Inżynieria oprogramowania

Badania i praktyka

Redakcja naukowa

Lech Madeyski
Mirosław Ochodek

Konferencje naukowe organizowane przez
Polskie Towarzystwo Informatyczne:

IX edycja Sejmiku Młodych Informatyków
XVI edycja Krajowej Konferencji Inżynierii Oprogramowania
XXI edycja konferencji „Systemy Czasu Rzeczywistego”

oraz współorganizowana przez Oddział Wielkopolski PTI

XVIII edycja konferencji „Przetwarzanie Sygnałów”

zostały dofinansowane
przez Ministra Nauki i Szkolnictwa Wyższego
w ramach programu związanego z realizacją zadań upowszechniających naukę
(decyzja nr 1187/P-DUN/2014 z dnia 7 lipca 2014 roku)

Dziękujemy!

Polskie Towarzystwo Informatyczne

Inżynieria oprogramowania

Badania i praktyka

Redakcja naukowa

Lech Madeyski
Mirosław Ochodek

Poznań-Warszawa 2014

WYDAWNICTWO
Nakom ● Poznań

Autorzy

Artur Kasprzyk, Anita Walkowiak – ***ROZDZIAŁ 1***
Tomasz Protasowicki – ***ROZDZIAŁ 2***
Romuald Hoffmann, Tomasz Protasowicki – ***ROZDZIAŁ 3***
Alicja Ciemniewska, Paweł Kędziora, Marek Lewandowski, Cezary Mazurek, Marcin Wolski – ***ROZDZIAŁ 4***
Walery Susłow, Jacek Kowalczyk, Marta Boińska, Janina Nowak, Michał Statkiewicz – ***ROZDZIAŁ 5***
Włodzimierz Dąbrowski, Andrzej Stasiak – ***ROZDZIAŁ 6***
Michał Pawłowski, Ziemowit Nowak – ***ROZDZIAŁ 7***
Tomasz Protasowicki, Jerzy Stanik – ***ROZDZIAŁ 8***
Tomasz Sitek, Michał Litka – ***ROZDZIAŁ 9***
Rafał Wojszczyk – ***ROZDZIAŁ 10***

Recenzenci

Bartosz Alchimowicz, Piotr Czapiewski, Włodzimierz Dąbrowski, Iwona Dubielewicz, Zbigniew Fryźlewicz, Janusz Górski, Marek Grobelny, Bogumiła Hnatkowska, Jarosław Hryszko, Sylwia Kopczyńska, Michał Maćkowiak, Marcin Mastalerz, Piotr Miklosik, Jan Mikosiński, Mirosław Ochodek, Cezary Orłowski, Jakub Rojek, Lech Tuzinkiewicz, Bartosz Walter, Jolanta Wrzuszczak-Noga, Piotr Zaremba, Janusz Żmudziński

Redakcja naukowa

Lech Madeyski
Mirosław Ochodek

ISBN 978-83-63919-15-3

Wydanie I
Wydawnictwo NAKOM, ul. Starołęcka 18A, lok. 303, 61-361 Poznań

Spis treści

Wstęp

Inżynieria oprogramowania od początku swojego istnienia nieodzownie łączy ze sobą świat nauki i praktyki. Samo określenie zostało użyte po raz pierwszy w latach 60-tych ubiegłego stulecia w sposób nieco przewrotny. Kontrastowało bowiem z obserwowaną ówcześnie niską dojrzałością procesów wytwarzania oprogramowania. Z drugiej strony miało ono na celu zasugerowanie dalszego kierunku rozwoju dziedziny, stawiając za wzór inne, bardziej dojrzałe nauki inżynierskie. Warto przypomnieć, że w owym czasie branża IT zmagała się z licznymi problemami, określanymi często zbiorczo mianem kryzysu oprogramowania. Taki stan rzeczy wynikał przede wszystkim z dynamicznie rosnącego zainteresowania produktami informatycznymi oraz postępującym wzrostem ich skomplikowania. W rezultacie czego wiele przedsięwzięć kończyło się fiaskiem, przekraczając znacząco założone ramy czasowe i finansowe.

Nadzieje pokładane w nurcie inżynierii oprogramowania wiązały się przede wszystkim z usystematyzowaniem procesu wytwarzania oprogramowania oraz dostarczeniem narzędzi wspomagających zespoły projektów. Od tego czasu możemy zaobserwować nieustający rozwój metod i narzędzi, jak również postęp prac badawczych w zakresie inżynierii oprogramowania. Z zadowoleniem dostrzegamy też zacieśnianie się współpracy pomiędzy środowiskami naukowymi oraz firmami z branży IT.

W niniejszej monografii zebraliśmy prace, których autorami są reprezentanci jednostek badawczych oraz praktycy zajmujący się zawodowo wytwarzaniem oprogramowania. Poruszają oni szerokie spektrum problemów obejmujących swoim zasięgiem różne etapy rozwoju produktu informatycznego. Z uwagi na to monografia została podzielona na cztery części, których tematyka postępuje zgodnie z umownym cyklem życia projektu i produktu informatycznego.

W pierwszej części zebrano opracowania omawiające zagadnienia z zakresu inżynierii wymagań. Rozdział 1, którego autorzy reprezentujący Politechnikę Wrocławską oraz firmę AION podejmują próbę zobrazowania korzyści wynikających ze stosowania standardów oraz kompleksowych procesów wytwórczych w analizie biznesowej oraz systemowej. W rozdziałach 2 i 3 badacze z Wojskowej Akademii Technicznej dzielą się swoim doświadczeniem i przemyśleniami związanymi z rozwojem systemów informatycznych. W rozdziale 2 poruszają oni problematykę budowy systemów teleinformatycznych dla Systemów Bezpieczeństwa Narodowego, poświęcając przy tym wiele uwagi aspektom prawnym funkcjonowania tego typu systemów. Natomiast w kolejnym rozdziale podejmują oni próbę oceny znaczenia identyfikacji wymagań dla realizacji systemów teleinformatycznych, dedykowanych wspomaganiu funkcjonowania

organizacji o charakterze federacyjnym. W rozdziale 4 pracownicy Poznańskiego Centrum Superkomputerowo - Sieciowego dzielą się swoimi doświadczeniami dotyczącymi roli inżynierii wymagań w modernizacji systemów informatycznych wspierających oświatę w Polsce.

Druga część monografii poświęcona została aspektom projektowania systemów informatycznych. Rozpoczynający ją rozdział 5 poświęcono sylwetce zawodowej projektanta oprogramowania. Zespół naukowców z Uniwersytetu Gdańskiego oraz Politechniki Koszalińskiej przedstawia wyniki swoich badań, dotyczących związków pomiędzy cechami osobowości a predyspozycjami do pracy w zawodzie projektanta systemów informatycznych. Rozdział 6 poświęcony został projektowaniu systemów sterowania ruchem. Autorzy z Wojskowej Akademii Technicznej oraz Politechniki Warszawskiej prezentują język dziedzinowy TransML służący do modelowania systemów kontroli i nadzorowania ruchem oraz prezentują rozwijane przez siebie środowisko TransCAD. W rozdziale 7 autorzy z Politechniki Wrocławskiej oraz z firmy SI2 przedstawiają platformę programistyczną (ang. *framework*), służącą do tworzenia wysokowydajnych aplikacji internetowych w języku PHP.

Trzecia część monografii dotyczy szeroko rozumianych aspektów zarządzania jakością procesu wytwarzania oprogramowania. W rozdziale 8 zaprezentowano symulacyjne podejście do badania jakości oprogramowania w zwinnym procesie produkcji. Natomiast w rozdziale 9 autorzy reprezentujący Politechnikę Gdańską oraz firmę Avena Technologie przedstawili swoje doświadczenia związane z pomiarem i oceną skutków wdrożenia narzędzi Business Intelligence w organizacji wsparcia informatycznego.

W czwartej i ostatniej części monografii zaprezentowano pracę poświęconą problemowi utrzymaniu systemów informatycznych, w której omówiono możliwości pozyskiwania struktury obiektowej z kodu oprogramowania przy wykorzystaniu metod inżynierii odwrotnej.

Życząc czytelnikom interesującej i wartościowej lektury, chcielibyśmy podziękować wszystkim osobom zaangażowanym w przygotowanie niniejszej monografii. W pierwszej kolejności podziękowania kierujemy do autorów rozdziałów, którzy zechcieli podzielić się wynikami swoich badań, przemyśleniami i obserwacjami procesu wytwarzania oprogramowania. Równie gorąco pragniemy podziękować zespołowi recenzentów za ich zaangażowanie i wkład w doskonalenie zebranych prac.

Lech Madeyski
Mirosław Ochodek

CZĘŚĆ I
INŻYNIERIA WYMAGAŃ

Rozdział 1

Biznesowe korzyści ze stosowania standardów oraz kompleksowych procesów wytwórczych w analizie biznesowej oraz systemowej

Obserwując przyrost standardów w obszarze inżynierii oprogramowania oraz architektury biznesowej i jednocześnie znikomą ich adaptację przemysłową w naturalny sposób pojawia się wątpliwość odnośnie ich użyteczności oraz możliwości przełożenia na rzeczywiste wartości biznesowe. Bez wskazania takowych, z dużym prawdopodobieństwem można przyjąć iż standardy te staną się tylko ciekawostką, a rozdźwięk pomiędzy działaniami instytucji standaryzujących a codziennością biznesową będzie się powiększał.
Celem niniejszego rozdziału jest zaprezentowanie wybranych standardów przez pryzmat spójnego, analitycznego procesu produkcyjnego. Bazą dla rozważań będzie metodyka firmy AION, wypracowywana od 1994 roku i stosowana zarówno w ramach projektów realizowanych przez AION jak i firmy, które w ramach współpracy oparły swoje procesy produkcyjne o omawianą metodykę.

Dynamiczny przyrost liczby nowych standardów umożliwiających specyfikowanie artefaktów architektury biznesowej oraz rozwiązań informatycznych skłania do refleksji odnośnie obserwowanej, nikłej adaptacji w środowiskach przemysłowych nawet tych standardów, które od lat są dostępne i rozwijane. Artykułowane argumenty odnośnie ich mnogości, złożoności, długiego procesu poznawania, braku wystarczającego wsparcia narzędziowego z pewnością odgrywają znaczenie, aczkolwiek w konfrontacji z powszechnym stosowaniem nietrywialnych środowisk wytwórczych tracą na sile.

Obserwując próby wdrożeń niektórych ze standardów – w szczególności *Unified Modeling Language* [OMG13c] i *Business Process Model and Notation* [OMG13a] – z perspektywy firmy doradczej i szkoleniowej często zauważalnymi są podnoszone obawy o:

- biznesową zasadność stosowania tak złożonych metod specyfikowania wiedzy,
- spadek wydajności zespołów analitycznych przez długi czas wdrażania nowych metod pracy,
- nieczytelność wiedzy zapisanej w dedykowanych, specjalizowanych językach dla osób nietechnicznych,
- skomplikowanie modeli i w związku z tym obawa przed koniecznością długotrwałego wdrażania projektantów, programistów, architektów w specyfikę zapisu efektów analizy biznesowej i systemowej,
- koszt utrzymania kompleksowej i szczegółowej dokumentacji.

Na powyższe obawy nakładają się dodatkowo doświadczenia – często negatywne – z wcześniejszych prób wdrażania standardów notacyjnych.

Czynnikiem mogącym zmniejszyć siłę obaw jest posłużenie się pragmatycznym, spójnym, kompleksowym procesem produkcyjnym, w którym poszczególne standardy mają swoje miejsce, jako techniki służące do: uzyskiwania wiedzy, jej analizy, opracowywania i komunikowania proponowanych rozwiązań, dokumentacji stanowiącej zarówno wsad do dalszych prac w ramach procesu produkcyjnego jak i, w przyszłości, podstawę do zarzadzania zmianą.

W dalszej części rozdziału zaprezentowane zostaną wybrane techniki tworzenia architektury biznesowej oraz modeli analitycznych rozwiązań informatycznych, ujęte w formie zstępującego procesu wytwórczego, prowadzącego od opisu przedsiębiorstwa, na opisie analitycznym systemu informatycznego zakończywszy. Szczególny nacisk zostanie położony na uzasadnienie wartości stosowania poszczególnych technik.

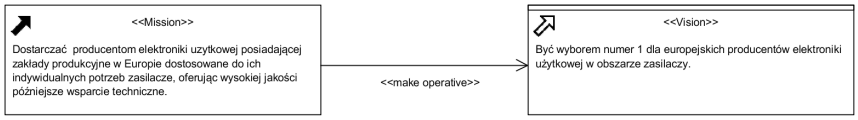
1.1 Strategia biznesu

Zmienność otoczenia, zmienność trendów rynkowych, zmniejszająca się lojalność klientów powodują, iż coraz większego znaczenia nabiera zdolność organizacji do szybkiego dostosowywania się do nowych warunków, w jakich funkcjonuje. Szybko, a jednocześnie świadomie i optymalnie skorygowany kierunek działania organizacji wymaga od osób zarządzających stałego dostępu do aktualnego modelu biznesu. W przypadku niedużych podmiotów gospodarczych, gdzie pryncypia biznesu najczęściej rezydują w głowach właścicieli, korekt modelu biznesu można dokonać mniej formalnie – bazując na ich znajomości rynku oraz wiedzy o każdym aspekcie funkcjonowania firmy. W średnich, a tym bardziej, dużych przedsiębiorstwach, w przypadku których mnogość zależności łączących różne aspekty planu biznesowego, opracowywanego często w sposób zstępujący, na różnych szczeblach zarządzania organizacją, przekierowanie ich na nowe tory wymaga kompleksowej analizy wpływu nowych czynników na aktualny stan przedsiębiorstwa oraz wypracowania optymalnego z punktu widzenia całości organizacji nowego planu biznesu, którego zapis będzie stanowił podstawę do przyszłych zmian. Nakładając na aspekt analityczny, aspekt zmienności kadry zarządzającej, posługującej się często autorskimi technikami zarządczymi, łatwo dojść do wniosku, iż zdolność organizacji do szybkiej adaptacji w nowym otoczeniu gospodarczym nie jest łatwa do uzyskania.

Pozytywnie na zdolność adaptacyjną organizacji wpłynąć może wdrożenie i konsekwentne stosowanie na wszystkich szczeblach zarządczych standardu

Business Motivation Model [OMG14b], firmowanego przez *Object Management Group* (OMG). Dzięki niemu organizacja „zmuszona" jest do ujednolicenia ontologii stosowanej do wypracowywania planów biznesowych, a zakres informacji wymaganych do skonstruowania poprawnego planu jest jasno określony. Tym samym, poprawie ulega spójność i kompletność opracowywanego planu, ułatwiona jest walidacja proponowanych rozwiązań oraz weryfikacja zgodności planów biznesowych jednostek podrzędnych z planami jednostek nadrzędnych.

Rysunek 1.1 przedstawia przykład najwyższego z punktu widzenia BMM poziomu planu biznesowego, czyli wizję i misję, wraz z łączącą je relacją. Rodzaje relacji przewidzianych do wiązania ze sobą wystąpień poszczególnych rodzajów elementów, stanowią integralną składową BMM.



Rysunek 1.1. Przykład wizji i misji[1].

Zgodnie z zaleceniami BMM, wizja, rozumiana jako cel nadrzędny organizacji, powinna zostać zdekomponowana na cele cząstkowe, nazywane w nomenklaturze standardu celami strategicznymi[2] (ang. *goal*) – Rysunek 1.2 – które na dalszym etapie tworzenia modelu powinny zostać rozbudowane o cele taktyczne (ang. *objective*) – Rysunek 1.3. Cele taktyczne, które powinny spełniać cechy SMART, można traktować jako miarę osiągniecia celów strategicznych.

Warto zwrócić uwagę na pomoc, jaką oferuje BMM[3] w aspekcie facylitacji procesu tworzenia planu biznesowego. Poprzez dostarczenie dobrze opisanego metamodelu, zawierającego, w większości przypadków, praktyczne definicje pojęć, osoba odpowiedzialna za prowadzenie działań mających na celu stworzenie planu biznesowego jest kierowana w swoich działaniach charakterem

[1] Standard BMM nie definiuje notacji dla ustanowionych pojęć. Zaprezentowane diagramy zostały przygotowane w narzędziu Visual Paradigm Enterprise Edition [VPEE] z wykorzystaniem opracowanej przez jego producenta notacji.

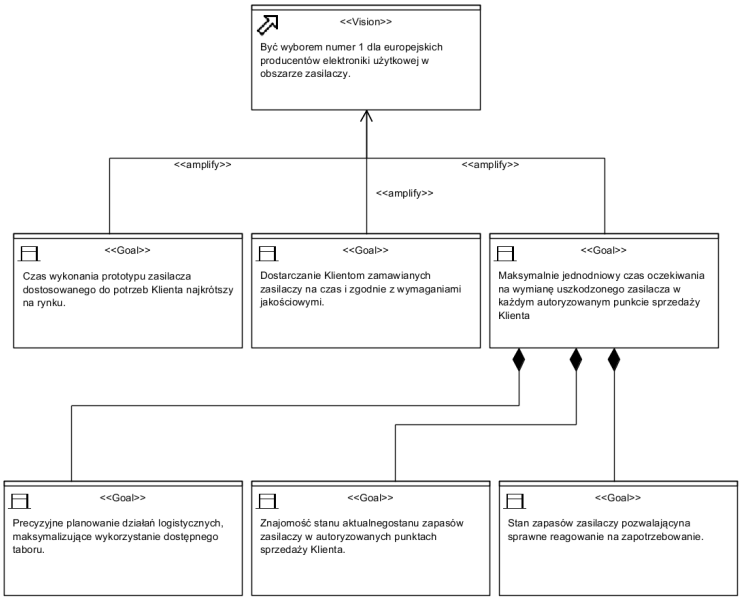
[2] W dokumencie wykorzystywane są opracowane w firmie AION tłumaczenia pojęć standardu. Nie stanowią one jednakże standardu w związku z czym w dostępnej literaturze można spotkać inne tłumaczenia oryginalnych pojęć.

[3] Uwaga ta dotyczy wszystkich, dedykowanych języków modelowania o właściwie określonym metamodelu.

opisanych relacji, z których, do pewnego stopnia, można wyprowadzić plan prac w ramach przedsięwzięcia.

Określone w planie biznesowym cele, wymagają zdefiniowania środków ich osiągania. Na najwyższym poziomie ogólności, misja organizacji prezentuje pomysł na dojście do celu określonego jej wizją. Stosując metodę zstępującą, zgodnie z BMM, celom strategicznym należy przyporządkować strategie postępowania (ang, *strategy*), natomiast celom taktycznym – taktyki (ang. *tactic*).

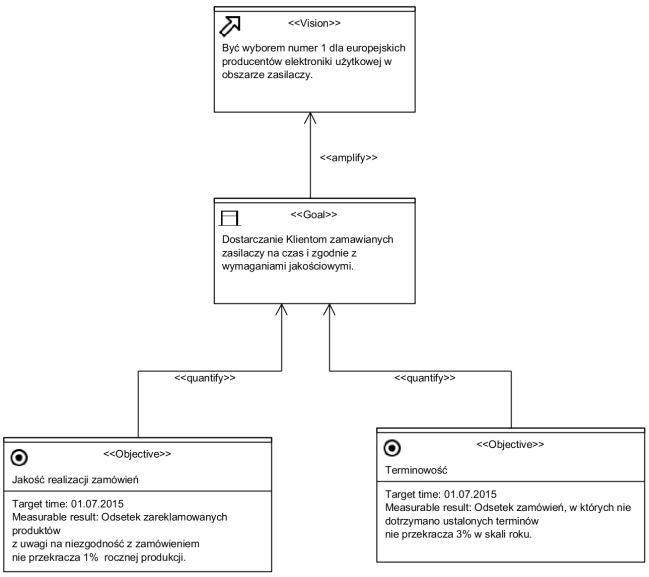
BMM pozostawia autorom planów biznesowych sporą dowolność łączenia poszczególnych odpowiedników celów oraz środków ich osiągania, określając liczność typów relacji jako wiele-do-wielu. Ponadto, poprzez wprowadzenie relacji dekompozycji celów strategicznych i taktycznych, BMM umożliwia kalibrowanie poziomu szczegółowości definicji celów do potrzeb organizacji (między innymi w aspekcie komunikacji celów, przekładania celów organizacji na cele procesów biznesowych czy jednostek organizacyjnych).



Rysunek 1.2. Przykład celów strategicznych.

Podobną elastyczność przewidziano w obszarze definiowania czynników wpływających na postać planów biznesowych. Dokonawszy ogólnego podziału na czynniki wewnętrzne i zewnętrzne, BMM nie precyzuje źródła ich pochodzenia czy też technik, jakimi powinno się je pozyskiwać. Dzięki temu, bez

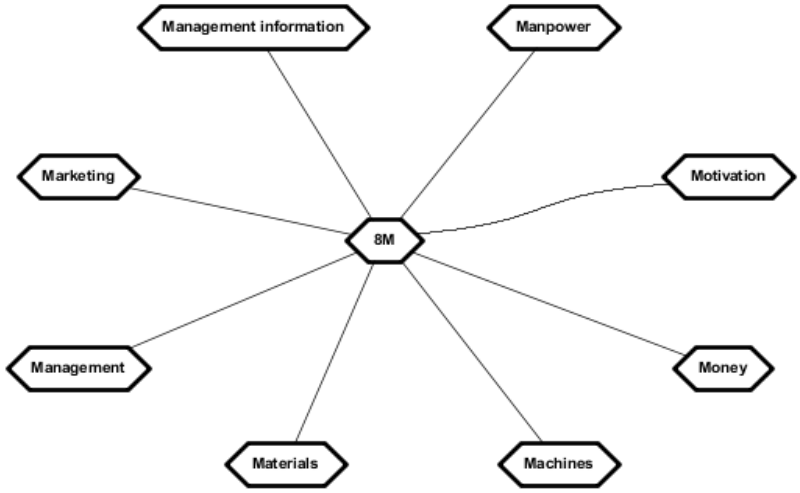
względu na stosowane metody dokonywania analiz wewnętrznych i zewnętrznych organizacji, teoretycznie[4] możliwe jest ich skojarzenie z modelem BMM. W firmie AION wykorzystywane jest narzędzie Visual Paradigm, w którym zaimplementowano między innymi BMM. Do wykonywania analiz marketingowych przystosowano Mapy Myśli (ang. *Mind Map*), konstruowane w sposób pokazany na rysunkach: Rysunek 1.4 oraz Rysunek 1.5.



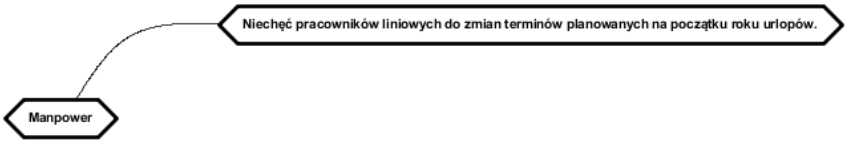
Rysunek 1.3. Przykład celów taktycznych.

Uzyskanie spójności modelu stało się możliwe dzięki mechanizmowi odpowiadającemu zależności «trace», udostępnianemu w narzędziu Visual Paradigm – Rysunek 1.6. Czynnik wynikający z analiz zapisanych w formie Mapy Myśli jest wiązany zależnością «trace» z odpowiednikiem na diagramie BMM. Mankamentem takiego rozwiązania jest redundancja zapisów tej samej informacji – mając jednakże na uwadze potencjalne ryzyko rozsynchronizowania modelu, wydaje się, iż takie podejście, przy stosowaniu odpowiedniej procedury zarządzania zmianą, jest wystarczająco użyteczne.

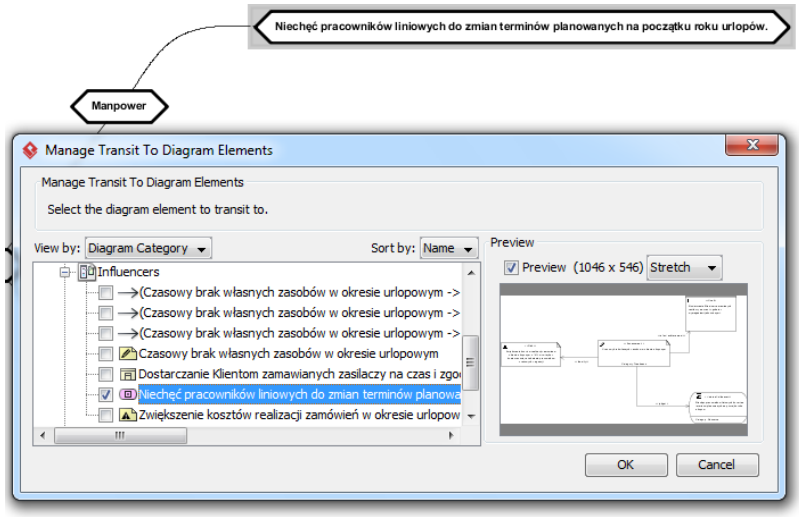
[4] Określenie „teoretycznie" zostało dodane z uwagi na fakt, iż autorom nie są znane narzędzia, które pozwalają na integrację typowych technik marketingowych z elementami BMM.



Rysunek 1.4. Analiza wewnętrzna 8M.



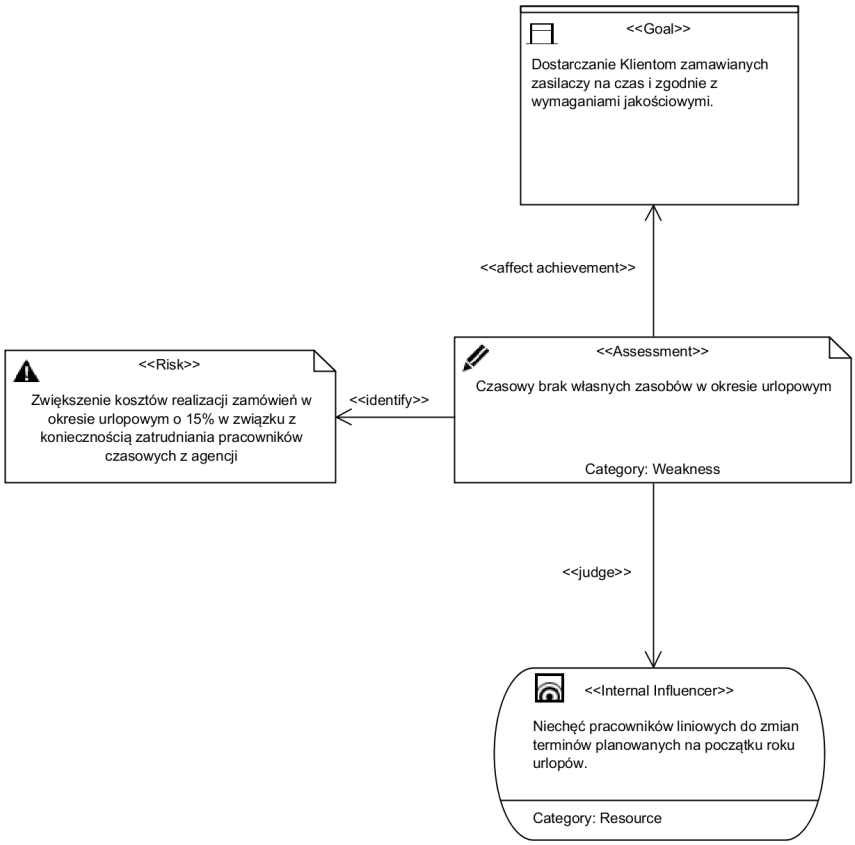
Rysunek 1.5. Czynnik wewnętrzny.



Rysunek 1.6. Mechanizm *Transit-To* narzędzia Visual Paradigm.

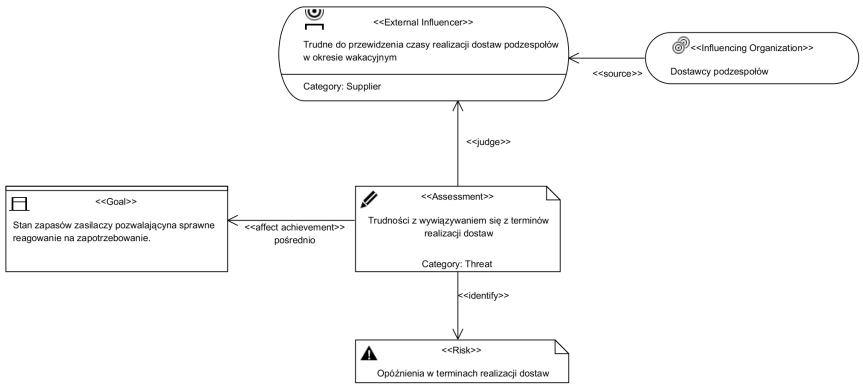
Identyfikację czynnika wpływającego należy, zgodnie z BMM, interpretować jako stwierdzenie faktu o neutralnym wydźwięku. Dopiero jego ocena (ang. *assesment*) wyraża wpływ czynnika na opracowywany plan biznesowy.

Rysunek 1.7 prezentuje przykład oceny wpływu czynnika wewnętrznego na plan biznesowy. Zaobserwowana niechęć pracowników liniowych do zmian terminów planowanych na początku roku w ocenie autorów planu przyczynia się do czasowego braku własnych zasobów produkcyjnych w okresie urlopowym, co należy uznać za słabą stronę aktualnego sposobu funkcjonowania firmy. W szczególnych sytuacjach, taki stan należy uznać za ryzykowny, gdyż może przyczynić się do 15% wzrostu kosztów realizacji zamówień, z uwagi na konieczność korzystania z pracowników tymczasowych, pozyskiwanych w ramach porozumień z agencjami pracy tymczasowej. Zidentyfikowane ryzyko stanowi wsad do całego obszaru zarządzania ryzykiem, nie wchodzącego w zakres BMM.



Rysunek 1.7. Wewnętrzny czynnik wpływający.

Warto w tym momencie ponownie zwrócić uwagę na elastyczność BMM. Z jednej strony, dość łatwo można było uzyskać integrację modelu BMM z, często postrzeganymi jako mało konkretne, analizami marketingowymi (zaprezentowany został jedynie przykład 8M; podobne zależności można określić dla analiz zewnętrznych – na przykład PEST, czy PESTEL). Z drugiej, wprowadzenie stopniowej oceny czynników, powoduje, iż osoby tworzące model muszą nawiązać do zidentyfikowanych neutralnych czynników, następnie dokonać oceny, by ostatecznie stwierdzić, czy z oceny wynika ryzyko, czy też potencjalna szansa dla organizacji. Warto zwrócić uwagę, iż w przypadku ocen czynników, BMM wyjątkowo wskazał rekomendowaną technikę – analizę SWOT. Na zamieszczonym niżej przykładzie, z uwagi na fakt oceny czynnika wewnętrznego, ocena wskazała na słabą stronę organizacji. W przypadku ocen czynników zewnętrznych, wskazywana jest albo szansa, albo zagrożenie – Rysunek 1.8.



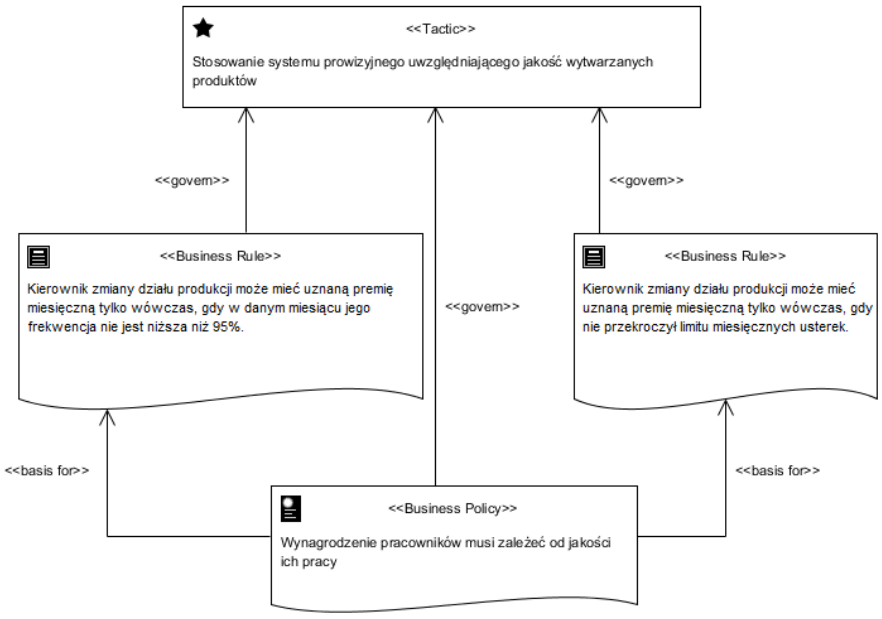
Rysunek 1.8. Zewnętrzny czynnik wpływający.

Kolejnym obszarem objętym standardem BMM są ogólnie pojęte dyrektywy (ang. *directive*), dzielące się na polityki (ang. *business policy*) oraz reguły biznesowe (ang. *business rule*). Dyrektywy, mają na celu zdefiniowanie zaleceń, ograniczeń, które należy brać pod uwagę przy opracowywaniu planów biznesowych. Pierwsza z wymienionych rodzajów dyrektyw – polityka, pozwala na określenie ogólnych zaleceń, których istnienie określa kierunek myślenia, aczkolwiek nie jest wystarczająca do regulowania bieżących działań organizacji. Doprecyzowaniem polityk na poziomie operacyjnym, są reguły biznesowe (Rysunek 1.9), pozwalające, z uwagi na swoją niepodzielność oraz precyzję sformułowania, na regulowanie działań na poziomie poszczególnych stanowisk

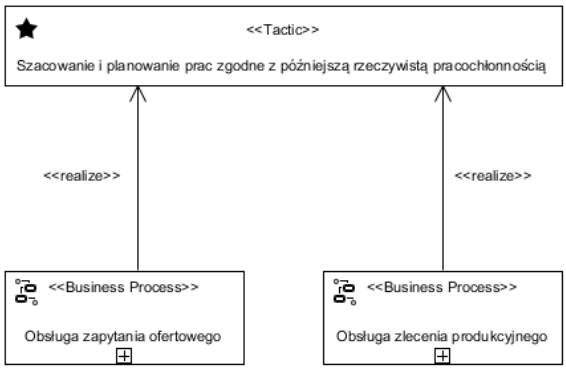
pracy oraz miejsc w procesach biznesowych, jeśli oczywiście takowe są zidentyfikowane i opisane. W modelu BMM, reguły biznesowe mogą być skojarzone ze strategiami i taktykami oraz procesami biznesowymi, które w myśl BMM, realizują strategiczne i taktyczne kierunki działań (Rysunek 1.10).

Uwzględnienie na etapie tworzenia planu biznesowego dyrektyw powoduje, iż kluczowe przesądzenia w tym obszarze mają stabilne uzasadnienie, wyrażone zależnościami z innymi składowymi planu. Szczególnie istotna jest możliwość powiązania reguł biznesowych z politykami, dzięki czemu uproszczone jest śladowanie pomiędzy regułą – rozumianą jako ograniczenie stopnia swobody w działaniach operacyjnych – a intencją, jaka jej przyświecała, reprezentowaną przez politykę, która najczęściej będzie posiadała związki z celami organizacji oraz kluczowymi dla ich osiągania kierunkami działań.

Warto, przy okazji omawiania reguł biznesowych zwrócić uwagę na odwołania pomiędzy standardami OMG. Wprawdzie BMM w metamodelu jawnie umieszcza regułę biznesową, aczkolwiek w treści specyfikacji wskazuje, iż źródłem wiedzy o regułach biznesowych jest standard *Semantics of Business Vocabulary and Business Rules* [OMG13b]. Podobnie, w przypadku procesów biznesowych, wskazany jest jawnie standard BPMN.



Rysunek 1.9. Polityki oraz reguły biznesowe.



Rysunek 1.10. Procesy biznesowe w BMM.

Przedstawione powyżej informacje nie wyczerpują całości standardu BMM, pokazują natomiast w stopniu wystarczającym możliwy zakres zastosowania. Na pierwszy plan wyraźnie wysuwa się możliwość zbudowania procesu wytwórczego (metodyki) na bazie przedstawionej ontologii oraz charakteru asocjacji, łączących klasy metamodelu BMM. Zaprezentowane w postaci hierarchii definicje celów, środków ich osiągania oraz łączących te dwa obszary asocjacje, pozwalają na określenie działań o charakterze zstępującym, mających na celu stworzenie fundamentów planu biznesowego. Następnie, niezależnie od siebie, można rozbudowywać model o elementy związane z dyrektywami, procesami biznesowymi oraz czynnikami wpływającymi na elementy konstruowanego planu biznesowego. Mając skonstruowany według jasno określonych zasad model, dużo łatwiejsza staje się jego analiza, której potrzeba może wynikać zarówno z etapów konstruowania określonej wersji modelu (przeglądy wewnętrzne, zatwierdzanie modelu) jak i późniejszych działań związanych z zarządzaniem zmianą.

Interesującą, z metodycznego punktu widzenia, jest możliwość odwoływania się w ramach BMM do składowych innych standardów. Oznaczać to bowiem może, konieczność koordynacji prac w obszarach objętych różnymi standardami. Aspekt ten będzie omawiany w kolejnych podrozdziałach.

1.2 Koncepcje biznesowe

Koncepcje biznesowe[5] stanowią wieloaspektowy fundament modelu biznesowego. Koncepcje biznesowe, rozumiane jako termin oraz jego znaczenie,

[5] Często określane mianem pojęć biznesowych.

stanowią składową języka, przy pomocy którego biznes jest opisywany. Przykładowo, w sformułowaniu *Obsługa zlecenia produkcyjnego,* następuje jawne odwołanie do terminu *Zlecenie produkcyjne*. Aby termin uznać za poprawnie zdefiniowaną koncepcję, należy uzupełnić go o definicję, na przykład *żądanie wyprodukowania określonego produktu, zawierające oczekiwania ilościowe, jakościowe, czasowe oraz wskazówki odnośnie procesu technologicznego.* Za koncepcję biznesową należy także uznać logiczny związek łączący różne koncepcje. Przykładowo, koncepcje *Zamówienie* i *Zlecenie produkcyjne* łączy koncepcja *powstaje na bazie*.

O znaczeniu tego aspektu opisu organizacji może świadczyć to, iż oparte są na nim inne elementy opisu obszaru biznesowego, w szczególności reguły biznesowe oraz reguły decyzyjne. Niektóre z technik tworzenia architektury procesów biznesowych oraz określania zdolności (ang. *capability*) organizacji także bezpośrednio bazują na zidentyfikowanych koncepcjach biznesowych. Precyzyjnie określone znaczenie terminu będzie także determinowało granice opisywanych procesów biznesowych.

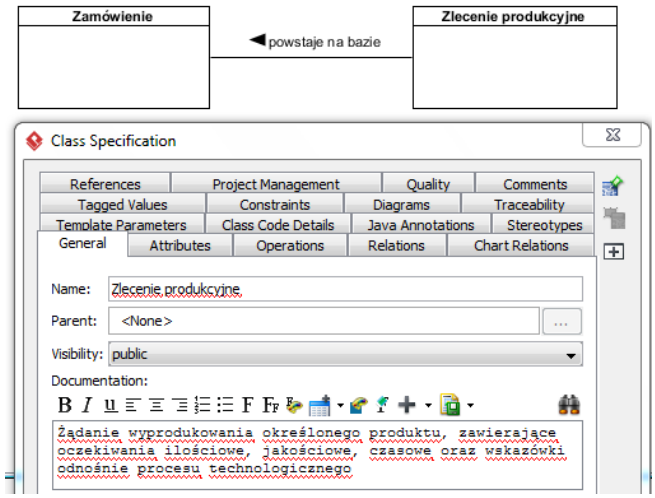
Standardem, narzucającym porządek w zakresie definiowania koncepcji jest SBVR, który w zakresie omówionym powyżej precyzuje:

- wyrażenia (ang. *expression*), rozumiane jako element służący komunikacji, którego przykładem jest wyrażenie tekstowe *Zlecenie produkcyjne*, oraz
- ich znaczenie (ang. *meaning*).

