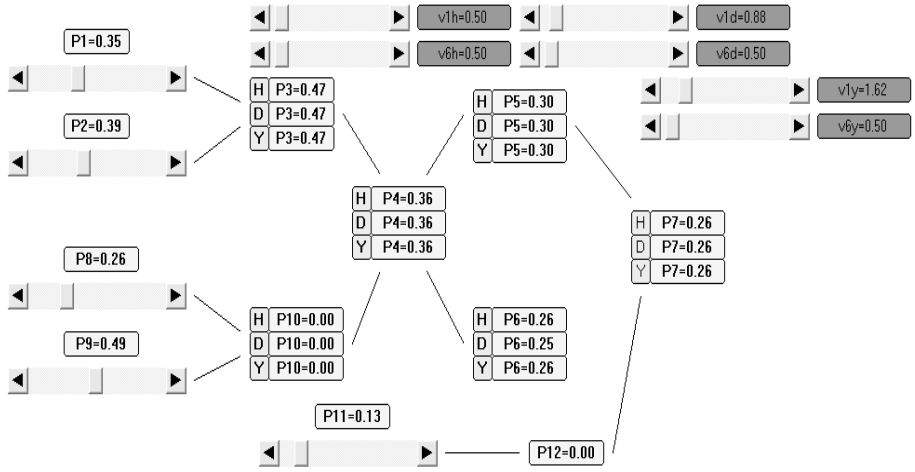
P1	P2	P8	P9	P11
0,35	0,39	0,26	0,49	0,13

Natomiast wartości wniosków, czyli pozostałych miejsc sieci powinny przyjąć wartości podane w tab. 9.

Tabela 9. Spodziewane wartości miejsc (wnioski) dla parametryzowanej rozmytej sieci Petriego

P3	P4	P5	P6	P7	P10	P12
0,8	0,36	0,3	0,26	0,26	0	0

Rysunek 8 przedstawia odpowiednio sparametryzowaną sieć, czyniącą zadość postawionym wcześniej wymaganiom.



Rysunek 8. Wizualizacja w TwinCat sieci dla wniosków wskazanych przez eksperta

Można zauważyć, że problem daje się rozwiązać dla norm Hamachera, Dombiego i Yagera, a parametry przyjmują wartości: v1d = 0,88, v1y = 1,62, pozostałe 0,5.

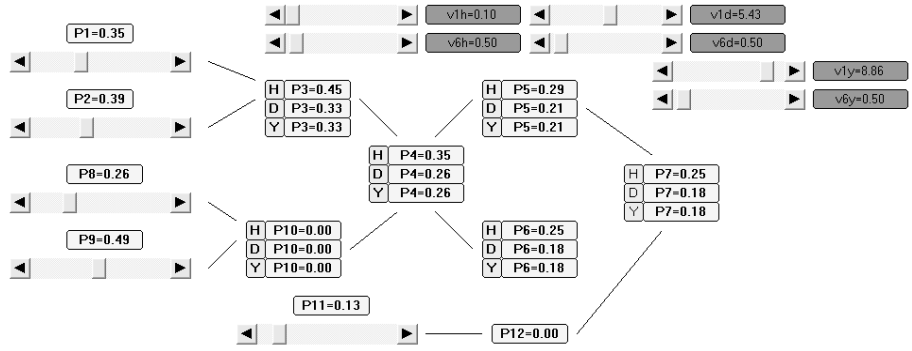
Załóżmy, że po zastanowieniu i konsultacji z innymi ekspertami, ekspert skorygował spodziewane wartości wniosków i określa je teraz tab. 10.

Tabela 10. Skorygowane przez eksperta wartości miejsc (wnioski) dla parametryzowanej rozmytej sieci Petriego

P3	P4	P5	P6	P7	P10	P12
0,32	0,25	0,21	0,18	0,18	0	0

Problem jest teraz następujący: czy możliwe jest takie *dostrojenie* wartości parametru *v*, aby przy niezmienionych regułach i niezmienionych wartościach progowych oraz takich jak poprzednio wartościach współczynników prawdziwości reguł uzyskać oczekiwane wartości wniosków. Intuicja podpowiada, że powinno być to możliwe. Skoro ekspert podał poprawne reguły i trafnie ocenił stopień ich prawdziwości oraz wartości progowe zostały wybrane nieprzypadkowo, to pod warunkiem, że wnioski sugerowane przez eksperta są poprawne, powinny wynikać z rozważanej sieci.

Rozwiązanie pokazuje rys. 9. Wymagania eksperta udało się spełnić dla norm Dombiego i Yagera ustawiając v1d=5,43, v6d=0,5 oraz v1y=8,86, v6d=0,5.



Rys. 9. Wizualizacja w TwinCat sieci dla wniosków skorygowanych przez eksperta

Wnioskiem cząstkowym jest stwierdzenie, że zastosowanie parametryzowanej rozmytej sieci Petriego pozwala *dostroić* sieć do wymagań i zaleceń eksperta bez ingerowania w reguły wnioskowania i wartości współczynników z nimi związanych.

5. Podsumowanie

W rozdziale zaprezentowano wyniki badań nad praktycznym zastosowaniem nowego typu sieci Petriego, a mianowicie parametryzowanych rozmytych sieci Petriego. W szczególności rozważono trzy przypadki takich sieci, dla s- i t-norm Hamachera, Yagera i Dombiego, uzasadniając celowość wyboru właśnie tych norm. Realizując system diagnozowania pracy silnika spalinowego pokazano, że parametryzowana rozmyta sieć Petriego z powodzeniem może zastąpić tzw. logiczną rozmytą sieć Petriego. Ponadto, możliwość *dostrajania* tego typu sieci pozwala ekspertowi definiować nie tylko reguły wnioskowania, stopień ich prawdziwości i wartości funkcji progowych, ale także wskazywać wartości wniosków przy zadanych wartościach przesłanek. Ponadto, taka właściwość sieci umożliwia uwzględnianie tzw. procesów starzeniowych, naturalnie występujących w praktyce (np. występujących we wszystkich procesach produkcyjnych, działaniu maszyn itp.). Parametryzowana rozmyta sieć Petriego jest szczególnie przydatna w przypadku problemów, gdzie uzyskiwane są wnioski przeciwstawne i niezbędne jest rozstrzygnięcie o trafności

jednego z nich. W takim przypadku zmiana wartości parametru sieci pozwala na uzyskanie większej różnicy między wartościami wniosków lub wręcz pozwala wyjść z sytuacji braku rozstrzygnięcia (identyczne wartości wniosków przeciwstawnych).

Przedstawione w rozdziale spostrzeżenia zachęcają do dalszych prac, które powinny objąć problem sposobu doboru wartości parametrów sieci, w przypadku gdy sieć jest wielowarstwowa, a ponadto chcemy zmieniać wszystkie dostępne w parametryzowanych sieciach Petriego parametry. Aby omawiany typ sieci faktycznie znalazł praktyczne zastosowanie, konieczne jest opracowanie zaleceń, a najlepiej kompletnego algorytmu, doboru wartości poszczególnych parametrów dla kolejnych warstw sieci. Odrębną kwestią jest wybór odpowiedniej rodziny parametryzowanej s- lub t-normy. W tym przypadku bardzo pomocne jest wykorzystanie panującego wśród norm częściowego porządku.

Literatura

[1] CENELEC, *EN 61131-3, Programmable controllers - Part 3: Programming languages (IEC 61131-3:2013), International Standard*, Brussels, May 2013.

[2] Lewis Robert W., *Programming Industrial Control Systems Using IEC 1131-3*, IEE Control Engineering Series, IEE, 1998.

[3] Looney, C.G.: *Fuzzy Petri Nets for Rule-Based Decision-making*, IEEE Transaction on Systems, Man, and Cybernetics, vol.18, no.1, Jun-Feb. 1988, pp.178-83.

[4] Suraj, Z.: *Parameterised Fuzzy Petri Nets for Approximate Reasoning in Decision Support Systems*. In: Proc. of AMLTA'2012, Dec. 8-10, 2012, Cairo, Egypt, Comm. in Comp. and Infor. Sci. series, Vol. 322, Springer, 2012, pp. 33-42

[5] Suraj, Z.: *Generalised Fuzzy Petri Nets for Approximate Reasoning in Decision Support Systems*. In: Proc. of Int. Workshop on Concurrency, Specification, and Programming (CS&P'2012)", Sept. 28-30, 2012, Humboldt University, Berlin, 2012, pp. 370-381.

[6] Suraj, Z.: *Parameterised Fuzzy Petri Nets for Approximate Reasoning in Decision Support Systems*. In: Proc. of Int. Conf. on Advanced Machine Learning Technologies and Applications (AMLTA'2012), Dec. 8-10, 2012, Cairo, Egypt, Communications in Computer and Information Science series, Vol. 322, Springer, 2012, pp. 33-42.

[7] Suraj, Z.: *Knowledge Representation and Reasoning Based on Generalised Fuzzy Petri Nets*. In: Proc. of Int. Conf. on Intelligent Systems Design and Applications (ISDA'2012), Nov. 27-29, 2012, Kochi, India, IEEE Press, 2012.

[8] Szpyrka Marcin. Sieci Petriego w modelowaniu i analizie systemów współbieżnych,2008.

[9] Yingping Zheng, Changjun Jiang. Fuzzy reasoning based on petri nets. Department of Computer Science and Engineering Tongji University, Shanghai, 2012.

Rozdział 10

AngularJS vs. Ember.js: analiza wydajności frameworków dla aplikacji webowych typu SPA

1. Wprowadzenie

Single Page Application (SPA), to termin charakteryzujący nowoczesne aplikacje webowe, w których, po załadowaniu strony i wykonaniu jakiejkolwiek akcji przez użytkownika, nie następuje klasyczne przeładowanie strony. Dzięki temu, podczas korzystania z aplikacji, można odnieść wrażenie, że ma sie do czynienia z natywną aplikacją, a nie stroną internetową.

W aplikacjach typu SPA, serwer udostępnia klientowi API (ang. *Application Programming Interface*), przy pomocy którego klient jest w stanie w dowolnym momencie zażądać dowolny zasób. Serwer odpowiedzialny jest za obsługę żądań klienta i przesyłanie odpowiedzi jako surowych danych, najczęściej w formacie JSON (ang. *JavaScript Notation Object*). JSON jako format wymiany danych stał sie swego rodzaju standardem w tego typu aplikacjach ze względu na swoja prostotę i niewielki wpływ na rozmiar przesyłanych danych.

Ponieważ odpowiedzialność za generowanie kodu HTML została przeniesiona z serwera na klienta, cała logika związana z obsługą interfejsu użytkownika, zdarzeń i akcji, jakie może wykonać użytkownik, musi mieć miejsce w kliencie. Dzięki temu serwer i aplikacje serwerowe mogą skupić się na odbieraniu, przetwarzaniu i wysyłaniu danych, a klient na generowaniu i obsłudze interfejsu użytkownika. Ze względu na bezpieczeństwo, to strona serwerowa nadal jest odpowiedzialna za uwierzytelnianie i autoryzację użytkowników, jak i ostateczną walidację danych, czy komunikację ze źródłami danych, zapewniając tym samym synchronizację.

Aplikacja SPA zmniejsza obciążenie serwera WWW i uproszcza aplikację serwerową, co często ma przełożenie na wzrost jej wydajności. Architektura tego typu określana jest jako gruby klient (ang. *fat client*) – cienki serwer (ang. *thin server*).

Kolejną zaletą SPA jest niezależny rozwój aplikacji serwerowej oraz klienckiej. Aplikacja kliencka nie ma wglądu w szczegóły dot. implementacji aplikacji serwerowej. Nie jest również w żaden sposób uzależniona od technologii użytej do utworzenia aplikacji serwerowej. Obie aplikacje mogą zostać umieszczone na odrębnych serwerach, pod różnymi domenami.

Analiza wydajności aplikacji webowych przez długi czas utożsamiana była wyłącznie z wydajnością serwera WWW i aplikacji serwerowej, skupiając się na czasie obsługi żądania HTTP i wygenerowania kodu HTML i przesłania odpowiedzi do klienta. Wydajność serwera WWW aplikacji serwerowej nadal jest ważnym czynnikiem w

odniesieniu do wydajności aplikacji webowych, niemniej w aplikacjach typu SPA to klient, czyli przeglądarka internetowa jest wąskim gardłem w kwestii postrzeganej wydajności. Szczególnie kosztowne są operacje związane z manipulacją modelu obiektowego – DOM (ang. *Document Object Model*) i renderowaniem widoków lub wybranych fragmentów widoku. Dlatego celem autorów było przebadanie dwóch najpopularniejszych frameworków JavaScript AngularJS i Ember.js pod katem wydajności postrzeganej przez użytkownika.

2. Porównanie frameworków AngularJS i Ember.js

W Internecie dostępnych jest kilkadziesiąt różnych frameworków JavaScript do tworzenia w aplikacji typu SPA. Jednak AngularJS i Ember.js, to obecnie dwa, kompletne i dojrzałe frameworki JavaScript, które mają zestaw funkcjonalności pozwalających na zbudowanie dowolnej wielkości Single Page Application. Rozwiązania te znalazły swoje zastosowanie z wielkim sukcesem w różnego rodzaju projektach internetowych i używane są przez takie firmy, jak Google, Yahoo, Zendesk, Groupon, Sky Store, czy MSNBC. Zdecydowano sie na wybór AngularJS i Ember.js ze względu na fakt, że oba frameworki pozwalają zrealizować ten sam cel w odmienny sposób. Realizują w inny sposób renderowanie widoków, warstwę przechowywania danych lub modeli, zarządzanie routingiem oraz wymuszają odmienny podział odpowiedzialności (ang. *separation of concerns*) wewnątrz samej aplikacji.

AngularJS i Ember.js maja wiele cech wspólnych, takich jak dwukierunkowe wiązanie danych, integrację z REST API, tworzenie własnych tagów HTML wraz ze zdefiniowaną funkcjonalnością, wbudowane wsparcie dla obiektów Promise (17), czy rozwiązanie kwestii routingu wewnątrz aplikacji. Dokładne porównanie zamieszczono w tabeli 1.

Do największych różnic pomiędzy frameworkami AngularJS i Ember.js, należą:

- realizacja dwukierunkowego wiązania danych,
- renderowanie widoków aplikacji i aktualizacja danych w modelu DOM,
- warstwa przechowywania danych.

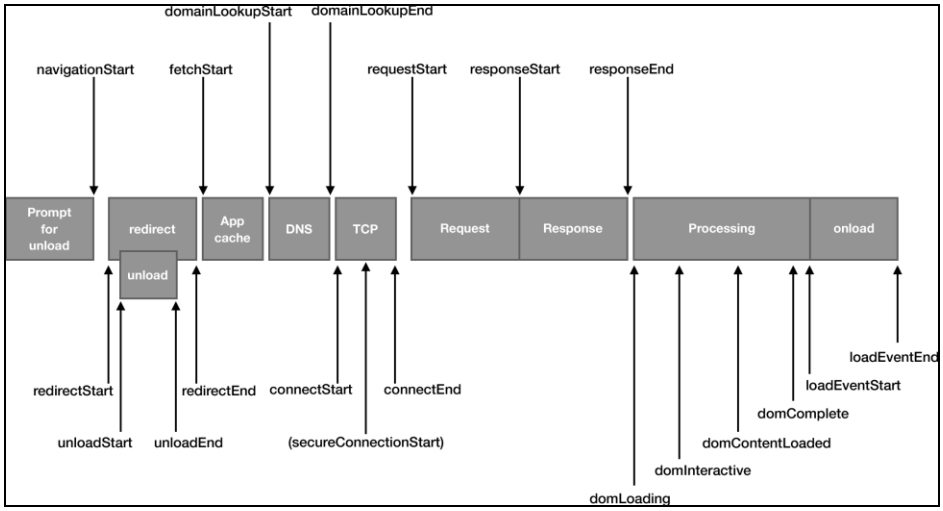
Tabela 1. Porównanie frameworków AngularJS i Ember.js. Wyszarzeniem zaznaczono różnice.