SBVR wprowadza jednocześnie elastyczność podczas kojarzenia wyrażeń i znaczeń, dopuszczając możliwość zdefiniowania wielu znaczeń dla określonego wyrażenia. W terminologii SBVR powiazanie wyrażenia z określonym znaczeniem jest nazywane reprezentacją (ang. *representation*).

SBVR nie definiuje jednakże żadnej notacji dla opisanych powyżej koncepcji; w gestii metodyki oraz wykorzystywanego narzędzia wspierającego tworzenie artefaktów architektury biznesowej pozostaje więc wskazanie środków umożliwiających ich realizację.

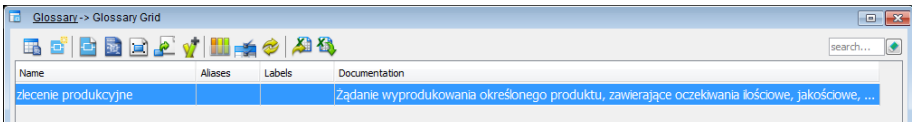
W firmie AION, do tworzenia modelu koncepcji, stosowany jest przede wszystkim diagram klas języka UML, dzięki czemu minimalizuje się liczbę środków wymaganych do realizacji działań w obszarach analizy biznesowej oraz systemowej. Zapis wyrażenia i znaczenia, przy tym podejściu, jest realizowany bezpośrednio poprzez nadanie klasie, właściwości nazwy oraz określenia ich definicji w stosowanym miejscu (Rysunek 1.11). Aspekt reprezentacji jest spełniany automatycznie poprzez określenie wartości dla dwóch wcześniejszych cech.



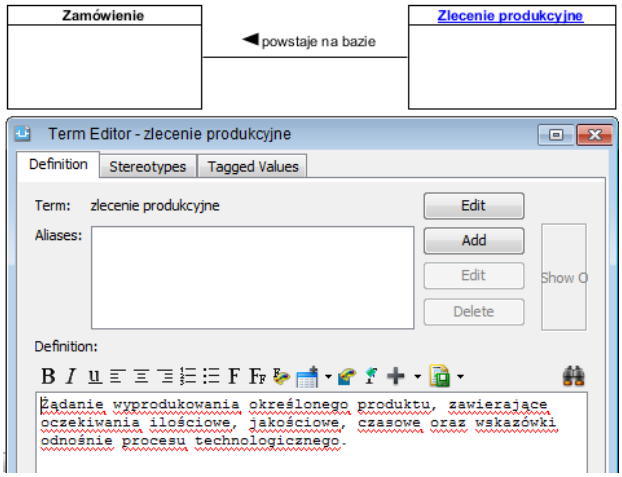
Rysunek 1.11. Wyrażenie oraz znaczenie.

W poszukiwaniu innych sposobów realizacji modelu koncepcji, wynikających w mniejszym stopniu z cech samego języka UML, należy brać pod uwagę możliwości narzędzi wykorzystywanych do opracowywania architektury biznesowej. Dość często spotykaną w popularnych narzędziach modelowania funkcjonalnością jest możliwość utrzymywania słownika pojęć (Rysunek 1.12), rozumianego jako para: wyrażenie – znaczenie. Zwykle, nazwy koncepcji zdefiniowanych w takim słowniku są rozpoznawalne w narzędziach przy okazji opisywania elementów składowych modelu, tworząc trwałe powiązania pomiędzy rozpoznanym łańcuchem tekstowym a pozycją słownika (Rysunek 1.13), co czyni model bardziej spójnym i łatwiejszym do czytania.

W sytuacjach, gdyby takie proste podejścia nie były wystarczające, należałoby stosować dedykowane rozwiązania informatyczne, służące zarządzaniu modelami koncepcji (np. *RuleXpress* [RX] czy *Sapiens Decision* [SD]).



Rysunek 1.12. Słownik.



Rysunek 1.13. Odwołanie do elementu słownika.

1.3 Procesowe spojrzenie na organizację

Procesowy paradygmat zarządzania organizacją wydaje się być aktualnie jedną z najbardziej dojrzałych koncepcji zarządczych. Konstrukcyjnie, procesowy model organizacji można podzielić na dwa wyraźne aspekty:

- architekturę procesów – identyfikującą procesy organizacji oraz wskazującą zależności je łączące, oraz
- przebiegi procesów – prezentujące przebieg prac realizowanych przez role określonego typu.

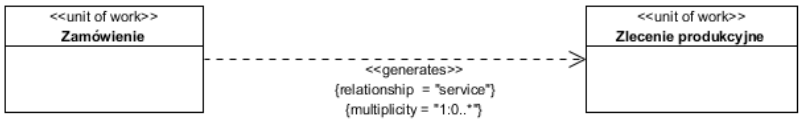
1.3.1 Architektura procesów

Architektura procesów jest pojęciem przyjmującym bardzo różne formy w metodykach związanych z szeroko pojętym zarządzaniem procesowym. W firmie AION, do definiowania architektury procesów, zaadaptowano podejście proponowane przez Martyna Oulda [Oul05], zmieniając jednakże oryginalną notację na UML oraz BPMN.

Zstępujące podejście do utworzenia architektury procesów oparto o diagram klas języka UML, dociążając klasy stereotypami:

- «essential business entity»,
- «designed business entity»,
- «mandatory business entity»,
- «essential unit of work»,
- «designed unit of work»,
- «mandatory unit of work».

W stosunku do oryginalnego podejścia, wprowadzono dodatkową kategorię przedmiotów pracy (ang. *unit of work*) – *mandatory*, reprezentującą te przedmioty, które muszą być obsługiwane przez organizację, pomimo faktu, iż nie stanowią jej kluczowej działalności. Relacje łączące przedmioty pracy pozostały w metodyce AION niezmienione w stosunku do pierwowzoru, aczkolwiek, rozszerzono ich interpretację oraz zastosowanie, proponując, by łańcuchy wartości stanowiące podstawę do procesowego zarządzania organizacją na poziomie makro, były tworzone na poziomie przedmiotów pracy, a nie procesów z nich wynikających. Przykład reprezentacji przedmiotów pracy oraz relacji je łączących przedstawia Rysunek 1.14.



Rysunek 1.14. Zależności pomiędzy przedmiotami pracy.

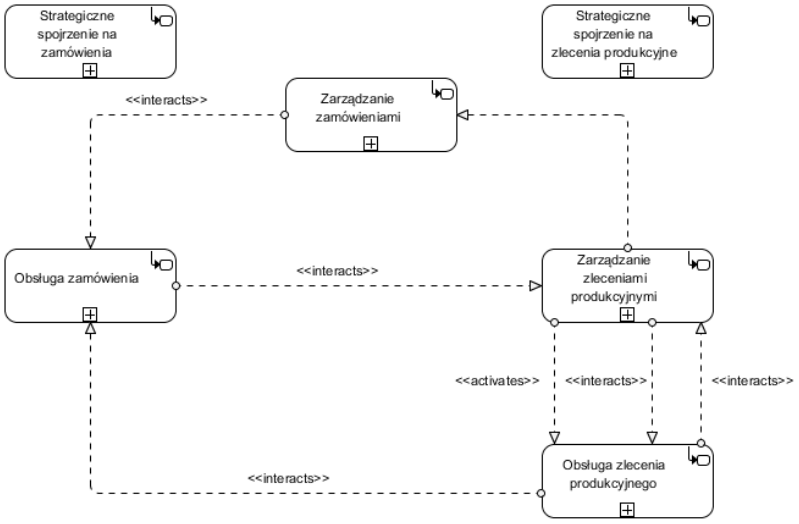
Na podstawie charakteru relacji łączących przedmioty pracy (*task-force*, *service*) oraz samych relacji, w promowanym przez AION podejściu definiuje się pierwsze przybliżenie docelowej architektury procesów, bazując na predefiniowanych przekształceniach. Przykład zdefiniowanej architektury procesów prezentuje Rysunek 1.15.

Pierwsza wersja architektury jest bazą dla prac nad opracowaniem wersji docelowej, która może różnić się od wersji pierwszej liczbą procesów (może się okazać, iż działania w ramach któregoś z określonych procesów są na tyle trywialne, iż nie warto utrzymywać osobnego bytu i działania te są dołączane do innego procesu) oraz liczbą wzajemnych zależności.

Opracowywana w opisany sposób architektura procesów niesie ze sobą wiele korzyści, spośród których warto wymienić:

- zstępujący sposób powstawania, pozwalający analitykom stopniowo zapoznawać się z obszarem biznesowym poddawanym analizie, jednocześnie tworząc wartościowe artefakty analityczne,
- możliwość analizy łańcuchów tworzenia wartości na podstawie prostych koncepcyjnie modeli przedmiotów pracy,
- jasne określenie odpowiedzialności poszczególnych procesów oraz instancji procesów,
- możliwość zautomatyzowania etapu tworzenia pierwszej wersji architektury przekształceniami *Model Driven Architecture* [OMG09].

Mankamentem, szczególnie w kontekście tematu przewodniego niniejszego rozdziału, jest brak jawnego wsparcia w ramach standardu BPMN dla takiego sposobu opisywania architektury procesów. Zaprezentowana w rozdziale metoda opisu łamie składnię BPMN w momentach, gdy podprocesy są łączone przepływami komunikatów. Wydaje się jednakże, iż wyjaśnienie docelowym odbiorcom modelu, charakteru odstępstwa oraz sposobu interpretacji zapisu nie powinno powodować problemów z interpretacją oraz późniejszym utrzymaniem modelu.



Rysunek 1.15. Architektura procesów dla omówionych przedmiotów pracy.

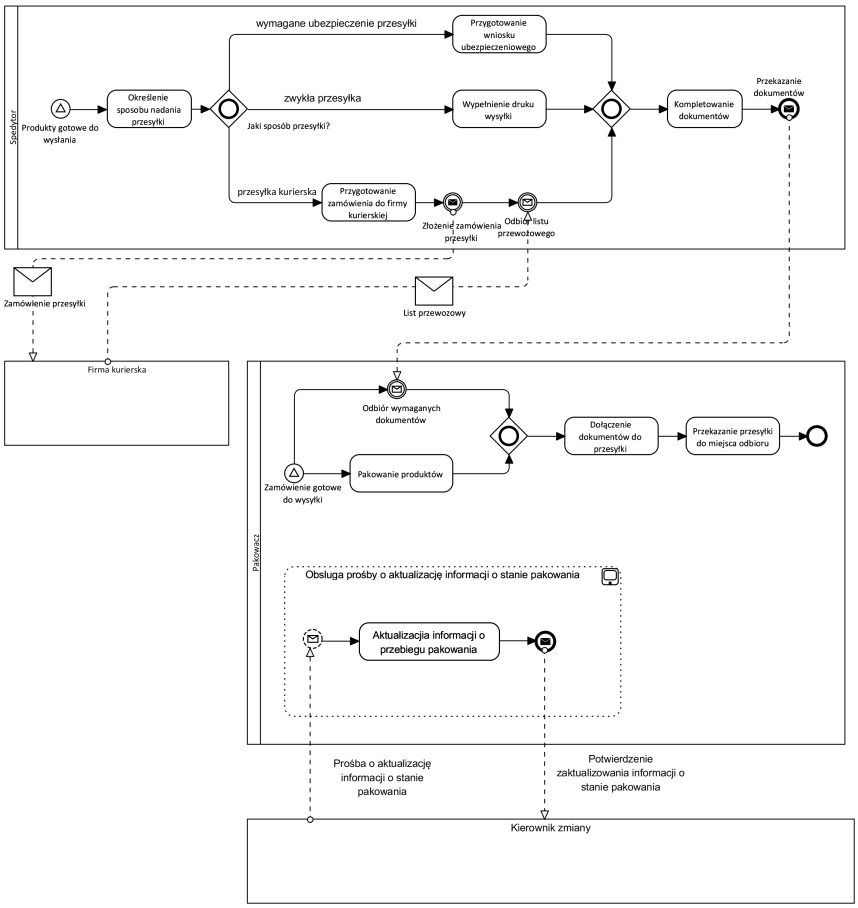
1.3.2 Przebiegi procesów w ramach ustalonej architektury

Ostro zdefiniowane w ramach architektury procesów biznesowych granice procesów w dużym stopniu przyczyniają się do ułatwienia konstruowania opisów ich przebiegów. W praktyce, zaobserwować to można podczas, najczęściej warsztatowych, dyskusji odnośnie: zdarzeń rozpoczynających proces biznesowy, zasadności umieszczania określonych działań w ramach danego procesu czy też momentu, w którym proces należy uznać za zakończony.

Przyjęta metodyka modelowania procesów biznesowych określać powinna także zasady utrzymania zgodności przebiegów procesów z przesądzeniami architektonicznymi oraz wskazywać sposób wykorzystania wybranych języków modelowania procesów. Praktyka pokazuje bowiem, iż pomimo wypracowania standardu notacyjnego dla tego obszaru architektury biznesowej, specyfika sto-

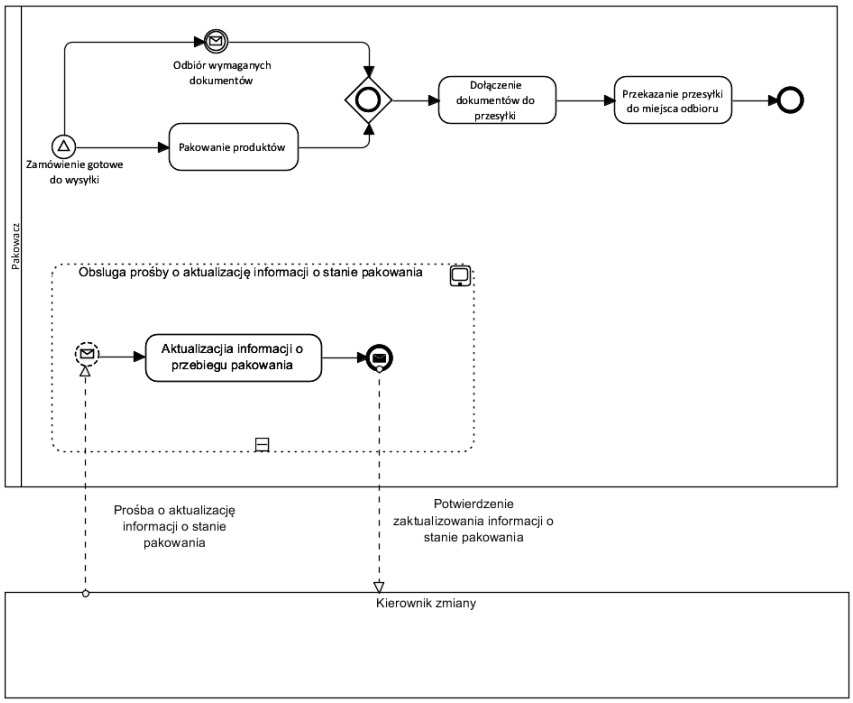
sowanych metod modelowania może wymagać konieczności dostosowania do nich dostępnych standardów. Przykłady takich konwencji prezentuje Rysunek 1.16, zawierający fragment opisu przebiegu procesu biznesowego.

Pierwszą rzeczą, która rzuca się w oczy praktykom BPMN jest wykorzystanie wielu basenów (ang. *pool*) w ramach opisu przebiegu jednego procesu biznesowego przy jednoczesnym braku torów (ang. *lane*). Podejście takie jest bezpośrednim efektem postrzegania procesu biznesowego jako zestawu działań realizowanych przez niezależnie od siebie funkcjonujące role, wymieniające między sobą komunikaty w celu koordynacji prac. Przy tego typu podejściu nieistotne jest, czy komunikacja odbywa się wewnątrz procesu czy pomiędzy procesami – w każdym przypadku sprowadza się ona do wymiany komunikatów pomiędzy rolami.



Rysunek 1.16. Sposób opisu przebiegów procesów biznesowych w metodyce AION.

Przykład tego typu sytuacji prezentuje Rysunek 1.17, na którym *Kierownik zmiany*, reprezentujący jeden proces, komunikuje się z *Pakowaczem*, uczestniczącym w realizacji drugiego procesu, będącego przedmiotem diagramu. Charakter komunikacji nakazuje reprezentowanie *Kierownika zmiany* w formie czarnej skrzynki (ang. *black box*); szczegóły jego działań będą opisane na diagramie prezentującym przebieg procesu, w którego realizacji uczestniczy.



Rysunek 1.17. Przykład komunikacji międzyprocesowej.

Podane powyżej przykłady konwencji dostosowujących standardy do potrzeb procesów wytwórczych mogłyby być interpretowane jako argument przeciwko standardom – z uwagi na brak bezpośredniego wsparcia językowego dla określonego rodzaju sytuacji – albo procesom wytwórczym, w których wykorzystano elementy nieopisane żadnymi standardami. Dotychczasowe doświadczenie firmy AION w dostosowywaniu dostępnych języków do potrzeb metodyki nakazuje jednakże zinterpretować taką sytuację jako naturalną. Z jednej bowiem strony, bogate w środki wyrazu języki oraz mechanizmy ich rozszerzeń (np. stereotypy) pozwalają na dostosowanie notacji do potrzeb określonego podejścia, co może znaleźć odzwierciedlenie, czy to w formalnym profilu języ-

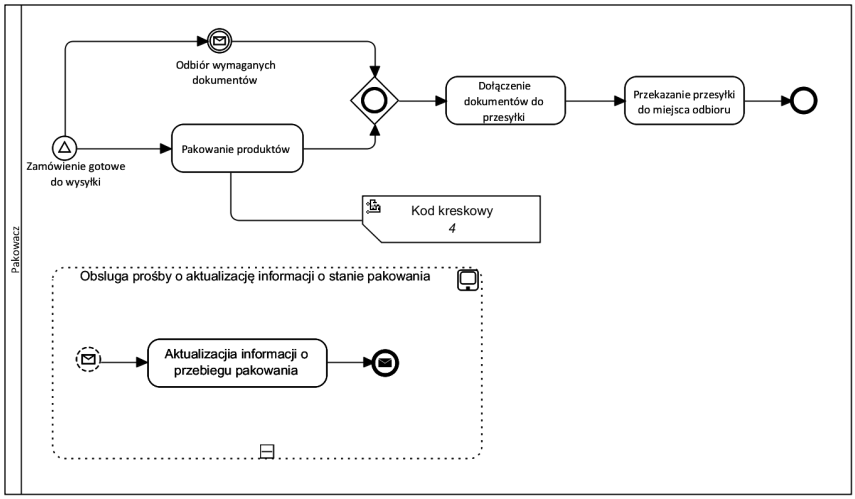
ka, czy też, mniej formalnym z punktu widzenia inżynierskiego, ale właściwym z punktu widzenia zastosowań biznesowych, dokumencie standaryzującym, opisującym rekomendowane rozszerzenia i konwencje, z drugiej, identyfikowana zostaje potrzeba w zakresie języka modelowania, mogąca znaleźć odzwierciedlenie w wersjach nowszych.

1.4 Reguły biznesowe jako operacyjne wsparcie działań biznesowych

Logika podejmowanych decyzji oraz skutki ich podjęcia, rozumiane jako praca do wykonania, stanowią dwa wyraźne, aczkolwiek ściśle związane ze sobą, aspekty funkcjonowania każdej organizacji. Aktualnie, jednym z najczęściej spotykanych podejść do opisywania tej synergii jest rozbudowywanie modeli przebiegów procesów biznesowych o rozwidlenia i warunki wynikające z logiki decyzyjnej oraz uwzględnianie, najczęściej w tekstowych komentarzach związanych z poszczególnymi krokami, zasad, do jakich powinny się stosować osoby uczestniczące w realizacji procesów. Podejścia takie, pomimo swojej, wydawałoby się, prostoty, docelowo prowadzić mogą do uzyskania nadmiernie rozbudowanych, mało czytelnych i trudnych do analizy modeli.

Opublikowane standardy *Semantics of Business Vocabulary and Business Rules* (SBVR) oraz wersja beta *Decision Model and Notation* [OMG14a] nakreślają kierunki, w jakich powinny być tworzone modele architektury biznesowej, by z jednej strony precyzyjnie oddać logikę biznesową, a z drugiej umiejscowić ją w kontekście opracowywanych modeli procesów biznesowych. Z uwagi na fakt, iż stanowią one integralną część architektury biznesowej, mogą być tworzone niezależnie od modelu procesowego organizacji, stanowiąc wartościowy element opisujący biznes. SBVR swoim zakresem obejmuje zarówno tworzenie modelu pojęć jak i reguł biznesowych, rozumianych jako ograniczenie stopnia swobody postępowania osób uczestniczących w pracy realizowanej w organizacji. Zgodnie z teorią, i stojącymi za nią zaleceniami praktycznymi (w szczególności opisanych w publikacjach dotyczących języka RuleSpeak™ [RST]), reguły biznesowe są definiowane niezależnie od miejsca ich wykorzystania; w szczególności uwaga ta dotyczy reguł operacyjnych (ang. *operative business rule*), rozumianych jako element ustanowionego ładu organizacyjnego, który może być przez ludzi złamany. Zaproponowanie wyraźnego rozgraniczenia pomiędzy działaniami realizowanymi przez organizację, a narzuconymi na to działanie ograniczeniami powoduje, iż każdym z wymienionych rodzajów artefaktów należy zarządzać osobno, dbając jednocześnie o utrzymywanie wzajemnych relacji.

SBVR nie precyzuje notacji dla reguł biznesowych umieszczanych w modelach architektury biznesowej. W gestii metodyki oraz wykorzystywanych narzędzi do modelowania leży więc zaproponowanie sposobu prezentacji reguł oraz relacji z innymi artefaktami. Rysunek 1.18 ilustruje notację dla reguł biznesowych umieszczanych na diagramie przebiegu procesów biznesowych BPMN, zaproponowaną przez producenta narzędzia Visual Paradigm. Zgodnie z zaleceniami SBVR oraz RuleSpeak, reguła operacyjna o treści *Produkt może być zapakowany tyko wówczas, gdy posiada kod kreskowy* została powiązana z tym miejscem przebiegu procesu, w którym reguła może zostać złamana.



Rysunek 1.18. Przykład umiejscowienia reguły biznesowej w modelu procesów.

Umiejscowienie reguł biznesowych w opisie biznesu, a nie wspierających biznes rozwiązań informatycznych, jasno wskazuje, iż wszelkie inicjatywy mające na celu wdrożenie maszyn regułowych (ang. *business rules management system*, *BRMS*) powinny być realizowane na poziomie biznesowym a automatyzowane reguły biznesowe winny pochodzić ze zdefiniowanych modeli biznesowych, zachowując jednocześnie wyraźne śladowanie pomiędzy warstwą logiki biznesowej a warstwą automatyzowanych reguł biznesowych.

Reguły decyzyjne (ang. *decision rule*), w odróżnieniu do operacyjnych reguł biznesowych, mają na celu opisanie kryteriów wyboru najlepszej / rekomendowanej opcji w kontekście wykonywanych rutynowych działań operacyjnych. Standard DMN określa obowiązkowe cechy decyzji operacyjnej oraz wskazuje rekomendowane metody jej specyfikowania. Każda ze wskazanych metod cał-

kowicie uniezależnią opis decyzji od miejsca, w jakim decyzja zostanie wykorzystana, aczkolwiek, w przypadku stosowania procesowego paradygmatu opisu realizowanych w organizacji prac z wykorzystaniem BPMN, DMN wskazuje możliwe związki modelu decyzyjnego oraz modelu procesów.

Zarówno reguły biznesowe jak i decyzyjne, wraz z wprowadzeniem stosownych standardów sankcjonujących ich istnienie oraz znaczenie, mają szansę stać się pełnoprawnym produktem zarządczym organizacji, pozwalającym na zwiększenie jednoznaczności proponowanych metod pracy oraz mogącym przyczynić się do tworzenia spójnego ładu organizacyjnego. Uzupełniając standardy o najlepsze praktyki branżowe, takie jak RuleSpeak™, czy The Decision Model™ [TDM], możliwie będzie zwiększenie jednolitości reguł oraz uproszczenie reguł decyzyjnych, tym samym zmniejszając zarówno koszty wprowadzania nowych rozwiązań menedżerskich jak i ich późniejszej aktualizacji.

1.5 Wsparcie biznesu rozwiązaniem informatycznym

1.5.1 Wymagania biznesu wobec funkcjonalności system IT

Projektowanie docelowego kształtu procesów biznesowych, bez względu na to, czy zostaną explicite osadzone w organizacji, czy też posłużą jedynie jako narzędzie analityczne, musi opierać się na dostępności zasobów mogących wesprzeć realizatora działań w osiągnięciu postawionych celów. Jednym z rodzajów zasobów, które należy wziąć pod uwagę, są rozwiązania informatyczne, objawiające się udostępnieniem określonej funkcjonalności w określonej sytuacji.

W metodyce AION, oczekiwane przez reprezentantów biznesowych funkcjonalności rozwiązań informatycznych, są identyfikowane oraz specyfikowane na bazie opisanych procesów biznesowych. Analiza potrzeb odbywa się na poziomie poszczególnych, niepodzielnych kroków w przebiegu procesów biznesowych, tym samym, nadając wymaganiom bardzo wąski kontekst biznesowy. Uzyskanie oczekiwanej jednolitości tworzonych wymagań, charakteryzującej się zbliżonym poziomem szczegółowości opisu, jednoznacznie określonymi cechami oraz jednoznacznie określonymi zasadami śladowania jest wypadkową stosowanego języka opisu wymagań oraz przyjętej metodyki prac.

Metodyka AION rekomenduje stosowanie diagramu wymagań języka *Systems Modeling Language* (SysML) [OMG12] do specyfikowania wymagań. Rekomendacja w tym zakresie jest efektem:

— polityki firmy do opierania się w jak największym stopniu na standardach,

— dostępności narzędzia (Visual Paradigm) umożliwiającego stosowanie diagramu wymagań języka SysML w zakresie szerszym, niż tylko wśród artefaktów określonych w specyfikacji SysML, co umożliwia zintegrowanie wymagań ze wszystkimi artefaktami architektury biznesowej oraz modeli rozwiązań informatycznych.

Rysunek 1.19 przedstawia fragment przebiegu procesu, który będzie w dalszej części poddany analizie pod kątem specyfikacji wymagań funkcjonalnych na poziomie *Computation Independent Model* (CIM), według nomenklatury MDA.

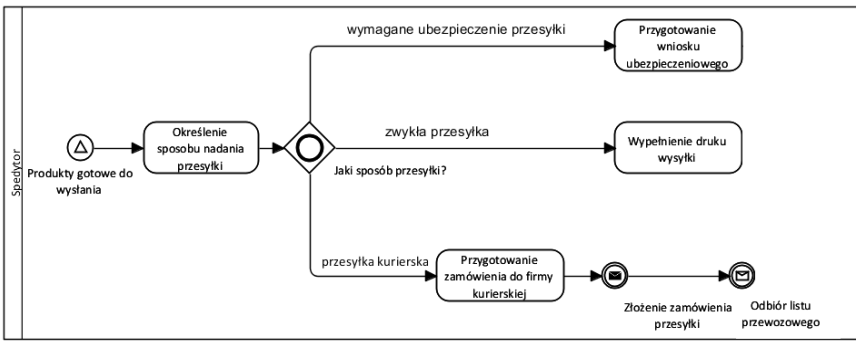
Analiza kroku *Określenie sposobu nadania przesyłki* zaowocowała, w omawianym przykładzie, wyspecyfikowaniem wymagań: *Umożliwienie zdefiniowania zawartości przesyłki* oraz *Automatyczne określenie sposobu nadania przesyłki* (Rysunek 1.20). Dla każdego z wymagań łatwo można podać uzasadnienie, wykazujące, iż realizacja wymagań pozytywnie wpłynie na sposób wykonywania kroku procesu biznesowego. Analogiczną analizę wykonuje się dla każdego kroku w procesie biznesowym. W szczególnych przypadkach może się okazać, iż wykorzystanie technologii informatycznych nie tylko polepszy parametry poszczególnych kroków procesu, ale także zmieni kształt przebiegu procesu.

Diagram wymagań języka SysML daje możliwość tworzenia różnego rodzaju związków i zależności pomiędzy wymaganiami. Związek kompozycji pozwala na dekomponowanie bardziej ogólnych wymagań, na wymagania o większym stopniu szczegółowości. Działanie takie może mieć uzasadnienie na przykład wówczas, gdy ogólne wymaganie będzie dostarczane w sposób przyrostowy, albo poszczególne aspekty wymagania ogólnego będą miały różne priorytety. Innym sposobem wzajemnego powiązania wymagań jest wykorzystanie zależności «deriveReqt», która określa, iż jedno wymaganie jest wyprowadzane z innego wymagania. W omawianym przykładzie[6], wymaganie *Umożliwienie definiowania szablonu wniosku ubezpieczeniowego* jest wyprowadzone z bardziej ogólnego wymagania *Umożliwienie definiowania szablonu dokumentu*. Analogicznie, warto rozpatrzyć model pod kątem zastosowania pozostałych rodzajów zależności, zdefiniowanych dla tej klasy problemu, co może się przyczynić do poprawy jego jakości.

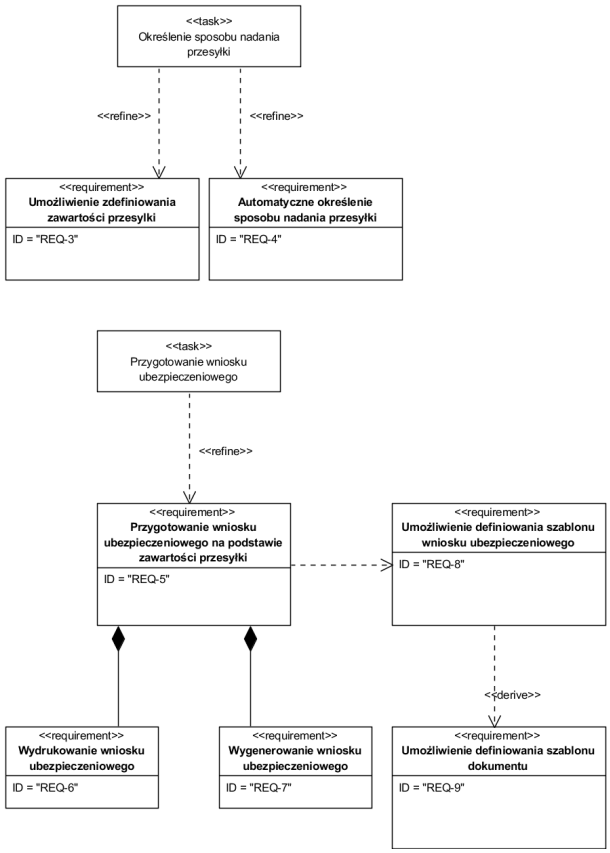
Z metodycznego punktu widzenia, warto zauważyć, iż wymagania funkcjonalne definiowane na poziomie modelu architektury biznesowej powinny być opisywane w sposób oddający intencję (potrzebę) biznesu, abstrahując jedno-

[6] W implementacji diagramu wymagań SysML w narzędziu Visual Paradigm stereotypowi «deriveReqt» odpowiada stereotyp «derive».

cześnie od sposobu ich realizacji. Za realizację postawionych wymagań odpowiedzialne będą kolejne fazy procesu produkcyjnego, z których jedna zostanie zaprezentowana w kolejnych podrozdziałach.



Rysunek 1.19 Fragment procesu poddany analizie pod kątem wymagań funkcjonalnych.



Rysunek 1.20 Fragment diagramu wymagań.

1.6 Realizacja wymagań biznesu – Analiza systemowa

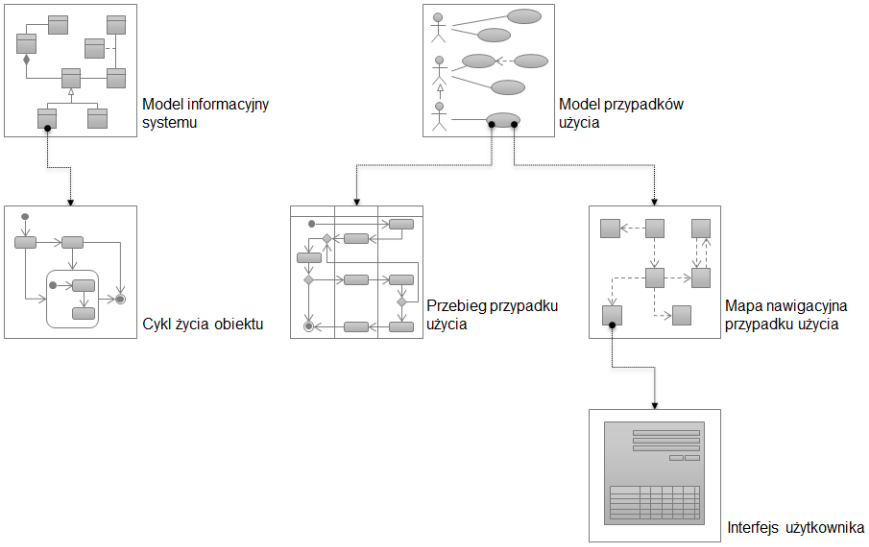
Omówione w poprzednich rozdziałach artefakty architektury biznesowej definiują biznesowy kontekst dla systemu wspierającego opisywaną organizację. Osadzone w tym kontekście wymagania – oczekiwania organizacji wobec systemu informatycznego, definiowane z perspektywy uczestników procesów biznesowych, które system ma wspierać – określają zakres przedsięwzięcia jakim jest wytworzenie tego systemu. Określenie zakresu pozwala przystąpić do realizacji kolejnych prac procesu wytwórczego, których celem jest opracowanie propozycji konceptualnej rozwiązania informatycznego realizującego wymagania – tj. opracowania modeli analizy systemowej.

Znalezienie optymalnego rozwiązania – przeniesienie, zdobytej na etapie modelowania biznesowego, wiedzy o dziedzinie problemu oraz zidentyfikowanych potrzeb uczestników procesów biznesowych, na poziom proponowanego rozwiązania informatycznego, mimo prostoty wykorzystywanych zwykle narzędzi, nie jest zagadnieniem trywialnym, wymaga od analityków kreatywnego myślenia, doświadczenia, a w kontekście efektywności prowadzenia prac przedsięwzięcia – świadomości konsekwencji jakie może nieść za sobą przyjęte rozwiązanie na dalsze prace analityczne. Stąd kluczowa rola dobrze zdefiniowanej metodyki prowadzenia prac.

Modele tworzone na poziomie analizy systemowej specyfikują zachowanie systemu abstrahując od rozwiązań implementacyjnych (doboru technologii czy metod implementacji). Odnosząc się do nomenklatury MDA można uznać, że są to modele na poziomie *Platform Independent Model* (PIM). W ramach prezentowanego podejścia projektowany system specyfikowany jest przez: *Model informacyjny systemu* oraz *Model przypadków użycia.* Model informacyjny definiuje strukturę zawartości informacyjnej systemu (wykorzystywanym środkiem wyrazu jest diagram klas języka UML) łącznie z cyklami życia kluczowych klas dziedziny (modelowanymi za pomocą diagramu maszyny stanów UML). Model przypadków użycia (wyrażony za pomocą diagramu przypadków użycia UML) stanowi zestawienie funkcjonalności realizowanych przez system, ze wskazaniem użytkowników systemu oraz zależności pomiędzy poszczególnymi funkcjonalnościami. Specyfikacja funkcjonalności uwzględnia jej przebieg, tj. zobrazowanie interakcji użytkownika z systemem oraz przetwarzanie danych realizowane w ramach funkcjonalności (modelowany za pomocą diagramu aktywności UML) oraz projekt interfejsu użytkownika – model obrazujący ścieżki nawigacyjne (przejścia) pomiędzy poszczególnymi elementami (np. ekranami) interfejsu użytkownika, którymi podąża użytkownik korzystający z funkcjonalności (specyfikowany za pomocą diagramu klas języka UML) oraz

projekt elementów graficznych interfejsu (wyrażony za pomocą konstrukcji dostępnych w stosowanym narzędziu modelowania).

Modele opracowywane na etapie analizy systemowej, prowadzonej zgodnie z podejściem stosowanym w firmie AION, przedstawiono na schemacie poniżej (Rysunek 1.21). W ramach kolejnych punktów niniejszego rozdziału zostaną one opisane, wskazany zostanie cel tworzenia, stosowane środki wyrazu oraz (opracowane na potrzeby realizacji przedsięwzięć) rozszerzenia z zakresu każdego z modeli. Zdefiniowane zostaną zależności zachodzące pomiędzy elementami modeli.



Rysunek 1.21. Modele analizy systemowej prowadzonej zgodnie z podejściem stosowanym w firmie AION.

Na potrzeby prowadzenia przedsięwzięć analizy zgodnie z prezentowanym podejściem, stosowany język modelowania został rozszerzony. Wykorzystanym mechanizmem rozszerzeń języka jest profil (ang. *profile*). Profil pozwala dostosować język do specyfiki prowadzonych przedsięwzięć projektowych – rozszerzyć (poprzez uściślenie i doprecyzowanie semantyki) elementy języka, a w konsekwencji zdefiniować własne metamodele oraz reguły ich poprawności (pod warunkiem zachowania zgodności z ograniczeniami przewidzianymi przez standard języka dla rozszerzanych elementów języka).

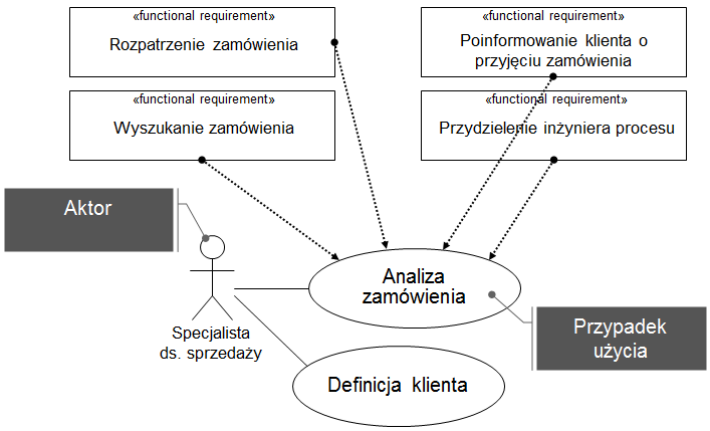
Na potrzeby prowadzenia analizy systemowej zgodnie z podejściem wypracowanym w firmie AION zdefiniowany został profil *Standardy Analizy AION*

(dla uproszczenia nazywany w dalszej części rozdziału profilem AION). Profil AION określa elementy języka UML wykorzystywane do prowadzenia przedsięwzięć analizy zgodnie z podejściem, wiąże się z definicją stereotypów (ang. *stereotype*) nakładanych na te elementy, przypisaniem im określonych właściwości – tzw. wartości etykietowanych (ang. *taggedValues*) oraz definicją ograniczeń (ang. *constraints*) oznaczających warunki i reguły, zgodnie z którymi elementy opatrzone stereotypami funkcjonują.

1.6.1 Model przypadków użycia

Pierwszym krokiem w kierunku pozyskania przez organizację pożądanego i użytecznego dla niej rozwiązania informatycznego jest precyzyjne określenie rzeczywistych oczekiwań w stosunku do cech funkcjonalnych systemu, czyli sprecyzowanie wymagań w terminach zadań, jakie przyszli użytkownicy docelowego systemu będą mogli w tym systemie wykonać – wymagań, które definiowane są z perspektywy użytkowników tworzonego systemu i realizują wyspecyfikowane na etapie analizy biznesowej wymagania biznesu.