Właściwość / Cecha	**AngularJS (1.3.12)**	**Ember.js (1.9.1)**
Wzorzec architektoniczny	MVW (ang. *Model View Whatever*), zbliżony do MVVM (ang. *Model View Viev-Model*)	MVC (ang. *Model View Controller*)
System szablonów	Oparty na fragmentach DOM	Zewnętrzna biblioteka Handlebars.js oparta na łączeniu łańcuchów znaków. Brak możliwości umieszczania kodu JavaScript wewnątrz szablonów HTML.
Warstwa przechowywania danych	Brak, możliwość zastosowania nieoficjalnego rozwiązania angular-data	Tak, Ember Data
Integracja z REST API	Tak, przy pomocy ng-resource	Tak, wbudowana w framework DSRestAdapter
Zagnieżdżone widoki (ang. *nested views*)	Tak	Tak
Możliwość tworzenia własnych tagów HTML	Tak, poprzez dyrektywy	Tak, poprzez komponenty
Dwukierunkowe wiązanie danych	Tak	Tak
Routing	Tak (opcjonalny)	Tak (wymagany)
Wsparcie animacji	Tak, ng-animate	Brak
Zależności	Brak	Tak, wymagana jest biblioteka Handlebars.js
Kompatybilność z innymi frameworkami	Tak	Tak
Biblioteka do obsługi obiektów Promise	Tak, bazująca na bibliotece Q.js	Tak, RSVP.JS
Możliwość obserwowania zmian modelu	Tak, poprzez watch expressions	Tak, z użyciem observables
Walidacja formularzy	Tak, wbudowana synchroniczna i asynchroniczna walidacja formularzy	Brak
Oficjalny system budowania aplikacji	Brak, możliwość skorzystania z narzędzia yeoman	ember-cli
Rozmiar frameworka wraz z zależnościami po minifikacji	123 kB	396,4 kB
Rozmiar frameworka wraz z zależnościami z użyciem kompresji gzip	47 kB	119,3 kB
Dodatkowe funkcjonalności	Moduł ułatwiający testowanie ngMock	Obliczane właściwości (computed properties)

3. Wydajność aplikacji webowych po stronie klienta

Wydajność strony internetowej powszechnie rozumiana jest jako czas potrzebny na załadowanie sie całej strony. Kiedy strona internetowa zostanie załadowana w całości, przeglądarka wyzwala zdarzenie *onload*, dzięki czemu można dokonać pomiaru czasu, potrzebnego na załadowanie całej strony.

Aby zrozumieć co dzieje się podczas pracy przeglądarki internetowej warto posłużyć się ilustracją przygotowaną przez W3C Web Performance Working Group [15] Na rysunku 1. wszystkie etapy zostały zilustrowane w postaci bloków wraz z nazwami granicznych lub wewnętrznych zdarzeń.



Rysunek 1. Proces wyświetlania strony internetowej od momentu uruchomienia przeglądarki lub nowej karty do kompletnego załadowania strony internetowej [15].

3.1. Proces wyświetlania strony internetowej

Zostanie teraz prześledzony proces od momentu uruchomienia przeglądarki internetowej, bądź otwarcia nowej karty, do wyświetlenia kompletnej strony internetowej. Opis ten dotyczy również sytuacji, w której użytkownik klikając w pewien odsyłacz (link), przechodzi na kolejna stronę.

Pierwszym etapem jest wyzwolenie zdarzenia *unload*, które usuwa z aktywnej karty lub okna przeglądarki aktualnie wygenerowany tam dokument (stronę internetową).

Po wyzwoleniu zdarzenia *unload* przeglądarka odczytuje URL z paska adresu, bądź klikniętego odsyłacza. Jeżeli wskazywany przez URL zasób nie zostanie znaleziony w pamięci podręcznej przeglądarki, klient łączy sie z serwerem DNS w celu uzyskania adresu IP serwera. Następnie klient zestawia z serwerem połączenie TCP, po czym wysyła do serwera żądanie HTTP (najczęściej GET lub POST). Serwer, po obsłużeniu żądania, zaczyna wysyłać klientowi odpowiedz zawierającą plik z kodem HTML, CSS, SVG, JavaScript lub binarny.

Przeglądarka internetowa, aby przetworzyć otrzymywaną odpowiedź, musi wykonać szereg działań ściśle zdefiniowanych przez specyfikację HTML [14] i CSS [13]. Warto nadmienić, że kod HTML i JavaScript interpretowane są na bieżąco. Miedzy innymi z tego powodu kod JavaScript powinien być umieszczony na dole strony, przed zamykającym znacznikiem `</body>`, ponieważ przeglądarka wstrzymuje proces wyświetlania w momencie przetwarzania kodu JavaScript.

W wyniku parsowania kodu HTML powstaje obiektowy model dokumentu określany jako DOM (ang. *Document Object Model*). Parsowanie przebiega w następujący sposób [5]:

1. **Konwersja** – przeglądarka odkodowuje bajt po bajcie zawartość dokumentu HTML, uwzględniając dane kodowanie pliku, np. UTF-8.
2. **Tokenizacja** – przeglądarka transformuje łańcuchy znaków na odrębne tokeny zgodnie ze standardem określonym w specyfikacji HTML, np. `<p>`, `<div>`. Każdy z tokenów ma specjalne znaczenie, określone pewnym zestawem reguł.

3. **Analiza leksykalna** – tokeny przekształcane są w obiekty z określonymi właściwościami i regułami.
4. **Tworzenie modelu DOM** – obiekty łączone są w drzewiastą strukturę danych odzwierciedlającą zależności pomiędzy elementami nadrzędnymi i podrzędnymi, określonymi w kodzie HTML (jedne tagi zawierają się wśród innych tagów). Obiekt HTML jest elementem nadrzędnym – korzeniem drzewa.

Po wygenerowaniu modelu DOM przeglądarka nie może wyświetlić strony, gdyż DOM jest tylko reprezentacją właściwości i zależności opisanych w kodzie HTML i nie zawiera żadnych informacji na temat wyglądu elementów, które należy wyświetlić. Odpowiedzialność za wygląd spoczywa na modelu CSSOM (ang. *Cascading Style Sheets Object Model*). Parsowanie kodu CSS ma taki sam przebieg, jak w przypadku parsowania kodu HTML. Bajty danych CSS przekształcane są w znaki, następnie w tokeny i węzły, a na końcu łączone są w strukturę drzewiastą CSSOM.

Drzewa DOM i CSSOM są strukturami niezależnymi od siebie, dlatego przeglądarka nie może jeszcze wyświetlić strony internetowej. Na podstawie tych struktur budowane jest drzewo renderowania. Wyznacza ono rozmieszczenie każdego widocznego elementu na stronie, a na jego podstawie proces rysowania renderuje piksele na ekranie. Drzewo renderowania zawiera tylko te węzły, które będą wyświetlane na stronie, np. nie zawiera elementu, którego styl określa go jako „niewidoczny" (display: none).

Proces tworzenia drzewa renderowania przebiega w następujący sposób:

1. **Odwiedzenie każdego widocznego węzła w drzewie DOM** – pomimo, że niektóre węzły są widoczne, przeglądarka je pomija, np. `<script>`, `<link>`.
2. **Przyporządkowanie reguł CSS z drzewa CSSOM** – reguły CSS, które zostaną przyporządkowane dotyczą tylko widocznych węzłów drzewa DOM.
3. **Wygenerowanie widocznych węzłów wraz z treścią i regułami CSS**.