Wykorzystywanym w prezentowanym podejściu narzędziem, które umożliwia zdefiniowanie zakresu funkcjonalności tworzonego systemu jest model przypadków użycia, wyrażony za pomocą diagramu przypadków użycia języka UML (Rysunek 1.22). Aktorzy modelu reprezentują przyszłych użytkowników systemu oraz inne systemy zewnętrzne współpracujące z modelowanym. Wymagania funkcjonalne wobec systemu są odwzorowywane w przypadki użycia. Uwzględnione w modelu zależności aktor – przypadek specyfikują dostępność funkcjonalności systemu dla poszczególnych użytkowników.



Rysunek 1.22. Diagram przypadków użycia wyprowadzony z wymagań biznesu.

Aktorzy modelu przypadków użycia są mapowani z uczestników procesów biznesowych odpowiedzialnych za realizację tych akcji, które stanowią źródło zidentyfikowanych wymagań biznesu. Identyfikacja aktorów umożliwia spojrzenie na system z perspektywy użytkowników i określenie jakich funkcji będą oni wymagać od systemu, a w konsekwencji – identyfikację przypadków użycia.

Liczba przypadków użycia zależy od przyjętej metodyki, ponieważ ta przekłada się na ich granulację. Wśród stosowanych podejść można wyróżnić dwa najczęściej stosowane. W pierwszym z podejść poszczególnym akcjom (*dodawania*, *modyfikacji*, itd.) odpowiadają osobne przypadki. W drugim podejściu, przypadek odpowiada zestawowi operacji wykonywanych na danym pojęciu, niekiedy w określonym stanie, dociążonym uprawnieniami określonego aktora. Z punktu widzenia aktora jest to skoncentrowanie usług na określonym pojęciu – przedmiocie pracy.

W pierwszym podejściu mamy większą liczbę przypadków użycia, ale każdy z przypadków jest opisany prostszym scenariuszem/diagramem. Dodatkowo konsekwencją częstych powtórzeń wspólnych fragmentów przebiegów i zastosowania relacji include/extend do reprezentacji tych powtórzeń, jest jeszcze większa liczba przypadków użycia. W drugim podejściu liczba przypadków oraz relacji jest zdecydowanie mniejsza aczkolwiek przebiegi są dużo bardziej złożone. Co jest lepsze? To zależy. Wybór podejścia stanowi aspekt procesowy realizacji metodyki – każdorazowo zależy od specyfiki prowadzonego przedsięwzięcia. W praktyce firmy AION zdecydowanie częściej stosowane jest podejście drugie – bardziej treściwe przypadki użycia. Jest to związane z, omówionym w dalszej części rozdziału, wykorzystaniem diagramu aktywności do wyrażenia przebiegów przypadków.

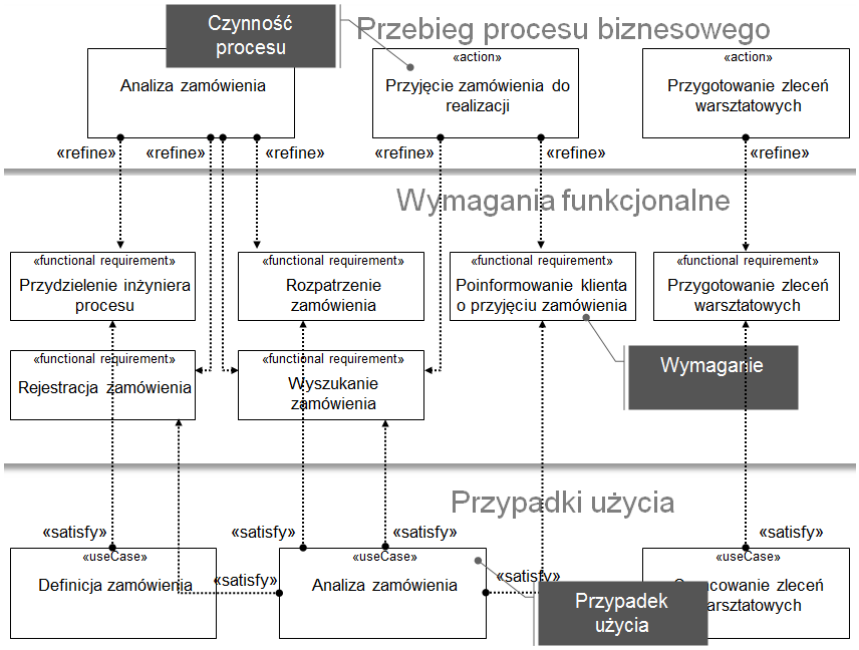
Niezależnie od zastosowanego podejścia liczba przypadków użycia rzeczywistego systemu biznesowego jest zwykle odpowiednio duża. Aby ułatwić pracę z modelem i zarządzanie przypadkami, dokonuje się podziału modelu. Stosowanym kryterium podziału jest definicja diagramu dla każdego z aktorów (lub kilku aktorów z mniejszą liczbą przewidzianych dla nich funkcjonalności) oraz dodatkowego diagramu prezentującego przyjętą hierarchię dziedziczenia aktorów.

Diagram przypadków użycia jest najprostszym diagramem UML (z najmniejszą liczbą konstrukcji składowych), ale jednocześnie jednym z najtrudniejszych diagramów języka – budzącym największe kontrowersje wśród praktyków. Źródłem największych problemów jest nieprecyzyjna definicja relacji

zdefiniowanych pomiędzy przypadkami użycia (include, extend oraz generalizacji).

Diagramy przypadków użycia, opracowywane w ramach pierwszej iteracji podejścia firmy AION nie uwzględniają relacji pomiędzy przypadkami użycia. Te pojawiają się na diagramach dopiero w wyniku strukturalizacji diagramów aktywności stanowiących zapis przebiegów przypadków użycia i to nie zawsze. Diagramy aktywności mają bowiem konstrukcję umożliwiającą wyodrębnienie fragmentu przebiegu (w szczególności części wspólnej kilku diagramów) i późniejsze odwoływanie się do niego. Konstrukcja pozwala na zamodelowanie odpowiedniej zależności jedynie na diagramach aktywności i nie wymaga jej przeniesienia na poziom diagramu przypadków użycia. Przyjęte rozwiązanie w praktyce okazało się bardziej użyteczne i czytelne dla biznesu. Prezentowany odbiorcy systemu diagram zawiera bowiem jedynie ‘czyste’ przypadki użycia niosące rzeczywistą, zauważalną/mierzalną wartość dla użytkowników systemu– jest zgodnie z definicją zbiorem deklarowanych funkcjonalności realizowanych przez system.

Uzgodniony za pomocą modelu przypadków użycia zakres funkcjonalności tworzonego systemu jest poddawany ocenie pod kątem jego zgodności z oczekiwaniami odbiorcy systemu – poddawany walidacji. Wykorzystywanym w tym celu narzędziem jest diagram wymagań języka SysML (Rysunek 1.23). Opracowany na etapie analizy biznesowej diagram, zostaje uzupełniony – zidentyfikowane wymagania biznesu (a w konsekwencji akcje procesów biznesowych, które stanowią źródło tych wymagań) zostają powiązane z przypadkami użycia, które realizują oczekiwaną funkcjonalność. Powiązanie ma znaczenie pragmatyczne – ustanawia zależność pomiędzy elementami opisu funkcjonowania organizacji a elementami opisu funkcjonalności tworzonego systemu, umożliwiając tym samym śledzenie wpływu zmian dokonanych na poziomie organizacji na projektowany system. Przyjęte rozwiązanie gwarantuje, iż opracowany model i zestaw wymagań stanowią spójną, wzajemnie wpływającą na siebie strukturę, pozwalającą na sprawne przeprowadzenie przedsięwzięcia budowy systemu IT.



Rysunek 1.23. Diagram wymagań – zależność pomiędzy przypadkami użycia, a elementami analizy biznesowej.

1.6.2 Specyfikacja funkcjonalności

1.6.2.1 Przebieg realizacji funkcjonalności

Specyfikacja języka UML pozostawia modelującemu pewną swobodę doboru środków wyrazu w zakresie modelowania przebiegu przypadku użycia, jednakże jednocześnie umiejscawia element przypadku użycia (*UseCase*) w konkretnej strukturze metamodelu wskazując szereg cech ograniczających te środki – definiuje *UseCase* jako specjalizację elementu typu *BehavioredClassifier*, którego zachowanie może być opisane elementem typu *Behavior*, z uwzględnieniem parametrów (*Parameter*) oraz ograniczenia (*Constraint*).

Wykorzystywanym w podejściu firmy AION narzędziem opisu przebiegu przypadku użycia (realizacji funkcjonalności projektowanego systemu) jest, budowany według określonych zasad, diagram aktywności języka UML (powiązany jako poddiagram ze specyfikowanym przypadkiem użycia). Przyjęte rozwiązanie jest w pełni zgodne ze standardem języka UML – diagram aktywności jest graficzną reprezentacja elementu typu *Activity* – specjalizacji *Behavior*.

Diagram dzielony jest na partycje odpowiadające odpowiednio: działaniom aktora, usługom warstwy prezentacji oraz usługom warstwy logiki systemu. Założony poziom szczegółowości opisu ustalają przyjęte zasady definicji poszczególnych działań/akcji jako atomowych kroków algorytmu przebiegu modelowanej funkcjonalności. Działania, za realizację których odpowiedzialny jest aktor to m. in. akcje wprowadzania/modyfikacji danych, wyboru obiektu do dalszego przetwarzania, wyboru opcji czy potwierdzenia decyzji. Działania uwzględnione w partycji warstwy prezentacji to akcje związane z wizualizacją elementów interfejsu użytkownika prezentujących dane i informacje, umożliwiających edycję danych, wybór obiektów czy opcji. Zgrupowane w warstwie logiki działania są związane z realizacją logiki biznesowej systemu, są to m in. akcje odczytu i zapisu danych trwałych, weryfikacji poprawności przetwarzanych danych, transformacji danych czy wywoływania usług zewnętrznych.

Przebieg realizacji opisywanej funkcjonalności rozpoczyna się od zdarzenia odzwierciedlającego wyzwalacz przypadku użycia wskazujący biznesowy kontekst uruchamiania funkcjonalności. Uwzględnienie w przebiegu zdarzenia-wyzwalacza dodatkowo wiąże się z ukryciem sposobu dostępu do funkcjonalności (modelowanym na opisanej w dalszej części rozdziału mapie nawigacyjnej przypadku użycia), a modelowaniem w przebiegu tylko akcji związanych z realizacją funkcjonalności.

Konstrukcje składowe diagramu aktywności umożliwiają modelowanie w ramach jednego diagramu wszystkich scenariuszy przebiegu przypadku użycia – głównych, alternatywnych oraz wyjątkowych. Rozdzielenie przebiegu na poszczególne scenariusze związane jest przede wszystkim z węzłami decyzyjnymi, z których część sterowana jest przetwarzanymi danymi, część wynika z wyboru przez użytkownika jednej z dostępnych opcji. W celu umożliwienia jawnego powiązania przebiegu przypadku z elementami interfejsu użytkownika odzwierciedlającymi dostępne opcje, w profilu AION zdefiniowano odpowiednie stereotypy oraz związane z nimi atrybuty (prezentowany w przykładzie stereotyp «button driven» umożliwia powiązanie przebiegu sterowania z przyciskiem zaprojektowanego interfejsu użytkownika).

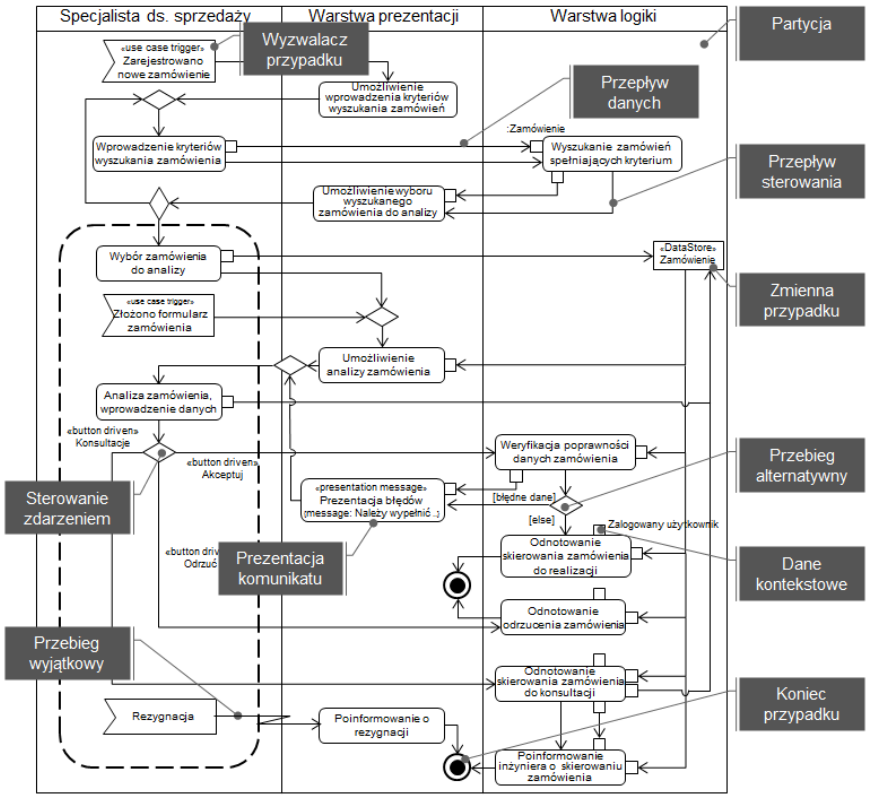
Opracowywane diagramy aktywności uzupełniane są o przesyłane pomiędzy poszczególnymi akcjami obiekty. W ramach prezentowanego podejścia stosowana jest notacja *pinowa* dla modelowania przepływu obiektów – wykorzystywaną konstrukcją są piny akcji (ang. *InputPin*, *OutputPin*). Przyjęty poziom szczegółowości modelowania danych to poziom klas modelu informacyjnego systemu (omówienie w dalszej części rozdziału) – jako typy pinów wskazywane

są klasy modelu odzwierciedlające przetwarzane dane. Dodatkowo, w celu zwiększenia czytelności diagramów, stosuje się szereg konwencji/rozszerzeń :

- w celu ułatwienia wielokrotnego odwoływania się do danych przetwarzanych w ramach opisywanej funkcjonalności (w szczególności wielokrotnej 'tymczasowej' – nie utrwalanej – modyfikacji danych) oraz zagwarantowania odwołania do aktualnych wartości, wykorzystywana jest konstrukcja składnicy danych (ang. *DataStore*) z nałożonym, zdefiniowanym w profilu AION, stereotypem «useCase variable»;
- dane kontekstowe wykorzystywane w przebiegu (np. informacja o zalogowanym użytkowniku korzystającym z modelowanej funkcjonalności) są modelowane jako pin wartości (ang. *ValuePina*) bez konieczności wskazania źródłowego przepływu;
- dla akcji, której celem jest odczytanie danych trwałych definiowane są: piny wejściowe (ang. *InputPin*) odwzorowujące parametry służące do identyfikacji odczytywanych danych oraz piny wyjściowe (ang. *OutputPin*) reprezentujące odczytane obiekty;
- dla akcji, której celem jest zapisanie danych trwałych definiowane są piny wejściowe odwzorowujące zapisywane obiekty, nie są modelowane piny wyjściowe chyba, że akcja zapisu wiąże się z modyfikacją obiektu, wówczas w celu aktualizacji danych przetwarzanych w ramach przypadku użycia modelowane są również piny wyjściowe reprezentujące zapisywane i jednocześnie modyfikowane obiekty;
- dla akcji, której celem jest utworzenie/zapisanie nowych obiektów definiowane są: piny wejściowe odwzorowujące parametry służące do utworzenia obiektu oraz piny wyjściowe reprezentujące utworzone/zapisane obiekty;
- dla akcji, której celem jest przetworzenie danych (np. wyliczenie pewnych wartości lub transformacja danych) definiowane są: piny wejściowe odwzorowujące parametry stanowiące podstawę do przetworzenia oraz piny wyjściowe reprezentujące obiekty stanowiące produkt przetworzenia;
- dla akcji realizowanej w partycji reprezentującej warstwę prezentacji definiowane są tylko piny wejściowe odwzorowujące obiekty prezentowane przez elementy interfejsu użytkownika;
- dla akcji realizowanej w partycji reprezentującej działania użytkownika definiowane są tylko piny wyjściowe odwzorowujące dane wprowadzone lub wskazane przez użytkownika.

Uzupełnienie przebiegu przypadku użycia o przetwarzane w ramach funkcjonalności dane związane jest z jednej strony z weryfikacją możliwości realizacji poszczególnych działań, czyli weryfikacją czy w modelu informacyjnym uwzględnione zostały wszystkie wymagane dane oraz ich właściwości, z drugiej dostarcza danych pozwalających na przeprowadzenie wymiarowania (szacowa-

nia złożoności) modelowanej funkcjonalności metodami Punktów Funkcyjnych (w tym [COSMIC14] bazującej na ‘przesunięciach’ danych).



Rysunek 1.24. Diagram aktywności opisujący przebieg przypadku użycia.

Konstrukcje diagramu aktywności są na potrzeby prezentowanej metodyki rozszerzane. Przykłady rozszerzeń pojawiły się już w opisie powyżej. Innym użytecznym rozszerzeniem jest wyróżnienie akcji warstwy prezentacji związanej z prezentacją komunikatu, którego celem jest poinformowanie użytkownika np. o statusie przetwarzania danych, w tym o błędach wprowadzonych danych lub możliwych opcjach wyboru. Akcję związaną z prezentacją komunikatu przeciąża się stereotypem «presentation message», a za pomocą zdefiniowanych dla stereotypu atrybutów wskazuje typ okna dialogowego (jeden ze zdefiniowanych dla systemu) oraz treść komunikatu (w szczególności parametryzowaną).

Przedstawiony na (Rysunek 1.24) diagram przedstawia przykładowy przebieg przypadku użycia zbudowany zgodnie z założeniami opisanymi w niniej-

szym rozdziale, przy wykorzystaniu stosowanych w prezentowanym podejściu konwencji i notacji.

Reguły i ograniczenia systemowe

Akcje przebiegu przypadku użycia realizowane w partycji reprezentującej logikę systemu są uzupełniane o informacje związane z regułami przetwarzania danych, w szczególności o reguły i ograniczenia systemowe nakładane na to przetwarzanie. Celem uzupełnienia jest doprecyzowanie logiki przetwarzania danych w ramach modelowanej funkcjonalności oraz jednoznaczne wskazanie reguł sterowania przebiegiem w oparciu o wartości cech przetwarzanych obiektów (sterowanie przebiegiem w oparciu o wybory użytkownika odwzorowywane jest za pomocą zdefiniowanych w profilu stereotypów nakładanych na krawędzie przepływu).

Reguły i ograniczenia systemowe wyznaczające sposób przetwarzania danych specyfikowane są jako wyrażenia języka *Object Constraint Language* [OMG14c] z zachowaniem zgodności z ograniczeniami przewidzianymi przez standard języka OCL dla definicji zachowania. Definiowanym zachowaniem jest precyzowana akcja przebiegu (z uwzględnieniem pinów akcji interpretowanych odpowiednio jako parametry wejściowe lub wyjściowe zachowania), klasyfikatorem kontekstowym zachowania jest modelowany przypadek użycia. W celu zwiększenia czytelności specyfikacji, stosuje się szereg konwencji/rozszerzeń notacji związanych z przyjętą definicją uniwersum dla specyfikowanego przetwarzania i wynikającą z tego interpretacją/rozróżnieniem pomiędzy przetwarzaniem danych trwałych a danych jeszcze nieutrwalonych (przetwarzanych w ramach realizacji funkcjonalności). Opracowane konwencje i rozszerzenia notacyjne nie stanowią przedmiotu niniejszego opracowania.

1.6.2.2 Mapa nawigacyjna

Struktura warstwy prezentacji oraz schemat nawigacji w podejściu firmy AION specyfikowane są za pomocą modelu *Mapy Nawigacyjnej*. Mapa obrazuje przyjęty podział interfejsu użytkownika na składowe (ekrany, panele, itp.) oraz ścieżki nawigacyjne (przejścia) pomiędzy poszczególnymi elementami interfejsu, którymi podąża użytkownik korzystający ze specyfikowanej funkcjonalności. W opisywanym podejściu mapa nawigacyjna tworzona jest dla każdego przypadku użycia. W sytuacji gdy jest zasadne, mapa nawigacyjna może być utworzona również dla całego systemu lub dla działań realizowanych w ramach określonej roli użytkownika.

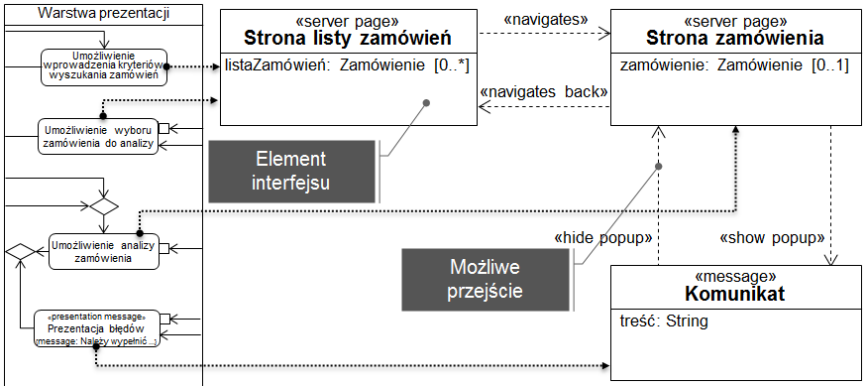
Stosowanym środkiem wyrazu modelu jest diagram klas języka UML, definiowany jako poddiagram specyfikowanego przypadku użycia, konstruowany zgodnie z przyjętymi w prezentowanym podejściu, opisanymi poniżej zasadami (Rysunek 1.25). Poszczególne klasy modelu odwzorowują elementy interfejsu użytkownika odpowiedzialne za realizację działań wynikających z przebiegu specyfikowanej funkcjonalności – są wnioskowane z akcji zawartych w partycji reprezentującej warstwę prezentacji diagramu aktywności specyfikującego przebieg przypadku użycia (śladowanie jest zapamiętywane w modelu poprzez powiązanie akcji ze składowymi mapy nawigacyjnej). Klasa reprezentująca element interfejsu użytkownika, uczestniczący w realizacji kilku przypadków użycia, jest definiowana raz w modelu, ale prezentowana na kilku diagramach map nawigacyjnych. Dla klas map nawigacyjnych zdefiniowane zostały w profilu AION stereotypy pozwalające na wyróżnienie typów poszczególnych elementów interfejsu użytkownika (stron webowych, formularzy desktopowych, okien, paneli dialogowych i innych komponentów będących zbiorami elementów graficznych wyświetlanych w tym samym czasie na ekranie). Możliwe ścieżki nawigacji pomiędzy elementami interfejsu modelowane są za pomocą skierowanych zależności (ang. *Dependency*) pomiędzy klasami odwzorowującymi te elementy, opisanych stereotypami zdefiniowanymi w profilu. Ustalone dla stereotypów właściwości pozwalają na definicję ewentualnych warunków przejść.

Zawartość informacyjna elementów interfejsu użytkownika czyli prezentowane na interfejsie dane są modelowane jako atrybuty klas map nawigacyjnych. Przyjęty poziom szczegółowości reprezentacji danych za pomocą atrybutów klas to poziom obiektów. Jeden z atrybutów (oznaczony ograniczeniem context/self) wskazuje obiekt główny (kontekstowy) interfejsu. Dla interfejsu prezentującego dane pojedynczego obiektu jest to rozważany obiekt, jeżeli dodatkowo na interfejsie prezentowane są dane powiązane z obiektem, dane te są wyliczane poprzez nawigację po modelu informacyjnym w specyfikacji prototypu interfejsu (omówienie w dalszej części rozdziału) – w ogólności nie stanowią atrybutów klasy mapy nawigacyjnej (wyjątek opisany poniżej). Dla interfejsu prezentującego kolekcję obiektów, obiektem głównym interfejsu jest rozważana kolekcja.

W sytuacji wielokrotnego odwoływania się do tych samych obiektów osiągalnych poprzez nawigację po modelu informacyjnym z obiektu głównego, w celu uproszczenia zapisu mapowania danych, zaleca się definicję atrybutów wnioskowanych klasy mapy nawigacyjnej – wyznaczonych za pomocą wyrażeń zdefiniowanych na podstawie atrybutów ‘zwykłych’ klasy.

Ponieważ elementy interfejsu użytkownika wynikają bezpośrednio z działań specyfikowanych za pomocą akcji przebiegów przypadków użycia, wymagana jest zgodność (co do typu oraz liczności) parametrów akcji z atrybutami powiązanej z akcjami klasy mapy nawigacyjnej:

- parametrów wejściowych akcji warstwy prezentacji odpowiedzialnej za prezentację danych,
- parametrów wyjściowych akcji warstwy reprezentującej działania użytkownika jeżeli poprzedzająca ją akcja warstwy prezentacji umożliwia edycję prezentowanych danych.



Rysunek 1.25. Mapa nawigacyjna powiązana z akcjami przebiegu przypadku użycia.

Mapy Nawigacyjnej nie utożsamia się z graficznym interfejsem użytkownika. Elementy graficzne są definiowane w ramach projektowania interfejsu użytkownika dla poszczególnych elementów mapy nawigacyjnej.

1.6.2.3 Projekt interfejsu użytkownika

Praktyka pokazała, iż istotną rolę w pozyskiwaniu rzeczywistych potrzeb użytkownika odgrywa interfejs graficzny. W związku z tym, prezentowana metodyka prowadzenia analizy systemowej została rozszerzona – w zakresie prac analitycznych zostało uwzględnione opracowywanie projektów interfejsów graficznych.

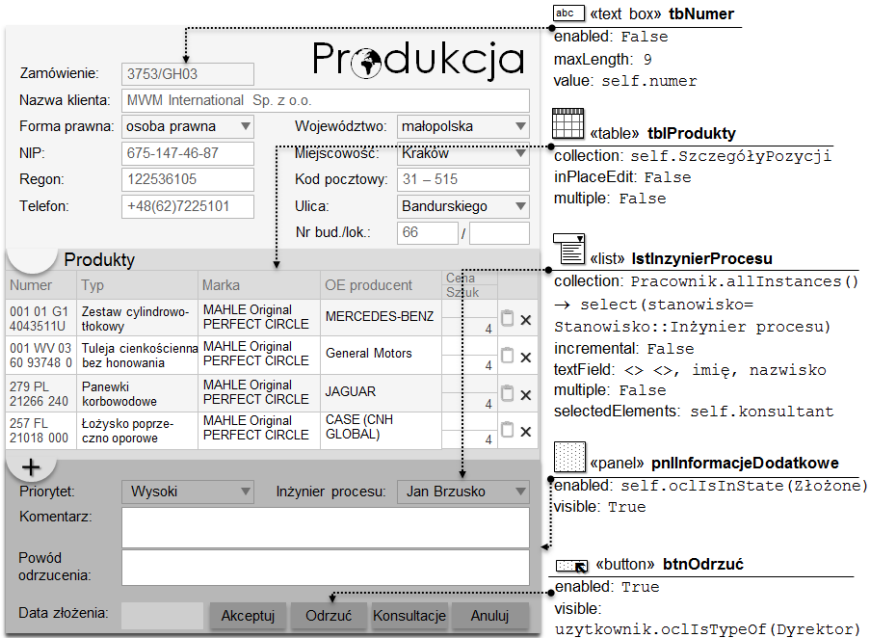
Projekt interfejsu jest poprzedzony podjęciem fundamentalnych decyzji co do środków technicznych jego implementacji, tj.: standardu GUI i/lub środowiska implementacji. Decyzje te warunkuje przede wszystkim specyfikacja wymagań niefunkcjonalnych projektu. W oparciu o przyjęte standardy i technologię projektowane są elementy graficzne interfejsu. Proces projektowania ele-

mentów ma charakter przyrostowy i iteracyjny. Zgodnie z założeniami metodyki rozpoczyna się on od opracowania kluczowych elementów graficznych interfejsu i polega na stopniowym uszczegóławianiu ich specyfikacji.

Stosowanym środkiem wyrazu jest dostępny w narzędziach modelowania, umożliwiający projektowanie elementów graficznych interfejsu, diagram interfejsu użytkownika. Diagram jest kojarzony z klasą mapy nawigacyjnej, reprezentującą specyfikowany element interfejsu – definiowany jako poddiagram danej klasy (Rysunek 1.26).

Zgodnie z przyjętymi w ramach prezentowanej metodyki zasadami modelowania prototypu interfejsu użytkownika, na diagramie uwzględniane są wszystkie wymagane kontrolki UI (również kontrolki widoczne warunkowo). Przykładowe dane są tak dobierane, aby odzwierciedlać rzeczywistą zawartość informacyjną interfejsu (dobrą praktyką jest prezentowanie w ramach przykładowych danych wartości ‘granicznych’).

W przypadku tabel z możliwą dynamiczną edycją danych projektowana jest również postać wiersza edytowanego. Wiersz projektowany jest jako osobna tabela (zachowująca zgodność kolumn jeżeli taka przy edycji zostaje) z jednym wierszem zawierającym kontrolki służące edycji danych.



Rysunek 1.26. Projekt interfejsu użytkownika ze specyfikacją elementów składowych.

Specyfikacja elementów składowych interfejsu użytkownika

Uwzględnione na diagramie interfejsu użytkownika kontrolki są doprecyzowywane – uzupełniana jest specyfikacja ich cech, np. precyzowany warunek widoczności lub dostępności kontrolki, określana maksymalną liczbę znaków możliwych do wprowadzenia w kontrolce pola tekstowego, ustalany typ kontrolki wyboru elementów z listy (wybór wielu czy tylko jednego elementu). W tym celu, w ramach profilu AION zdefiniowane zostały stereotypy odzwierciedlające właściwości poszczególnych typów kontrolek (np. «button», «combo box», «text box», «table») rozszerzone o atrybuty specyfikujące cechy charakterystyczne kontrolek i umożliwiające ich uszczegółowienie (np. *visible*, *enabled*, *multiple*, *maxLength*). Założono, iż ograniczenia nałożone na kontrolki poprzez ustalenie wartości atrybutów mają naturę inwariantów – muszą być zawsze spełnione dla projektowanego interfejsu. Przyjętą formą zapisu wartości atrybutów są wyrażenia języka OCL.

Jednym z kluczowych elementów projektowania interfejsu użytkownika jest zdefiniowanie powiązania elementów graficznych interfejsu z danymi dostępnymi w systemie. Przyjęty w ramach prezentowanej metodyki kontekst dla danych powiązanych z kontrolkami stanowi klasa mapy nawigacyjnej, z którą skojarzony jest projektowany interfejs. Prezentowane na interfejsie dane specyfikowane są jako atrybuty klasy.

Do definicji powiązania zawartości poszczególnych kontrolek interfejsu z danymi (z właściwościami obiektów wskazanych przez atrybuty klasy mapy nawigacyjnej) wykorzystywane są wyróżnione atrybuty zdefiniowanych w ramach profilu AION stereotypów zdefiniowanych dla kontrolek – w zależności od typu kontrolki jest to: dla pola tekstowego atrybut *value*, dla kontrolek wprowadzania danych logicznych (pola wyboru oraz przycisku opcji) – *checked*, dla listy wyboru – *selectedElements*, dla tabeli – *collection*, dla kolumny tabeli – *textField*. Dodatkowo wyróżniono atrybut *defaultValue* specyfikujący wartość domyślną jaką przyjmuje kontrolka w sytuacji kiedy skojarzona z nią cecha prezentowanych danych nie może być wyznaczona. Zakłada się, że zdefiniowana wartość będzie prezentowana przez kontrolkę przy wyświetleniu interfejsu w przypadku tworzenie nowego obiektu.

W przypadku tworzenia w ramach przypadku użycia nowego obiektu, atrybuty kontrolek służące mapowaniu danych wskazują pola, którym podczas zapisu tworzonego obiektu przypisywane są wartości z kontrolek. W przypadku modyfikacji istniejącego obiektu, atrybuty mapowania danych wskazują pola źródłowe, z których dane są pobierane do prezentacji przy uruchomieniu interfejsu oraz pola, którym przypisywane są wartości podczas zapisu obiektu.

Zgodnie z założeniami metodyki zagadnieniem, które również należy uwzględnić podczas projektowania interfejsu użytkownika są warunki poprawności danych wprowadzanych przez użytkownika za pomocy projektowanych elementów interfejsu. Ogólne zagadnienia związane z kontrolą poprawności danych, np.:

- sposób prezentacji komunikatów informujących o błędach (np. prezentacja komunikatów obok kontrolek wprowadzania danych niespełniających zdefiniowanych warunków poprawności czy prezentacja listy błędów w oknie dialogowym),
- szablon konstrukcji komunikatu informującego o błędach,
- czas walidacji (np. walidacja następuje po przejściu do innej kontrolki czy przy próbie wyjścia ze strony/formularza),

są ustalane jako standard dla całej aplikacji. Niezależnie modelowane są tylko ewentualne odstępstwa od przyjętego, standardowego rozwiązania.

Dla zagadnień poprawności danych związanych z poszczególnymi kontrolkami zdefiniowano w profilu AION stereotypy rozszerzające właściwości kontrolek wprowadzania danych. Zdefiniowane stereotypy pozwalają m.in. zapisać ograniczenie obligatoryjności danych («required field validator»), wymaganie wprowadzenia wartości mieszczącej się w zadanym zakresie («range validator»), czy wymaganie wprowadzenia tekstu spełniającego wyrażenie regularne («regular expression validator»). Zdefiniowane dla stereotypów walidacji danych atrybuty pozwalają sprecyzować nakładane na kontrolki ograniczenia (np. atrybut *errorMessage* pozwala zdefiniować komunikat prezentowany jeżeli wpisana przez użytkownika wartość nie spełni postawionych wymagań poprawności).

Zaproponowanym w ramach metodyki sposobem reprezentacji szczegółowej specyfikacji elementów interfejsu użytkownika (ograniczeń nakładanych na kontrolki, wymagań poprawności czy zasad mapowania danych prezentowanych przez kontrolki z danych dostępnych w systemie) są wymagania języka SysML (ang. *Requirement*). Dla kontrolek, które wymagają uszczegółowienia definiowane są wymagania opisujące ich cechy. Na zdefiniowane wymagania nakładane są stereotypy profilu zgodne z typem opisywanej kontrolki oraz ustalane wartości powiązanych ze stereotypami atrybutów, w szczególności atrybutów mapowania danych (Rysunek 1.26).

1.6.3 Model informacyjny

Równolegle z opracowywaniem modelu przypadków użycia tworzony jest model informacyjny systemu. Model pozwala na wyrażenie aspektu statycznego

modelowanego systemu – odwzorowanie struktury informacji/danych przetwarzanych w systemie niezbędnych do realizacji funkcjonalności oferowanych przez system.

Model zawiera charakterystyki elementów należących do dziedziny problemu objętej przedsięwzięciem informatycznym wraz z definicjami statycznych relacji zachodzących pomiędzy nimi. Tworząc model podejmowane są kluczowe decyzje o kształcie systemu i sposobie, w jaki będzie on odwzorowywał rzeczywistość biznesową, a w konsekwencji wspierał swoich użytkowników.

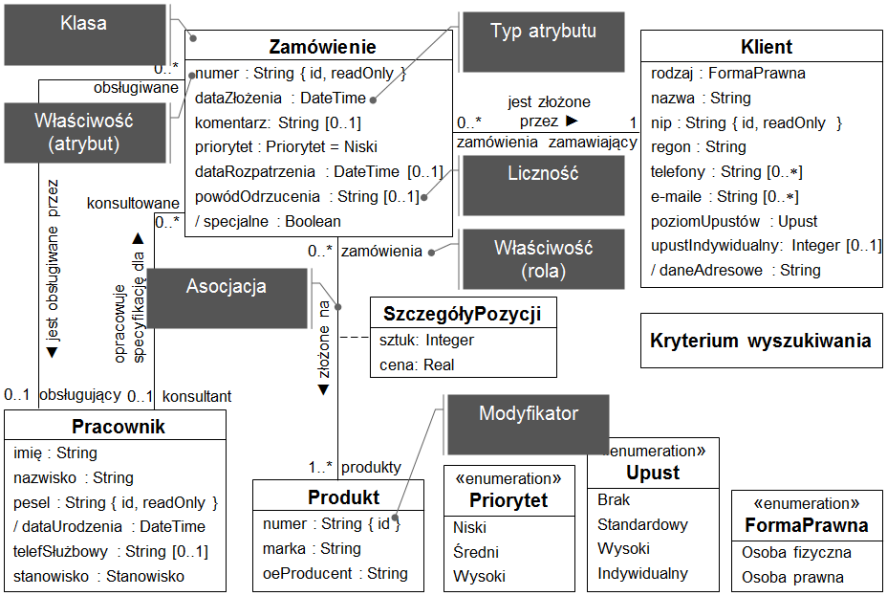
Identyfikacja elementów modelu informacyjnego systemu ma charakter przyrostowy i iteracyjny. Zgodnie z założeniami metodyki rozpoczyna się od wskazania kluczowych elementów dziedziny problemu wokół, których realizowane jest przetwarzanie w systemie (zwykle na podstawie opracowanego na etapie analizy biznesowej modelu koncepcji biznesowych) i polega na stopniowym uzupełnianiu modelu o nowe, identyfikowane podczas analizy funkcjonalności elementy niezbędne do jej realizacji oraz uszczegóławianiu specyfikacji tych elementów. Zawarte w modelu elementy oraz ich specyfikacje są ściśle powiązane z pozostałymi artefaktami analizy systemowej – wynikają bezpośrednio z przebiegów przypadków użycia, zaprojektowanych interfejsów użytkownika oraz zdefiniowanych reguł i ograniczeń systemowych determinujących przetwarzanie systemu.

Model informacyjny systemu uwzględnia przede wszystkim dane trwałe, aczkolwiek mogą być uwzględnione w nim również dane tymczasowe jeżeli są one wykorzystywane w realizacji funkcjonalności systemu (np. kryteria wyszukiwania, struktury danych raportów).

Wykorzystywanym środkiem wyrazu umożliwiającym zapis przyjętej struktury zawartości informacyjnej systemu jest diagram klas języka UML (Rysunek 1.27). Klasa odzwierciedla przyjętą definicję rozważanego pojęcia dziedziny problemu, opracowaną z uwzględnieniem przyjętej perspektywy oraz odpowiedniego (wynikającego z dostarczanej przez system funkcjonalności) poziomu szczegółowości. Stanowi zapis/model tej definicji. Klasa precyzuje odzwierciedlane pojęcie poprzez wskazanie jego znaczenia (semantyki) oraz cech strukturalnych bytów specyfikowanych (klasyfikowanych) przez pojęcie. Cechy strukturalne pojęcia określane są za pomocą właściwości klasy. Zestaw właściwości klasy wyznacza zawartość informacyjną jaką niosą ze sobą obiekty klasy przetwarzane w projektowanym systemie.

Przyjęty w prezentowanej metodyce poziom szczegółowości opisu właściwości klas uwzględnia specyfikację ograniczeń nakładanych na modelowane cechy, w konsekwencji sygnatury właściwości (atrybutów oraz ról wynikają-

cych z asocjacji) uzupełniane są o: typ – precyzujący dopuszczalne wartości, jakie mogą być przypisane właściwości; liczność – definiującą liczbę tych wartości; predefiniowane w języku UML modyfikatory – precyzujące wymaganie unikalności, niezmienności właściwości, ograniczenie podzbioru czy wnioskowalność.



Rysunek 1.27. Model informacyjny systemu.

Ważnym elementem modelu opracowywanego zgodnie z zasadami prezentowanej metodyki są prawidłowo konstruowane nazwy elementów modelu (klas, właściwości – atrybutów i ról oraz asocjacji). Nazwy powinny precyzować znaczenie etykietowanego elementu w ramach dziedziny problemu, tak by prezentowana informacja była interpretowana jednoznacznie, a diagram zrozumiały, czytelny i możliwy do interpretacji w sposób intuicyjny. Do zapisu nazw stosowane są konwencje zgodne z zaproponowanymi w specyfikacji języka UML.