Po utworzeniu drzewa renderowania, następuje proces wyznaczania układu strony (ang. layout). Przeglądarka rozpoczyna wyznaczanie geometrii (położenie, rozmiar) każdego elementu, zaczynając od korzenia drzewa renderowania i przechodzi przez całe drzewo, obliczając geometrię każdego elementu zgodnie ze specyfikacja modelu pudełkowego [12] (ang. *box model*). Wszystkie położenia względne, korzystające z takich miar jak: em, rem, %, vh, vw, konwertowane są do bezwzględnych wartości w pikselach (px).

Ostatnim etapem jest wyrysowanie i rasteryzacja wszystkich elementów. Na podstawie drzewa renderowania zostają wówczas wyznaczone rzeczywiste piksele na ekranie użytkownika.

3.2. Reflow i repaint

Należy zwrócić uwagę, że z różnych przyczyn może dojść do zmiany wyglądu strony, np.:

- dodanie, przeniesienie, usunięcie, animacja elementu DOM,
- ukrycie elementu DOM przy użyciu CSS,
- zmiana wielkości okna przeglądarki internetowej,
- zmiana wielkości rozmiaru czcionki,
- przewinięcie strony internetowej.

Każda z powyższych akcji może zmienić dane wejściowe wykorzystane do utworzenia drzewa renderowania, wywołując proces reflow lub repaint.

Reflow (określany również jako *re-layout*), to proces polegający na przebudowaniu drzewa renderowania lub jego poddrzewa. W momencie, gdy drzewo renderowania zostanie przebudowane i zmieni się geometria elementów, nastąpi ponowne wyrysowanie (*repaint*) tych części strony, które uległy zmianie. Przykładowo, w przypadku ukrycia elementu przy wykorzystaniu reguły `display: none;` nastąpi *reflow* i *repaint*, ponieważ element zostanie usunięty z drzewa renderowania. W przypadku użycia reguły `visibility: hidden;` nastąpi tylko *repaint*.

Zarówno *reflow* jak i *repaint* mogą być procesami kosztownymi. Mogą negatywnie wpływać na postrzeganą przez użytkownika wydajność aplikacji webowej.

4. Pomiary wydajności aplikacji webowych po stronie klienta

Kwestia pomiarów wydajności aplikacji webowych po stronie klienta jest znana od dawna. W ramach World Wide Web Consortium (W3C) problemem tym zajmuje się Web Performance Working Group. Misją grupy jest opracowanie metod, które pozwolą na mierzenie różnych aspektów wydajnościowych aplikacji webowych po stronie klienta [16].

Grupa ta wypracowała specyfikację Navigation Timing API definiującą specjalny interfejs dla aplikacji webowych. Pozwala on na pomiar przepustowości łącza, latencji oraz czasu wymaganego do załadowania całej strony internetowej. Dostęp do danych możliwy jest poprzez interfejs PerformanceTiming [9]. Aby uzyskać dane na temat wybranego zasobu, należy posłużyć sie Resource Timing API. Zaprezentowany wcześniej rysunek 1. specyfikuje zdarzenia, jakie są rejestrowane przez Navigation Timing API. Google, Mozilla, Opera, Microsoft i Apple rozpoczęły już w swoich przeglądarkach proces wdrażania rozwiązań pozwalających na mierzenie wydajności po stronie klienta na bazie Navigation Timing API.

Dla badacza ważną kwestią jest łatwy dostęp do danych pomiarowych gromadzonych przez przeglądarkę. Z pomocą przychodzi tu zaproponowany przez Google – Remote Debugging Protocol [3], który umożliwia dostęp do informacji kolekcjonowanych podczas korzystania z przeglądarki na bazie JSON API.

Jednym z narzędzi, które wykorzystuje ten protokół jest *browser-perf* [11]. Jest to aplikacja umożliwiająca mierzenie procesów zachodzących wewnątrz przeglądarki internetowej podczas korzystania z aplikacji webowej. *Browser-perf* działa w środowisku node.js wykorzystując *Selenium WebDriver* (WS.js), który przy pomocy Remote Debugging Protocol symuluje działania prawdziwego użytkownika. Dzięki temu możliwy jest pomiar wydajności podczas wykonywania określonych akcji, takich jak: wpisywanie tekstu w pole tekstowe, zmiana zakładki, czy przejście na inną podstronę. *Browser-perf* to aplikacja typu CLI (ang. *Command Line Interface*), co oznacza, że wszystkie polecenia wykonywane są z poziomu konsoli (wiersza poleceń). Aktualnie umożliwia wykonywanie pomiarów wydajności w następujących przypadkach:

- środowisko lokalne (komputer PC) – Google Chrome, Safari, Firefox, Internet Explorer,
- urządzenia mobilne i emulatory – Mobile Safari (iOS), Google Chrome (Android),
- aplikacje hybrydowe – Aplikacje Cordova/PhoneGap.

Browser-perf wspiera następujące metryki [11]:

- Google Chrome Timeline Metrics [4],

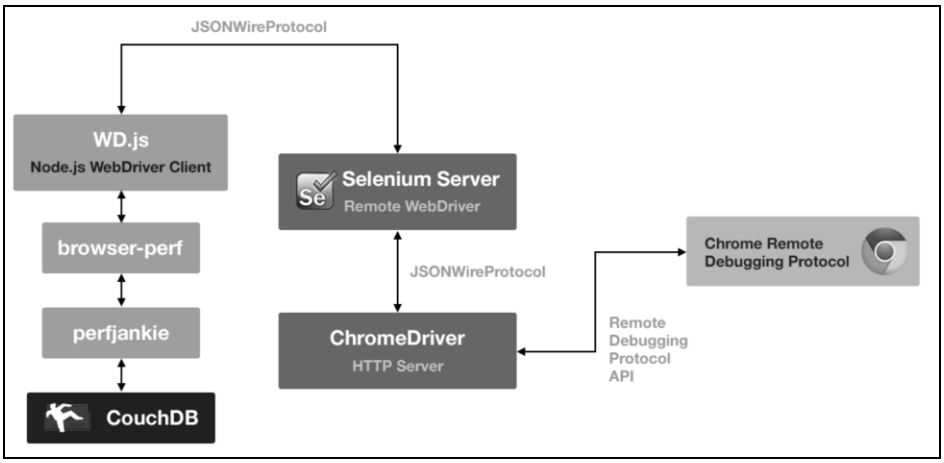
- Google Chrome Tracing Metrics,
- Request Animation Frame Metrics,
- PerfMetrics,
- Navigation Timing API.

Jak widać, alternatywą dla metryk Navigation Timing API mogą być metryki Google Chrome Timeline Metrics. Metryki te podzielone są na grupy: Scripting, Loading, Rendering i Painting. W tabeli 2. zestawiono najważniejsze metryki należące poszczególnych grup.

Tabela 2. Wybrane metryki udostępniane przez Google Chrome Timeline Metrics.

Grupa	Nazwa	Opis
Scripting	FunctionCall	Czas potrzebny na wykonywanie kodu JavaScript
Scripting	EvaluateScript	Czas potrzeby na interpretację kodu JavaScript
Scripting	GCEvent	Czas potrzebny na odśmiecenie pamięci
Scripting	TimerFired	Czas potrzebny na przetwarzanie funkcji wywołanych przy pomocy setTimeout() i setInterval()
Loading	ParseHTML	Czas potrzebny na konwersję, tokenizację, analizę leksykalną i tworzenie modelu DOM
Rendering	Layout	Czas potrzebny na tworzenie układu strony (tworzenie drzewa renderowania na podstawie drzewa DOM)
Rendering	RecalculateStyles	Czas potrzebny na przetwarzanie kodu CSS i generowanie modelu CSSOM
Painting	CompositeLayers	Czas potrzebny na scalanie warstw (poddrzewo drzewa renderowania) przez GPU i wyrenderowanie pikseli na ekranie
Painting	Paint	Czas potrzebny na rasteryzację wyświetlanej strony do bitmapy

Aplikacją uzupełniającą możliwości *browser-perf* jest *perfjankie* [10]. Aplikacja ta również uruchamiana jest w środowisku node.js i służy do zapisywanie danych zebranych przy pomocy *browser-perf* w bazie danych *Apache CouchDB*. Ważną funkcjonalnością *perfjankie* jest możliwość automatyzacji wykonywania pomiarów poprzez ustalenie liczby powtórzeń wykonywanego badania. *Perfjankie*, to aplikacja typu CLI (uruchamiana z konsoli). Istnieje też aplikacja webowa, która umożliwia przeglądanie zebranych pomiarów, zapisanych w bazie danych *CouchDB*. Zależności pomiędzy omówionymi wyżej aplikacjami zostały zilustrowane na rysunku 2.