W kolejnych krokach etapu analizy model informacyjny systemu jest uszczegóławiany tak, by docelowo uwzględniać:

- specyfikacje ograniczeń-niezmienników nałożonych na elementy modelu: klasy, właściwości, asocjacje,
- definicje wartości początkowych/domyślnych dla właściwości,
- definicje algorytmów wyliczenia dla właściwości wnioskowanych.

Powyższe elementy definiowane są jako ograniczenia (ang. *Constraint*), opisujące odpowiednio klasy, atrybuty lub role, z treścią zapisaną w postaci wyrażeń języka OCL.

Dla klas modelu informacyjnego nie są definiowane operacje. Wyjątek stanowią operacje pomocnicze definiowane w kontekście wyrażeń OCL – operacje «OclHelper».

W celu uproszczenia, złożony lub obszerny model informacyjny może być podzielony na części. Do reprezentacji przyjętej struktury modelu wykorzystuje się diagram pakietów.

1.7 Podsumowanie

Powołanie się w niniejszym dokumencie na standardy, oparcie rozważań o dobrze określony analityczny proces produkcyjny i przywołanie często stosowanych technik miało na celu pokazanie, iż stosowanie spójnych i dedykowanych narzędzi do prowadzenia prac analitycznych może przyczynić się do osiągania różnego rodzaju korzyści biznesowych, począwszy od poprawy jakości produktów, a na docelowym obniżeniu kosztów realizacji prac analitycznych zakończywszy.

Wszystkie z omówionych w dokumencie standardów, z punktu widzenia prowadzenia prac analitycznych, dostarczają ontologie służące do opisu dedykowanego aspektu rzeczywistości. Zakładając, iż ontologia jest kompletna, ewentualnie w wystarczającym, z praktycznego punktu widzenia, zakresie pokrywa obszar zastosowań, wówczas wysiłek, jaki należy poświęcić na wypracowanie metod opisu danego obszaru skraca się do minimum potrzebnego do zdobycia kompetencji w stosowaniu standardu. W dalszej kolejności, posiadając wystarczająco kompletną ontologię, w dużym stopniu zmniejszamy ryzyko wykonania niepełnych prac analitycznych i wszystkich tego późniejszych konsekwencji. Następną korzyścią o jakiej warto wspomnieć przy okazji niektórych standardów, jest możliwość skonstruowania procesu wytwórczego wokół rodzajów artefaktów opisanych w metamodelu języka. Przykładem takiego standardu jest BMM, opisany w rozdziale Strategia biznesu. Zdefiniowane pojęcia misji, wizji, celów strategicznych i taktycznych, strategii i taktyk, połączone dobrze określonymi relacjami narzucają określoną kolejność prowadzenia prac mających na celu stworzenie planu biznesowego. Uzupełnienie tych pojęć o techniki pozwalające na uzyskiwanie informacji wymaganych do stworzenia modelu skutkuje powstaniem narzędzi facylitacyjnych, mogących znaleźć zastosowanie podczas warsztatów analitycznych. Oparcie się o jednolity standard powoduje, iż powstające narzędzia są stabilne, jednolite, spójne, a co za tym idzie, możli-

we do stosowania przez różne zespoły, co z czasem zaowocuje nabyciem wysokich kompetencji przez wszystkich interesariuszy, wymaganych do aktywnego uczestnictwa w projektach realizowanych w organizacji, tym samym zmniejszając koszty prowadzenia prac.

Powiązanie (bezpośrednio, bądź pośrednio) komplementarnych standardów w ramach jednego procesu produkcyjnego, uzupełnionego o techniki wymagane do wypracowania artefaktów opisanych standardami daje organizacji dodatkowe korzyści. Pierwszą z nich jest znaczne zwiększenie przewidywalności prac, co jest możliwe do osiągnięcia dzięki temu, iż prace prowadzone są według jednolitych zasad i z wykorzystaniem jednolitych i logicznie spójnych środków wyrazu a czego korzyści są trudne do przecenienia. Ta cecha procesu produkcyjnego jest istotna zarówno dla zespołów roboczych, realizujących prace jak i pozostałych interesariuszy, którzy w pracach uczestniczą, bądź muszą być zaznajamiani z niektórymi z ich efektów. Drugą, jest minimalizacja wysiłku potrzebnego do stałego rozwoju procesu produkcyjnego poprzez korzystanie z wypracowanych już rozwiązań cząstkowych. W pewnym sensie, Centra Doskonałości odpowiedzialne za rozwój stosowanego procesu wytwórczego, stają się integratorami standardów, ograniczając wysiłek potrzebny na wypracowywanie cząstkowych rozwiązań językowych.

Bibliografia

[COSMIC14] The Common Software Measurement International Consortium FFP, Measurement Manual Version 4.0, 2014,

[Oul05] Martyn Ould. Business Process Management – A rigorous approach., Meghan Kiffer Press, 30 Styczeń, 2005,

[OMG09] OMG, Model Driven Architecture, Guide Version 1.0.1, 2009

[OMG12] OMG, Systems Modeling Language, Version 1.3, 2012

[OMG13a] OMG, Business Process Model and Notation, Version 2.0, 2013

[OMG13b] OMG, Semantics of Business Vocabulary and Business Rules, Version 1.2, 2013

[OMG13c] OMG, Unified Modeling Language, Version 2.5, 2013

[OMG14a] OMG, Decision Model and Notation, Version 1.0, 2014

[OMG14b] OMG, Business Motivation Model, Version 1.2, 2014

[OMG14c] OMG, Object Constraint Language, Version 2.4, 2014

[RST] RuleSpeak, http://www.rulespeak.com/en/

[RX] RuleXpress , http://www.rulearts.com/rulexpress

[SD] Sapiens DECISION . http://www.sapiensdecision.com/

[TDM] The Decision Model, http://kpiusa.com/index.php/The-Decision-Model/the-decision-model.html

[VPEE] Visual Paradigm Enterprise Edition, www.visual-paradigm.com,

Rozdział 2

Wybrane aspekty realizacji rozwiązań teleinformatycznych dla Systemu Bezpieczeństwa Narodowego RP

Istniejące obecnie w Polsce środowisko systemów wsparcia teleinformatycznego organów państwa, administracji rządowej i samorządowej, służb i innych podmiotów realizujących zadania w ramach Systemu Bezpieczeństwa Narodowego RP (SBN RP) było budowane niezależnie w ramach inicjatyw projektowych prowadzonych przez poszczególne sektory czy resorty. Jedynie niewielka część tych projektów była koordynowane centralnie. Wynika to z wieloletnich zaniedbań w zakresie działań zmierzających do uporządkowania i stosownego umocowania prawnego kwestii związanych z budową nowych i rozwijaniem istniejących systemów teleinformatycznych w Polsce oraz brakiem jednolitej metodyki dedykowanej realizacji tego typu systemów. W niniejszym rozdziale dokonano analizy wybranych aspektów mających najbardziej istotny wpływ na proces realizacji systemów teleinformatycznych dedykowanych SBN RP oraz na powodzenie związanych z tym przedsięwzięć. Dokonano również przeglądu i oceny praktyk stosowanych w tym zakresie w różnych państwach świata. Zaprezentowano także wyniki oceny kierunku ewolucji metodyk realizacji tego typu systemów. Umożliwiło to m.in. określenie, które spośród stosowanych obecnie na świecie podejść i metodyk realizacji systemów teleinformatycznych mogą stanowić podstawę zbudowania metodyki dedykowanej realizacji tego typu systemów wsparcia teleinformatycznego Systemu Bezpieczeństwa Narodowego RP.

Zapewnienie bezpieczeństwa państwa oraz jego obywateli jest jednym z nadrzędnych interesów narodowych Rzeczypospolitej Polskiej. Obecna organizacja bezpieczeństwa narodowego RP, określana umownie jako „System Bezpieczeństwa Narodowego RP" (SBN RP) polega na istnieniu odrębnych systemów operacyjnych, których integracja jest bardzo utrudniona ze względu na ich rozłączność organizacyjną i funkcjonalną [PR13a]. Ponadto niemal w każdym z tych systemów występują te same lub prawie te same elementy funkcjonujące według odrębnych reguł ustalonych w przepisach prawa krajowego. W przypadku wystąpienia sytuacji wykraczających poza możliwości poszczególnych służb resortowych istnieje potrzeba sprawnego kierowania tymi służbami i ich aparatem wykonawczym poprzez odpowiednio zorganizowany system bezpieczeństwa działający na różnych szczeblach (gminy, powiatu, województwa i centralnym), składający się z organów kierowania i sfery wykonawczej, który będzie sprawnie funkcjonował zarówno w czasie pokoju, kryzysu, jak i wojny [KI13].

W świetle powyższego konieczne staje się z jednej strony dokonanie zmian w istniejących uregulowaniach prawnych, które powinny zmierzać do stworzenia sieciowej struktury bezpieczeństwa, poprawy skuteczności działania, zwięk-

szania ekonomiczności wprowadzanych rozwiązań oraz jakościowego rozwoju systemu [BB13]. Z drugiej strony konieczne jest zbudowanie odpowiedniego podejścia do zarządzania transformacją podmiotów realizujących zadania na rzecz SBN RP, które pozwoli na skuteczne przeprowadzenie tego procesu zarówno w warstwie organizacyjnej, informacyjnej, jak i technicznej. Docelowym efektem tej transformacji będzie budowa zdefiniowanego prawnie, spójnego organizacyjnie, funkcjonalnie i informacyjnie SBN RP zdolnego do realizacji misji na rzecz ochrony i obrony państwa jako instytucji politycznej i zapewniania trwałego oraz wolnego od zakłóceń istnienia i rozwoju społeczeństwa. Wymaga to m.in. opracowania odpowiedniej metodyki transformacji SBN RP, która uwzględni w sposób całościowy kwestie związane z zapewnianiem interoperacyjności tworzonych rozwiązań i zapewni ich zgodność z powstającym jednolitym systemem informacyjnym państwa [PR13a] oraz pozwoli na zbudowanie zintegrowanego systemu teleinformatycznego wspomagania kierowania SBN RP.

W niniejszym rozdziale dokonano analizy wybranych aspektów mających najbardziej istotny wpływ na proces realizacji systemów teleinformatycznych dedykowanych SBN RP oraz na powodzenie związanych z tym przedsięwzięć.

2.1 System Bezpieczeństwa Narodowego w Polsce

Termin bezpieczeństwo pochodzący od łacińskiego słowa *sine cura = securitas* (bez pieczy), w potocznym rozumieniu jest ujmowany negatywnie, jako brak zagrożeń, zaś w definicjach słownikowych zazwyczaj występuje w ujęciu pozytywnym utożsamiając bezpieczeństwo z pewnością, jako stanem przeciwstawnym zagrożeniom. Z kolei w nauce termin ten jest definiowany w sposób zależny od poglądów konkretnego autora i ściśle związany z wyznawanym przez niego podejściem teoretycznym z zakresu nauki o stosunkach międzynarodowych [ZZ10].

Bezpieczeństwo narodowe to zdolność państwa i jego społeczeństwa do zapewnienia warunków jego istnienia i rozwoju, integralności terytorialnej, niezależności politycznej, stabilności wewnętrznej oraz jakości życia. Zdolność ta jest kształtowana poprzez działania polegające na wykorzystaniu szans, podejmowaniu wyzwań, redukowaniu ryzyka oraz eliminowaniu zagrożeń zewnętrznych i wewnętrznych, co zapewnia trwanie, tożsamość, funkcjonowanie i swobody rozwojowe państwa i społeczeństwa [ZZ10].

Na podstawie Białej Księgi Bezpieczeństwa Narodowego RP można stwierdzić, że: "*system bezpieczeństwa narodowego (bezpieczeństwa państwa) stanowi całość sił (podmiotów), środków i zasobów przeznaczonych przez państwo do*

realizacji zadań w dziedzinie bezpieczeństwa, odpowiednio do tych zadań zorganizowana (w podsystemy i ogniwa), utrzymywana i przygotowywana. Składa się z podsystemu (systemu) kierowania i szeregu podsystemów (systemów) wykonawczych, w tym podsystemów operacyjnych (obronny i ochronne) i podsystemów wsparcia (społeczne i gospodarcze)" [BB13]. W toku przeprowadzonych prac badawczych, bazujących m.in. na wyżej cytowanej Białej Księdze Bezpieczeństwa Narodowego RP, Pawłowski i in. dochodzą z kolei do następującej szerszej konkluzji: "*System Bezpieczeństwa Narodowego (SBN) to celowo wyodrębniony z systemu państwowego, kolektywny zbiór organów władzy i administracji publicznej, innych organów państwowych, sił zbrojnych, przedsiębiorców i innych jednostek organizacyjnych, organizacji społecznych i obywateli wykonujących działania na rzecz zapewnienia bezpieczeństwa narodowego, wewnętrznie skoordynowanych i wzajemnie powiązanych ze sobą za pomocą relacji porządkujących, ze względu na realizowaną misję, którą jest obrona i ochrona państwa jako instytucji politycznej, terytorialnej i społecznej, a także zapewnienie wolnych od zakłóceń warunków bytu i rozwoju jednostek i całego społeczeństwa oraz ochrona życia i zdrowia ludzi, ich dóbr (materialnych i niematerialnych) i środowiska naturalnego we wszystkich stanach funkcjonowania państwa (w czasie normalnym, w tym w czasie kryzysu oraz w stanach nadzwyczajnych*)" [PA14].

Rolą SBN RP jest zapewnienie nienaruszalnego przetrwania państwa jako instytucji politycznej oraz trwałego i wolnego od zakłóceń istnienia i rozwoju społeczeństwa poprzez efektywne zaangażowanie i wykorzystanie dostępnych sił, środków i zasobów do realizacji działań zmierzających do redukowania ryzyka w dziedzinie bezpieczeństwa, eliminacji zagrożeń oraz prowadzenie aktywnej polityki wykorzystywania pojawiających się szans [PR14a]. Z kolei do poprawnej realizacji tych zadań niezbędne jest stworzenie zintegrowanego systemu teleinformatycznego wspomagania kierowania SBN RP.

2.2 Uwarunkowania prawne realizacji systemów ICT na rzecz SBN

Wywieranie wpływu na sposób realizacji systemów ICT jest możliwe m.in. poprzez prawne uregulowanie stosowania metodyk, technik i narzędzi służących temu celowi. Wiele państw posiada stosowne regulacje legislacyjne, które oddziałują na inicjatywy związane z budową i rozwojem systemów wsparcia teleinformatycznego dedykowanych administracji publicznej oraz szeroko pojętemu sektorowi bezpieczeństwa. W niniejszym punkcie przedstawione zostały praktyki stosowane obecnie na świecie w celu zapewnienia ram prawnych dla

realizacji systemów ICT na rzecz administracji publicznej oraz sektora bezpieczeństwa.

2.2.1 Rozwiązania stosowane w wybranych państwach świata

Pionierami w zakresie stosowania rozwiązań legislacyjnych w obszarze realizacji systemów ICT są Stany Zjednoczone, Korea Południowa, Australia i Nowa Zelandia. Rozwiązania funkcjonujące w tym zakresie w USA stanowią swoisty punkt odniesienia, który stanowił bazę do podjęcia analogicznych działań, zmierzających do wprowadzenia stosownych regulacji prawnych, zarówno na poziomie Unii Europejskiej, jak również poszczególnych państw Europy. Najbardziej dojrzałe regulacje w zakresie stosowania metodyk realizacji systemów wsparcia teleinformatycznego zarówno administracji, jak również sił zbrojnych i służb federalnych posiadają Stany Zjednoczone. Kwestie te są uregulowane w tamtejszym prawodawstwie w formie następujących ustaw:

- Government Performance and Reform Act z 1993;
- Paperwork Reduction Act z 1995;
- Clinger-Cohen Act z 1996;
- The Government Paperwork Elimination Act z 1998;
- Federal Information Security Management Act 2002;
- Electronic Government z 2002

oraz następujących okólników:

- Preparation, Submission and Execution of the Budget;
- Management of Federal Information Resources.

Przytoczone powyżej akty prawne nakładają na jednostki rządowe USA obowiązek podporządkowania celom strategicznym państwa stosowanych przez siebie rozwiązań teleinformatycznych. Nakazują one również usprawnienie procedur związanych z wyborem i zarządzaniem systemami teleinformatycznymi przez poszczególne jednostki. Przekłada się to na powiązanie realizacji systemów wsparcia teleinformatycznego (w tym również działających w zakresie SBN Stanów Zjednoczonych) z planowaniem strategicznym, planowaniem efektywności, planowaniem finansowym oraz procesem tworzenia budżetu państwa. Jako narzędzie osiągnięcia tego celu w zakresie realizacji systemów wsparcia teleinformatycznego wskazywane są ramy architektury korporacyjnej FEA (ang. *Federal Enterprise Architecture*).

2.2.2 Rozwiązania stosowane w Unii Europejskiej

W ramach Unii Europejskiej istnieje szereg rozwiązań o charakterze prawnym, które oddziałują na zasady zastosowania metodyk realizacji systemów

wsparcia teleinformatycznego SBN. Punkt wyjścia do jest zaprezentowana w 2010r. Agenda Cyfrowa [KE10a], która została opracowana przez Komisję Europejską i stanowi jeden z filarów Strategii Europa 2020 [KE10b]. Autorzy Agendy Cyfrowej wskazują na potrzebę dostosowania europejskich ram prawnych, w zakresie ustalania standardów dotyczących realizacji systemów teleinformatycznych na poziomie lokalnym i paneuropejskim, do szybkozmiennych warunków technologicznych, co pozwoli na zapewnianie interoperacyjności systemów tworzonych na w/w poziomach. W związku z tym Komisja Europejska dokonała przeglądu europejskiej polityki normalizacyjnej zgodnie z wytycznymi przedstawionymi w Białej Księdze *"Modernizacja normalizacji technologii informacyjno-komunikacyjnych w UE"* [KE09]. Zaowocowało to ogłoszeniem Europejskiej Strategii Interoperacyjności (ang. *European Interoperability Strategy - EIS*) [KE10c] oraz Europejskich Ram Interoperacyjności (ang. *European Interoperability Framework - EIF*) [KE10d].

Zadaniem EIS jest zapewnianie ładu (ang. *governance*) w zakresie tworzenia i rozwoju systemów teleinformatycznych administracji poszczególnych krajów członkowskich poprzez podejmowanie wynikających z niej działań zmierzających do opracowania i wdrożenia reguł i dobrych praktyk w tym obszarze. Z kolei zaproponowane przez Komisję Europejską ramy EIF koncentrują się na kwestiach związanych z podejściem do projektowania europejskich usług publicznych, pozostawiając kwestie szczegółowych rozwiązań poszczególnym państwom członkowskim. Państwa członkowskie odpowiadają tym samym za stworzenie spójnych z EIF krajowych ram interoperacyjności (ang. *National Interoperability Frameworks - NIF*), a Komisja Europejska monitoruje ich działania za pośrednictwem inicjatywy NIFO, publikującej cykliczne raporty podsumowujące stan budowy podstaw prawnych interoperacyjności w poszczególnych państwach UE.

2.2.3 Rozwiązania stosowane w Polsce

Pierwsze realne działania związane z narzuceniem porządku prawnego w zakresie tworzenia systemów wsparcia teleinformatycznego państwa zostały podjęte z dniem 1 lipca 2002r. poprzez zmianę ustawy o działach administracji rządowej (Dz. U. z 2003r., nr 159, poz. 1548), co spowodowało, że informatyzacja stała się jednym z nich. Następnym krokiem było wprowadzenie pojęcia interoperacyjności do ustawy *prawo telekomunikacyjne* (Dz. U. z 2004r., nr 171, poz. 1800). Ustawa ta zawężała jednak to zagadnienie wyłącznie do kwestii technicznych. W dalszej kolejności powstała ustawa *o informatyzacji działalności podmiotów realizujących zadania publiczne* (Dz. U. z 2005r.,

nr 64, poz. 565), której projekt rządowy złożony 26 sierpnia 2003r. został poddany krytyce przez autorów opinii zamówionych przez Kancelarię Sejmu. Ustawa o informatyzacji zmieniła siedemnaście innych ustaw, natomiast jej gruntowna nowelizacja została dokonana 12 lutego 2010r. Przepisy tej ustawy stosuje się do wszystkich określonych w niej podmiotów realizujących zadania publiczne. Nie stosuje się ich jednak do przedsiębiorstw państwowych, spółek handlowych, służb specjalnych w rozumieniu art. 11 ustawy z dnia 24 maja 2002 r. *o Agencji Bezpieczeństwa Wewnętrznego oraz Agencji Wywiadu* (Dz. U. z 2010r. Nr 29, poz. 154, z późn. zm.), Kancelarii Sejmu, Kancelarii Senatu, Kancelarii Prezydenta Rzeczypospolitej Polskiej oraz Narodowego Banku Polskiego, poza przypadkami, gdy w związku z realizacją zadań przez te podmioty istnieje obowiązek przekazywania informacji do i od podmiotów niebędących organami administracji rządowej. Normy dotyczące realizacji systemów wsparcia teleinformatycznego SBN RP są rozproszone pomiędzy wiele różnych aktów prawnych. Dodatkowo akty te traktują sprawy z zakresu budowy potencjału teleinformatycznego państwa w sposób wycinkowy. Jednak nawet pomimo obserwowanego w ostatnich latach dużego nasilenia prac w tym zakresie nadal brakuje spójnego "prawa informatycznego", które w sposób całościowy porządkowałoby przestrzeń związaną z uregulowaniem zasad budowy systemów wsparcia teleinformatycznego wykorzystywanych w Polsce w sektorze administracji publicznej oraz w obszarze szeroko rozumianego bezpieczeństwa państwa. Należy zauważyć, że Polska nie doczekała się również do tej pory stworzenia dedykowanych metod i narzędzi pozwalających na zbudowanie kompletnej i spójnej architektury obejmującej podmioty biorące udział w realizowaniu funkcji SBN, wspomagającej podejmowanie racjonalnych decyzji w zakresie organizacji SBN RP i jego wsparcia teleinformatycznego.

2.3 Cyfrowe Państwo i systemowa interoperacyjność SBN

Zagadnienia dotyczące interoperacyjności w obszarze administracji publicznej oraz w obszarze obronnym i bezpieczeństwa państwa stanowią obecnie jedno z głównych wyzwań związanych transformacją SBN RP. Z jednej strony wiąże się to z rozwojem w Polsce idei tzw. Cyfrowego Państwa, zaś z drugiej strony łączy się z koniecznością dotrzymania narzucanego RP tempa i zakresu transformacji narodowej zmierzającej w kierunku osiągnięcia zdolności sieciocentrycznych i zapewnienia interoperacyjności z NATO i UE. Dotyczy to interoperacyjności z NATO i/lub UE zarówno jako instytucjami ponadnarodowymi w szeroko pojętym obszarze administracyjnym, jak również w ramach wspólnych działań prowadzonych pod ich egidą (np. z zakresu zarządzania i reago-

wania kryzysowego, ochrony zdrowia obywateli, działań o charakterze obronnym itp.).

Polska zmierza nieuchronnie w kierunku realizacji idei Cyfrowego Państwa, które jest postrzegane i funkcjonuje jako jedna, spójna całość sterowana misją i zorientowana na dostarczanie wartości publicznej zarówno dla obywateli jak i przedsiębiorstw [SO12]. Cyfrowe Państwo bazuje na czterech podstawowych ideach [BO12]:

- pozbawionym barier przepływie informacji - rozumianym jako praktyczne wdrożenie interoperacyjności manifestujące się na poziomie celów jednostek wchodzących w skład państwa;
- postawieniu obywatela w centrum uwagi - rozumianym jako zdolność państwa do świadczenia obywatelowi/przedsiębiorcom kompletnych usług, przynoszących mu wymierne korzyści i rozwiązujących jego konkretne problemy życiowe i/lub biznesowe;
- otwartości i partycypacji - wyrażającej się poprzez przejrzystość działań podejmowanych przez władzę publiczną i możliwość uczestniczenia obywateli, przedsiębiorców i innych organizacji w tworzeniu przepisów prawa i podejmowaniu decyzji publicznych;
- prawie do prywatności - manifestującym się poprzez ochronę zintegrowanych danych o obywatelach przed niewłaściwym użyciem, zarówno na poziomie przepisów prawa, jak również w warstwie technicznej.

Realizacji tych idei sprzyjają różne inicjatywy podejmowane od ponad 20 lat, w tym zarówno te zmierzające do zmian kulturowych i organizacyjnych w administracji, jak również te wiążące się z zastosowaniem technologii informatycznych. Należy zauważyć, że przez ten czas informatyzacja w Polsce przechodziła różne fazy realizowane według różnych modeli - od rozproszonej realizacji projektów, po próby skoordynowania podejmowanych działań. Ponadto w tym okresie zrealizowano kilkaset projektów informatycznych dedykowanych wsparciu funkcjonowania działalności państwa. W przypadku wielu z nich korzyści uzyskiwane z ich przeprowadzenia nie były podporządkowane wizji na poziomie strategicznym, dominowało natomiast skupienie uwagi na zakupie sprzętu i technologii [BO12]. Takie podejście spowodowało powstanie rozwiązań silosowych, które nie zapewniały wymiany danych i nie przyczyniały się do wzrostu interoperacyjności w szerszym sensie, ani też nie wpłynęły pozytywnie na wzrost sprawności i efektywności całego państwa. Obecnie cały wysiłek związany z informatyzacją skupia się na wdrożeniu paradygmatu tzw. Informatyzacji Zintegrowanej. Koncepcja ta zmaterializowała się w 2012r. w pracach prowadzonych pod egidą Ministerstwa Administracji i Cyfryzacji. Stanowi ona próbę całościowego podejścia do informatyzacji zakładającego uczestnictwo

wszystkich instytucji publicznych w budowie, rozwoju i utrzymaniu Systemu Informacyjnego Państwa oraz wdrożenie standardów zarządzania procesowego w administracji publicznej [BO12]. Działania te muszą odbywać się ze zrozumieniem przez instytucje publiczne całości procesów, w których uczestniczą oraz zachowaniem świadomości własnej roli w ich realizacji, a także poczucia odpowiedzialności za obsługiwaną funkcjonalność biznesową i dostarczane w ten sposób wartości. Takie podejście musi się wiązać z prawidłowym przydziałem ról w przedsięwzięciach związanych z informatyzacją. Pozwoli to wyeliminować patologiczne zjawisko związane z przypisywaniem roli właściciela działom informatyki w poszczególnych instytucjach. Rola właściciela jako podmiotu, który jest odpowiedzialny merytorycznie za dany proces lub grupę procesów musi być bowiem przypisywana tym osobom, które posiadają wiedzę na temat celów stojących przed instytucją i sposobów ich realizacji. Informatyzacja Zintegrowana opiera się na czterech podstawowych zasadach [BO12]:

- obieg informacji musi być logiczny i skuteczny;
- obieg informacji musi być zdefiniowany w oparciu o potrzeby informacyjne osób i jednostek zaangażowanych w realizację procesów biznesowych;
- informatyzacja musi być przejrzysta, skuteczna i podporządkowana uzyskaniu jak najlepszej relacji otrzymanych wyników w stosunku do zaangażowanych nakładów;
- neutralność technologiczna państwa musi gwarantować swobodę wyboru technologii służących do realizacji usług publicznych.

Działania w zakresie przekształcenia RP w Cyfrowe Państwo są w dniu dzisiejszym faktem. Sprawia to, że wszelkie aktywności realizowane w zakresie transformacji SBN RP (rozpatrywanej zarówno w wymiarze organizacyjnym, jak również w wymiarze systemów teleinformatycznych) muszą uwzględniać przedstawione w niniejszym punkcie fundamentalne założenia koncepcyjne Cyfrowego Państwa oraz podstawowe zasady Informatyzacji Zintegrowanej. Pozwoli to na zachowanie spójności podejmowanych w tym zakresie działań z pozostałymi przedsięwzięciami o charakterze transformacyjnym, których podmiotem są organy państwa, jednostki administracji publicznej i inne szeroko rozumiane systemy funkcjonujące w Polsce [PR14a].

Z ideą Cyfrowego Państwa silnie wiąże się pojęcie interoperacyjności. Ulegało ono na przestrzeni ostatnich lat silnej ewolucji, a jego pojmowanie zmieniało się w tym czasie obejmując coraz szersze spektrum zagadnień począwszy od wymiany informacji w systemach informatycznych na poziomie technicznym, aż do uwzględnienia różnorodnych aspektów współdziałania na poziomie organizacyjnym. Ze względu na dużą rozpiętość definicji interoperacyjności

funkcjonujących obecnie poniżej przedstawione zostały wybrane z nich oraz wyprowadzona została propozycja definicji interoperacyjności w kontekście SBN [PR13b].

Według definicji zaproponowanej przez organizację IEEE (*Institute of Electrical and Electronics Engineers*) interoperacyjność oznacza *"zdolność dwóch lub większej liczby systemów informatycznych lub ich komponentów do wymiany informacji i do jej użycia"* [IE90]. Interoperacyjność zdefiniowana przez firmę Microsoft Corporation to *"możliwość sprawnej, efektywnej i spójnej komunikacji i wymiany informacji pomiędzy różnymi systemami informatycznymi, aplikacjami i sieciami komputerowymi oraz wykorzystywania otrzymanych w ten sposób informacji"* [MI08]. Firma Gartner określiła interoperacyjność jako *"zdolność dwóch lub większej liczby różnych podmiotów do wymiany informacji i do wykorzystania informacji między nimi wymienianej"* [GA07]. Rząd Nowej Zelandii przyjmuje, że interoperacyjność stanowi *"zdolność instytucji rządowych do współdzielenia informacji i integracji informacji oraz procesów biznesowych poprzez użycie wspólnych standardów"* [MA01]. Rząd Australii z kolei w opracowanych przez siebie ramach interoperacyjności wskazuje, że interoperacyjność jest to *"zdolność do transferu i użycia informacji w zunifikowany i efektywny sposób pomiędzy wieloma organizacjami i systemami informatycznymi, przy czym należy rozważać ją na trzech poziomach: procesów biznesowych, informacyjnym i technicznym"* [AU05]. W pracach Komisji Europejskiej (KE) pojęcie to zostało zdefiniowane jako *"możliwość współdziałania różnych odrębnych organizacji na rzecz osiągnięcia uzgodnionych i korzystnych dla wszystkich stron celów, przy jednoczesnym dzieleniu się informacjami i wiedzą pomiędzy tymi organizacjami poprzez wspierane przez nie procesy biznesowe, za pomocą wymiany danych za pośrednictwem odpowiednich systemów technologii informacyjno-komunikacyjnych"* [KE10d]. W Polsce zagadnienie interoperacyjności zostało zdefiniowane w ustawie z dnia 12 lutego 2010 roku *o zmianie ustawy o informatyzacji działalności podmiotów realizujących zadania publiczne oraz niektórych innych ustaw* (Dz.U. z 2010r., nr 40, poz. 230) jako *"zdolność różnych podmiotów oraz używanych przez nie systemów teleinformatycznych i rejestrów publicznych do współdziałania na rzecz osiągnięcia wzajemnie korzystnych i uzgodnionych celów, z uwzględnieniem współdzielenia informacji i wiedzy przez wspierane przez nie procesy biznesowe realizowane za pomocą wymiany danych za pośrednictwem wykorzystywanych przez te podmioty systemów teleinformatycznych"*.

Mnogość oraz dużą rozpiętość funkcjonujących obecnie definicji interoperacyjności może powodować istotne różnice w pojmowaniu przez różnych intere-

sariuszy zagadnień związanych z tym obszarem. Z tego względu zaproponowana została następująca definicja, uwzględniająca specyfikę SBN, która mówi, że: *"interoperacyjność w kontekście SBN należy rozumieć jako wszystkie przypadki komunikacji pomiędzy podmiotami pełniącymi określone role w SBN danego państwa, gdzie następuje wymiana informacji zachodząca pomiędzy tymi podmiotami (m.in. z wykorzystaniem systemów informatycznych i telekomunikacyjnych) w drodze do osiągnięcia przez nie założonych celów poprzez ich sprawne i efektywne współdziałanie zarówno na poziomie prawnym, organizacyjnym, semantycznym jak i technicznym"* [PR13b]. Przypadki komunikacji, o której mowa w powyższej definicji, nie są ograniczone jedynie do komunikacji wewnętrznej w ramach SBN, ale obejmują również współdziałanie podmiotów tworzących SBN z partnerami zewnętrznymi. Zarówno interoperacyjność wewnętrzna poszczególnych elementów tworzących SBN RP, jak również interoperacyjność zewnętrzna SBN RP jako całości biorącej udział w działaniach ponadnarodowych musi być postrzegana jako jego immanentna cecha systemowa. Wymagane jest zatem jej uwzględnienie i wbudowanie w nowy model SBN RP tworzony obecnie w ramach działań zmierzających do jego transformacji. Interoperacyjność musi być ponadto rozpatrywana wielopoziomowo i obejmować kontekst polityczny oraz aspekty prawne, organizacyjne, semantyczne i techniczne (por. Rysunek 2.1).

Działania związane z transformacją SBN RP i jego środowiska rozwiązań teleinformatycznych muszą ostatecznie zmierzać do osiągnięcia wysokiego poziomu dojrzałości w zakresie interoperacyjności. Manifestuje się on na poziomie celów organizacji wchodzących w skład SBN RP, a wyznacznikami jego osiągnięcia jest m.in.:

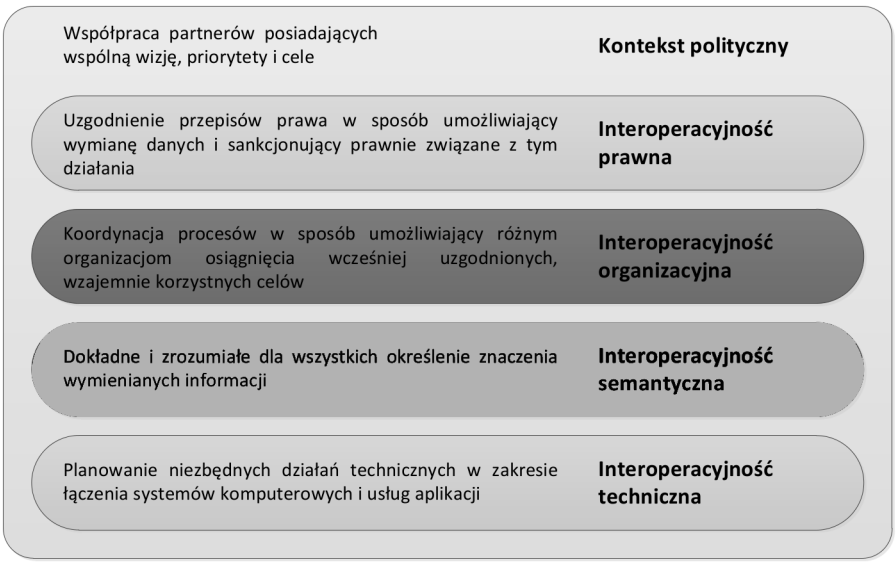
- dopasowanie strategiczne współdziałających ze sobą organizacji;
- brak konfliktów celów i wspólne planowanie;
- istnienie jednolitego systemu wartości i stylu zarządzania oraz wspólnej bazy wiedzy;
- istnienie procesów do wspierania ciągłego doskonalenia interoperacyjności.

Analizując kwestie związane z osiąganiem interoperacyjności przez podmioty realizujące zadania na rzecz SBN konieczne jest rozważenie głównych barier związanych z wdrażaniem tego podejścia. Zasadnicze przeszkody występujące w praktyce, w niemal każdym państwie wdrażającym koncepcję interoperacyjności, to ograniczenia [EI07]:

- prawne,
- kulturowe,
- organizacyjne,

– informacyjne,
– techniczne,
– ekonomiczne.

Bez ich przezwyciężenia praktycznie nie można mówić o implementacji interoperacyjności zarówno w kontekście SBN jak i szeroko pojętej e-Administracji, nie wspominając już o realizacji wizji Cyfrowego Państwa.



Rysunek 2.1. Poziomy interoperacyjności wg EIF
Źródło: opracowanie na podstawie: *European Interoperability Framework (EIF) for European public services*, European Commission, Bruxelles 2010.

2.4 Ewolucja metodyk realizacji systemów ICT

Budowa systemów wsparcia teleinformatycznego SBN stanowi zazwyczaj długotrwały i wieloetapowy proces związany z koniecznością realizacji wielu wzajemnie powiązanych czynności [SZ01]. Podstawą budowy i wdrożenia systemu wsparcia teleinformatycznego w dowolnej organizacji, w tym realizującej funkcje w ramach SBN, jest zastosowanie odpowiedniej metodyki definiującej sposób organizacji procesu wytwórczego. W literaturze spotykanych jest wiele definicji metodyki budowy systemów wsparcia teleinformatycznego. Pressman definiuje ją jako *"schemat wykonywania pewnych zadań, koniecznych do stworzenia dobrego systemu teleinformatycznego"* [PS04]. Według Wesołowskiego pojęcie to można przedstawić jako *"zbiór zasad i sposobów działania zmierza-*

jących do rozwiązania określonych problemów i osiągnięcia wyznaczonych celów" [WE99]. Wrycza z kolei definiuje je jako *"spójny, logicznie uporządkowany zestaw procedur o charakterze technicznym i organizatorskim pozwalających zespołowi wykonawczemu zrealizować cykl życia systemu"* [WR99]. Zbliżoną definicję proponują również Olszak i Sroka, mówiąc, że: *"metodyką budowy systemu informatycznego określa się spójny, logicznie uporządkowany zbiór pojęć, metod, technik, zasad i sposobów postępowania wykorzystywanych w procesie realizacji cyklu życia systemu"* [OS03]. Metodyka realizacji systemów wsparcia teleinformatycznego pozwala zatem na takie zorganizowanie procesów wytwarzania, wdrażania i utrzymywania oprogramowania, które gwarantuje dostarczenie, w ramach określonego harmonogramu i budżetu, wysokiej jakości produktu spełniającego potrzeby swoich użytkowników oraz umożliwia zaadresowanie po stronie zespołu realizacyjnego pytań *"kto"* robi *"co"*, *"jak"* i *"kiedy"*. Metodyka ta ustala zatem:

- fazy projektu,
- role i odpowiedzialności uczestników projektu;
- modele tworzone w każdej z faz;
- scenariusze postępowania w każdej z faz;
- reguły przechodzenia do następnej fazy realizacji projektu;
- notacje, których należy używać do dokumentowania i komunikowania wyników prac;
- dokumentację powstającą w każdej z faz.