Rysunek 2. Wybrane narzędzia pomiarowe

Alternatywną platformą do badania wydajności aplikacji webowych po stronie klienta może być *WebPageTest.org*. Jest to serwis internetowy rozwijany i wspierany przez firmę Google. Narzędzie umożliwia przetestowanie strony internetowej przy pomocy metryk Google Chrome Timeline Metrics, Google Chrome Tracing Metrics oraz Navigation Timing API. Jedną z najważniejszych funkcji *WebPageTest.org* jest możliwość nagrania filmu ukazującego postęp ładowania strony internetowej oraz możliwość porównania dwóch serwisów na podstawie klatek z nagrania video. Funkcja ta jest bardzo przydatna do ustalania wpływu wprowadzanych na stronie internetowej zmian na czas potrzebny do wyświetlania widocznego fragmentu strony.

Okazuje się, że metryka zwracająca czas całkowitego załadowania strony nie odzwierciedla dobrze rzeczywistości. Dlatego inżynierowie z *WebPageTest.org* opracowali metrykę *SpeedIndex* [7], która określa średnią ilość czasu (w milisekundach) potrzebną do wyświetlenia widocznego fragmentu strony internetowej. Jej wartość zależna jest od wielkości okna przeglądarki. *WebPageTest.org* wykorzystuje algorytm obliczający kompletność załadowania widocznego fragmentu strony. Algorytm ten bazuje na obrazach pobieranych z nagrania video o szybkości 10 klatek na sekundę. Co 0,1 s obliczana jest kompletność załadowania widocznego fragmentu strony wyrażona w procentach. Na podstawie tych danych tworzony jest wykres przedstawiający procent załadowania widocznego fragmentu strony od czasu. Wartość *SpeedIndex* wyznaczana jest na podstawie pola nad wykresem. Im mniejsza wartość *SpeedIndex*, tym lepiej – oznacza to, że widoczna część strony została przedstawiona użytkownikowi szybciej.

SpeedIndex i czas potrzebny na całkowite załadowanie aplikacji maja bardzo duże znaczenie w przypadku klasycznych aplikacji webowych. Aplikacje webowe typu SPA różnią sie diametralnie od klasycznych aplikacji. Zarówno czas potrzebny na załadowanie aplikacji webowej jak i *SpeedIndex* jest ważny. Ważniejsze jest jednak, jak szybko wyświetlane są poszczególne widoki już po uruchomieniu aplikacji typu SPA, jak długo trwa wykonywanie kodu JavaScript oraz jak reaguje interfejs użytkownika. Ma to szczególne znaczenie w przypadku rozstrzygnięcia, który framework JavaScript jest wydajniejszy. Z tego powodu, w przypadku aplikacji typu SPA, badania powinny skupić sie na „pracy" jaką musi wykonać przeglądarka internetowa już po załadowaniu całej aplikacji webowej (po zdarzeniu *onload*).

Oprócz *SpeedIndex* oraz "standardowych" metryk grupowych: Scripting, Loading, Rendering i Painting, najważniejszymi metrykami oferowanymi platformę WebPageTest.org są:

- **LoadTime** – czas, który upływa do pełnego załadowania sie strony internetowej (zdarzenie *onload*).
- **Time to first byte (TTFB)** – czas, który upływa od momentu wysłania zadania do serwera WWW do otrzymania pierwszego bajtu odpowiedzi przez przeglądarkę internetową.
- **StartRender** – czas, po którym nastąpiło pierwsze wymalowanie strony internetowej (zdarzenie paint).
- **FullyLoaded** – czas, po którym po zdarzeniu *onload* przeglądarka internetowa przez minimum 2 sekundy nie zarejestrowała żadnej aktywności sieciowej. Wartość tej metryki równa jest czasowi odpowiadającemu ostatniej aktywności sieciowej.
- **Time to visually complete (TTVC)** – czas, po którym widoczna cześć strony internetowej została w całości załadowana. Jej wartość jest zawsze większa niż SpeedIndex.

5. Aplikacje

Przegotowując się do analizy wydajności frameworków AngularJS i Ember.js, autorzy zwrócili uwagę, że istnieje wiele *microbenchmarków*, które badają tylko niewielki ich wycinek i skupiają się, np., na kwestii renderowania kilkudziesięciotysięcznej listy elementów. Autorom zależało jednak, aby badanie nie polegało na mierzeniu czasu „pojedynczych operacji", tylko na wyznaczaniu czasu, jaki przeglądarka wykorzystuje na wykonanie „całego kodu".

5.1. Aplikacje klienckie

W celu przeprowadzenia badań, na bazie obu farameworków JavaScript zostały skonstruowane osobne aplikacje webowe typu SPA. Aplikacje oferują takie same funkcjonalności, korzystają z tych samych stylów CSS i komunikują się w ten sam sposób z aplikacją serwerową udostępniającą JSON REST API.

Aplikacje opierają sie na koncepcji systemu umożliwiającego sprzedaż biletów na różnego rodzaju festiwale muzyczne. Utworzono tylko cześć systemu: panel administracyjny, który pozwala na zarządzanie festiwalami, przydzielanie puli dostępnych biletów, przeglądanie listy sprzedanych biletów oraz przeglądanie listy osób sprzedających bilety. Założono, że sprzedaż biletów będzie obsługiwać odrębna aplikacja webowa.

Obie aplikacje skonstruowano zgodnie z techniką RWD (ang. *Responsive Web Design*). Dzięki temu strony dostosowują się do rozmiaru okna przeglądarki zmieniając rozmieszczenie elementów i dostrajając nawigację.

Aplikacje składają się z trzynastu odrębnych widoków, jednak do badań zakwalifikowano trzy: dashboard (strona główna), lista sprzedawców oraz lista sprzedanych biletów. Kryterium wyboru był stopień zaawansowania widoku wyrażony liczbą zmiennych przekazywanych do widoku oraz liczbą generowanych elementów DOM.

W widoku dashboard użytkownik ma dostęp do podstawowych statystyk przedstawiających liczbę wszystkich sprzedanych biletów i liczbę sprzedawców. Na dole strony wyświetla sie informacja na temat postępu konfiguracji aplikacji, o ile konfiguracja nie została zakończona.

Widok listy sprzedawców ma formę tabeli. Po kliknięciu na wybrany wiersz w tabeli, odsłania sie dodatkowe menu, które umożliwia przejście do widoku profilu sprzedawcy oraz zmianę statusu sprzedawcy na pasywny. U góry strony umieszczono kontrolki pozwalające filtrować listę na podstawie imienia i nazwiska, statusu i numeru telefonu sprzedawcy. Na dole umieszczono stronicowanie (paginację) pozwalające przemieszczać sie miedzy stronami. Domyślnie, na jednej stronie wyświetlana jest lista dziesięciu sprzedawców. Liczbę wyświetlanych osób można modyfikować korzystając z parametru *limit* w URL. Przy przejściu na kolejna stronę, wykonywane jest zapytanie do aplikacji serwerowej. Po otrzymaniu odpowiedzi aktualizowana jest lista wyświetlanych sprzedawców.