Analizując dostępną literaturę przedmiotu prezentującą sposoby realizacji systemów wsparcia teleinformatycznego SBN spotykane w poszczególnych państwach należących do UE i/lub NATO można postawić tezę, że dotychczas nie udało się wypracować jednej spójnej metodyki dedykowanej wytwarzaniu tego typu systemów. W dalszym ciągu w różnych krajach świata (oraz w samych organizacjach jakimi są NATO i UE) lokalne grupy reprezentujące interesy organizacji gospodarczych, społecznych i badawczych promują stosowanie własnych metodyk realizacji systemów wsparcia teleinformatycznego. Dotyczy to budowy systemów realizowanych zarówno na potrzeby podmiotów cywilnych, jak również sił zbrojnych i różnego rodzaju służb i formacji funkcjonujących w ramach SBN. Z dotychczasowej praktyki wynika jednak, że w realizacji systemów wsparcia teleinformatycznego SBN zastosowanie znajdują w zdecydowanej większości przypadków metodyki formalne. Wynika to wprost ze skali problemów związanych z ciągłym wzrostem stopnia skomplikowania struktury tych systemów uwarunkowanym m.in. odejściem od rozwiązań monolitycznych i typu klient-serwer na rzecz systemów realizowanych w architekturze wielowarstwowej lub architekturze zorientowanej na usługi (SOA). Jest to sytuacja

wymuszona planowanym zarówno w UE jak i w NATO wielofazowym procesem transformacji zmierzającej do utworzenia środowiska ukierunkowanego na osiąganie zdolności sieciocentrycznych. Proces ten jest wieloaspektowy, a jego realizacja przebiega zarówno w wymiarze biznesowym (procesów operacyjnych), jak również w wymiarze zarządczym (procesów dowódczych) i odbywa się w perspektywie sięgającej do roku 2025 [SU11]. Metodyki stosowane do ich realizacji muszą uwzględniać wymagania związane m.in. z koniecznością:

- integracji systemów, danych i procesów wielu suwerennych podmiotów pełniących zarówno różne jak i tożsame role w funkcjonowaniu całego SBN;
- doskonalenia procesów podejmowania decyzji i ich przekazywania w warunkach rozproszenia odpowiedzialności;
- zmierzania do elastyczności funkcjonalnej i strukturalnej przy uwzględnieniu stale rosnącego zaawansowania merytorycznego i technicznego związanego z kompleksowym charakterem SBN;
- unifikacji funkcji cząstkowych systemów i sposobów ich wykorzystania przez podmioty tworzące SBN w drodze budowy modułowych systemów dziedzinowych o otwartym charakterze, które będą pozwalały na podnoszeniu poziomu interoperacyjności i zwiększały tym samym siłę efektu synergii;
- zapewnienia zgodności ze zmieniającymi się elementami otoczenia systemowego, a zwłaszcza z aktualnym stanem prawnym, ewoluującym zgodnie z przyjętymi procedurami legislacyjnymi.

Próby opracowania spójnej metodyki dedykowanej wytwarzaniu systemów wsparcia teleinformatycznego, która będzie spełniała przedstawione powyżej wymagania oraz znajdzie zastosowanie w tworzeniu systemów teleinformatycznych SBN, były podejmowane zarówno w NATO jak również w UE i bazowały na koncepcji Architektury Korporacyjnej (ang. *Enterprise Architecture*) wprowadzonej przez Zachmana [ZA92].

Prace w tym obszarze podejmowane przez NATO od lat 90-tych doprowadziły do powstania dwóch generacji metodyki realizacji systemów wsparcia teleinformatycznego dowodzenia i kierowania uwzględniających szczególne wymagania wojska w tym zakresie [SU07]. Pierwsza generacja metodyki jest zorientowana systemowo i została ukierunkowana zstępująco, zgodnie z klasycznym cyklem życia systemu. Umożliwiała ona przedstawienie przekrojów pewnego wyróżnionego systemu lub podsystemu w sposób całościowy, a zapewnienie powiązań pomiędzy przekrojami systemu (podsystemu) było zagwarantowane poprzez przestrzeganie zdefiniowanej metodyki tworzenia architektury. Druga generacja metodyki jest zorientowana usługowo i uwzględnia wie-

lokierunkowość działań prowadzących do wytwarzania nowych rozwiązań oraz zmian w już istniejących systemach teleinformatycznych. Bazuje ona na dorobku wiodących ośrodków przemysłowych, akademickich oraz wojskowych i opiera się na segmentacji opisu na mniejsze, niezależne fragmenty. Pierwsza generacja metodyki została zainicjowana przez pojawienie się programu zarządzania interoperacyjnością NIMP (ang. *NATO Interoperability Management Plan*), który określał cykl życia systemu oraz definiował ramy architektoniczne umożliwiające tworzenie opisów architektonicznych systemów. NIMP określał również struktury NATO odpowiedzialne za tworzenie tych opisów na różnych etapach cyklu życia systemów teleinformatycznych [SU07]. W kolejnym kroku NIMP został zastąpiony dyrektywą interoperacyjności NID (ang. *NATO Interoperability Directive*), natomiast same ramy architektoniczne NATO zostały wydzielone jako osobne dokumenty: NAF 2.0 (ang. *NATO Architecture Framework*) oraz metodyka tworzenia architektury NATO - AEM (ang. *Architecture Engineering Methodology*). Na tym etapie użyteczność ram architektonicznych NAF 2.0 została poddana krytycznej ocenie. W rezultacie pojawiły się wątpliwości, czy zstępująca realizacja systemów, kładąca nacisk na początkowe fazy klasycznego cyklu życia systemów, nadaje się do zastosowania w przedsięwzięciach doskonalących realizowany w odniesieniu do istniejących rozwiązań. Spowodowało to powstanie kolejnej wersji ram architektonicznych NAF 3.0 oraz połączonych z nimi standardów i profili interoperacyjności NISP (ang. *NATO Interoperability Standards and Profiles*), a także aktualizacją metodyki tworzenia architektury NATO do kolejnej wersji - AEM 2.0.

Działania podejmowane w Unii Europejskiej mają swoją genezę w podejściu zmierzającym z jednej strony do zapewnienia interoperacyjności systemów informacyjnych funkcjonujących w różnych dziedzinach związanych z wypełnieniem wyzwań stojących przez Komisją Europejską jako globalną organizacją i jej przekształcenia w sprawnie działającą e-Administrację Unii Europejskiej [KE05]. Z drugiej strony wiążą się one z koniecznością wypracowania podejścia do sprawniejszego dowodzenia i kierowania w operacjach zarządzania kryzysowego UE. W pierwszym obszarze próby opracowania jednolitej metodyki realizacji systemów wsparcia teleinformatycznego zaowocowały powstaniem ogłoszonych przez Komisję Europejską ram architektonicznych CEAF [KE05]. W drugim obszarze podjęte prace doprowadziły do powstania studium realizowalności "NEC Implementation Study" przygotowanego przez konsorcjum EURONEC na zlecenie Europejskiej Agencji Obrony (ang. *European Defence Agency - EDA*) [EU09].

Na tle przedstawionych powyżej działań podejmowanych zarówno w ramach NATO, jak i UE można postawić tezę, że koncepcja Architektury Korporacyjnej powinna być obecnie postrzegana jako podstawa metodycznego podejścia do realizacji systemów teleinformatycznego wsparcia SBN. Realizacja systemów teleinformatycznego wsparcia SBN obejmuje wszystkie fazy jego cyklu życia. Zatem odnosi się ona również do przedsięwzięć doskonalących w stosunku do istniejących systemów oraz działań związanych z zapewnianiem ich bieżącego funkcjonowania po wdrożeniu. Z tego względu dokonano również przeglądu spotykanych obecnie na świecie podejść do zarządzania eksploatacją systemów wsparcia teleinformatycznego SBN. Wnioski płynące z tego przeglądu zostały przedstawione w kolejnym punkcie niniejszego opracowania.

2.5 Zarządzanie eksploatacją systemów ICT w środowisku SBN RP

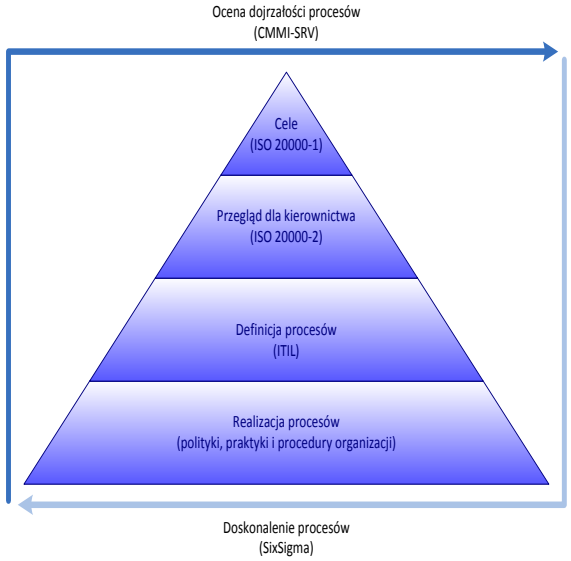
Problematyka zarządzania eksploatacją systemów wsparcia teleinformatycznego jest dziedziną rozwijającą się równolegle z obszarem związanym z ich wytwarzaniem. W chwili obecnej w zarówno w zakresie praktyki inżynierskiej (występującej w przedsiębiorstwach komercyjnych i w administracji publicznej), jak również w pracach badawczych wyraźnie odznacza się trend charakteryzujący się postrzeganiem zagadnień związanych z eksploatacją złożonych środowisk IT przez pryzmat podejścia usługowego. Powoduje to, że jednostki organizacyjne pełniące określone role w SBN, które korzystają z systemów teleinformatycznych w celu wspomagania swoich procesów biznesowych, coraz częściej nie są zainteresowane informatyką jako taką, lecz zdolnościami, jakie mogą budować przy jej aktywnym wykorzystaniu. W takim rozumieniu usługa IT oznacza sposób dostarczenia wartości jej odbiorcy poprzez umożliwienie mu uzyskania oczekiwanych rezultatów przy wykorzystaniu zasobów jej dostawcy. Należy zauważyć przy tym, że ryzyko oraz koszty związane z przygotowaniem i wdrożeniem usług, utrzymaniem ciągłości działania, budową infrastruktury technicznej, zarządzaniem usługami itp. przenoszone są na ich dostawcę. Odbiorca ponosi koszt wykorzystania usług do wsparcia swoich procesów biznesowych. W przypadku, gdy te procesy biznesowe przecinają silosy jednej lub wielu organizacji zaangażowanych w realizację zadań na rzecz SBN, to możliwe jest podejmowanie działań zmierzających do optymalizacji kosztów eksploatacji systemów teleinformatycznych poprzez budowę wspólnych centrów usług IT.

Jak pokazują badania naukowe, prowadzone obecnie przez różne ośrodki akademickie, opisane powyżej tendencje występują na całym świecie i dotyczą zarówno państw zaawansowanych technologicznie, jak i rozwijających się.

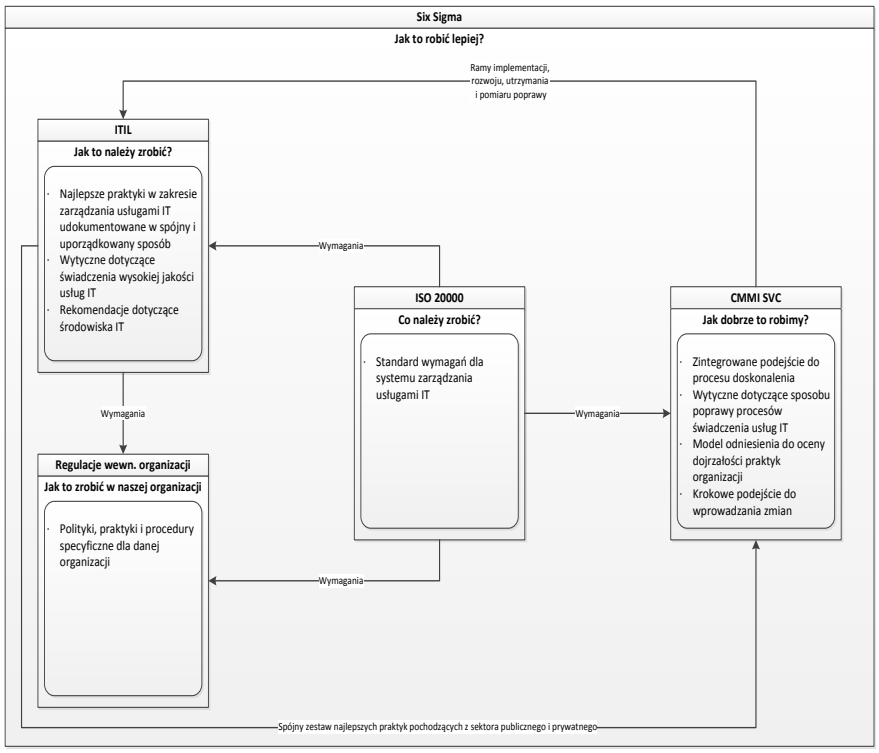
Stosowane w praktyce podejścia do zarządzania eksploatacją systemów wsparcia teleinformatycznego bazują m.in. na:

- normie ISO/IEC 20000;
- bibliotece ITIL;
- modelu CMMI for Services;
- metodzie Six Sigma.

Wszystkie w/w podejścia bazowe podkreślają znaczenie nastawienia procesowego, jednak operują na różnych poziomach abstrakcji i udzielają odpowiedzi na różne pytania oraz dostarczają różnych elementów składowych niezbędnych do całościowego zarządzania procesami eksploatacji systemów teleinformatycznych w modelu usługowym. Powstaje zatem problem stworzenia takiego systemu zarządzania eksploatacją IT w organizacjach biorących udział w realizacji procesów w ramach SBN, który pozwoli na wsparcie pełnego cyklu życia usług IT oraz zapewni możliwość ciągłego doskonalenia procesów związanych z ich świadczeniem. Koncepcja takiego modelu zarządzania została przedstawiona na poniższych rysunkach (por. Rysunek 2.2, Rysunek 2.3). Zakłada ona takie wykorzystanie potencjału wymienionych wcześniej podejść bazowych, które pozwoli uzyskać efekt synergii i przełoży się na: redukcję ryzyka, kosztów, błędów i usterek, podwyższenie efektywności i jakości oraz uzyskanie wspólnego rozumienia aspektów związanych z eksploatacją rozwiązań IT w organizacji.



Rysunek 2.2. Koncepcja całościowego zarządzania procesami eksploatacji systemów IT.



Rysunek 2.3. Relacje pomiędzy elementami koncepcji całościowego zarządzania procesami eksploatacji systemów IT.

2.6 Podsumowanie

SBN RP jest systemem obejmującym w sposób całościowy wszystkie obszary działalności państwa w zakresie przeciwdziałania potencjalnym zagrożeniom bezpieczeństwa narodowego. Jest on złożony oraz rozproszony funkcjonalnie i terytorialnie oraz powiązany z analogicznymi systemami innych państw wchodzących w skład ponadnarodowych sojuszy, których uczestnikiem jest również Polska. Stanowi on zatem system niezwykle ważny dla sprawnego funkcjonowania zarówno samego państwa jak i jego obywateli, który musi być wspierany przez odpowiednie systemy teleinformatyczne. Ich rolą jest dostarczenie odpowiedniego zestawu usług elektronicznych zapewniających kompleksową obsługę organów administracji, służb i innych podmiotów odpowiedzialnych za realizację działań w ramach SBN RP. Konieczne jest zatem podejmowanie działań, które pozwolą zbudować kompleksowy system informacyjny SBN RP spójny z tworzonym systemem informacyjnym państwa

[MA13]. W trakcie szeroko zakrojonych prac analitycznych prowadzonych w oparciu o dostępną literaturę i otwarte źródła wiedzy, wskazano aspekty, które zostały uznane za najistotniejsze z punktu widzenia projektowania systemów IT, jak również stworzenia przyszłej koncepcji wsparcia teleinformatycznego SBN RP. W świetle uzyskanych wyników dla zapewnienia powodzenia przedsięwzięć związanych z realizacją systemów wsparcia teleinformatycznego SBN najbardziej znaczące wydają się być:

- zastosowanie odpowiedniej metodyki realizacji tych systemów uwzględniającej m.in. kwestie związane z zapewnieniem ich interoperacyjności oraz różne strategie postępowania względem istniejących systemów spadkowych (ang. *legacy*);
- odpowiednie umocowanie prawne stosowanej metodyki realizacji tych systemów;
- podejmowanie działań zmierzających do pokonania ewentualnych barier w osiąganiu interoperacyjności.

Przeprowadzona analiza w zakresie ewolucji metodyk realizacji systemów wsparcia teleinformatycznego SBN stosowanych współcześnie w różnych państwach świata oraz w organizacjach ponadnarodowych, jakimi są Unia Europejska i NATO, pozwala stwierdzić, że nastąpiła zmiana orientacji tych metodyk z podejścia systemowego na podejście usługowe. Powoduje to, że jednostki organizacyjne pełniące określone role w SBN, korzystające z systemów teleinformatycznych w celu wspomagania swoich procesów biznesowych, nie są zainteresowane informatyką jako taką, lecz zdolnościami, jakie mogą budować przy jej aktywnym wykorzystaniu. Przeprowadzona analiza wskazuje również, że współcześnie problematyka realizacji systemów wsparcia teleinformatycznego SBN opiera się na trzech zasadniczych filarach:

- ramach interoperacyjności (narodowych lub ponadnarodowych);
- koncepcji Architektury Korporacyjnej i wywodzących się z niej ramach architektonicznych oraz metodykach tworzenia architektury;
- metodach, technikach i narzędziach służących do całościowego zarządzania procesami eksploatacji systemów wsparcia teleinformatycznego.

Zachodzą one na siebie tworząc pewną część wspólną, przez co można postawić tezę, że współczesna metodyka realizacji systemów wsparcia teleinformatycznego SBN musi uwzględniać te zależności, aby umożliwić zbudowanie i dostarczenie zestawu usług elektronicznych zapewniających kompleksową obsługę organów państwa, jednostek administracji publicznej, służb i innych podmiotów odpowiedzialnych za realizację działań w ramach SBN RP. Ponadto metodyka ta musi umożliwiać prawidłowe i pełne określenie wymagań względem tworzonych systemów na wszystkich poziomach abstrakcji (biznesowym,

użytkownika, systemowym) oraz zapewniać właściwe procesy zarządzania wymaganiami w warunkach współpracy wielu rozłącznych organizacji tworzących SBN RP uwzględniające zastosowanie adekwatnych metod, narzędzi i technik umożliwiających identyfikację kluczowych wymagań i eliminowanie konfliktów pomiędzy wymaganiami różnych interesariuszy.

Bibliografia

[AU05] *Australian Government Technical Interoperability Framework*, Australian Government Department of Finance, 2005.

[BB13] *Biała Księga Bezpieczeństwa Narodowego Rzeczypospolitej Polskiej*, BBN, Warszawa, 2013.

[BO12] Boni M. i in., *Państwo 2.0, nowy start dla e-administracji*, Ministerstwo Administracji i Cyfryzacji, Warszawa 2012

[EI07] Studium interoperacyjności na szczeblu lokalnym i regionalnym, European Institute of Public Administration (EIPA), Bruksela 2007.

[EU09] *NEC Implementation Study*, EURONEC, 2009.

[GA07] *Preparation for Update European Interoperability Framework 2.0 - Final Report*, Gartner, 2007.

[IE90] *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*, IEEE, New York 1990.

[KE05] *The Commission Enterprise IT Architecture Framework [CEAF] Version 1 explained*, Komisja Europejska, Bruksela 2005.

[KE09] *Biała Księga. Modernizacja normalizacji technologii informacyjno-komunikacyjnych w UE*, Komisja Europejska, Bruksela 2009.

[KE10a] *Europejska Agenda Cyfrowa*, Komisja Europejska, Bruksela 2010.

[KE10b] *Europa 2020. Strategia na rzecz inteligentnego i zrównoważonego rozwoju sprzyjającego włączeniu społecznemu*, Komisja Europejska, Bruksela 2010.

[KE10c] *European Interoperability Strategy (EIS) for European public services*, Komisja Europejska, Bruksela 2010.

[KE10d] *European Interoperability Framework (EIF) for European public services*, Komisja Europejska, Bruksela 2010.

[KI13] Kitler W. i in., *Ocena obecnej struktury bezpieczeństwa narodowego RP*, raport z realizacji podzadania badawczego nr 1.2.1.2. projektu nr O ROB/0076/03/001 pt. "System Bezpieczeństwa Narodowego RP" w zakresie obronności i bezpieczeństwa państwa finansowanego ze środków NCBiR, Warszawa 2013.

[MA07] Mallard T., *New Zealand E-government Strategy*, Wellington 2001.

[MA13] *Program Zintegrowanej Informatyzacji Państwa*, Ministerstwo Administracji i Cyfryzacji, Warszawa 2013.

[MI08] *Ramy interoperacyjności systemów administracji publicznej*, Microsoft Corporation, Warszawa 2008.

[OS03] Olszak C., Sroka H., *Informatyka w zarządzaniu*, Akademia Ekonomiczna w Katowicach, Katowice, 2003.

[PA14] Pawłowski J. i in., *Misja, cele i zadania SBN RP*, raport z realizacji podzadania badawczego nr 5.1. projektu nr O ROB/0076/03/001 pt. "System Bezpieczeństwa

Narodowego RP" w zakresie obronności i bezpieczeństwa państwa finansowanego ze środków NCBiR, Warszawa 2014.

[PR13a] Protasowicki T. i in., *Organizacja wspomagania teleinformatycznego kierowania bezpieczeństwem narodowym RP*, raport z realizacji podzadania badawczego nr 1.3. projektu nr O ROB/0076/03/001 pt. "System Bezpieczeństwa Narodowego RP" w zakresie obronności i bezpieczeństwa państwa finansowanego ze środków NCBiR, Warszawa 2013.

[PR13b] Protasowicki T., *Przegląd metodyk realizacji systemów wsparcia teleinformatycznego SBN w innych krajach, w tym z NATO i UE*, raport z realizacji podzadania badawczego nr 6.1. projektu nr O ROB/0076/03/001 pt. "System Bezpieczeństwa Narodowego RP" w zakresie obronności i bezpieczeństwa państwa finansowanego ze środków NCBiR, Warszawa 2013.

[PR14a] Protasowicki T., *Koncepcja wsparcia teleinformatycznego SBN RP*, raport z realizacji podzadania badawczego nr 6.2. projektu nr O ROB/0076/03/001 pt. "System Bezpieczeństwa Narodowego RP" w zakresie obronności i bezpieczeństwa państwa finansowanego ze środków NCBiR, Warszawa 2014.

[PR14b] Protasowicki T., *Wybrane aspekty zastosowania koncepcji architektury korporacyjnej w transformacji Systemu Bezpieczeństwa Narodowego RP*, Zeszyty Naukowe Uniwersytetu Szczecińskiego Ekonomiczne Problemy Usług, vol. 112, 2014

[PS04] Pressman S., *Praktyczne podejście do inżynierii oprogramowania*, WNT, Warszawa, 2004.

[SO12] Sobczak A., *Architektura korporacyjna państwa jako narzędzie zarządzania cyfrową transformacją organizacji sektora publicznego*, Roczniki Kolegium Analiz Ekonomicznych, 2012, nr 24/2012.

[SU07] Szulik W., *Metodyka realizacji systemów dowodzenia NATO*, sprawozdanie z realizacji zadania badawczego nr 13201, program badawczy zamawiany nr PBZ–MNiSW–DBO–02/I/2007 pt. *Zaawansowane metody i techniki tworzenia świadomości sytuacyjnej w działaniach sieciocentrycznych*, Konsorcjum Operacji Sieciocentrycznych, Warszawa 2007.

[SU11] Szulik W., *Wybrane zagadnienia zarządzania kryzysowego w środowisku sieciocentrycznym*, [w:] *Zarządzanie kryzysowe w systemie bezpieczeństwa narodowego*, red. Sobolewski G., Majchrzak D., AON, Warszawa 2011.

[SZ01] Szyjewski Z., *Wykorzystanie metody analizy strategicznej SWOT w tworzeniu strategii informatyzacji*, [w:] *Efektywność zastosowań systemów informatycznych. Tom III*, red. Grabara J. K., Nowak J. S., WNT, Warszawa 2001.

[WE99] Wesołowski W., *Inżynieria zarządzania*, Wyższa Szkoła Handlu i Prawa, Warszawa, 1999.

[WR99] Wrycza S., *Analiza i projektowanie systemów informatycznych zarządzania. Metodyki, techniki i narzędzia*, PWN, Warszawa, 1999.

[ZA92] Zachman J., *A Framework for Information Systems Architecture*, IBM Systems Journal, 1981, vol. 26, no. 3; por. Zachman J., *Extending and Formalizing the Framework for Information Systems Architecture*, IBM Systems Journal, 1992, vol. 31, no. 3.

[ZZ10] Zięba R., Zając J., *Budowa zintegrowanego systemu bezpieczeństwa narodowego Polski, Ekspertyza sporządzona na zlecenie Ministerstwa Rozwoju Regionalnego w ramach prac nad aktualizacją średniookresowej strategii rozwoju kraju*, ` Warszawa 2010.

Rozdział 3

Znaczenie identyfikacji wymagań dla realizacji systemów teleinformatycznych dedykowanych wspomaganiu funkcjonowania organizacji o charakterze federacyjnym

Wymagania są pewnymi atrybutami systemu teleinformatycznego pozwalającymi na sformułowanie jego funkcjonalności i cech które musi on posiadać, aby spełnić potrzeby interesariuszy. Współcześnie określa się je na wielu poziomach abstrakcji, które obejmują różny zakres oraz angażują interesariuszy reprezentujących poszczególne szczeble organizacji. Identyfikacja wymagań nabiera szczególnego znaczenia w przypadku inicjatyw związanych z budową systemów teleinformatycznych dedykowanych wspomaganiu funkcjonowania organizacji o charakterze federacyjnym. Tego typu organizacje składające się z wielu suwerennych podmiotów wspólnie realizujących pewną jednolitą misję i wynikające z niej procesy biznesowe. Prawidłowa identyfikacja wymagań w warunkach konieczności zapewnienia zgodności tworzonych systemów teleinformatycznych z potrzebami różnych interesariuszy reprezentującymi niezależne podmioty, jest możliwa wyłącznie poprzez właściwy dobór i zastosowanie metod, narzędzi i technik wywodzących się z inżynierii wymagań. Dodatkowo decyzja o wyborze spośród zidentyfikowanych wymagań tych, które mają zostać zrealizowane w systemie teleinformatycznym jest jednym z najbardziej krytycznych rozstrzygnięć mającym wpływ na powodzenie całego przedsięwzięcia. W niniejszym rozdziale przedstawiono znaczenie identyfikacji wymagań dla realizacji systemów teleinformatycznych dedykowanych wspomaganiu funkcjonowania organizacji o charakterze federacyjnym na przykładzie Systemu Bezpieczeństwa Narodowego RP (SBN RP). Na tym tle zaprezentowano m.in. występujące współcześnie problemy związane z inżynierią wymagań dotykające organizacji federacyjnych, a także zaproponowano ogólną strategię wyboru metod odkrywania wymagań i przedstawiono listę metod oceny ważności wymagań pozwalających na nadawanie im priorytetów przy uwzględnieniu podejścia grupowego i siły głosu poszczególnych interesariuszy.

Technologie telekomunikacyjne i informatyczne są obecne w Polsce od wielu lat. Odgrywają one istotną rolę w codziennym funkcjonowaniu różnych jednostek administracji publicznej oraz podmiotów należących do szeroko pojętego sektora bezpieczeństwa i obronnego. Tworzenie rozwiązań w oparciu o te technologie staje się wraz z upływem czasu coraz bardziej skomplikowane. W szczególności dotyczy to realizacji systemów teleinformatycznych dedykowanych wspomaganiu funkcjonowania organizacji o charakterze federacyjnym. W niniejszej rozdziale przedstawiono znaczenie identyfikacji wymagań dla realizacji systemów teleinformatycznych dedykowanych wspomaganiu funkcjonowania takich organizacji na przykładzie Systemu Bezpieczeństwa Narodowego RP (SBN RP). Na tym tle zaprezentowano m.in. występujące współcześnie problemy związane z inżynierią wymagań dotykające organizacji federacyjnych, a także zaproponowano ogólną strategię wyboru metod odkrywania

wymagań i przedstawiono katalog metod oceny ważności wymagań pozwalających na nadawanie im priorytetów przy uwzględnieniu podejścia grupowego i siły głosu poszczególnych interesariuszy.

3.1 Budowa systemów ICT dla organizacji federacyjnych

Organizacja federacyjna stanowi sieć niezależnych podmiotów, która jest tworzona doraźnie lub trwale w celu realizowania wspólnych działań. Istotą ich istnienia jest nastawienie na wspólne realizowanie pewnej uzgodnionej misji w przyjętym horyzoncie czasowym. Tego typu organizacje zbudowane są w sposób zapewniający możliwość dzielenia się, przez wchodzące w ich skład podmioty, potencjałem technicznym i ludzkim, *know-how*, kosztami, ryzykiem itp. Oprócz tego są one nastawione na uzyskiwanie korzyści, które nie są możliwe do osiągnięcia przez poszczególne podmioty działające niezależnie. Jednak budowa organizacji federacyjnej na bazie autonomicznych podmiotów daje im dużą elastyczność i zdolność do rekonfiguracji. Oznacza to również, że realizowane w niej procesy biznesowe mogą ulegać dynamicznym zmianom. Organizacja federacyjna różni się od innych typów współcześnie spotykanych organizacji nie tylko w warstwie dotyczącej jej kształtu i sposobu działania, ale również kultury organizacyjnej i leżącej u jej podstaw filozofii działania, która determinuje styl koordynowania działań. Organizacje tego typu mogą być zatem scharakteryzowane z perspektywy: czasu ich istnienia, konfiguracji i sposobu koordynacji.

"System Bezpieczeństwa Narodowego (bezpieczeństwa państwa) stanowi całość sił (podmiotów), środków i zasobów przeznaczonych przez państwo do realizacji zadań w dziedzinie bezpieczeństwa, odpowiednio do tych zadań zorganizowana (w podsystemy i ogniwa), utrzymywana i przygotowywana. Składa się z podsystemu (systemu) kierowania i szeregu podsystemów (systemów) wykonawczych, w tym podsystemów operacyjnych (obronny i ochronne) i podsystemów wsparcia (społeczne i gospodarcze)" [BB13]. Pawłowski i in. Dochodzą z kolei do następującej szerszej konkluzji: *"System Bezpieczeństwa Narodowego RP (SBN RP) to celowo wyodrębniony z systemu państwowego, kolektywny zbiór organów władzy i administracji publicznej, innych organów państwowych, sił zbrojnych, przedsiębiorców i innych jednostek organizacyjnych, organizacji społecznych i obywateli wykonujących działania na rzecz zapewnienia bezpieczeństwa narodowego, wewnętrznie skoordynowanych i wzajemnie powiązanych ze sobą za pomocą relacji porządkujących, ze względu na realizowaną misję, którą jest obrona i ochrona państwa jako instytucji politycznej, terytorialnej i społecznej, a także zapewnienie wolnych od zakłóceń warunków bytu*

i rozwoju jednostek i całego społeczeństwa oraz ochrona życia i zdrowia ludzi, ich dóbr (materialnych i niematerialnych) i środowiska naturalnego we wszystkich stanach funkcjonowania państwa (w czasie normalnym, w tym w czasie kryzysu oraz w stanach nadzwyczajnych)" [PA14]. Na tle w/w definicji można stwierdzić, że System Bezpieczeństwa Narodowego RP (SBN RP) jest organizacją o charakterze federacyjnym. W ramach tego systemu następuje intensywna wymiana informacji zachodząca pomiędzy tworzącymi go podmiotami (m.in. z wykorzystaniem systemów informatycznych i telekomunikacyjnych) w drodze do osiągnięcia przez nie założonych celów. Identyfikacja wymagań ma na tym tle zasadnicze znaczenie dla prawidłowej realizacji systemów teleinformatycznych dedykowanych wspomaganiu funkcjonowania SBN RP. Powodem tego stanu rzeczy jest m.in. konieczność zapewnienia szeroko rozumianej interoperacyjności poszczególnych podmiotów i rozwiązań technicznych realizujących wspólnie pewne funkcje w ramach mniej lub bardziej sformalizowanych związków powoływanych (trwale lub w sposób dynamiczny) w celu realizacji misji SBN RP. Wyzwanie w takich warunkach stanowi zapewnienie zgodności systemów ICT z potrzebami różnych interesariuszy reprezentujących niezależne podmioty, która jest możliwa wyłącznie poprzez właściwy dobór i zastosowanie metod, narzędzi i technik wywodzących się z inżynierii wymagań. Stanowi ona fundament realizacji wszelkich współczesnych systemów wsparcia teleinformatycznego, eksploatowanych przez różne jednostki w obszarze administracji publicznej, obronnym i bezpieczeństwa państwa, jak również w sektorze przedsiębiorstw na całym świecie. Należy również zaznaczyć, że decyzja o wyborze spośród zidentyfikowanych wymagań tych, które mają zostać zrealizowane w systemie teleinformatycznym jest jednym z najbardziej krytycznych rozstrzygnięć mającym wpływ na powodzenie całego przedsięwzięcia. Niezbędne jest w tym celu podejmowanie odpowiednich działań mających na celu ustalenie istotności wymagań oraz wyeliminowanie konfliktów pomiędzy nimi i optymalizację pod kątem korzyści uzyskiwanych z wdrożenia systemu dla całej organizacji federacyjnej jaką jest SBN RP. Działania te muszą jednocześnie uwzględniać potrzeby federacji w zakresie zapewnienia swobodnej, bezpiecznej i efektywnej wymiany informacji będącej podstawą sukcesu jej funkcjonowania. Umożliwi to połączenie niezależnie zarządzanych podmiotów w taki sposób, który pozwoli na uzyskanie efektu synergii.

3.2 Wybrane aspekty inżynierii wymagań

Inżynieria wymagań (ang. *requirements engineering*) pozwala na dostarczenie w usystematyzowany sposób wiarygodnego opisu określającego, co ma być

wytworzone w ramach przedsięwzięcia związanego z realizacją danego rozwiązania teleinformatycznego, który jednocześnie abstrahuje od opisu sposobu implementacji systemu. Termin "wymaganie" (ang. *requirement*) może być rozumiany w różny sposób, zależny m.in. od kontekstu w jakim został użyty. Podstawowa definicja mówi, że pojęciem tym określamy [IE98]:

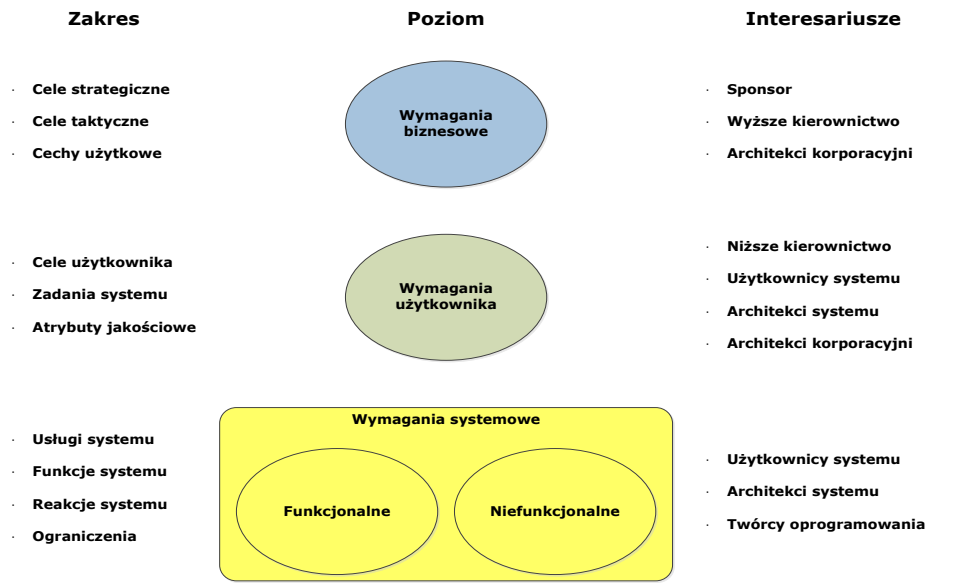
- stan lub zdolności wymagane przez użytkownika do rozwiązania problemu lub osiągnięcia celu;
- stan lub zdolności, które muszą występować w systemie lub jego elementach, aby umożliwić spełnienie warunków umowy, standardów, specyfikacji lub innego formalnie nałożonego zobowiązania;
- udokumentowaną reprezentację stanu lub zdolności określonych w punktach (a) lub (b) powyżej.

Inaczej można stwierdzić, że wymagania są pewnymi atrybutami systemu, które muszą zostać odkryte przed jego zbudowaniem, pozwalającymi na sformułowanie jego funkcjonalności i cech które musi on posiadać, aby spełnić potrzeby interesariuszy.

Inżynieria wymagań jest postrzegana jako najbardziej krytyczna i złożona dziedzina spośród innych związanych z budową skomplikowanych systemów społeczno-technicznych. Wynika to, z jednej strony, z jej znaczącego wpływu na powstające rozwiązanie, gdyż niepoprawnie określone wymagania powodują dostarczenie systemu nie spełniającego oczekiwań odbiorcy. Z drugiej strony wiąże się to z multidyscyplinarnym charakterem samej inżynierii wymagań, która oprócz zagadnień z zakresu inżynierii systemów obejmuje również zagadnienia dotyczące interakcji społecznych i metod zarządczych. Inżynieria wymagań stanowi zatem systematyczne podejście, pozwalające zbierać wymagania z różnych źródeł i implementować je w procesach rozwoju oprogramowania w całym jego cyklu życia z uwzględnieniem aspektów technicznych i społecznych [SB98, KS98, SA90].

3.2.1 Poziomy opisu wymagań

Kolejne rozszerzenia pojęcia "wymaganie", uwzględniające poziomy abstrakcji na jakich są one opracowywane zostały przedstawione na poniższym rysunku (Rysunek 3.1) i są omówione w dalszej części niniejszego punktu [WI03]. Należy zauważyć, że różne poziomy wymagań są przeznaczone dla różnych kategorii interesariuszy. Powiązanie pomiędzy poziomami wymagań a typami interesariuszy również przedstawia przywołany rysunek.



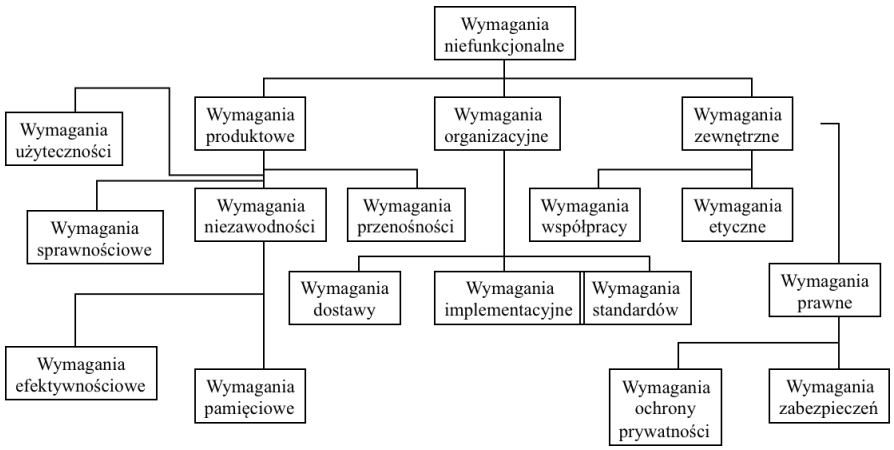
Rysunek 3.1. Poziomy, zakres wymagań i interesariusze.

Pierwszy poziom abstrakcji stanowią wymagania biznesowe (ang. *business requirements*), które są podstawą rozpoczęcia każdego przedsięwzięcia związanego z budową lub wdrożeniem systemów wsparcia teleinformatycznego. Wymagania te wyrażają na ogólnym poziomie potrzeby interesariuszy, których zaspokojenie przełoży się na osiągnięcie założonych przez organizację celów biznesowych (np. zapewnić efektywne wykorzystanie sił i środków, umożliwić sprawne zarządzanie akcją ratowniczą, dostarczyć spójnej i aktualnej informacji o zagrożeniu terrorystycznym itp.). Wymagania biznesowe muszą zatem opierać się na solidnym uzasadnieniu biznesowym pozwalającym sponsorom na racjonalne podjęcie decyzji o budowie systemu.

Kolejny poziom abstrakcji to wymagania użytkownika (ang. *user requirements*), które mają postać opisu usług oczekiwanych od systemu oraz pożądanych atrybutów jakościowych. Wyrażone są one zazwyczaj w języku naturalnym i stanowią warstwę pośrednią pomiędzy wymaganiami biznesowymi a wymaganiami systemowymi. Wymagania użytkownika informują o celach użytkowników lub zadaniach, które użytkownicy muszą być w stanie wykonać z wykorzystaniem systemu (np. system musi zapewnić mechanizmy dostępu do plików zawierających informacje o dostępnych siłach i środkach tworzonych przez inne narzędzia, system musi zapewniać możliwość wykonania optymalizacji tras przejazdu podczas akcji ratowniczej, system musi umożliwiać stwo-

rzenie mapy ryzyka ataków terrorystycznych itp.). Wymagania użytkownika muszą być sformułowane w taki sposób, aby były możliwe do zrozumienia przez osoby nie posiadające szczegółowej wiedzy technicznej [SO03].

Następny poziom abstrakcji reprezentują wymagania systemowe, które wyprowadzone są na podstawie wymagań użytkowników i stanowią precyzyjny opis szczegółowo przedstawiający wszystkie usługi oczekiwane od systemu oraz występujące ograniczenia. Wymagania systemowe zależą m.in. od rodzaju tworzonego systemu i grupy docelowej użytkowników. Jednocześnie muszą one być wyrażone w sposób umożliwiający twórcom systemu zrozumienie oczekiwań stawianych przed systemem. Dzielą się one dodatkowo na wymagania funkcjonalne i niefunkcjonalne. Wymagania funkcjonalne opisują szczegółowo usługi jakie ma oferować system, sposób w jaki ma on reagować na określone dane wejściowe lub zdarzenia itp. (np. system umożliwi wygenerowanie raportu historii zmian w planie reagowania kryzysowego, w przypadku braku środka gaśniczego w magazynie system wygeneruje automatycznie zamówienie tego środka w ilości wynikającej z iloczynu normy zużycia i prognozowanej liczby akcji gaśniczych dla okresu 30 dni, system umożliwi śledzenie działań jednostek biorących udział w akcji ratowniczej poprzez zobrazowanie ich pozycji i tras przejazdu na mapie cyfrowej). Wymagania niefunkcjonalne opisują właściwości i ograniczenia systemu (np. niezawodność, czas odpowiedzi, zasady bezpieczeństwa itp.). W złożonych systemach często dochodzi do konfliktów pomiędzy wymaganiami funkcjonalnymi i niefunkcjonalnymi [SO03]. Przykład dalszego podziału wymagań niefunkcjonalnych zaproponowany przez Sommerville'a został przedstawiony na kolejnym rysunku (Rysunek 3.2).



Rysunek 3.2. Typy wymagań niefunkcjonalnych [SO03].

3.2.2 Proces inżynierii wymagań

Zastosowanie metod, narzędzi i techniki wywodzących się z inżynierii wymagań pozwala na przeprowadzenie procesu opracowywania i zarządzania wymaganiami, pochodzącymi od wielu interesariuszy, w sposób zapewniający zbudowanie rozwiązań teleinformatycznych:

- spełniających cele biznesowe,
- dostarczających oczekiwanych funkcjonalności,
- charakteryzujących się wysoką jakością.

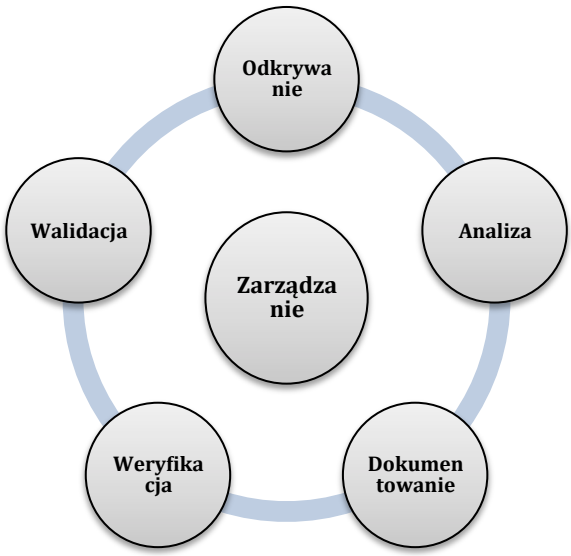
Proces inżynierii wymagań dzieli się na dwie główne grupy działań i został przedstawiony na poniższym rysunku (Rysunek 3.3) [SO03, PS10]:

- opracowywanie wymagań oraz
- zarządzanie wymaganiami.

Opracowywanie wymagań obejmuje aktywności związane z ich:

- odkrywaniem,
- analizą,
- dokumentowaniem,
- weryfikacją,
- walidacją.

Zarządzanie wymaganiami, rozumiane ogólnie, obejmuje działalność związaną ze śledzeniem zmian wymagań, oceną ich skutków i zarządzaniem tymi zmianami. Może ono być dodatkowo rozpatrywane w szerokim i wąskim znaczeniu. W szerokim sensie, zarządzanie wymaganiami obejmuje działania prowadzone w trakcie całego cykl życia systemu, tzn. zarówno w okresie jego rozwoju, jak również po wdrożeniu. W wąskim znaczeniu, zarządzanie wymaganiami ogranicza się tylko do zarządzania ich zmianą w kontekście inżynierii wymagań. Oznacza to, że wystąpienie błędów w dokumencie wymagań lub potrzeba zmiany niektórych wymagań powoduje konieczność podjęcia odpowiednich działań zarządczych. Mają one na celu zapewnienie utrzymania spójności, zintegrowania i poprawności dokumentu wymagań. Oprócz tego istnieje jeszcze koncepcja leżąca pomiędzy rozumieniem szerokim i wąskim, która odnosi się do zarządzania wymaganiami rozumianego jako metody systematyzacji procesu odkrywania, analizy i dokumentowania wymagań. Zarządzanie wymaganiami może być prowadzone z wykorzystaniem różnych metod zależnych od sposobu rozumienia tego pojęcia. Każdorazowo odpowiedni zbiór wykorzystywanych w tym celu metod powinien być dopasowany do szczegółowych uwarunkowań przedsięwzięcia budowy systemu wsparcia teleinformatycznego SBN RP.



Rysunek 3.3. Proces inżynierii wymagań.

3.2.3 Cechy dobrej specyfikacji wymagań

Dokumentowanie wymagań umożliwia ich utrwalenie w taki sposób, aby było możliwe ich komunikowanie interesariuszom oraz późniejsze zarządzanie nimi i ich utrzymywanie. Dobra dokumentacja wymagań dostarcza odpowiedzi na pytania twórców oprogramowania w zakresie umożliwiającym im szybkie osiąganie założonych celów związanych z budową systemu i minimalizację związanych z tym kosztów. Cechy dobrego dokumentu wymagań to [IE98]:

- poprawność (ang. *correct*);
- jednoznaczność (ang. *unambiguous*);
- kompletność (ang. *complete*);
- spójność (ang. *consistent*);
- uporządkowanie wg ważności lub stabilności (ang. *ranked for importance or stability*);
- weryfikowalność (ang. *verifiable*);
- modyfikowalność (ang. *modifiable*);
- umożliwienie śledzenia powiązań (ang. *traceable*).

3.2.4 Metody zbierania wymagań

Posiadanie przekrojowej wiedzy z zakresu inżynierii wymagań stanowi bardzo ważny czynnik umożliwiający inżynierom wymagań / analitykom skutecz-

ne określanie zakresu i istotnych właściwości tworzonego systemu wsparcia teleinformatycznego. Interesariusze reprezentujący różne szczeble w ramach podmiotów tworzących organizację federacyjną, którzy są zaangażowani w budowę systemu, często w odmienny sposób rozumieją i komunikują swoje problemy. Powoduje to konieczność wypracowania najlepszego dla danej organizacji podejścia do opracowywania wymagań oraz umiejętnego doboru właściwych metod, które będą wykorzystane do pracy z różnymi interesariuszami i dopasowane do ich poziomu. Przyczynia się to również do aktywnego przeciwdziałania niektórym z problemów opisanych w punkcie 3.3 niniejszego opracowania. Oprócz interesariuszy reprezentujących ludzi, źródło wymagań mogą stanowić również elementy środowiska organizacyjnego i legislacyjnego organizacji federacyjnej, w którym osadzony będzie system teleinformatyczny. Ponadto należy pamiętać, że kontekst i otoczenie budowanego systemu będą zmieniać się w czasie zgodnie z naturą funkcjonowania i rozwoju tego typu organizacji. Również inżynierowie wymagań / analitycy posiadają różną wiedzę i doświadczenie w dziedzinie inżynierii wymagań, które ewoluują w czasie. Wszystko to powoduje, że budowy złożonych systemów wsparcia teleinformatycznego nie sposób opierać na wykorzystaniu do opracowywania wymagań jednej metody ogólnego przeznaczenia i przeprowadzeniu z jej użyciem pojedynczej sesji wymagań. Dlatego opracowywanie wymagań opiera się najczęściej na przeprowadzeniu szeregu takich sesji, które odbywają się równolegle lub sekwencyjnie. Podczas każdej z nich stosowana jest metoda lub zbiór metod dopasowane do źródła wymagań, uczestników i kontekstu danej sesji. Sesje te powtarzane są aż do momentu uzyskania konsensusu w drodze uzgodnienia i poprawnego udokumentowania wymaganych funkcji i właściwości oczekiwanych od systemu. Syntetyczny przegląd spotykanych w praktyce procesów opracowywania wymagań został przedstawiony w poniższej tabeli (Tabela 3.1).

Tabela 3.1. Procesy opracowywania wymagań.

Proces	**Charakterystyka**	**Zastosowanie**
Liniowy	Proces ten odznacza się małą złożonością i składa się z sekwencji pięciu aktywności: konceptualizacja problemu, analiza problemu, studium wykonalności i wybór opcji, analiza i modelowanie, dokumentowanie wymagań.	Małe przedsięwzięcia
Liniowo-iteracyjny	Proces ten kładzie nacisk na dostarczenie dokładnych specyfikacji wymagań i zakłada ich wielokrotną walidację przez interesariuszy. Składa się on z czterech aktywności tj. wydobywania, analizy i	Średnie przedsięwzięcia

Proces	Charakterystyka	Zastosowanie
	negocjowania, dokumentowania i walidacji wymagań. Aktywności te często nachodzą na siebie i są realizowane w iteracjach, jednak wymagania nadal podlegają liniowym przekształceniom. Proces trwa aż do uzyskania ostatecznej akceptacji wymagań przez interesariuszy.	
Iteracyjny	Proces ten zawiera aktywności podobne do tych, które istnieją w wyżej wymienionych procesach, lecz sam posiada iteracyjny i cykliczny charakter. W procesie tym występują interakcje pomiędzy działaniami związanymi z wydobywaniem, specyfikowaniem i walidacją wymagań oraz użytkownikiem i dziedziną problemową. Ponadto działania realizowane są nieliniowo i zakłada się pomiędzy nimi istnienie silnych związków przyczynowo-skutkowych.	Duże przedsięwzięcia charakteryzujące się cyklicznym wydawaniem kolejnych wersji (w tym prowadzone w oparciu o metodyki zwinne)
Spiralny	Proces ten wykonywany jest w cyklach, gdzie każdy kolejny przebieg dostarcza pełną wersję wymagań, na podstawie których powinien zostać zbudowany system. Każda spirala jest dodatkowo podzielona na cztery ćwiartki tj. wydobycie, analizę i negocjację, dokumentowanie i walidację wymagań. Proces ten jest w stanie zaadresować pojawiające się zagrożenia, mogące zwiększyć koszty projektu i obniżyć jakość dostarczanego produktu, takie jak np. opóźnienia w specyfikowaniu wymagań, zmiany wymagań, niski ROI itp.	Duże i złożone przedsięwzięcia

Należy podkreślić, że w rzeczywistości dość rzadko spotyka się implementację opisanych powyżej procesów opracowywania wymagań w ich modelowej formie. Najczęściej jeżeli formalne procesy są w ogóle stosowane w organizacjach o charakterze federacyjnym podejmujących inicjatywy związane z budową systemów teleinformatycznych, to mają one tendencję do traktowania standardowych procesów jako zbioru najlepszych praktyk, adaptowanych lokalnie na potrzeby danej organizacji.

W ramach procesów opracowywania wymagań można zastosować metody związane z odkrywaniem wymagań należące do kilku kategorii. Najczęściej występujący w literaturze przedmiotu podział wyróżnia następujące grupy metod:

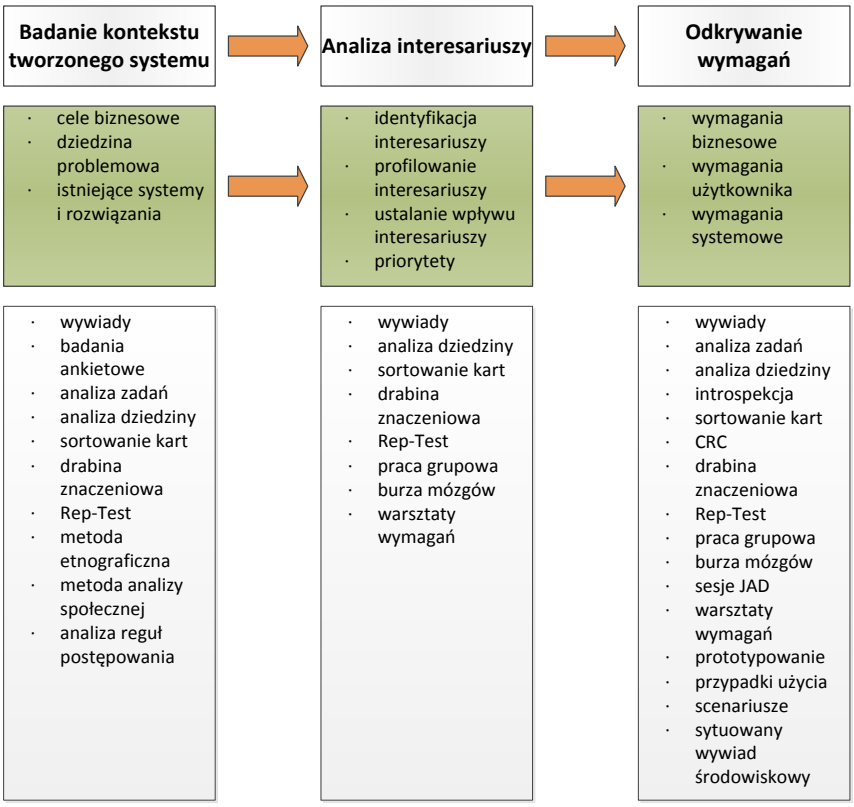
- klasyczne (ang. *classic / traditional*),
- poznawcze (ang. *cognitive*),
- kontekstowe (ang. *contextual*),
- współczesne (ang. *modern*).

Zostały one przedstawione w syntetyczny sposób w poniższej tabeli 3.2.

Tabela 3.2. Katalog metod odkrywania wymagań.

Metoda	**Grupa metod**			
	Klasyczne	**Poznawcze**	**Kognitywne**	**Współczesne**
analiza domeny (ang. *domain analysis*)	+			
analiza reguł postępowania (ang. *protocol analysis*)				+
analiza zadań (ang. *task analysis*)	+			
analizy społecznej (ang. *social analysis*)			+	
badania ankietowe (ang. *surveys*)	+			
burze mózgów (ang. *brainstorming*)				+
CRC (ang. *Class-Responsibility-Collaboration*)		+		
drabiny znaczeniowej (ang. *laddering*)		+		
etnograficzną (ang. *etnography*)			+	
introspekcja (ang. *introspection*)	+			
praca grupowa (ang. *group work*)				+
prototypowanie (ang. prototyping)				+
przypadki użycia (ang. *use cases*)				+
Rep-Test (ang. *repertory grids*)		+		
scenariusze (ang. *scenarios*)				+
sesje JAD (ang. *joint application development sessions*)				+
sortowanie kart (ang. *card sorting*)		+		
sytuowany wywiad środowiskowy (ang. *contextual inquiry*)				+
warsztaty wymagań (ang. *requirements workshops*)				+
wywiady (ang. *interviews*)	+			

Na wstępie niniejszego punktu przedstawiono powody, dla których budowa złożonych systemów teleinformatycznych dedykowanych wspomaganiu funkcjonowania organizacji o charakterze federacyjnym nie może opierać się na wykorzystaniu do opracowywania wymagań pojedynczej metody odkrywania wymagań. Rodzi to pytanie w jaki sposób analitycy mają dokonać wyboru właściwych metod, które umożliwią im uzyskanie spójnego zbioru rzeczywistych wymagań na dany system ICT, odwzorowującego potrzeby różnych interesariuszy. Niestety literatura przedmiotu nie dostarcza jednoznacznych odpowiedzi na to pytanie. Praktyka inżynierii wymagań wskazuje za to, że najlepsze efekty uzyskuje się dzięki łączeniu omówionych metod w taki sposób, aby uwzględniał on cel ich zastosowania, posiadane informacje dotyczące dziedziny problemowej, istniejące wymagania i ich jakość, dostępność interesariuszy. Ogólną strategię doboru metod na tej podstawie przedstawiono na poniższym rysunku (Rysunek 3.4).



Rysunek 3.4. Ogólna strategia wyboru metod odkrywania wymagań.

3.2.5 Metody oceny ważności wymagań

Decyzja o wyborze spośród zidentyfikowanych wymagań tych, które mają zostać zrealizowane w systemie wsparcia teleinformatycznego SBN RP jest jednym z najbardziej krytycznych rozstrzygnięć mającym wpływ na powodzenie całego przedsięwzięcia. Systemy wsparcia teleinformatycznego SBN RP, tworzone zarówno w Polsce jak i na świecie, są z reguły rozwiązaniami o dużej złożoności, zakresie funkcjonalnym i terytorialnym, w których budowę zaangażowanych jest wielu interesariuszy. Sytuacja ta rodzi szereg problemów związanych z ustaleniem wymagań mających znaczenie dla SBN RP jako całości, a nie tylko dla interesów ograniczonej grupy interesariuszy (np. pojedynczych ministrów, kierowników jednostek administracji rządowej i samorządowej, szefów służb itd.). W związku z tym w fazie analizy wymagań niezbędne jest podjęcie odpowiednich działań mających na celu ustalenie ich istotności oraz wyeliminowanie konfliktów pomiędzy nimi i optymalizację pod kątem korzyści uzyskiwanych z wdrożenia systemu. Działania te powinny zostać przeprowadzone z punktu widzenia SBN RP rozumianego jako jednej organizacji i obejmują [YE92, WI03, SS97, KO97]:

- wsparcie interesariuszy w wyborze głównych wymagań dla systemu;
- opracowanie uporządkowanej listy reprezentującej optymalny zbiór wymagań funkcjonalnych planowanych do zrealizowania w kolejnych wydaniach systemu;
- uzyskanie kompromisu dotyczącego wymagań wchodzących w zakres projektu względem ograniczeń takich jak: harmonogram, budżet, zasoby i jakość;
- zrównoważenie korzyści z każdego wymagania włączonego do realizacji względem kosztów jego implementacji.
- zrównoważenie oddziaływania wymagań wybranych do realizacji na architekturę oprogramowania i jego przyszły rozwój oraz związane z tym koszty.
- oszacowanie oczekiwanej satysfakcji interesariuszy i włączenie do realizacji tylko tej część wymagań pozwalających stworzyć system spełniający ich oczekiwania, który będzie przydatny w realizacji określonych celów w ramach SBN RP;
- uzyskanie przewagi względem dotychczas eksploatowanych rozwiązań technicznych;
- zminimalizowanie kosztów związanych z koniecznością wprowadzania poprawek wynikających z błędnego wyboru wymagań przeznaczonych do implementacji oraz zmaksymalizowanie stabilność planów budowy systemu;

- obsługę sprzecznych wymagań w procesie ich negocjacji i rozwiązywania nieporozumień pomiędzy interesariuszami;
- ustalenia istotności każdego wymagania, aby budowany system dostarczał największej wartości przy najniższych kosztach jego opracowania.

Powyższa lista pokazuje jak istotne jest znaczenie ustalania priorytetów wymagań i podejmowania decyzji o tym, które z nich zostaną zaimplementowane w systemie wsparcia teleinformatycznego SBN RP. Ruhe podsumowuje to następująco: *"wyzwaniem jest, aby wybrać właściwe wymagania z początkowego zbioru wymagań kandydujących tak, aby spełnione zostały różne kluczowe interesy, ograniczenia techniczne i preferencje krytycznych interesariuszy, a ogólna wartość produktu była zmaksymalizowana"* [RE02]. Niektórzy współcześni autorzy uważają, że proces nadawania priorytetów wymaganiom jest trywialny [SS97], inni z kolei traktują go jako zadanie o średniej trudności [WI03], a część uważa za jedno z najbardziej złożonych działań w inżynierii wymagań [LP93]. Jednak wszyscy z nich uważają prawidłowe nadanie priorytetów wymagań za zasadniczą aktywność przekładającą się bezpośrednio na ostateczny sukces projektu. Jednocześnie niektóre podręczniki dotyczące inżynierii wymagań nie podnoszą tej kwestii w jakimkolwiek stopniu [RR02].

Dotychczas opisano wiele metod umożliwiających ustalenia priorytetów poszczególnych obiektów w zbiorze. Celem każdej z tych metod jest przypisanie do poszczególnych obiektów wartości, które pozwalają na ustalenie relacji porządkujących między nimi. W przypadku rozważań będących przedmiotem niniejszego punktu, obiektami tymi są wymagania. Priorytety wymagań mogą zostać określone przy użyciu różnych metod i skal pomiarowych. Poniżej wymienione zostały najczęściej przywoływane w literaturze metody ustalania priorytetów:

- metoda AHP;
- binarne drzewo poszukiwań;
- metoda grupowania;
- metoda rankingowania;
- głosowanie kumulatywne;
- metoda dziesięciu głównych wymagań;
- metoda QFD;
- zaadaptowany model CAPM;
- metoda DCPT;
- metoda EVOLVE;
- metoda Cost-Value;
- metoda Quantitative Win-Win;

Część z wymienionych wyżej metod zakłada, że każde wymaganie ma przypisany pewien priorytet, natomiast inna część grupuje wymagania ze względu

na ich istotność. Metody te w większości nie uwzględniają podejścia grupowego do podejmowania decyzji, ani siły głosu poszczególnych interesariuszy. Abstrahują one również od aspektów związanych z zależnościami pomiędzy wymaganiami a kosztami, wartością, ludźmi i ich kompetencjami, aspektami technicznymi itd. Wyjątek stanowią metody DCPT, EVOLVE, Cost-Value oraz Quantitative Win-Win, które stanowią próbę uwzględnienia bardziej złożonych, wieloaspektowych zagadnień w obszarze ustalania priorytetów wymagań interesariuszy. Na polskim gruncie badań dotyczących tego obszaru ciekawą alternatywę do tych metod zaproponował Sobczak w postaci Techniki Wielopłaszczyznowej Priorytetyzacji Wymagań (TWPW). Metoda ta integruje zagadnienia technologiczne, organizacyjne oraz ekonomiczne i oparta jest o koncepcję strategicznej karty wyników (ang. *balanced scorecard*) Kaplana-Nortona[SA05].

3.3 Praktyczne problemy związane z inżynierią wymagań w organizacjach o charakterze federacyjnym

W organizacjach o charakterze federacyjnym wciąż występują problemy związane z wymaganiami w obszarze rozwiązań ICT, które w organizacjach tradycyjnych wydają się być łatwe do złagodzenia lub całkowitego wyeliminowania. Niestety w odniesieniu do organizacji federacyjnych nierzadko prowadzą one do niepowodzenia przedsięwzięć w obszarze budowy rozwiązań teleinformatycznych. Występujące współcześnie problemy związane z inżynierią wymagań można zaliczyć do następujących grup [SO03, SS97, CK92, FI07]:

- problemy z zakresem;
- problemy z komunikacją i zrozumieniem;
- problemy z jakością;
- problemy z zarządzaniem.

Ich występowanie jest tym częstsze im więcej podmiotów składa się na organizację federacyjną oraz im więcej interesariuszy jest zaangażowanych w tworzenie danego rozwiązania - szczególnie w obszarze problemów związanych z komunikacją i zrozumieniem.

Problemy z zakresem stanowią poważną przeszkodę w realizacji projektów związanych z tworzeniem systemów wsparcia teleinformatycznego. Wynikają one najczęściej z niedookreślonych potrzeb instytucji będącej odbiorcą tworzonego rozwiązania i zmian oczekiwań odbiorcy w trakcie trwania projektu związanych ze zdobywaniem przez niego doświadczenia i budowaniem nowej wiedzy w danej dziedzinie problemowej lub zmianami w samej organizacji federacyjnej. Przyczynia się to do pojawienia się zjawiska tzw. "pełzania zakresu" (ang. *scope creep*), który oznacza wprowadzanie w sposób niekontrolowany

dużej liczby pojedynczych, drobnych zmian w projekcie. Zmiany te rozpatrywane w oderwaniu od siebie nie wydają się w sposób istotny oddziaływać na tworzone rozwiązanie. Jednak występujące łącznie mogą powodować wymknięcie się zakresu projektu spod kontroli i przekroczenie założonego harmonogramu i/lub budżetu. Zakres systemu może ulegać zmianie w trakcie trwania projektu również na skutek zmian w funkcjonowaniu organizacji zamawiającej system. Ma to szczególne znaczenie w przypadku organizacji o charakterze federacyjnym, w których procesy operacyjne, sposób zarządzania, model współpracy z podmiotami zewnętrznymi itp. mogą ulegać dynamicznym i częstym przeobrażeniom.

Problemy związane z komunikacją i zrozumieniem stanowią grupę najpoważniejszych przeszkód spotykanych w organizacjach o charakterze federacyjnym. Jeśli nie występuje jednakowe zrozumienie dziedziny problemowej pomiędzy dostawcą systemu a zamawiającą go organizacją oraz użytkownicy i analitycy / inżynierowie wymagań mówią różnymi językami , wówczas zbudowanie systemu spełniającego oczekiwania odbiorcy jest niemożliwe. Problemy te związane są m.in. z brakiem świadomości reprezentantów odbiorców w zakresie własnych potrzeb i leżących u ich źródeł motywacji wynikających z celów organizacji federacyjnej oraz możliwości i ograniczeń technologicznych. Inne istotne aspekty polegają na istnieniu różnicy zdań pomiędzy różnymi przedstawicielami tej samej organizacji stojącej po jednej ze stron oraz definiowaniu wymagań niejasnych i nie dających się przetestować. Również analitycy / inżynierowie wymagań biorący udział w procesie opracowywania wymagań często nie mają wystarczającej wiedzy dziedzinowej umożliwiającej prawidłową komunikację z odbiorcą systemu. Problemy z komunikacją wiążą się również z liczbą stron zaangażowanych w proces opracowywania wymagań i ich poziomem abstrakcji w postrzeganiu różnych problemów. Informacje uzyskane tylko od jednej z grup na danym poziomie (np. kierowników średniego szczebla), będą cechować się stronniczością i stanowić reprezentację wyłącznie ich problemów i oczekiwań. Niezbędne jest włączenie w proces tworzenia systemu reprezentantów wszystkich zainteresowanych stron, ponieważ tylko takie podejście umożliwia uzyskanie kompleksowego obrazu budowanego rozwiązania. Niestety rodzi ono również problemy związane z ustaleniem priorytetów wymagań oraz określeniem "ważności" zdania poszczególnych interesariuszy w tym zakresie.

Problemy z jakością wiążą się, z jednej strony, z niskim poziomem wymagań zapisanych w specyfikacji, które często są dwuznaczne, niespójne, niekompletne, nielogiczne, niepoprawne, nieaktualne, niewymagane, wyspecyfikowane

z użyciem żargonu technicznego, wkraczające w sferę projektu poprzez wskazanie rozwiązań na poziomie technicznym, niemożliwe do spełnienia itp. Z drugiej strony problemy te wynikają z niedostatecznych kompetencji inżynierów wymagań / analityków, którzy nie mają odpowiedniego przeszkolenia w tym zakresie i nie posiadają dostatecznej znajomości metod, technik i narzędzi inżynierii wymagań. Brak umiejętności odkrywania wymagań może prowadzić do istotnych luk w ich specyfikacji, przekładających się na powstanie istotnych braków funkcjonalnych systemu. Osoby odpowiedzialne za opracowywanie wymagań, które nie mają odpowiedniego doświadczenia w tym zakresie, również często przeceniają możliwości poznanych technik i narzędzi. Nie zawsze są one adekwatne do rozwiązania napotykanych problemów, lecz brak doświadczenia uniemożliwia obiektywne stwierdzenie takiego stanu rzeczy.

Problemy z zarządzaniem wiążą się m.in. z brakiem stosowania lub nieprawidłowym wykorzystaniem dobrych praktyk, metod, technik i narzędzi w procesie odkrywania, analizy i dokumentowania wymagań. Najczęściej spotykane w praktyce problemy dotyczą organizacji samego procesu inżynierii wymagań, wykorzystania narzędzi wspomagających zarządzanie wymaganiami, sposobu śledzenia wymagań i zarządzania zmianą. Często spotykaną sytuacją jest wykorzystywanie nieadekwatnych takich jak np. arkusze kalkulacyjne, dokumenty tekstowe, lokalne bazy danych itp. Przechowywanie wymagań w takiej formie, zamiast w wyspecjalizowanym repozytorium zarządzanym przy użyciu odpowiedniego narzędzia, uniemożliwia prawidłowe zarządzanie ich cyklem życia i ma szczególnie negatywne konsekwencje w przypadku tworzenia rozwiązań ICT dedykowanych dla organizacji federacyjnych. Ponadto brak scentralizowanego i zautomatyzowanego narzędzia do zarządzania wymaganiami powoduje m.in. ograniczenia w zakresie wykorzystania metryk takich jak np. stabilność, dojrzałość, kompletność wymagań i raportowania ich wartości. Wykorzystanie źle dobranych narzędzi wpływa równocześnie na sposób śledzenia wymagań i zarządzania ich zmianą, co samo w sobie stanowi kolejne duże wyzwanie zarządcze. Śledzenie wymagań obejmuje m.in. dokumentowanie ich źródeł pochodzenia, powiązania z elementami architektury systemu i jego projektu. Nieprawidłowości w zakresie śledzenia wymagań utrudniają lub wręcz uniemożliwiają prawidłową ocenę proponowanych zmian wymagań pod kątem ich wpływu na tworzone rozwiązanie, jego architekturę, projekt, implementację i testowanie. Istnienie wyżej wymienionych problemów z zakresu zarządzania przyczynia się w istotny sposób do niepowodzeń projektów związanych z budową systemów teleinformatycznych dedykowanych dla organizacji federacyjnych.

3.4 Podsumowanie

Identyfikacja wymagań w organizacjach o charakterze federacyjnym stanowi wyzwanie, którego podjęcie powinno przyczynić się zarówno do identyfikacji systemu wartości i strategii rozwoju samej organizacji, jak również do określenia oczekiwań w stosunku do systemów ICT, których zadaniem będzie wspieranie jej funkcjonowania. Na tym tle zaprezentowano m.in. wybrane aspekty inżynierii wymagań i występujące współcześnie problemy dotykające organizacji federacyjnych w tym obszarze, a także zaproponowano ogólną strategię wyboru metod odkrywania wymagań i przedstawiono katalog metod oceny ważności wymagań pozwalających na nadawanie im priorytetów oraz na uwzględnienie podejścia grupowego i siły głosu poszczególnych interesariuszy. Podsumowując uzyskane dotychczas wyniki można stwierdzić, że płaszczyzna inżynierii wymagań w odniesieniu do organizacji o charakterze federacyjnym stanowi obecnie bardzo ważny obszar badań z dużym potencjałem rozwoju, dostarczający wielu wyzwań i potrzeby udzielenia odpowiedzi na szereg wciąż otwartych pytań. Otwarte kwestie, które należy uwzględnić w dalszych pracach z tego zakresu obejmują m.in.:

- lepsze zrozumienie reguł rządzących zachowaniem i sposobem funkcjonowania złożonych organizacji o charakterze federacyjnym,
- ustalenie szczegółowych potrzeb w zakresie rozwoju wieloaspektowego podejścia do ustalania priorytetów wymagań pochodzących od interesariuszy reprezentujących autonomiczne podmioty tworzące organizacje federacyjne,
- opracowania strategii radzenia sobie przez organizacje federacyjne ze zjawiskami mającymi negatywny wpływ na stabilność i jakość wymagań w sytuacji, kiedy same mogą ulegać dynamicznym przemianom za którymi muszą podążać ich systemy ICT.

Bibliografia

[BB13] *Biała Księga Bezpieczeństwa Narodowego Rzeczypospolitej Polskiej*, BBN, Warszawa, 2013.

[CK92] Christel M. G., Kang K. C., *Issues in Requirements Elicitation. Technical report CMU/SEI-92-TR-012*, Software Engineering Institute Carnegie Mellon University, Pittsburgh 1992

[FI07] Firesmith D., *Common Requirements problems, their negative consequences, and the industry best practices to help solve them*, Journal of object technology, 2007, vol. 6, no. 1, s. 17 - 23.

[IE98] *Standard IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications*, Institute of Electrical and Electronics Engineers, 1998.

[KO97] Karlsson J., Olsson S., Ryan K., *Improved Practical Support for Large-Scale Requi-*

rements Prioritizing, Requirements Engineering, 1997, vol. 2, issue 1, s. 51 - 60.

[KS98] Kotonya G., Sommerville I., *Requirements Engineering: Process and Techniques*, John Wiley & Sons, 1998;

[LP93] Lubars M., Potts C., Richter C., *A Review of the State of Practice in Requirements Modeling*, [w:] *Proceedings of IEEE International Symposium on Requirements Engineering*, IEEE Computer Society, Los Alamitos 1993

[PA14] Pawłowski J. i in., *Misja, cele i zadania SBN RP*, raport z realizacji podzadania badawczego nr 5.1. projektu nr O ROB/0076/03/001 pt. "System Bezpieczeństwa Narodowego RP" w zakresie obronności i bezpieczeństwa państwa finansowanego ze środków NCBiR, Warszawa 2014.

[PR13] Protasowicki T., *Przegląd metodyk realizacji systemów wsparcia teleinformatycznego SBN w innych krajach, w tym z NATO i UE*, raport z realizacji podzadania badawczego nr 6.1. projektu nr O ROB/0076/03/001
pt. "System Bezpieczeństwa Narodowego RP" w zakresie obronności i bezpieczeństwa państwa finansowanego ze środków NCBiR, Warszawa 2013.

[PR14a] Protasowicki T., *Koncepcja wsparcia teleinformatycznego SBN RP*, raport z realizacji podzadania badawczego nr 6.2. projektu nr O ROB/0076/03/001
pt. "System Bezpieczeństwa Narodowego RP" w zakresie obronności i bezpieczeństwa państwa finansowanego ze środków NCBiR, Warszawa 2014.

[PR14b] Protasowicki T., *Wybrane aspekty zastosowania koncepcji architektury korporacyjnej w transformacji Systemu Bezpieczeństwa Narodowego RP*, Zeszyty Naukowe Uniwersytetu Szczecińskiego Ekonomiczne Problemy Usług, vol. 112, 2014

[PS10] Pandey D., Suman U., Ramani A. K., *An Effective Requirement Engineering Process Model for Software Development and Requirements Management*, [w:] *ARTCOM '10 Proceedings of the 2010 International Conference on Advances in Recent Technologies in Communication and Computing*, IEEE Computer Society, Washington 2010

[RE02] Ruhe G., Eberlein A., Pfahl D., *Quantitative WinWin – A New Method for Decision Support in Requirements Negotiation*, [w:] *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE'02)*, ACM Press, New York 2002

[RR02] Robertson S., Robertson J., *Mastering the Requirements Process. Third Edition*, Addison-Wesley Professional, London 2012; por. Bray I. K., *An Introduction to Requirements Engineering*, Addison Wesley, London 2002

[SA05] Sobczak A., *Identyfikacja kluczowych wymagań w procesie budowy systemów informatycznych*, [w:] *Nowe koncepcje i metody w systemach informatycznych*, red. Januszewski A., Drelichowski L., Akademia Techniczno-Rolnicza w Bydgoszczy, Bydgoszcz 2005

[SA90] Sailor J.D., *System engineering: An introduction*, [w:] *System and Software Requirements Engineering*, red. Thayer R.H., Dorfman M., IEEE Computer Society Press, Los Alamitos 1990.

[SB98] Stevens R., Brook P., Jackson K., Arnold S., *Systems Engineering - Coping with Complexity*, Prentice Hall, London 1998.;

[SO03] Sommerville I., *Inżynieria Oprogramowania*, WNT, Warszawa 2003.

[SS97] Sommerville I., Sawyer P., *Requirements Engineering: A Good Practice Guide*, John Wiley & Sons, Chicheter 1997.

[WI03] Wiegers K., *Software Requirements 2*, Microsoft Press, Washington 2003.

[YE92] Yeh A.C., *REQUirements Engineering Support Technique (REQUEST) – A Market*

Driven Requirements Management Process, [w:] *Proceedings of Second Symposium of Quality Software Development Tools*, IEEE Computer Society, Piscataway 1992

Rozdział 4

Inżynieria wymagań w modernizacji systemów informatycznych na przykładzie systemów wspierających oświatę w Polsce

Niniejszy rozdział prezentuje metody inżynierii wymagań zastosowane podczas pracy nad I etapem projektu „Modernizacja systemów informatycznych do obsługi egzaminów zewnętrznych i nadzoru pedagogicznego". Instytucje odpowiedzialne za kształtowanie polityki oświatowej w Polsce korzystają z szeregu systemów informatycznych. Systemy te, mimo iż operują na tych samych danych, nie komunikują się ze sobą. Przedmiotem I etapu projektu była analiza istniejących systemów oraz opracowanie koncepcji nowego, zintegrowanego systemu wspierającego kształtowanie polityki oświatowej. W rozdziale zaprezentowano metody inżynierii wymagań zastosowane podczas pracy nad projektem: spotkania z ekspertami, wywiady z użytkownikami, słowne opisy wymagań, prototypowanie, analiza użyteczności. Omówiono zalety i wady poszczególnych metod oraz przedstawiono uogólnione przesłanki, którymi można się kierować przy definicji i analizie wymagań dla realizacji projektu o podobnej skali.

Celem niniejszego rozdziału jest przedstawienie doświadczeń zdobytych podczas realizacji pierwszego etapu projektu "Modernizacja systemów informatycznych do obsługi egzaminów zewnętrznych i nadzoru pedagogicznego" (współfinansowanego z Europejskiego Funduszu Społecznego w ramach Programu Operacyjnego Kapitał Ludzki 2007-2013). Projekt realizowany był przez konsorcjum, w skład którego weszły: Centrum Informatyczne Edukacji (CIE), Okręgowa Komisja Egzaminacyjna w Krakowie (OKE Kraków), Poznańskie Centrum Superkomputerowo - Sieciowe afiliowane przy Instytucie Chemii Bioorganicznej Polskiej Akademii Nauk (PCSS). Projekt prowadzony był od 04.2013 do 05.2014.

Polityka oświatowa w Polsce kształtowana jest przez szereg instytucji centralnych, np. Ministerstwo Edukacji Narodowej (MEN), Centralna Komisja Egzaminacyjna (CKE), Instytut Badań Edukacyjnych (IBE), Ośrodek Rozwoju Edukacji (ORE) oraz lokalnych, jak np. okręgowe komisje egzaminacyjne (OKE), jednostki samorządu terytorialnego (JST), kuratoria oświaty (KO), szkoły i placówki oświatowe (PO), poradnie psychologiczno – pedagogiczne (PPP). Zarówno centralne jak i lokalne instytucje korzystają z wielu systemów informatycznych. Mimo, że systemy te operują na tych samych danych (np. dane osobowe uczniów, historia edukacji uczniów, dane teleadresowe placówek oświatowych) i wymagają ich spójności, nie są one ze sobą zintegrowane, co skutkuje koniecznością wielokrotnego wprowadzania tych samych danych przez użytkowników oraz koniecznością kosztownego uspójniania danych. Po-

nadto, niezależność lokalnych instytucji skutkuje duplikacją systemów (np. systemy w OKE wspierające procesy planowania, przeprowadzania i oceniania egzaminów). Analizy prowadzone przez MEN [MOD13] wykazały, że eliminacja redundancji baz danych oraz wprowadzenie centralnego rejestru danych referencyjnych usprawniłoby działanie już istniejących systemów informatycznych wspierających kształtowanie polityki oświatowej.