Widok listy sprzedanych biletów ma również formę tabeli. W wierszach umieszczono informacje dotyczące sprzedawcy biletów, dane kupującego kwotę za jaką kupił bilety. U góry strony umieszczono kontrolki pozwalające filtrować listę na podstawie imienia i nazwiska kupującego, statusu transakcji oraz metody płatności. Umożliwiono też filtrowanie transakcji, które zostały już spłacone w całości. Filtrowanie odbywa sie natychmiast, przez przeszukanie aktualnie załadowanej listy sprzedanych biletów, bez wykonywania zapytań do aplikacji serwerowej. Domyślnie wyświetlanych jest 10

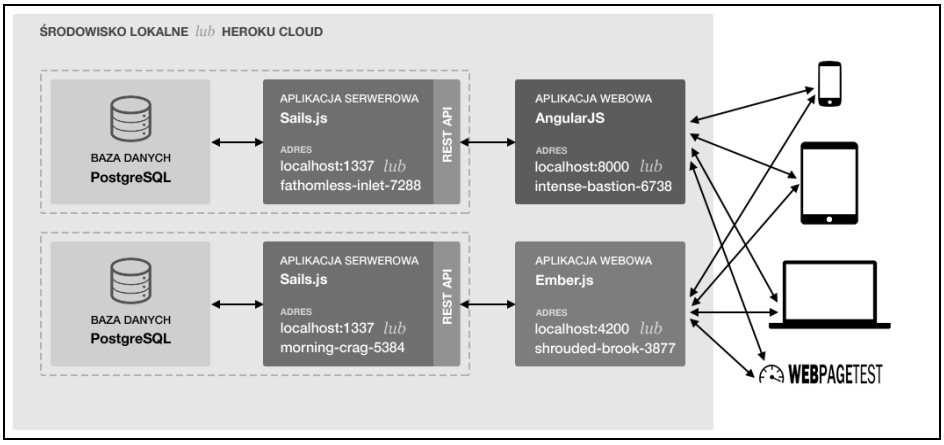
ostatnich sprzedanych biletów. Liczbę tę można modyfikować podając parametr *limit* w URL. Widok ten nie ma zaimplementowanej paginacji.

5.2. Aplikacja serwerowa i zewnętrzny hosting

Aplikacje webowe typu SPA komunikują sie z aplikacją serwerową skonstruowaną przy pomocy frameworka JavaScript Sails.js [8]. Framework Sails działa w środowisku node.js i implementuje wzorzec architektoniczny MVC, umożliwiając szybkie tworzenie aplikacji webowych w oparciu o REST API [2]. Posiada wbudowany mechanizm generowania różnych warstw aplikacji: modeli, widoków, kontrolerów z możliwością tworzenia odpowiednich *endpointów* zgodnych z REST API. Sails.js korzysta z serwera HTTP Express.js. Wykorzystuje też rozwiązanie typu ORM (ang. *Object-Relational Mapping*) o nazwie Waterline, umożliwiające integrację z dowolną baza danych. Na potrzeby aplikacji wybrano bazę PostgreSQL.

Aby przeprowadzić badania należało wypełnić bazę aplikacji serwerowej danymi. W tym celu skonstruowano narzędzie umożliwiające wygenerowanie dowolnej liczby rekordów. W ten sposób aplikacje webowe typu SPA mogły wyświetlać dowolnej długości listy sprzedawców lub sprzedanych biletów. Do generowania losowych danych, takich jak: imiona i nazwiska, adresy, numery telefonów, narzędzie wykorzystuje bibliotekę *chance.js*.

W zależności od rodzaju badania, kod obu aplikacji typu SPA został umieszczony na osobnych lokalnych serwerach lub serwerach *chmurowej* usługi firmy Heroku. Dla aplikacji klienckich uruchomiono osobne aplikacje serwerowe z własnymi bazami danych PostgreSQL. W ten sposób uzyskano całkowitą niezależność aplikacji webowych skonstruowanych przy pomocy frameworków AngularJS i Ember.js. Na rysunku 3. pokazano uproszczoną architekturę obu aplikacji.



Rysunek 3. Uproszczona architektura aplikacji webowych uruchomionych lokalnie i w usłudze chmurowej Heroku.

6. Badania

Zaplanowano badania wydajności z wykorzystaniem następujących widoków:

- dashboard,
- lista sprzedawców,
- lista sprzedanych biletów.

Badania podzielono na dwie części. W pierwszej, przy pomocy narzędzi *browser-perf* i *perfjankie*, przebadano widoki list sprzedanych biletów, natomiast w drugiej, do przebadania widoku dashboard oraz widoków list sprzedawców wykorzystano serwis *WebPageTest.org*. Widoki list sprzedawców i sprzedanych biletów zostały przebadane dla 100, 400 i 800 wyświetlanych elementów. Uzyskano je poprzez zmianę parametru *limit* w URL.

6.1. Badania z wykorzystaniem browser-perf i perfjankie

Badania te zostały przeprowadzone na laptopie, tablecie oraz smartfonie. Na wszystkich urządzeniach zainstalowana była przeglądarka Google Chrome 40.0.2214.111. Parametry techniczne każdego z urządzeń zostały wyspecyfikowane w tabeli 3.

Tabela 3. Specyfikacja techniczna urządzeń, na których wykonano badania.

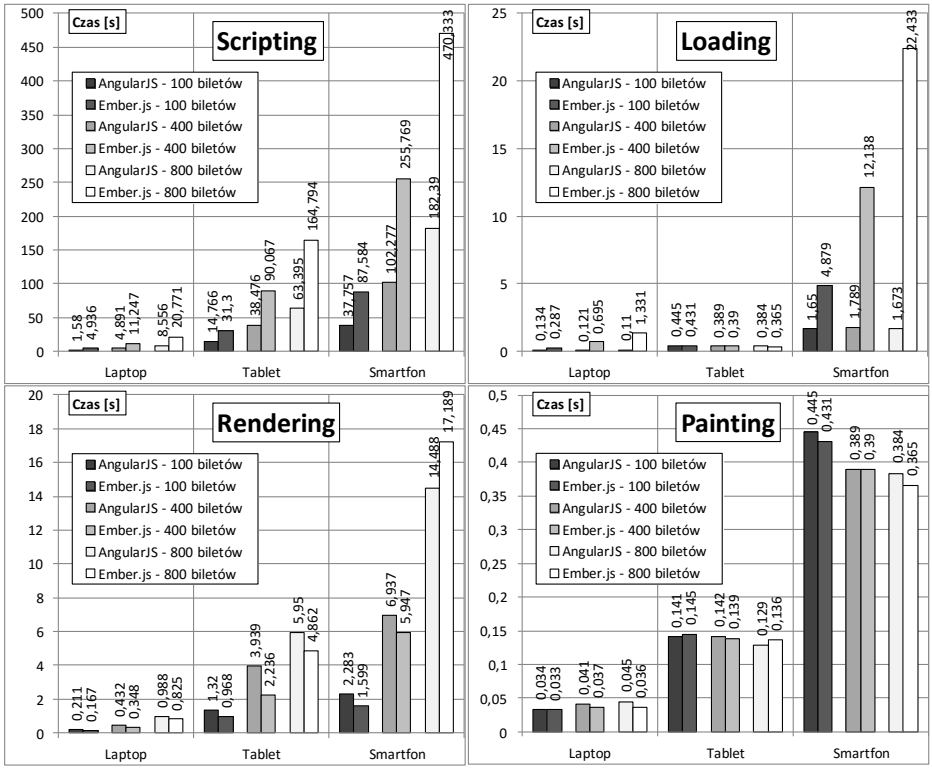
	Komputer	**Tablet**	**Smartfon**
Marka, model	MacBook Pro 11,2 (Retina, 15 calowy, koniec 2013 r.)	Asus Nexus 7 (1 generacja)	Huawei Ascend P6
Procesor	Intel Core i7 2.0 GHz	Quad-core 1,3 GHz nVidia Tegra 3	Quad-core 1,5 GHz Cortex A9
Pamięć	8 GB 1600 MHz DDR3	1 GB	2 GB
Karta graficzna	Intel Iris Pro 1536 MB	Twelve-core 416 MHz	480 MHz Vivante GC4000
Dysk	Apple SSD 256 GB	8 GB	8 GB
System operacyjny	OS X 10.10.2	Android 5.0.2	Android 4.4.2

W celu zautomatyzowania badań, korzystając z biblioteki WD.js utworzono skrypt sterujący przeglądarką internetową. Skrypt ten wypełniał pole tekstowe w widoku listy sprzedanych biletów, co powodowało natychmiastowe przefiltrowanie tej listy. Dzięki takiemu rozwiązaniu skupiono się na metrykach związanych przede wszystkim z wykonywaniem się kodu JavaScript.