W projekcie wyróżniono następujące zadania:

1. Inwentaryzacja systemów używanych w oświacie i określenie ich aktualnej oraz potencjalnej współpracy.
2. Określenie możliwości technicznych i prawnych integracji powyższych systemów.
3. Inwentaryzacja użytkowników powyższych systemów.
4. Zebranie wymagań od użytkowników.
5. Opracowanie koncepcji zintegrowanego (w wyniku modernizacji) systemu oraz przeprowadzenie dowodu wykonywalności zintegrowanego systemu.
6. Testy użyteczności dowodu wykonywalności.

W rozdziale skupimy się na technikach inżynierii wymagań wykorzystanych podczas zbierania wymagań od użytkowników oraz zaprezentujemy, opracowany w PCSS, wzorzec interoperacyjności wspierający określanie możliwości technicznych integracji systemów informatycznych. Zaprezentujemy również metody wykorzystane podczas testów użyteczności opracowanego dowodu wykonywalności.

4.1 Zakres prac

Projekt rozpoczął się od inwentaryzacji systemów używanych przez centralne i lokalne instytucje kształtujące politykę oświatową w Polsce. W rezultacie zidentyfikowano:

- ponad 60 różnych systemów wspomagających obsługę egzaminów zewnętrznych w 8 OKE
- 20 systemów wspomagających pracę 16 KO
- 6 niezależnych systemów wykorzystywanych przez instytucje centralne
- dużą (nieokreśloną) liczbę systemów wykorzystywanych w JST oraz w PO.

Mimo iż każda jednostka odpowiedzialna za kształtowanie polityki oświatowej w Polsce (jednostek tych jest ponad 70.000) jest użytkownikiem Systemu

Informacji Oświatowej (SIO), to, z uwagi na jego funkcjonalność, szczegółowo zdefiniowaną w ustawie o systemie informacji oświatowej [SIO11], do sprawnego realizowania swoich zadań wykorzystuje dodatkowe systemy informatyczne. Z uwagi na szeroki zakres narzędzi komercyjnych oraz brak powszechnie uznanych standardów (np. do opisu danych ucznia), dokładna liczba systemów używanych w JST oraz PO jest niemożliwa do oszacowania. Bez wątpienia jednak jest to liczba duża. Inwentaryzacji systemów dokonano na podstawie ankiet rozsyłanych do instytucji zajmujących się kształtowaniem polityki oświatowej oraz od ekspertów współpracujących z projektem. Równolegle do inwentaryzacji systemów informatycznych prowadzona była inwentaryzacja użytkowników tychże systemów.

W wyniku analizy funkcjonalności zidentyfikowanych systemów stwierdzono, iż SIO jest systemem, który ma największy potencjał jeśli chodzi o integrację systemów w oświacie. Potencjał ten wynika przede wszystkim z powszechności SIO i ustawowej obligatoryjności jego użycia przez instytucje kształtujące politykę oświatową oraz ze specyfiki danych przetwarzanych przez ten system. W celu określenia technicznych możliwości integracji SIO z innymi systemami używanymi w oświacie, posłużono się tzw. wzorcem interoperacyjności. Zdecydowano również o przeprowadzeniu analizy technicznej, która polegała m.in. na statycznej analizie oraz na inspekcji kodu źródłowego.

Po inwentaryzacji użytkowników zebrano od nich wymagania względem docelowego, zintegrowanego systemu. Powstały one na podstawie:

- analizy wymagań inwestora (MEN),
- słownych opisów (ang. *user stories*) użytkowników i ekspertów współpracujących z projektem,
- prototypowania oraz zbierania opinii od użytkowników i ekspertów współpracujących z projektem,
- testów użyteczności badających doświadczenie użytkownika (ang. *user experience*).

Na podstawie analizy systemów wykorzystywanych w kształtowaniu polityki oświatowej oraz na podstawie analizy wymagań użytkowników opracowano koncepcję zintegrowanego systemu i przeprowadzono dowód jego wykonywalności. Dowód wykonywalności przygotowany został metodą prototypowania, gdzie kolejne wersje prezentowane były wybranej grupie użytkowników docelowego systemu. Podczas tych prezentacji przeprowadzano również testy użyteczności, których rezultaty były modelowane jako kolejne wymagania względem opracowywanego, zintegrowanego systemu.

4.2 Metodyka inżynierii wymagań

Realizacja zadań w pierwszym etapie projektu "Modernizacja systemów informatycznych do obsługi egzaminów zewnętrznych i nadzoru pedagogicznego" nakładała konieczność zastosowania szeregu metod inżynierii wymagań: analizę ekspercką (analizę istniejących systemów w połączeniu z badaniem interoperacyjności za pomocą wzorca); wywiady z ekspertami i użytkownikami (w połączeniu z zastosowaniem ustrukturyzowanego opisu wymagań oraz ankiet), prototypowanie oraz zastosowanie analizy użyteczności. Podział ten wynikał z określonych uwarunkowań i motywacji a każda z metod posiadała swoje zalety i wady. Przed rozpoczęciem prac dokonano ewaluacji zastosowanych metod inżynierii oprogramowania w odniesieniu do warunków projektu i założonych celów. Poniższy rozdział zawiera podsumowanie przeprowadzonej ewaluacji.

4.2.1 Analiza ekspercka

Analiza ekspercka uzasadniona jest wtedy, jeżeli istnieje konieczność dogłębnego badania technicznego systemu informatycznego (np. w przypadku próby określenia jego bezpieczeństwa, wydajności, możliwości integracji). Z uwagi na dużą czasochłonność i kosztowność analizy eksperckiej, powinna być ona stosowana tylko do niewielkiej liczby krytycznych systemów. Analiza ekspercka pozwala na uzyskanie wyników (przede wszystkim wymagań niefunkcjonalnych) o bardzo wysokiej jakości. W projekcie będącym tematem niniejszego rozdziału, analiza ekspercka została zastosowana w przypadku jednego, kluczowego systemu, jakim był system SIO.

Analiza ekspercka wymaga dostępu do kodu źródłowego oraz do statystyk środowiska produkcyjnego (np. obciążenie serwera aplikacji, obciążenie serwera bazy danych, itp. Sporym ułatwieniem na każdym, nie tylko początkowym, etapie pracy jest dostępność i możliwość konsultacji z wykonawcami badanego systemu. Naturalnym jest, że analiza ekspercka, będąca swego rodzaju audytem, jest analizą krytyczną. Niedostateczne umotywowanie wniosków czy nieodpowiednio dobrane przykłady, mogą stać się zarzewiem rozbieżności, czy wręcz konfrontacji, pomiędzy analitykami a aktualnym wykonawcą

4.2.2 Wywiady z użytkownikami

Wywiady zapewniają możliwość bezpośredniego spotkania i rozmowy z rzeczywistymi użytkownikami systemu informatycznego. Dzięki dynamice takiego spotkania (przede wszystkim możliwość eksploracji wybranych obsza-

rów problemu, moderowania dyskusji i odbiegania od przyjętego skryptu czy szablonu) możliwe jest poznanie potrzeb użytkownika, które są trudne do nazwania i opisania (np. motywacje, przyzwyczajenia, subiektywne odczucia, uprzedzenia, itp.). Z drugiej strony spotkania takie są czasochłonne, zarówno z uwagi na czas trwania spotkania jak i na kosztowność opracowania wyników. Komfortową sytuacją jest możliwość przeprowadzenia wielokrotnych wywiadów z użytkownikiem, analiza i ewentualna redefinicja wcześniej zebranych wymagań. W takim przypadku minimalizuje się prawdopodobieństwo niezrozumienia lub niewłaściwej interpretacji wymagań użytkownika.

Bezpośrednie spotkanie z użytkownikiem w miejscu jego pracy daje również możliwość podejrzenia narzędzi, z których na co dzień on korzysta, organizacji pracy oraz charakterystyki użytkowanych urządzeń końcowych (np. wielkość monitora, szybkość łącza internetowego). Informacje te są niedocenionym źródłem wymagań niefunkcjonalnych, których zdefiniowanie jest często dla użytkownika nieoczywiste.

Bezpośrednie wywiady z użytkownikami pozwalają na uzyskanie dużej liczby wymagań, zatem powinny być stosowane, gdy liczba użytkowników nie jest duża, lub gdy spośród wszystkich użytkowników można wyselekcjonować reprezentatywną grupę.

4.2.3 Ankiety

W przeciwieństwie do zbierania wymagań podczas bezpośrednich spotkań z użytkownikami, skorzystanie z ankiet nie zapewnia interakcji i nie pozwala na pozyskanie dodatkowej wiedzy (np. motywacji, przyzwyczajeń czy też subiektywnych odczuć). Z drugiej jednak strony możliwe jest zebranie wymagań od bardzo dużej liczby ankietowanych. Zamknięta formuła tej metody ułatwia opracowywanie wyników. W zależności od zastosowanej metodyki (ankiety drukowane, ankiety online) możliwa jest automatyczna analiza wyników, wraz z wygenerowaniem statystyk. Brak bezpośredniego kontaktu z użytkownikiem pociąga za sobą brak możliwości wyjaśnienia wątpliwości w przypadku niezrozumienia pytania przez ankietowanego.

Przygotowanie dobrej ankiety wymaga od autora uwagi i zaangażowania. Szczególny nacisk należy położyć na opracowanie spójnej i jednoznacznej terminologii. Warto również, aby pierwsza wersja ankiety została wypełniona przez wybraną, zwykle małą, grupę użytkowników. Na podstawie doświadczeń zebranych w małej próbie, można do ankiety wprowadzić wymagane poprawki. Szerokiemu gronu użytkowników zaprezentować należy poprawioną wersję ankiety.

Skorzystanie z ankiet jest zatem metodą oszczędzającą czas i koszty zebrania wymagań, lecz potencjalnie może mieć negatywny wpływ na jakość wyników. Należy również położyć odpowiednio duży nacisk na formę ankiety i jej złożoność, tak aby w rezultacie uzyskać oczekiwany obraz wymagań użytkowników.

4.2.4 Ustrukturyzowany opis wymagań

Ustrukturyzowany opis wymagań zakłada, że użytkownicy przekazują wymagania w ujednoliconej, spisanej formie. Zwinne metodyki wytwarzania oprogramowania [COH04] zalecają, żeby opis taki miał formę tzw. historii użytkownika (ang. *user stories*). Każda z historii składa się z trzech części:

— użytkownik,
— czynność,
— wartość biznesowa.

W uproszczeniu, historię użytkownika można przedstawić jako wypowiedź typu: „Jako użytkownik chciałbym móc wykonać czynność, aby osiągnąć założony cel". Opcjonalnie, każda historia użytkownika może zawierać tzw. kryteria akceptacyjne. Przykładem historii użytkownika byłaby zatem wypowiedź: „Jako dyrektor szkoły chciałbym mieć możliwość wydrukowania listy zdających, aby sprawdzić obecność na egzaminie."

Uznaje się, że najważniejszą częścią historii użytkownika jest wartość biznesowa, gdyż to zbiór wartości biznesowych formułuje listę pożądanych funkcjonalności. Tworzenie ustrukturyzowanych opisów wymagań w formie historii użytkownika wymaga dużego zaangażowania ze strony tegoż użytkownika. W rezultacie minimalizuje się jednak prawdopodobieństwo otrzymania nieprzemyślanego czy zbędnego wymagania. W przypadku dużej liczby użytkowników znacząco może wzrosnąć koszt analizy zgłoszonych wymagań (z uwagi chociażby na potencjalnie powielające się historie).

Korzystanie z ustrukturyzowanego opisu wymaga od użytkownika doświadczenia w formalnej definicji wymagań dla systemu i chęci zaangażowania w ten proces. Można tę metodę włączyć do wywiadu z użytkownikiem, co pozwoli na uzyskanie jeszcze lepszych rezultatów (zwiększając jednocześnie czasochłonność). W pożądanym przypadku, docelowa grupa odbiorców systemu ma doświadczenie w wypełnianiu dokumentów z opisem wymagań a ponadto potrafi zdefiniować kryteria akceptacyjne. Wówczas zebrane wymagania stanowią doskonałe wejście do rozpoczęcia prac programistycznych w ramach zwinnej metodyki, gdzie zdefiniowane przez użytkowników historie można łatwo rozdzielić na zadania programistyczne.

4.2.5 Prototypowanie

Prototypowanie zakłada wielokrotne tworzenie nowych wersji oprogramowania (ale również innego dowolnego produktu) w celu zaprezentowania go użytkownikom i uzyskania natychmiastowej weryfikacji wymagań i założeń projektowych. Prototypowanie stwarza możliwość bezpośredniej, iteracyjnej i przyrostowej pracy z użytkownikami znacznie minimalizując ryzyko niezrozumienia ich oczekiwań. Inną wartością jest możliwość uzyskania dodatkowych wymagań i zgłoszeń nieprawidłowości na wczesnym etapie projektu.

Prototypowanie pociąga jednak za sobą zwiększony koszt fazy zbierania wymagań (wynikający z konieczności zatrudnienia programistów już na wczesnym etapie projektu). Bez wątpienia jednak umożliwia zebranie bardzo szczegółowych wymagań dodatkowo dostarczając ich wstępnej weryfikacji.

W przypadku projektów informatycznych, prototypowanie ograniczyć można do zbudowania makiety interfejsu użytkownika. Makieta ta powinna ukazywać możliwości docelowego systemu, odwzorowywać układ graficzny i pozwalać na interakcję umożliwiającą symulację przeprowadzenia czynności, które wspierane będą przez docelowy system.

4.2.6 Analiza użyteczności prototypu

Analiza użyteczności prowadzona jest podczas prezentacji użytkownikom kolejnych wersji prototypu i polega na obserwacji następujących elementów: reakcje (gesty, mimika) użytkownika, komentarze użytkownika wypowiadane podczas interakcji z prototypem, sekwencje działań podejmowanych przez użytkownika w celu wykonania zadania, czas potrzebny na wykonanie zadania oraz analiza ruchu gałek ocznych (ang. *eye tracking*). W celu dokładnego opracowania wyników zaleca się rejestrację sesji testowej, czyli zarejestrowanie twarzy użytkownika, wypowiadanych komentarzy oraz zdarzeń prezentowanych na ekranie

Analizę użyteczności można wspomagać za pomocą urządzenia zwanego okulografem (ang. *eye-tracker*). Okulograf pozwala na śledzenie ruchu gałek ocznych, co w rezultacie daje możliwość analizy ścieżek wzroku (ang. *scan paths*) oraz tzw. map termicznych (ang. *heat maps*). Analiza ścieżek wzroku pozwala na odtworzenie kolejności przeglądania elementów interfejsu użytkownika, natomiast mapy termiczne pozwalają na określenie czasu, przez jaki badany patrzył na poszczególne elementy interfejsu użytkownika. Warto podkreślić, że analiza użyteczności wykonana przy użyciu okulografu powinna być połączona z pozostałymi informacjami zebranymi w toku badania, jak komenta-

rze i pytania badanego czy notatki nadzorującego testy. Inaczej wyniki takiego badania mogą okazać mało wartościowe.

Analiza użyteczności zwiększa koszt fazy zbierania wymagań, pośrednio (tzn. po przeprowadzeniu interpretacji rezultatów) dostarczając nowych wymagań niefunkcjonalnych, które mogą mieć znaczący wpływ na intuicyjność korzystania z systemu.

4.3 Inżynieria wymagań w projekcie

4.3.1 Analiza interoperacyjności

Elementem zadania projektowego obejmującego usprawnienie funkcjonowania SIO było przeprowadzenie analizy technologicznej zastanego rozwiązania informatycznego. Celem analizy było scharakteryzowanie systemu z punktu widzenia interoperacyjności i możliwości integracji z innymi rozwiązaniami teleinformatycznymi.

Dla usystematyzowania analizy technicznej posłużono się wzorcem otwartości i interoperacyjności autorstwa PCSS (rozdział 4.3.1.1). Stanowi on zestaw pytań i wskazówek mających na celu stworzenie tzw. „karty interoperacyjności" analizowanego systemu. Wzorzec pozwolił na inwentaryzację istniejącego systemu w celu określenia jego spójnego obrazu (technicznego), jak i możliwości jego rozszerzenia. Biorąc pod uwagę priorytety analizy technicznej, zakres prac objął architekturę systemu, interfejsy komunikacyjne do innych systemów, opis modelu danych i podstawowych zbiorów danych przechowywanych w systemie. Wyniki analizy technologicznej w oparciu o wzorzec interoperacyjności stanowiły jeden z punktów wyjścia do podjęcia decyzji o zakresie w jakim system SIO może zostać wykorzystany jako centralny element całego środowiska IT oświaty. Ponadto zebrane dane miały także pośredni wpływ na definicje wymagań niefunkcjonalnych, jak i wybór komponentów i cech zastanego systemu, które zostały uwzględnione w architekturze zmodernizowanego systemu.

4.3.1.1 Wzorzec interoperacyjności

Wzorzec interoperacyjności PCSS został zaprojektowany celem usprawnienia procesu inwentaryzacji złożonych systemów z punktu widzenia rozwoju całego środowiska aplikacyjnego organizacji. Ułatwia on określenie bieżącego poziomu dojrzałości analizowanego systemu, jak również wspomaga wyznaczenie poziomu docelowego, czyli poziomu, do którego w krótkiej perspektywie organizacja zamierza się rozwijać.

Wzorzec interoperacyjności wpisuje się w model OSIMM (ang. *Open Group Service Integration Maturity Model*) [OSI11], określający poziom dojrzałości instytucji pod względem zaawansowania i przystosowania do ogólnych trendów rozwoju systemów teleinformatycznych opartych o paradygmat architektury zorientowanej na usługi. OSIMM jest modelem przedstawiającym poziomy dojrzałości instytucji z organizacyjnego punktu widzenia, natomiast wzorzec interoperacyjności uzupełnia ten model z perspektywy pojedynczych aplikacji i systemów teleinformatycznych.

Analiza interoperacyjności przy użyciu wzorca pozwala przedstawić charakterystykę techniczną systemów teleinformatycznych w trzech obszarach:

- aplikacji - zawiera kryteria, które dotyczą systemu jako całości i traktują go jako tzw. „czarną skrzynkę".
- architektury - wnika w wewnętrzną strukturę systemu, analizuje zaprojektowane i zaimplementowane mechanizmy wpływające na komunikację i zgodność systemu z paradygmatami architektury zorientowanej na usługi.
- informacji i danych - skupia się na ujęciu danych i informacji w systemie, co ma główny wpływ na semantyczną interoperacyjności systemów.

Wzorzec, przedstawiony na rysunku 4.1, ma postać struktury hierarchicznej zawierającej dla każdej płaszczyzny szereg pytań i wskazówek pomagających scharakteryzować interoperacyjność analizowanego systemu.

W każdej płaszczyźnie zdefiniowano serię kryteriów, wpływających na interoperacyjność systemu.

Płaszczyzna aplikacji:

- Metodyka utrzymywania systemu i zarządzanie zmianami.
- Charakter komunikacji z innymi systemami.
- Dostępność systemu.

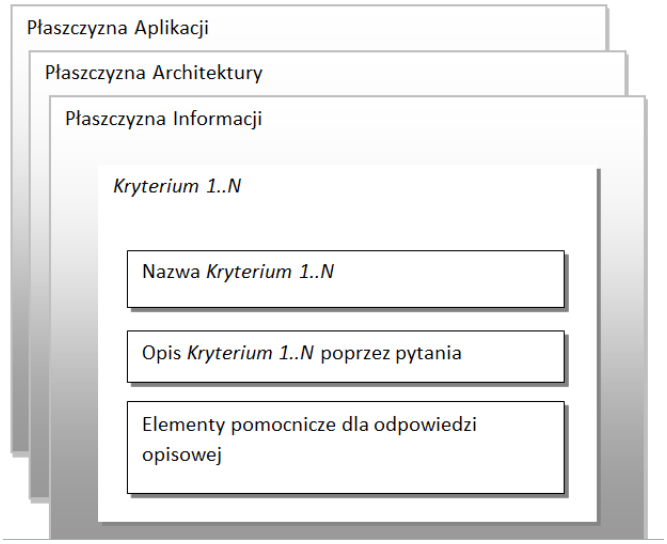
Płaszczyzna architektury:

- Typ architektury.
- Warstwy logiczne w architekturze systemu.
- Podział funkcji systemu (podsystemy/komponenty/ usługi).
- Interfejsy i elementy integracyjne architektury.
- Mechanizmy bezpieczeństwa.

Płaszczyzna Informacji:

- Metodyka opracowania modelu danych.

— Opis i reprezentacja modelu danych.
— Podstawa opracowania modelu.
— Charakter danych przetwarzanych w systemie.
— Warstwa danych - sposób implementacji.
— Formaty wymiany danych.
— Utrzymanie danych.



Rysunek 4.1: Struktura wzorca interoperacyjności

Dla każdego z kryterium zdefiniowano pytania naprowadzające dotyczące systemu oraz elementy pomocnicze obejmujące podpowiedzi i przykłady rozwiązań, technologii, metodyk, które mogły zostać zastosowane i które należałoby uwzględnić w opisie systemu w karcie interoperacyjności. Przykład fragmentu opisu kryterium „Charakter komunikacji z innymi systemami" znajduje się w tabeli 4.1.

Tabela 4.1. Przykładowe kryterium wzorca interoperacyjności

CHARAKTER KOMUNIKACJI Z INNYMI SYSTEMAMI	
Pytania:	• Czy system komunikuje się z innymi systemami? • Jaka jest architektura/styl komunikacji (uwzględniając systemy wewnętrzne i zewnętrzne)? • Jakie są zidentyfikowane problemy lub ograniczenia w wymianie danych z tymi systemami?

CHARAKTER KOMUNIKACJI Z INNYMI SYSTEMAMI	
	• ...
Elementy pomocnicze:	• (opis, z jakimi systemami komunikuje się system i kierunek komunikacji per system). • System A jest częścią większego systemu B, w którym pełni rolę X. • Dane są pozyskiwane/synchronizowane na żądanie / okresowo (np. co tydzień), ręcznie / automatycznie. • ...

4.3.2 Definicja wymagań

Wymagania dotyczące nowego, zintegrowanego systemu wspierającego zarządzanie oświatą były zbierane wieloetapowo z udziałem kilku grup użytkowników. Zarówno wymagania MEN jak i wymagania użytkowników z instytucji lokalnych zostały zgromadzone w rezultacie serii spotkań ze specjalistami i użytkownikami obecnie używanych systemów. Spotkania te prowadzone były w formie moderowanych dyskusji, gdzie moderator sterował rozmową aby uzyskać następujące informacje:

- Jakie (oraz w jaki sposób i z jaką częstotliwością) systemy informatyczne są wykorzystywane?
- Z jakich danych (i w jaki sposób pozyskanych) korzystają te systemy?
- Jakich wymagań aktualnie używane systemy nie spełniają?
- Czy są jakieś obszary lub działania, które nie są wsparte narzędziami informatycznymi?
- Czy są jakieś działania, które muszą być wielokrotnie wykonywane w różnych systemach?

Informacje pozyskane podczas spotkań z przedstawicielami MEN stanowiły podstawę dalszych prac analitycznych prowadzonych w projekcie. Eksperci współpracujący z projektem (de facto użytkownicy systemów informatycznych z PO, JST i PPP) oprócz odpowiedzi na powyższe pytania, zostali poproszeni o opracowanie pożądanych, nowych funkcjonalności w formie ustandaryzowanych opisów przygotowanych w oparciu o materiały opisujące elementy zwinnych metodyk wytwarzania oprogramowania. Opisy te zawierały: słowny opis funkcjonalności, oczekiwaną wartość biznesową, nazwę użytkownika oraz kryteria akceptacji. Chociaż wartość biznesowa stanowi najważniejszy element historii użytkownika, w opisanym studium przypadku kryteria akceptacyjne również odgrywały kluczową rolę. Kładąc duży nacisk na ten element, uzyska-

no zbiór dokładnych kryteriów, które w fazie implementacyjnej stanowiły podstawę do tworzenia przypadków testowych, jak i testów akceptacyjnych dla interfejsu użytkownika.

Wymagania od specjalistów z wszystkich ośmiu OKE i z CKE zostały zebrane w formie ankiet, w których użytkownicy odpowiadali na następujące pytania:

- Jaka jest funkcjonalność aktualnie używanych systemów?
- Jakie są słabe i mocne strony aktualnie używanych systemów?
- Jakie specyficzne rozwiązania organizacyjne są zastosowane w danej instytucji?
- Jakie wolumeny danych są przechowywane i przetwarzane?

Sumarycznie w fazie definicji wymagań zebrano 171 wymagań funkcjonalnych i 11 niefunkcjonalnych. Tabela 4.2 przedstawia liczbę zebranych wymagań funkcjonalnych w zależności od sposobu komunikacji z użytkownikami.

Tabela 4.2. Liczba zebranych wymagań funkcjonalnych w zależności od sposobu komunikacji z użytkownikami

Sposób komunikacji	Liczba zebranych wymagań
Ankiety	87
Ustrukturyzowane opisy	53
Bezpośrednie spotkania	31

Zebrane wymagania funkcjonalne i niefunkcjonalne posłużyły do analizy możliwych przepływów danych pomiędzy systemami i do opracowania koncepcji nowego, zintegrowanego systemu.

4.3.3 Prototypowanie i ewaluacja dowodu wykonywalności

Zebrane wymagania oraz informacje posłużyły do budowy dowodu wykonywalności systemu, który następnie został pokazany współpracującym z projektem ekspertom, w celu zebrania ich opinii. Prototyp przygotowany był dwuetapowo. W pierwszym etapie opracowano zbiór statycznych ekranów użytkownika prezentujących widoki modułów zintegrowanego systemu odpowiedzialnych za wsparcie procesów planowania, przeprowadzenia i oceny sprawdzianu szóstoklasisty oraz obsługi egzaminatorów - przypisywania do określonych sesji egzaminacyjnych. Zbiór widoków ekranów, dzięki częstemu prezentowaniu ich ekspertom i regularnemu zbieraniu ich uwag i komentarzy, podlegał ciągłej ewolucji. Drugi etap prototypowania polegał na opracowaniu dowodu wykony-

walności systemu. Dowód ten zakładał pobranie danych testowych z zewnętrznych systemów oraz uzupełnienie ich (w scenariuszach planowania, przeprowadzenia i oceny egzaminu szóstoklasisty) przez ekspertów współpracujących z projektem.

Eksperci reprezentujący docelowych użytkowników zintegrowanego systemu wzięli udział w serii indywidualnych spotkań z projektantami, podczas których proszeni byli o interakcję z zaprezentowanym prototypem i przekazywanie uwag. Interakcja odbywała się według ustalonych scenariuszy. Scenariusze te zostały przygotowane na podstawie wcześniejszych doświadczeń zdobytych podczas realizacji złożonych systemów informatycznych. Zakładały one realizację dwóch sesji testowych odbywających się w kilkutygodniowych odstępach czasowych. Każde ze spotkań było rejestrowane – zebrano materiał prezentujący mimikę i komentarze głosowe użytkownika oraz widok ekranu komputera testowego. Zebrane podczas tych testów materiały - reakcje i działania użytkownika, jego komentarze, czas odszukania odpowiednich pól i przycisków, poruszanie się po formularzach - posłużyły do analizy użyteczności zaprezentowanego prototypu. Sesje z ekspertami podzielono na dwie części: wolne nawigowanie i część zadaniową. Podczas części zadaniowej użytkownicy realizowali elementy głównego scenariusza, skupiając się na zgodności każdego widoku z rozwiązaniem projektowanym w pierwszej fazie prototypowania, poprawnym funkcjonowaniu, oraz ich ergonomii i czytelności. Pytania pomocnicze zadawane przez osobę prowadzącą motywowały eksperta do dodatkowych działań i przemyśleń. Wyniki przeprowadzonych sesji stanowiły materiał pomocniczy do weryfikacji i modyfikacji wymagań docelowego systemu. Zostały one zgrupowane i zarejestrowane w katalogu cech prototypu, a ich realizacja włączona w kolejne iteracje prac programistycznych.

4.4 Podsumowanie

Wybór właściwiej metodyki dla definicji, analizy i oceny wymagań nie jest sprawą trywialną. Doświadczenia zebrane w trakcie pracy nad opisanym projektem wskazują, iż kluczowe jest zaangażowanie użytkowników końcowych we wszystkich etapach prowadzonego projektu. Typowym błędem jest koncentracja aktywności użytkowników tylko podczas fazy zbierania wymagań i brak wczesnej oceny cząstkowych wyników projektu. Jest to widoczne dla projektów prowadzonych według metodyki kaskadowej. Zastosowane w pierwszym etapie projektu "Modernizacja systemów informatycznych do obsługi egzaminów zewnętrznych i nadzoru pedagogicznego" podejście zwinne zakłada ścisłą interakcję z użytkownikami i ciągłą ewaluację przyrostów oprogramowania.

Przy projektach dużej skali, gdzie występuje wielu beneficjentów systemu posiadających często rozbieżne interesy, właściwie prowadzona inżynieria wymagań jest jednym z kluczowych elementów decydujących o sukcesie całego przedsięwzięcia. Inżynieria wymagań może pochłonąć nawet 40% zasobów projektu [COM14]. Metody zbierania wymagań powinny zatem być dostosowane do możliwości budżetowych projektu oraz do liczby użytkowników mogących te wymagania dostarczyć. Bezpośrednie metody zbierania wymagań (wywiady z użytkownikami, spotkania, prototypowanie) zwiększają prawdopodobieństwo pozyskania wysokiej jakości wymagań. Metody te są czasochłonne oraz wymagają odpowiednich zdolności interpersonalnych ze strony inżyniera wymagań. Metody niewymagające bezpośredniego kontaktu z użytkownikiem (np. ankiety) są mniej czasochłonne, niosą jednak ze sobą ryzyko nieporozumienia i w konsekwencji obniżenia jakości zebranych wymagań.

W opisanym projekcie proces zbierania i analizy wymagań był prowadzony wieloetapowo ze względu na skalę docelowego rozwiązania oraz różnorodność jego beneficjentów. Analiza zebranych wymagań doprowadziła do opracowania koncepcji zintegrowanego systemu. Ponadto kluczową decyzją było przygotowanie wczesnego prototypu aplikacji w postaci szkiców interfejsu użytkownika. Pozwoliło to na pierwszą weryfikację poczynionych na etapie analizy wymagań założeń a tym samym na sprawne zbudowane dowodu wykonywalności ilustrującego działanie wybranych scenariuszy użytkowych systemu.

Bibliografia

[COH04] Cohn, M. User Stories Applied: For Agile Software Development. Addison-Wesley: Crawfordsville, 2004

[COM14] http://www.computerworld.pl/artykuly/395371/Inzynieria.wymagan.wychodzi.z.ukrycia.html

[MOD13] Modernizacja systemów informatycznych do obsługi systemu egzaminów zewnętrznych i nadzoru pedagogicznego - I etap http://efs.men.gov.pl/images/Broszura/Modernizacja%20systemw%20informatycznych%20do%20obsugi%20systemu%20egzaminw%20zewntrznych_www.pdf

[OSI11] Model OSIMM, http://www.opengroup.org/soa/source-book/osimmv2/

[SIO11] Ustawa z dnia 15.04.2011 r. o systemie informacji oświatowej, Dz. U. z 2011 r. Nr 139, poz. 814

CZĘŚĆ II
PROJEKTOWANIE OPROGRAMOWANIA

Rozdział 5

Osobowość a predyspozycje zawodowe przyszłych projektantów oprogramowania

Rozdział przedstawia relację z pierwszego etapu badań prowadzonych w ramach interdyscyplinarnego projektu InfoPsycho, który został zainicjowany w roku 2013 na Politechnice Koszalińskiej i na Uniwersytecie Gdańskim. Celem badania było wskazanie zmiennych psychologicznych, które cechują odnoszących sukcesy akademickie kandydatów na stanowiska projektantów oprogramowania. Jako kryterium umiejętności zawodowych kandydatów wybrano syntetyczne wskaźniki jakości wykonania przez nich zadań projektowych oraz ćwiczeniowych. Do pomiaru cech osobowości zastosowano kwestionariusz NEO-FFI oparty na pięcioczynnikowym modelu Costy i McCrae. Wstępne wyniki badań wskazują, że rokujący młodzi projektanci cechują się wyższą neurotycznością i introwersją w porównaniu z tymi, kto osiąga niższe wyniki na skali jakości dokumentacji projektowej oraz cechują się wyższą sumiennością, gdy brana jest pod uwagę ocena z wykonania programistycznych zadań ćwiczeniowych.

Jednym z najważniejszych wyzwań stojących przed systemem edukacji w Polsce jest dostosowanie oferty edukacyjnej wyższych uczelni do potrzeb rynku. Projektantów oprogramowania (developerów) stale przybywa, zaś popyt zdaje się nie słabnąć, a wręcz niezmiennie i systematycznie rośnie. Stąd tak ważnym zadaniem zdaje się być zbadanie, jakie psychologiczne korelaty predyspozycji zawodowych występują w tej grupie zawodowej, które z nich są dobrymi wskaźnikami przyszłej efektywności pracy jednostki oraz czy możemy na ich podstawie przewidywać sukces komercyjny oprogramowania, które zostanie stworzone przez danego projektanta.

Pytania te stawiane są od stosunkowo krótkiego czasu w literaturze przedmiotu. Pomimo coraz częstszego podejmowania badań nad czynnikami psychologicznymi w dziedzinie informatyki na arenie międzynarodowej, nie dostrzega się tego trendu w polskiej literaturze, co wydaje się o tyle zaskakujące, że specjaliści z dziedziny IT z obszaru Europy środkowowschodniej są oceniani jako jedni z najbardziej efektywnych programistów na świecie.

Z perspektywy praktyki ważne jest, by projektant oprogramowania (wiodący specjalista w zespole programistycznym) posiadał wysokie kompetencje komunikacyjne oraz szczególne predyspozycje osobowościowe, które wspierałyby proces tworzenia produktu wspólnie z klientem. Można by rzec, że projektant oprogramowania powinien łączyć podejście humanistyczno-społeczne z wysokimi kwalifikacjami technicznymi, by efektywnie wykonywać swoją pracę. To założenie wynikające z praktycznej refleksji skłoniło nas do połączenia sił badaczy z zakresu nauk społecznych i informatycznych, by interdyscyplinarnie

ocenić relacje między zmiennymi psychospołecznymi a zawodowymi predyspozycjami programistów.

Jakość oprogramowania znajduje się pod istotnym wpływem intuicji i doświadczenia projektanta. Używając sformułowań opublikowanej typologii projektów informatycznych [Wry10] możemy stwierdzić, że w najwyższym stopniu wpływu kwalifikacji projektanta będą doświadczać przedsięwzięcia analityczne i projektowe (zgodnie z kryterium fazy cyklu życia) oraz przedsięwzięcia pierwotne i rozwojowe (zgodnie z kryterium charakteru wprowadzanych zmian).

Praca nad projektami programistycznymi to bardzo specyficzne zajęcie, ponieważ projektant powinien być jednocześnie wysoce kompetentnym w zakresie metodologii i technologii informatycznych oraz dobrze rozumieć dziedzinę problemu, którą się zajmuje. W praktyce albo stosuje się twarde metodyki, gdzie najpierw powstaje projekt techniczny i cała odpowiedzialność zostaje skupiona na projektantach, albo programiści tworzą produkt ściśle współpracując z klientem, zdobywając niezbędną wiedzę o dziedzinie problemu, poszukując kompromisu między rozwiązaniem idealnym technicznie a rozwiązaniem idealnie przyjaznym użytkownikowi. Najbardziej dramatycznie projekt rozwija się w małych zespołach ze znacznie ograniczonym budżetem i brakiem doradców-ekspertów, gdzie niestety sytuacja wymusza często na programiście by sam sobie był architektem systemu i specjalistą dziedziny. Podobne problemy występują w zespołach większych, ale rozproszonych i słabo skoordynowanych, jakie często istnieją przy projektach typu Open Source.

Aby oprogramowanie odniosło sukces komercyjny powinno ono być produktem o wysokiej jakości. Bynajmniej nie chodzi tutaj o rozumienie jakości w sensie stopnia doskonałości, jak definiował to Platon. Owszem, dobre oprogramowanie powinno być niezawodne, stabilne, wydajne, bezpieczne, ergonomiczne, przenośne. Ale oprogramowanie wyróżnia się swoją jakością, gdy jest zgodne z wymaganiami sformułowanymi w ramach projektu, a jednocześnie spełnia wszystkie wymagania odbiorcy, zaspokaja potrzeby i oczekiwania użytkownika. Tak więc, akceptacja oprogramowania przez odbiorcę i użytkownika, która następuje na etapie wdrożenia gotowego produktu, stanowi ocenę jakości produktu informatycznego. Zmusza to projektantów do kojarzenia nie tylko własnych wewnętrznych norm i ustaleń, ale również nastawień, potrzeb, oczekiwań szeroko rozumianego klienta.

Analiza specyfiki oprogramowania jako produktu rynkowego doprowadza do wniosku o konieczności celowego przygotowania, specjalizacji informatyków do pełnienia funkcji projektanta. Projektant potrzebuje dobrych „interfej-

sów" socjalnych do komunikacji i współpracy z zespołem i z szerokim gronem interesariuszy, a także zdolności uczenia się nowych dziedzin, zdolności przełączania swojego pola widzenia na perspektywę klienta i przyszłego użytkownika oprogramowania, potrzebuje tego, co potocznie określamy podejściem humanistycznym przy zachowaniu znakomitego przygotowania technicznego. Niestety, tych aspektów przygotowania zawodowego projektantów oprogramowania brakuje w planach studiów na uczelniach wyższych technicznych. A o niektórych aspektach można sądzić, że nie da się ich skutecznie wykształcić u wszystkich studiujących kierunek informatyka, można by jedynie spróbować dobierać studentów zgodnie z posiadanymi przez nich predyspozycjami psychospołecznymi, a zatem pracować z nimi w ramach specjalizacji przygotowującej do pracy na stanowisku projektanta oprogramowania.

5.1 Przegląd literatury

Badania nad osobowością w populacji programistów korzystały z różnych modeli teoretycznych i psychometrycznych z różnym efektem. W poniższych podrozdziałach staramy się przybliżyć krótki opis instrumentów psychometrycznych oraz modeli, a następnie podsumować dotychczasowe wyniki badań nad osobowością programistów z wykorzystaniem instrumentów do badania osobowości.

5.1.1 Modele osobowości i ich operacjonalizacja w badaniach nad cechami osobowości programistów

W ostatnim czasie wielokrotnie były podejmowane próby badania oraz oceniania cech indywidualnych z zakresu zmiennych psychospołecznych, w tym osobowości, pośród populacji programistów [KFA14, FTAS08, HAES10]. Techniki stosowane w tych badaniach, będąc skutkiem przyjęcia konkretnego modelu teoretycznego opisującego osobowość, są bardzo zróżnicowane [Cap03, SMG14]. Istnieje więc potrzeba pogłębiania i replikowania stosowanych badań dla populacji programistów.

Najczęściej stosowaną metodą pomiaru cech osobowości w badaniach programistów dotychczas była MBTI (ang. Myers-Briggs Type Indicator) [GL04, CA06, CG07]. Jest ona oparta na teorii Junga opisującej typy psychologiczne. Kwestionariusz MBTI dotyczy czterech czynników dwubiegunowych lub wymiarów: *ekstrawersja-introwersja*, *poznanie-intuicja*, *myślenie-odczuwanie* i *osądzanie-obserwacja*. Bardzo popularnym kwestionariuszem w badaniu programistów był również KTS (ang. Keirsey Temperament Sorter) [SSAD06].