Tabela 4. Liczba wygenerowanych elementów DOM dla widoku listy sprzedanych biletów w zależności od liczby wyświetlanych sprzedanych biletów

Długość listy biletów	100		400		800	
Framework	AngularJS	Ember.js	AngularJS	Ember.js	AngularJS	Ember.js
Liczba elementów DOM	2060	1438	7760	5338	15360	10538

Na wstępie obliczono liczbę generowanych węzłów w drzewie DOM dla widoku listy sprzedanych biletów wyświetlającej 100, 400 i 800 sprzedanych biletów (tabela 4). Następnie wykonano badania dla listy sprzedanych biletów. Każdy pomiar został powtórzony dziesięć razy, po czym wyznaczono ich wartość przeciętną. Na rysunku 4. przedstawiono uzyskane wyniki w formie wykresów.



Rysunek 4. Wyniki pomiarów grupowych metryk dla list o długości 100, 400 i 800 biletów

Od razu rzuca się w oczy dwu, a nawet trzykrotnie dłuższy czas wymagany na wykonanie kodu JavaScript i odmiecenie pamięci (*Scripting*) aplikacji Ember.js w stosunku do czasu uzyskanego przez aplikację AngularJS. Duże różnice pomiędzy uzyskanymi wartościami występują również w przypadku grupy metryk Loading.

Wnioski

Aplikacja skonstruowana przy pomocy frameworka AngularJS jest ponad dwukrotnie szybsza podczas wykonywaniu kodu JavaScript. Należy jednak zwrócić uwagę, że AngularJS w większości przypadków był wolniejszy w kwestii grupy metryk Rendering wywołując częściej reflow. Najprawdopodobniej ma to związek z implementacją Run Loop w Ember.js, który „zarządza" kolejnością wykonywania wewnętrznych procesów Ember.js, co w rzeczywistości chroni przed zbyt wczesnym przekazywaniem danych do widoku i zmienianiem modelu DOM.

Ember.js wykorzystuje Handlebars.js – bibliotekę, która odpowiedzialna jest za zamianę łańcuchów znaków w elementy DOM. Widać tutaj ogromną różnicę pomiędzy podejściem „angularowym", które operuje bezpośrednio na elementach DOM klonując

utworzony wcześniej fragment DOM, a podejściem opartym o generowanie kodu HTML i późniejszą transformację tego kodu na fragment DOM.

Grupa Loading, do której należy metryka ParseHTML, odzwierciedla czas spędzony na parsowaniu i tworzeniu modelu DOM przez przeglądarkę internetową. Na widoku listy sprzedanych biletów, podczas wyświetlania 800 elementów, AngularJS był trzynaście razy szybszy w generowaniu fragmentów DOM. Aplikacja uruchomiona na smartfonie, w przypadku Ember.js potrzebowała aż 22 sekund, gdzie czas spędzony na tym zadaniu przez AngularJS wyniósł 1,5 sekundy.

6.2. Badania z wykorzystaniem serwisu WebPageTest.org

W tabeli 5. zebrano ustawienia, w oparciu o które zostały przeprowadzone badania wydajności z wykorzystaniem serwisu *WebPageTest.org*. Nie wykazano tam ustawień, które pozostawiono w stanie domyślnym.

Tabela 5. Zmiany ustawień w serwisie WebPageTest.org podczas prowadzonych badań

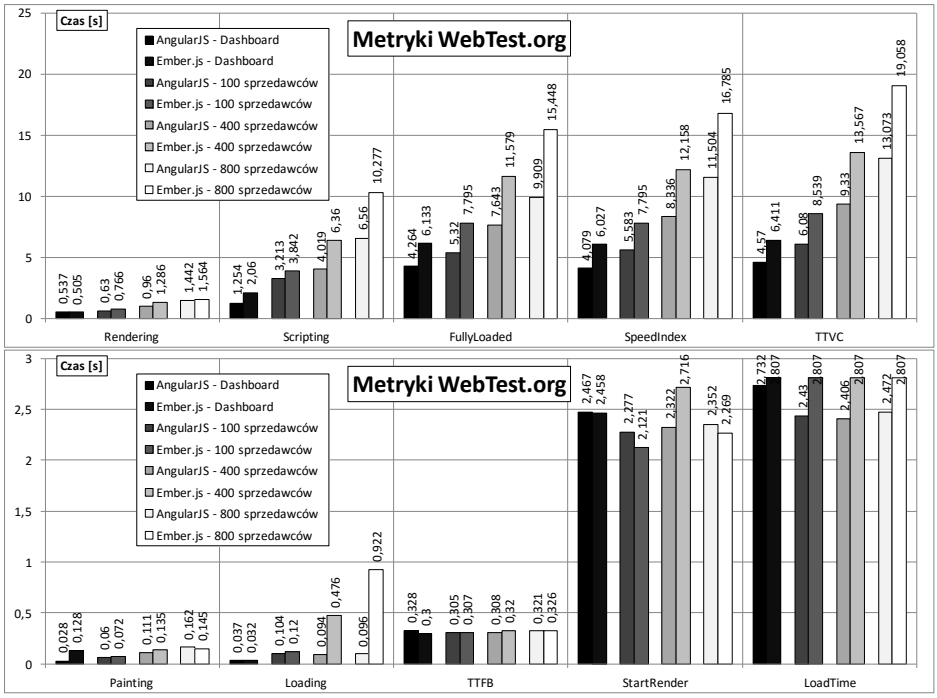
WebPageTest	Test location	Europe – Manchester, UK (IE 11, Chrome, Firefox)
	Browser	Chrome
Test settings	Connection	Cable (5/1 Mbps 28ms RTT)
	Number of tests to run	9
Advanced	Minimum test duration (seconds)	20
Chrome	Capture DevTools Timeline	Yes
	Capture Chrome Trace (about://tracing)	Yes
	Capture Network Log	Yes

Badania dotyczyły aplikacji uruchomionych w ramach *chmurowej* usługi firmy Heroku. Serwis *WebPageTest.org* umożliwia pomiar czasu wymaganego do załadowania całej aplikacji, tak więc rozmiar aplikacji skonstruowanych z użyciem AngularJS i Ember.js miał znaczenie. Kod aplikacji został zbudowany do pojedynczego pliku JavaScript i skompresowany z użyciem narzędzia UglifyJS. Rozmiary poszczególnych plików zostały zestawione w tabeli 6.

Tabela 6. Rozmiary plików z kompletnym kodem obu aplikacji

	Bez kompresji		**Gzip**	
Framework	AngularJS	Ember.js	AngularJS	Ember.js
Rozmiar pliku [kB]	494	840	149	213

Każdy pomiar został powtórzony dziewięć razy, po czym wyznaczono ich wartość przeciętną. Wyniki pomiarów przedstawiono w formie wykresów na rysunku 5.



Rysunek 5. Wyniki pomiarów metryk WebPageTest.org dla dashbord i list o długości 100, 400 i 800 sprzedawców

Największe różnice pomiędzy frameworkami wykazują grupy metryk: Scripting i Loading w przypadku listy 400 i 800 sprzedawców. Podobnie, jak w przypadku poprzednich badań wykonanych narzędziami *browser-perf* i *perfjankie*, również i tym razem Ember.js przetwarza kod HTML blisko dziesięciokrotnie dłużej niż AngularJS oraz blisko półtorakrotnie dłużej trwa wykonywanie kodu JavaScript.

Na wykresie zaobserwowano również (FullyLoaded), że AngularJS jest w stanie załadować widok listy sprzedawców z dwukrotnie większą liczbą osób do wyświetlenia, w czasie krótszym, niż Ember.js potrzebuje na wyświetlenie dwukrotnie krótszej listy.

W przypadku mniej skomplikowanych widoków, takich jak dashboard, czy lista 100 sprzedawców, użytkownik jest zmuszony czekać przynajmniej o 2 s dłużej na załadowanie się aplikacji utworzonej w oparciu o Ember.js.

Wnioski

Umieszczenie aplikacji webowych na dostępnych przez sieć, profesjonalnych serwerach WWW, pozwoliło na zbadanie wpływu rozmiaru kodu JavaScript aplikacji webowej na czas ładowania sie tej aplikacji. Różnica w rozmiarze plików po wykonaniu kompresji gzip wyniosła 64,45 kB, co przełożyło się sie na dłuższe ładowanie całej aplikacji o 400 ms (metryka Load time) w przypadku Ember.js. Wykazano w ten sposób, że rozmiar frameworka lub biblioteki nie ma większego znaczenia w przypadku aplikacji webowych typu SPA, ponieważ prawdziwym wąskim gardłem są operacje na drzewie DOM i szybkość wykonywania kodu JavaScript.

7. Podsumowanie

Analiza wydajności aplikacji webowych, to szeroki obszar, który dotyczy zarówno wydajności po stronie serwera jak i klienta. Autorzy skupili się tylko na wydajności po stronie klienta. Na uwagę zasługuje fakt, że coraz więcej elementów specyfikacji W3C Performance Timing API jest implementowanych przez producentów przeglądarek internetowych, co pozwala dokładniejsze wykonywanie pomiarów.

Celem autorów była analiza wydajności aplikacji webowych typu SPA skonstruowanych przy użyciu frameworków JavaScript. Na podstawie otrzymanych wyników badań stwierdzono, że aplikacja utworzona przy użyciu AngularJS była dwukrotnie szybsza pod względem wykonywania kodu JavaScript, czy przetwarzania widoków, które zawierały wiele elementów, w stosunku do Ember.js. Różnica była jeszcze bardziej widoczna w przypadku urządzeń mobilnych, gdzie czasy wykonywania kodu JavaScript wzrosły do nieakceptowanych rozmiarów. Należy mieć świadomość, że na podstawie przeprowadzonych badań nie można założyć, że w każdym przypadku AngularJS będzie szybszy niż Ember.js. Badania pokazały raczej, że bardziej wydajne jest operowanie na elementach DOM, niż na łańcuchach znaków, na podstawie których tworzone są nowe elementy DOM.

Twórcy Ember.js pracują intensywnie nad nowym silnikiem generowania szablonów HTML o nazwie HTMLBars, który w przyszłości zastąpi używany obecnie Handlebars.js. HTMLBars opiera swoje działanie na koncepcji wirtualnego modelu DOM (ang. *virtual DOM*), czyli struktury danych odzwierciedlającej rzeczywiste drzewo DOM. Pomysłodawcą wirtualnego modelu DOM jest Facebook, który utworzył bibliotekę JavaScript o nazwie *React* realizującą tę koncepcję. Ponieważ biblioteka *React* została udostępniona publicznie, istnieje możliwość jej integracji z frameworkiem AngularJS [1], gdyż oba rozwiązania operują bezpośrednio na modelu DOM. Zatem w momencie wydania nowej wersji Ember.js [6] wraz z silnikiem HTMLBars, można byłoby przeprowadzić ponowne badania obu frameworków, już z nową warstwą widoków operujących na koncepcji wirtualnego modelu DOM.

Literatura

[1] Bevacqua N., AngularJS' Internals In Depth, 2015. http://www.smashingmagazine.com/2015/01/22/angularjs-internals-in-depth/
[2] Fielding R. T., Taylor R. N., Principled Design of the Modern Web Architecture, ACM Transactions on Internet Technology, Vol. 2, No. 2, 2002. http://www.ics.uci.edu/~taylor/documents/2002-REST-TOIT.pdf
[3] Google, Chrome Remote Debugging Protocol, 2015. https://developer.chrome.com/devtools/docs/debugger-protocol
[4] Google, Performance profiling with the Timeline, 2015. https://developer.chrome.com/devtools/docs/timeline
[5] Grigorik I., Krytyczna ścieżka renderowania, 2015. https://developers.google.com/web/fundamentals/performance/critical-rendering-path/?hl=pl
[6] Haagen Skeie J., Ember.js in Action, Manning Publications Co., 2013.
[7] Irish P., Delivering the goods, Fluent 2014 Keynote, O'Reilly, 2014. https://www.youtube.com/watch?v=R8W_6xWphtw
[8] McNeil M., Sails.js, 2012. http://sailsjs.org/
[9] Mozilla Developer Network, Performance Timing, 2015. https://developer.mozilla.org/en-US/docs/Web/API/PerformanceTiming
[10] Narasimhan P., Making Frontend Performance Testing a Part of Continuous Integration – PerfJankie, Velocity, 2014. http://velocityconf.com/velocity2014/public/schedule/detail/33676
[11] Narasimhan P., Metric, broser-perf, 2014. https://github.com/axemclion/browser-perf/wiki/Metrics

[12] World Wide Web Consortium, Box dimensions, 2005. http://www.w3.org/TR/CSS2/box.html #box-dimensions
[13] World Wide Web Consortium, Descriptions of all CSS specifications, 2015. http://www.w3.org/Style/CSS/specs
[14] World Wide Web Consortium, HTML 4.01 Specification, 1999. http://www.w3.org/TR/html401/
[15] World Wide Web Consortium, Navigation Timing – W3C Recommendation, 2012. http://www.w3.org/TR/ navigation-timing/
[16] World Wide Web Consortium, Web performance working group, 2010. http://www.w3.org/2010/webperf/

Autorzy i afiliacje

Jakub Swacha** – **Rozdział 1
Wydział Nauk Ekonomicznych i Zarządzania, Instytut Informatyki w Zarządzaniu, Uniwersytet Szczeciński, Szczecin, Polska, jakubs@uoo.univ.szczecin.pl

Tomasz Protasowicki**– **Rozdział 2
Wydział Cybernetyki, Wojskowa Akademia Techniczna, Warszawa, Polska, tprotasowicki@wat.edu.pl

Jerzy Stanik** – **Rozdział 2,3
Wydział Cybernetyki, Wojskowa Akademia Techniczna, Warszawa, Polska, jstanik@wat.edu.pl

Jarosław Napiórkowski** – **Rozdział 3
Wojskowa Akademia Techniczna, Wydział Cybernetyki, jnapiorkowski@wat.edu.pl

Bogusz Jeliński** – **Rozdział 4
bogusz.jelinski@gmail.com

Rafał Wojszczyk** – **Rozdział 5
Politechnika Koszalińska, Koszalin, Polska, rafal.wojszczyk@tu.koszalin.pl

Lech Tuzinkiewicz** – **Rozdział 6
Wydział Informatyki i Zarządzania, Politechnika Wrocławska, Wrocław, Polska, lech.tuzinkiewicz@pwr.edu.pl

Emila Zakrawacz** – **Rozdział 6
Wydział Informatyki i Zarządzania, Politechnika Wrocławska, Wrocław, Polska, 183660@student.pwr.edu.pl

Bożena Śmiałkowska** – **Rozdział 7
Wydział Informatyki, Zachodniopomorski Uniwersytet Technologiczny w Szczecinie, Szczecin, Polska, bsmialkowska@wi.zut.edu.pl

Monika Łobaziewicz** – **Rozdział 8
OPTeam S.A., mlobaziewicz@opteam.pl

Lucjan Pelc** – **Rozdział 9
Państwowa Wyższa Szkoła Techniczno-Ekonomiczna w Jarosławiu, Jarosław, Polska, lucjan.pelc@pwste.edu.pl

Dawid Karabin – Rozdział 10
Wydział Informatyki i Zarządzania, Politechnika Wrocławska, Wrocław, Polska, mail@hinok.net

Ziemowit Nowak – Rozdział 10
Wydział Informatyki i Zarządzania, Politechnika Wrocławska, Wrocław, Polska, ziemowit.nowak@pwr.edu.pl