Z kolei w bardziej aktualnych badaniach zaczęto stosować pięcioczynnikowy model osobowości zwany w skrócie Big-Five [HAES10, SMG14].

Psychologowie wiedzą, że kwestionariusz do pomiaru cech osobowości powinien być dobrany do badania na podstawie licznych czynników, o których mówi metodologia badań psychologicznych, ale najważniejszymi parametrami dobrego kwestionariusza jako narzędzia badawczego powinny być jego rzetelność i trafność. Kwestionariusz MBTI, oparty na psychodynamicznej koncepcji osobowości Junga, pomimo jego popularności, jest krytykowany właśnie ze względu na wspomniane powyżej parametry [BM95]. Stąd należy przyjąć, iż powinno się eksplorować inny model teoretyczny, a co się z tym wiąże – inne techniki badawcze [Cho04, CDI08, SMG11].

Ze względu na wyżej wymienione ograniczenia psychometryczne MBTI, zdecydowaliśmy się na zastosowanie *Pięcioczynnikowego modelu osobowości* (PMO) autorstwa Paula Costy i Roberta McCrae. Model ten jest ostatnio często stosowany i wielokrotnie testowany w badaniach akademickich i klinicznych [BM91]. Różnicuje jednostki w stopniu znacznym, przez co jest powszechnie uważany za model dostarczający najbardziej użyteczną skale osobowości [BM91, SSAD09, LA05]. Ta skala jest hierarchicznie zorganizowanym inwentarzem osobowości dla pięciu cech, a mianowicie: *ekstrawersji*, *ugodowości*, *sumienności*, *otwartości* i *neurotyczności*. Oto krótki opis tych cech:

1. Ekstrawersja – wskazuje stopień, w jakim jednostka jest towarzyska, asertywna, aktywna w rozmowie. Osoba ekstrawertyczna dobrze czuje się w relacjach społecznych, postrzegając je jako przyjemność wynikającą z aktywności własnej.
2. Ugodowość – odnosi się do takich cech jednostki jak: życzliwość, zaufanie i ciepło. Osoba, która jest mało ugodowa jest samolubna, wroga i pełna wątpliwości.
3. Sumienność – dotyczy czyjejś orientacji na osiągnięcie. Osoby, które uzyskują wysokie wyniki, są pracowite, niezawodne, zorganizowane na tyle, by wypełniać powierzone im zdania na czas. Z drugiej strony, niski wynik dla sumienności charakteryzuje osoby impulsywne, nieuporządkowane i nieodpowiedzialne.
4. Otwartość – opisuje gotowość jednostki do szerokiego spektrum aktywności intelektualnych i kulturalnych. Jednostka o wysokiej otwartości, ma szerokie horyzonty, jest gotowa do podejmowania ryzyka celem stymulacji. Na przeciwnym krańcu znajduje się osoba o niskiej otwartości na doświadczenie, wykazująca małą wrażliwość estetyczną, kulturową, preferująca zachowania rytualistyczne, co z kolei sprzyja utrwalaniu konserwatywnych wartości.

5. Neurotyczność – zamiennie opisywana jako *stabilność emocjonalna* [MJ92], jest często skorelowana z poczuciem własnej skuteczności [Sch08]. Osoby z niskim wynikiem są pewne siebie, czują się bezpieczne, względnie stałe w nastrojach, spokojne. Wysoka neurotyczność jest wskaźnikiem częstej zmienności nastrojów, nerwowości, niepokoju oraz niepewności [DSGO06].

Jak każdy model teoretyczny, PMO ma swoje ograniczenia i spotkał się z opiniami krytycznymi, głównie odnoszącymi się do ilości czynników opisujących osobowość. Tym niemniej PMO jest dobrze oceniany w przestrzeni akademickiej i praktycznej, przez co jest standardowo wybieranym modelem osobowości do licznych badań [KFA14]. Do badania PMO istnieje wiele różnych narzędzi, np. NEO-PI-R, NEO-FFI, IPIP. NEO-FFI jest instrumentem, który wybraliśmy, gdyż jest dokładnie przetestowanym pod względem rzetelności i trafności, przez co powszechnie stosowanym kwestionariuszem do pomiaru osobowości [Con06, MRMH+96].

Podsumowując, wnikliwa analiza literatury pozwoliła autorom na podjęcie decyzji o przyjęciu PMO jako modelu teoretycznego, zoperacjonalizowanego w postaci NEO-FFI. Kolejna sekcja przeglądu opisuje opublikowane badania, które były ukierunkowane na znalezienie związków i relacji między osobowością programisty a osiąganiem przez niego wysokich wskaźników informatycznych.

5.1.2 Badania nad osobowością programistów i osiągnięciami informatycznymi

Coraz to większym zainteresowaniem wśród badaczy cieszy się tematyka relacji cech psychospołecznych i osobowościowych programisty z wskaźnikami jakości jego pracy. Główne obszary prowadzonych dotąd badań dotyczyły zagadnień programowania w parach (PwP), efektywności zespołów programistycznych. Wyniki tych badań nie dostarczają niestety jeszcze jednoznacznych odpowiedzi i kierunków relacji osobowości z jakością wykonywanych zadań programistycznych [BM95].

W poniżej przywołanych trzech badaniach testowano związek osobowości programisty z efektywnością programowania w parach. Uzyskane wyniki nie zawsze potwierdzały postawione hipotezy. Zaledwie jedno badanie w pełni wykazało, że w procesie programowania w parach odpowiednie dopasowanie osobowości wiąże się z efektywnością [CDI08]. W tej publikacji Choi stwierdza, że jeśli dwie osoby, bez wcześniej nabytego doświadczenia w programowaniu, są podobne pod względem głównych cech osobowości bądź uzupełniają

się w modelu MBTI, to ich poziom wydajności (w kontekście jakości tworzonego oprogramowania) będzie znacznie wyższy niż w przypadku innych kombinacji. Z kolei w badaniach Stalleh [SMG14] badacze sugerują, że różne poziomy sumienności (rozumianej jako cecha osobowości) nie wpływają na sukces naukowy sparowanych studentów-informatyków w toku wspólnego programowania. Badania prowadzone przez Sfetsos et al. [SSAD06], w których stosowano KTS, sugerują, że pary składające się z niejednorodnych osobowości lepiej wypadają pod względem parametrów programowania niż pary o tym samym typie osobowości. Do tej pory większość badań dostarcza niespójne wyniki lub takie, które nie potwierdzają istnienia istotnych różnic pod względem osobowości w PwP.

Równie często poruszanymi w kontekście osobowości programistów tematami są zarządzanie zespołem programistycznym (ZP) i wydajność wspólnej pracy członków ZP. Jednym z ciekawszych badań w tym zakresie jest badanie przeprowadzone przez Acuna i współpracowników, w którym podjęto próbę określenia wpływu osobowości na wyniki ZP. Badania wykazały, że zespoły mogą pracować w sposób zadowalający mimo znacznych różnic etnicznych, religijnych oraz osobowościowych między poszczególnymi członkami [AGJ09]. Z drugiej strony mamy wyniki, które raportują istotny związek czynników osobowościowych z zadowoleniem z zespołowej pracy programistów [KC06]. Natomiast Chao i Atli [CA06] badali osobowość 60 respondentów, zaś dane przez nich uzyskane, nie pozwoliły na potwierdzenie tezy, że istnieje różnica w jakości kodu między różnymi sposobami kojarzenia typów osobowości w zadaniach programowania parami.

Podczas przeglądu literatury natrafiliśmy na trudność w odnalezieniu precyzyjnych danych dotyczących osobowości programistów w kontekście pracy indywidualnej. Nieliczne raporty z badań przedstawiają niespójne wyniki. Tak Capretz L. F. [Cap03] wskazuje, że wśród programistów występuje – według klasyfikacji Junga w modelu MTBI – 57% introwertyków, zaś 43% ekstrawertyków, 81% myślących vs 19% odczuwających, 58% osądzających vs 42% obserwujących. Najczęściej prezentowanym zespołem cech osobowości jest *introwertyk-percepcjonista-myśliciel-sędzia* (ISTJ) – 24% dla całej próby badanych programistów, gdzie w badaniach na populacji ogólnej nie informatyków ISTJ osiąga zaledwie 11.6%. Osoby o tym typie są „systematyczne, solidne, praktyczne, realistycznie podchodzące do rzeczywistości, dotrzymujące obietnic, szanujące obowiązki, ceniące fakty, dobrze zorganizowane, ofiarne" [Gra13].

Pytanie jednak brzmi: czy taki zespół cech jest korzystny dla jakości tworzonego oprogramowania? Cunha i Greathead [CG07] starali się odnaleźć odpowiedź na to pytanie, badając uczniów pod względem nasilenia cech w modelu MBTI. Okazało się, że osoby o wyższych wynikach dla wymiaru *intuicja* uzyskały znacznie lepsze wyniki w zadaniu wymagającym przeglądu 282 linii kodu Java. Z kolei Acun z współpracownikami wskazuje, że to *ekstrawersja* jest istotnie związana z lepszą jakością oprogramowania przy tworzeniu oprogramowania w metodologii Agile [LP07].

5.2 Relacja z przebiegu projektu InfoPsycho

5.2.1 Koncepcja i hipoteza projektu

Celem prezentowanego etapu badań było sprawdzenie, czy i jakie predyspozycje psychologiczne (wymiary osobowości) młodych projektantów oprogramowania, rejestrowane na wczesnym etapie ich edukacji zawodowej, wiążą się z umiejętnością opracowania dobrego, przyjaznego dla użytkownika i zgodnego z wymaganiami klienta oprogramowania. Ponieważ literatura przedmiotu nie pozwala na postawienie konkretnych hipotez badawczych, niniejsze badanie ma charakter eksploracyjny (pilotażowy). Zadaliśmy następujące pytanie badawcze: "Jakie cechy osobowości charakteryzują młodych projektantów, odnoszących sukcesy w dziedzinie oprogramowania na szczeblu akademickim?" Do tego pytania postawiliśmy dwie ogólne hipotezy:

1. Istnieje związek między wybranymi wymiarami osobowości projektanta a jakością tworzonego przy jego udziale oprogramowania.
2. Możliwe jest zdiagnozowanie istotnych dla jakości oprogramowania cech osobowości projektanta, przed rozpoczęciem przez niego pracy zawodowej.

Aby zweryfikować powyższe hipotezy, przebadano dwie różne wiekowo grupy studentów-informatyków, uwzględniając inne wskaźniki efektywności pracy studentów nad oprogramowaniem w trybie ćwiczeniowym i trybie projektowym. Aby zapewnić badanym anonimowość, zastosowano kody, które pozwoliły połączyć wyniki poszczególnych wskaźników bez imiennej identyfikacji poszczególnych uczestników.

5.2.2 Grupy badanych

Zbiór informatyków badanych dotychczas w ramach projektu InfoPsycho składał się ze 128 mężczyzn i 12 kobiet będących studentami studiów pierwszego i drugiego stopnia kierunku Informatyka w Wydziale Elektroniki i Informatyki Politechniki Koszalińskiej.

Struktura badań pilotażowych przeprowadzonych w okresie od listopada 2013 do lutego 2014 przewidywała kompletację osób badanych w dwie grupy. Pierwsza grupa składała się z 65 osób (jedną osobę wykluczono ze względu na braki danych), z czego mężczyźni stanowili 89,4%. Byli to studenci drugiego roku, intensywnie zajmujący się programowaniem komputerów i technologiami informatycznymi, ale jeszcze niezaangażowani w prace projektowe. Średnia wieku dla tej grupy wyniosła 22 lata.

Druga grupa składała się z 73 osób (jedną osobę wykluczono ze względu na braki danych), z czego mężczyźni stanowili 93,2%. Łączyła studentów starszych roczników pierwszego stopnia i studentów z drugiego stopnia studiów informatycznych, wszyscy oni byli wykonawcami projektów oprogramowania i pełnili role projektantów, analityków i kierowników projektów oprogramowania w trakcie przeprowadzanych pomiarów psychologicznych. Średnia wieku w tej grupie wyniosła 24 lata.

5.2.3 Techniki badawcze

Jako narzędzie psychologiczne zastosowano kwestionariusz NEO-FFI Costy i McCrae w polskiej adaptacji Zawadzkiego, Strelaua, Szczepaniaka i Śliwińskiej [28], mierzący pięć wymiarów osobowości w koncepcji Costy i McCrae: neurotyczność, ekstrawersję, otwartość na doświadczenie, ugodowość i sumienność. Narzędzie zawiera 60 pozycji – po 12 dla każdej z 5 podskal. Badany udzielał odpowiedzi na pięciostopniowej skali Likerta: od „zdecydowanie się nie zgadzam” do „zdecydowanie się zgadzam”.

Wymienione powyżej wymiary osobowości były mierzone w formie samoopisu. Wyniki przedstawiają pełny opis osobowości badanego i antycypowanie jego możliwości adaptacyjnych do środowiska zawodowego [ZSSŚ13]. Czas wypełniania kwestionariusza wynosił ok. 15 minut.

Wskaźniki informatyczne, charakteryzujące jakość prac wykonywanych przez studentów, zostały sprowadzone do ocen trójpoziomowych w skali 1-2-3 od poziomu najgorszego do najlepszego. Przy wystawianiu ocen za ćwiczenia programistyczne i za prace projektowe oraz za poziom komunikacji w zespole projektowym użyto podejścia eksperckiego, ocenę aktywności dokonano na podstawie rejestracji wszystkich zdarzeń związanych z konsultowaniem i dostarczaniem rozwiązań. Jakość wykonania prac projektowych oceniana była za pomocą listy kontrolnej. Przedstawiona poniżej lista zawiera wymagania krytyczne (pozycje 1-2), które musiały zostać spełnione przez projektanta, natomiast stopień realizacji pozostałych wymagań (pozycje 4-6) decydował o przypisaniu konkretnej oceny:

1. Specyfikacja dokumentów do opracowania (SRS, model przypadków użycia, model domeny problemowej, konceptualny model danych, dokument detaliczny projektu itd.);
2. Poprawność składniowa diagramów w sensie zgodności z UML 2.0;
3. Poprawność konceptualna, spójność funkcjonalna projektowanego systemu;
4. Efektywność i oryginalność stworzonego rozwiązania;
5. Czytalność dokumentacji projektowej (przyjazny odbiorcy sposób opisu i realizacji diagramów, obecność glosariusza);
6. Rytmiczność wykonania zadań projektowych, zgodność faktycznych postępów z harmonogramem projektu.

Natomiast przy ocenianiu ćwiczeń programistycznych brano pod uwagę następujące aspekty:

1. Poprawność zrealizowania przydzielonego zadania (rozwiązania postawionego problemu w postaci działającego programu);
2. Stopień rozumienia i umiejętność analizy wykorzystanego kodu;
3. Zastosowanie dobrych praktyk programowania (paradygmaty obiektowe, czytelność kodu, obsługa błędów);
4. Efektywność rozwiązania.

Oceny aktywności studentów dokonano na podstawie rejestracji wszystkich zdarzeń związanych z konsultowaniem i dostarczaniem rozwiązań. Dla każdej grupy została ustalona aktywność minimalna (proporcjonalnie do ilości wydawanych zadań), a następnie na zasadzie rankingu studentom przypisano wartości numeryczne (oceny aktywności) z zakresu 1-3, gdzie 1 to faktyczna aktywność poniżej wyznaczonego minimum, 2 to aktywność od poziomu minimum do 50% od maksymalnej w danej grupie, a 3 to aktywność powyżej 50%.

Opis specyfiki prac informatycznych oraz pojęcia jakości projektu oprogramowania zostały przedstawione w części teoretycznej niniejszego rozdziale.

5.2.4 Wyniki

Wyniki analizy korelacji wskaźników psychologicznych oraz informatycznych przedstawiono w tabelach 1-2. Zastosowano dwa różne współczynniki korelacji ze względu na różny poziom pomiaru zmiennych (skala przedziałowa i porządkowa):

- *Tau b Kendalla* dla badania związku między jakością komunikacji, jakością projektu a zmiennymi osobowości (1), oceną, aktywnością a wymiarami osobowości (2).
- *Rho Spearmana* dla badania związku między jakością komunikacji i jakością projektu (3) oraz oceną i aktywnością (4).

Na potrzeby analiz przyjęliśmy alfa < 0.05. Wartości zmiennych opisowych do badanych zbiorów (średnią i odchylenie standardowe) również umieściliśmy w tabelach 5.1-5.2.

Tabela 5.1. Średnie, odchylenia standardowe i współczynniki korelacji dla oceny z ćwiczeń i aktywności studenta z podskalami NEO-FFI, N=65.

	M	SD	1	2	3	4	5	6	7
1. Ocena			1,00	0,70**	-0,05	0,38**	-0,16	0,10	0,01
2. Aktywność				1,00	-0,07	0,21*	0,01	0,00	0,10
3. Ekstrawersja	39,51	7,41			1,00	0,16*	-0,29**	0,23**	0,09
4. Sumienność	42,28	7,95				1,00	-0,18*	0,13	0,01
5. Neurotyczność	32,05	9,18					1,00	-0,20*	0,03
6. Otwartość na doświadczenie	38,57	6,87						1,00	-0,14
7. Ugodowość	38,68	4,52							1,00

* $p < 0,05$ ** $p < 0,01$

Tabela 5.2. Średnie, odchylenia standardowe i współczynniki korelacji pomiędzy oceną jakością projektu i komunikacji dot. studenta z podskalami NEO-FFI, N=73.

	M	SD	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1. Jakość projektu			1,00	0,60**	- 0,25**	0,08	0,21*	0,04	0,04
2. Jakość komunikacji				1,00	- 0,19*	0,12	0,11	0,03	0,27**
3. Ekstrawersja	39,70	6,78			1,00	0,21**	- 0,29**	0,05	0,12
4. Sumienność	44,21	8,05				1,00	- 0,25**	0,14*	0,22**
5. Neurotyczność	29,16	7,95					1,00	0,50	-0,13
6. Otwartość na doświadczenie	37,97	6,03						1,00	-0,00
7. Ugodowość	45,14	6,89							1,00

* $p < 0,05$ ** $p < 0,01$

W pierwszej grupie badanych zarówno ocena, jak i aktywność korelują pozytywnie z sumiennością, przy czym w pierwszym wypadku siła tego związku jest dużo wyższa. Obydwa wskaźniki korelują ze sobą bardzo wysoko (0,7, $p < 0,01$), co zgodne jest z kryterium ich oceniania – wspólna wariancja wynika z liczby poprawnych odpowiedzi udzielonych przez studenta, wchodzących w skład każdego z nich.

Uzyskane w drugiej grupie badanych rezultaty wskazują, że jakość projektu koreluje pozytywnie z neurotycznością (0,21, $p < 0,05$) oraz negatywnie z ekstrawersją (-0,25, $p < 0,01$). Jakość komunikacji natomiast koreluje pozytywnie z ugodowością (0,27, $p < 0,01$) oraz negatywnie z ekstrawersją. Jakość projektu

i jakość komunikacji korelują ze sobą pozytywnie (0,6, $p < 0,01$) – warto zauważyć, że siła tego związku jest bardzo duża.

W pozostałych przypadkach korelacje okazały się nieistotne statystycznie.

5.2.5 Dyskusja wyników

Rezultaty przeprowadzonych badań potwierdzają pierwszą hipotezę. Osiągnięcie wysokich wskaźników informatycznych koreluje z konkretnymi wymiarami osobowości. Pozytywną korelację uzyskaliśmy pomiędzy wskaźnikami jakości dokumentacji projektowej a poziomem neurotyczności i introwersji w ujęciu Costy i McCrae. Studenci o najwyższych wskaźnikach efektywności pracy w obszarze projektowania oprogramowania charakteryzują się mniejszą stabilnością emocjonalną (5 sten) niż studenci gorzej rokujący jako programiści (3 sten). Różnice stają się istotne przy porównaniach wewnątrzgrupowych, ale studenci lepiej rokujący nadal mieszczą się w normie w ich grupie wiekowej. Różnice wewnątrzgrupowe zaobserwowano również w obszarze ekstrawersji: studenci osiągający wyższe wskaźniki w ocenie jakości dokumentacji i poziomu komunikacji byli mniej ekstrawertywni (5 sten) niż studenci osiągający niższe rezultaty (6 sten).

Wiele dotychczasowych badań wykazało, że poziom neurotyczności jest skorelowany z przeżywaniem negatywnych emocji, niestabilnością emocjonalną [LP07]. Ale wyższy poziom neurotyczności świadczy również o tym, że jednostka potrafi myśleć w sposób niekonwencjonalny, posiadając niekiedy innowacyjne pomysły [RC03]. Wydaje się to zwłaszcza pożądane w przypadku przyszłych projektantów oprogramowania, u których znaczącą zaletą jest umiejętność odejścia od powszechnie obowiązujących schematów (w miarę potrzeb, nie ignorując jednak sprawdzonych metodyk i rozwiązań), nawet jeśli taki tok myślenia wydaje się początkowo niezrozumiały albo nielogiczny [TR04]. Wiele uwagi poświęciła neurotyczności Karen Horney [Hor11], która opisała model neurotycznego współzawodnictwa. Składowymi takiego współzawodnictwa są: wieczne porównywanie się z innymi, chęć bycia wyjątkowym i niepowtarzalnym, element wrogości. Stałe dążenie do bycia lepszym w pasjonujących dziedzinach wymaga często pracy w samotności, stronienia od innych ludzi. Charakteryzujące taką jednostkę przekonanie, że tylko nieliczni osiągną sukces w danej dziedzinie sprawia, że wszelka kolektywna praca staje się niewskazana czy nawet szkodliwa.

Występowanie wyższych wyników w obszarze neurotyczności i introwersji u przyszłych programistów jest spójne z wynikami uzyskanymi w podobnej konwencji przez Cegielskigo i Hall'a [CH06]. Może to oznaczać, że takie osoby

są bardziej nieśmiałe w kontaktach z innymi ludźmi. Relacje ludzkie w ocenie osób z wyższymi wskaźnikami introwersji są czasochłonne, dlatego więcej czasu poświęcają na pracę nie związaną z relacjami. Preferują samotność, mają mniejszą potrzebę poszukiwania doznań. Każda ich wypowiedź jest przez nich skrupulatnie analizowana [Lan02]. Należy zaznaczyć, że grupą docelową w naszych badaniach byli studenci, niewystarczająco uświadomieni w kwestii istotności kontaktów międzyludzkich. W programie studiów informatycznych nie realizuje się zajęć ukierunkowanych na rozwijanie umiejętności miękkich. Praca akademicka tych studentów kierunku technicznego podlega ocenie innych informatyków (specjalistów technicznych), co sprawia, że studenci funkcjonują na Uczelni w dość hermetycznym środowisku.

Jednym z czynników wpływających na odpowiedni poziom komunikacji charakterystyczny dla jednostki jest wysoki poziom ugodowości. Wyższy poziom ugodowości wskazuje, że takie jednostki potrafią współpracować z innymi, są przy tym empatyczne, przyjazne, kompromisowe [DT81], a także sympatyczne [Hog83]. Do komponentów ugodowości zalicza się takie wymiary jak prostolinijność, altruizm, ustępliwość, skromność, skłonność do rozczulania się [CM80]. Osoby z wyższym poziomem ugodowości mają naturalne skłonności do neutralizowania sytuacji konfliktowych i podkreślania korzyści wynikających z funkcjonowania w grupie [Hog83]. Warto też zaznaczyć, że osoby z wysokimi wynikami w tym obszarze potrafią kontrolować poziom gniewu, co może mieć również temperamentalne podłoże [RB98] i sprzyjać poprawnej komunikacji.

Występująca w naszych wynikach korelacja sumienności z oceną z zajęć zgodna jest z badaniami, które wskazują, że osiągnięcia akademickie są pozytywnie związane właśnie z tym wymiarem osobowości. Ponadto znaczna ilość badań wskazuje, że wysokie wskaźniki w obszarze sumienności pozwalają przewidywać funkcjonowanie jednostki w miejscu pracy [HPLP07]. Taka osoba jest postrzegana jako odpowiedzialna, wytrwała w dążeniu do celu, rozważna w planowaniu i podejmowaniu nowych zadań. Niska sumienność wiąże się ze skłonnościami do działań o charakterze kunktatorskim [DS02]. Sumienni pracownicy są bardziej wiarygodni, bardziej zmotywowani, mają też niższe wskaźniki absencji i rzadziej przypisuje im się szkodliwe zachowania w pracy, takie jak kradzież i agresja w stosunku do innych pracowników [MBS98].

W świetle uzyskanych wyników potwierdziła się również hipoteza numer dwa. Dobrze rokujący projektanci charakteryzują się określonymi wymiarami osobowości. Są bardziej introwertywni niż przyszli informatycy, którzy nie są uznawani jako dobrze rokujący programiści. Przyszli programiści, jak wspo-

mniano wcześniej, lubią działać na własną rękę, z dala od różnych dystraktorów. Praca w pojedynkę jest dla nich bardziej przewidywalna i wszystko jest pod kontrolą jednostki. Jeśli coś podlega zmianie, to tylko autor kodu może to zmienić, w sposób szybki, bezpieczny. Wyniki pracy są weryfikowane na bieżąco. Takie cechy są szczególnie pożądane przy spełnieniu przyunajmniej jednego z dwóch warunków: organizacji pracy zespołu IT w trybie ścisłej modularyzacji kodu, a także wykonania pracy w trybie zdalnym.

Dobrze rokujący projektanci osiągają też wyższe wyniki w obszarze neurotyczności niż studenci, którzy prawdopodobnie nie będą się specjalizować w obszarze projektowania. Jednostki bardziej neurotyczne mają większe skłonności do adekwatnego interpretowania swojego nastroju, analizowania siebie, i krytycznego spojrzenia na rzeczywistość, co może się wiązać z odpowiednio niższą samooceną, poczuciem bezradności. Wiedzą, że poprawa nastroju musi mieć realne podłoże, a nie wynikać z myślenia życzeniowego, chwilowej potrzeby bycia szczęśliwym. Bardziej realistyczne i krytyczne spojrzenie na rzeczywistość jest atutem osób wykonujących żmudną i czasochłonną pracę programisty-kodera, dla którego najważniejszy jest efekt końcowy. Nawet jeśli będzie wymagało to powrotu do wcześniejszych etapów, reorganizacji pracy, weryfikacji niedostrzegalnych dla innych szczegółów, co może prowadzić do krótkotrwałego bądź długotrwałego obniżenia samooceny, taka jednostka będzie zdeterminowana, żeby wykonać swoją pracę należycie i dokładnie.

Należy podkreślić, że nasze badanie miało charakter pilotażowy. Jednym z jego ograniczeń była relatywnie mała próba osób badanych, które z racji przyjętej na Uczelni organizacji procesu dydaktycznego nie podlegały jednakowym kryteriom oceny informatycznej. W kolejnych etapach nasz zespół będzie dążył do ujednolicenia kryteriów oceny informatycznej i zwiększenia populacji osób badanych pod względem wieku i doświadczenia w programowaniu. Ponadto nasze badania obejmą bardziej szczegółowe cechy osobowości znajdujące się w szerokim spektrum neurotyczności, sumienności i ekstrawersji Costy i McCrae. Pozwoli to na zbudowanie bardziej kompletnego i szczegółowego modelu osobowości skutecznego projektanta oprogramowania. Z naszego punktu widzenia, niezbędne jest również rozszerzenie badań na tych projektantów, którzy już zawodowo pracują nad oprogramowaniem. Dzięki temu będziemy mogli zweryfikować, czy cechy osobowości korelujące w przypadku programistów-studentów z jakością dokumentacji projektowej, będą również diagnostycznymi wymiarami ich sukcesu na rynku pracy.

5.3 Podsumowanie

Uzyskane w badaniach pilotażowych wyniki wskazują wprost na istnienie psychospołecznych korelatów predyspozycji zawodowych projektantów oprogramowania. Po wstępnym rozeznaniu tematu już wiadomo, jakie dodatkowo prace badawcze należy poprowadzić, aby mieć metodologiczne podstawy do praktycznego wykorzystania testów psychologicznych w celu wspomagania procesu dydaktycznego na kierunku Informatyka. Można by spodziewać się wdrożenia na wydziałach informatycznych dynamicznego zarządzania specjalizacjami, bazującego na przywiązaniu odkrywanych już na drugim roku studiów predyspozycji zawodowych aktualnych studentów do preferowanych dla nich ról w przemysłowych zespołach projektowych, a nie tak jak dotychczas planowania specjalizacji wyłącznie w oparciu o kojarzenie segmentów rynku oprogramowania lub platform implementacyjnych. Możliwość aktywnego propsychologicznego profilowania zawodowego studentów (dynamiczne specjalizacje) na podstawie prognozy zapotrzebowania na specjalistów IT może stać się naszą odpowiedzią na konieczność dostosowania oferty edukacyjnej wyższych uczelni do potrzeb rynku. Implikacją praktyczną przeprowadzonych badań może być poprawa w zakresie umiejętności komunikacyjnych studentów-informatyków. Należałoby się zastanowić nad możliwościami bardziej efektywnego wykorzystania w tym kierunku godzin dydaktycznych planowanych w module humanistycznym.

Bibliografia

[AGJ09] Acuńa S. T., Gómez M., Juristo N., How do personality, team processes and task characteristics relate to job satisfaction and software quality?, *Inf Softw.Technol*., 51 (3): 627–639, 2009.

[BM91] Barrick M. R., Mount M. K. The big five personality dimensions and job performance: a meta-analysis. *Personality Psychology*, 44: 1–26, 1991.

[BM95] Boyle G. J., Myers-Briggs type indicator (MBTI): some psychometric limitations. *Australian Psychologist*, 30(1): 71–74, 1995.

[CA06] Chao J., Atli G., Critical Personality Traits In Successful Pair Programming. In: *Proceedings of Agile 2006 Conference*, pp. 89–93, 2006.

[Cap03] Capretz L. F., Personality types in software engineering. *International Journal of Human–Computer Studies,* 58 (2): 207–214, 2003.

[CD08] Choi K. S., Deek F. P., Im I., Exploring the underlying aspects of pair programming: the impact of personality. *Information and Software Technology*, 50(11): 1114–1126, 2008.

[CG07] Cunha A. D. D., Greathead D., Does personality matter? An analysis of code-review ability. *Communications of the ACM*, 50(5): 109–112, 2007.

[CH06] Cegielski C. G., Hall D. J., What makes a good programmer? *Comm. ACM*, 49 (10): 73–75, Oct. 2006.

[Cho04] Choi K. S., A discovery and analysis of influencing factors of pair programming. Unpublished Ph.D. Dissertation, New Jersey Institute of Technology, USA 2004.

[CM80] Costa P. T. Jr., McCrae R. R., Influence of extraversion and neuroticism on subjective well-being: Happy and unhappy people. *Journal of Personality and Social Psychology*, 38: 668–678, 1980.

[Con06] Conard M. A., Aptitude is not enough: how personality and behavior predict academic performance. *J Res Pers*, 40: 339–346, 2006.

[DS02] Dewitt S., Schouwenburg H. C., Procrastination, temptations, and incentives: The struggle between the present and the future in procrastinators and the punctual. *European Journal of Personality*, 16 (6): 469–489, 2002.

[DS96] De Raad B., Schouwenburg H. C., Personality in learning and education: a review. *Eur J Pers*, 10: 303–336, 1996.

[DSGO06] Driskell J. E., Salas E., Goodwin F. F., O'Shea P. G., What makes a good team player? Personality and team effectiveness. *Group Dynamics: Theory, Research, and Practice*, 10(4): 249–271, 2006.

[DT81] Digman J. M., Takemoto-Chock N. K., Factors in the Natural Language of Personality: Re-analysis, Comparison, and Interpretation of Six Major Studies, *Multivariate Behavioral Research,* 16: 149–170, 1981.

[FTAS08] Feldt R., Torkar R., Angelis L., Samuelsson M., Towards individualized software engineering: empirical studies should collect psychometrics. In: *Proceedings of the 2008 International Workshop on Cooperative and Human Aspects of Software Engineering*, pp. 49–52, ACM, May 2008.

[GL04] Gorla N., Lam Y. W., Who should work with whom? Building effective software project teams. *Communications of the ACM,* 47(6): 79–82, 2004.

[Gra13] Grabowska G. H., Bliskość emocjonalna w tworzeniu zespołów projektowych. *Zeszyty Naukowe Wydziału Informatycznych Technik Zarządzania Wyższej Szkoły Informatyki Stosowanej i Zarządzania „Współczesne Problemy Zarządzania”*, 1, 2013.

[HAES10] Hannay J. E., Arisholm E., Engvik H., Sjoberg D. I., Effects of personality on pair programming. *IEEE Trans. Softw. Eng.*, 36 (1): 61–80, 2010.

[Hog83] Hogan R., Socioanalytic theory of personality. In M. M. Page (Ed.), 1982 Nebraska Symposium on Motivation: Personality—current theory and research, pp.55–89. Lincoln: University of Nebraska Press, 1983.

[Hor11] Horney K., Neurotyczne osobowości naszych czasów. Rebis, Poznań 2011, s. 62–68.

[HPLP07] Higgins D. M., Peterson J. B., Lee A., Pihl R. O., Prefrontal cognitive ability, intelligence, Big Five personality and the prediction of advanced academic and workplace performance, *Journal of Personality and Social Psychology*, 93: 298–319, 2007.

[KC06] Karn J., Cowling T., A follow up study of the effect of personality on the performance of software engineering teams. In: *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, ACM, pp. 232–241, 2006.

[KFA14] Kosti M. V., Feldt R., Angelis L., Personality, emotional intelligence and work preferences in software engineering: An empirical study. *Information and So-*

ftware Technology, 56 (8): 973–990, 2014.

[LA05] Lee K., Ashton M. C., Psychopathy, Machiavellianism, and narcissism in the Five-Factor Model and the HEXACO model of personality structure. *Personality Individ. Differ.,* 38 (7): 1571–1582, 2005.

[Lan02] Laney M. O., The Introvert Advantage: How to Thrive in an Extrovert World. NY: Workman Publishing, 2002.

[LP07] Lee A., Pihl R. O., Prefrontal cognitive ability, intelligence, Big Five personality and the prediction of advanced academic and workplace performance. *Journal of Personality and Social Psychology*, 93: 298–319, 2007.

[MBS98] Mount M. K., Barrick M. R., Stewart G. L., Five-factor model of personality and Performance in jobs involving interpersonal interactions. *Human Performance*, 11 (2): 145–165, 1998.

[MJ92] McCrae R. R., John O. P., An introduction to the five-factor model and its applications. *Journal of Personality*, 60 (2): 175–215, 1992.

[MRMH+08] Matzler K., Renzl B., Muller J., Herting S., Mooradian T. A., Personality traits and knowledge sharing. *J Econ Psychol*, 29: 301–313, 2008.

[RB98] Rothbart M. K., Bates J. E., Temperament. In W. Damon (Series Ed.), N. Eisenberg (Vol. Ed.), Handbook of child psychology: Vol. 3. Social, emotional and personality development, (5th Ed), New York: Wiley, 1998, pp. 105–176.

[RC03] Rothmann S., Coetzer E. P., The big five personality dimensions and job performance, *SA Journal of Industrial Psychology*, 29(1): 68–74, 2003.

[Sch08] Schmitt N., The interaction of neuroticism and gender and its impact on self-efficacy and performance. *Human Performance*, 21: 49–61, 2008.

[SMG11] Salleh N., Mendes E., Grundy J., Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review. *IEEE Trans Software Eng*, 37(4): 509–525, 2011.

[SMG14] Salleh N., Mendes E., Grundy J., Investigating the effects of personality traits on pair programming in a higher education setting through a family of experiments. *Empirical Software Engineering*, 19 (3):714–752, Jun 2014.

[SSAD06] Sfetsos P., Stamelos I., Angelis L., Deligiannis I., Investigating the impact of personality types on communication and collaboration-viability in pair programming – an empirical study. *Proceedings of the 7th International Conference on Extreme Programming and Agile Processes in Software Engineering (XP 2006)*, pp. 43–52, 2006.

[SSAD09] Sfetsos P., Stamelos I., Angelis L., Deligiannis I., An experimental investigation of personality types impact on pair effectiveness in pair programming. *Empir. Softw. Eng.*, 14 (2): 187–226, 2009.

[TR04] Tamir M., Robinson M. D., Knowing good from bad: The paradox of neuroticism, negative affect, and evaluative processing. *Journal of Personality and Social Psychology*, 87: 913–935, 2004.

[Wry10] Wrycza S., Informatyka ekonomiczna. PWE, Warszawa 2010, s. 345.

[ZSSŚ98] Zawadzki B., Strelau J., Szczepaniak P., Śliwińska M., Inwentarz osobowości NEO-FFI Costy i McCrae. Pracownia Testów Psychologicznych PTP, Warszawa 1998, s. 7–34.

Rozdział 6

Wsparcie procesu projektowania systemów sterowania ruchem

Inżynieria wymagań jest ważnym działem inżynierii oprogramowania. Znaczenie inżynierii wymagań i prawidłowego przebiegu procesów zarządzania wymaganiami jest często kluczowym czynnikiem sukcesu dla projektów informatycznych. Wsparcie procesów projektowych odpowiednimi narzędziami sprzyja poprawie jakości projektowanych systemów i umożliwia projektantowi koncentrację na dziedzinie problemowej. Jednym z narzędzi wykorzystywanych w inżynierii oprogramowania do tego rodzaju wsparcia są dedykowane języki dziedzinowe. W rozdziale przedstawiono zastosowanie języka TransML i środowiska TransCAD jako narzędzia usprawniającego procesy wytwórcze dla systemów wspomagających sterowanie ruchem ulicznym. Posługując się wspomnianym językiem i środowiskiem projektant systemów sterowania ruchem może skoncentrować się na określeniu i zapisaniu wymagań na system za pomocą elementów piktogramowych zrozumiałych dla inżyniera ruchu. Dzięki osadzeniu środowiska projektowego na platformie IBM RSA możliwa jest weryfikacja budowanych modeli pod względem ich poprawności semantycznej oraz obsługa procesów transformacji modelu do kolejnych warstw abstrakcji projektowej.

Problem efektywnego sterowania ruchem ulicznym jest przedmiotem ciągłych badań. Badania te mają też bezpośrednie przełożenie na praktyczne zastosowania w regulacji ruchem w aglomeracjach miejskich. Niniejsze praca stanowi kontynuację badań autorów nad tą problematyką. W pracach [ADKM11, DKAK11] zostały przedstawione wyniki badań uzyskanych w ramach projektu badawczego EUREKA E!4492 "Wspomagane komputerowo oprogramowanie do sterowania ruchem ulicznym" realizowanego przez Instytut Sterowania i Elektroniki Przemysłowej Politechniki Warszawskiej na zlecenie Narodowego Centrum Badań i Rozwoju. Zaproponowano w nich struktury danych oraz interfejsy GUI do baz danych dla systemu wspomagającego projektowanie planów świateł z wykorzystaniem programowania urządzeń sterujących zainstalowanych na skrzyżowaniach.

Istotnym problemem z punktu widzenia inżynierii procesów wytwórczych jest zapewnienie wsparcia dla inżynierów projektujących systemy sterowania ruchem.

W rozdziale przedstawiona jest metoda wpierająca pracę projektanta systemów sterowania ruchem oparta o język modelowania TransML przeznaczonego do budowy systemów kontroli i nadzorowania ruchu drogowego. Zaproponowana metoda powinna ułatwić projektantom rozwiązań wspomagających sterowanie ruchem oraz inżynierom ruchu, budowę i walidację systemów nadzoru i sterowania ruchem. Autorzy zaproponowali oryginalne podejście do projekto-

wania systemów sterowania ruchem z wykorzystaniem autorskiego środowiska TransCAD (Computer Aided Design).

6.1 Język dziedzinowy TransML

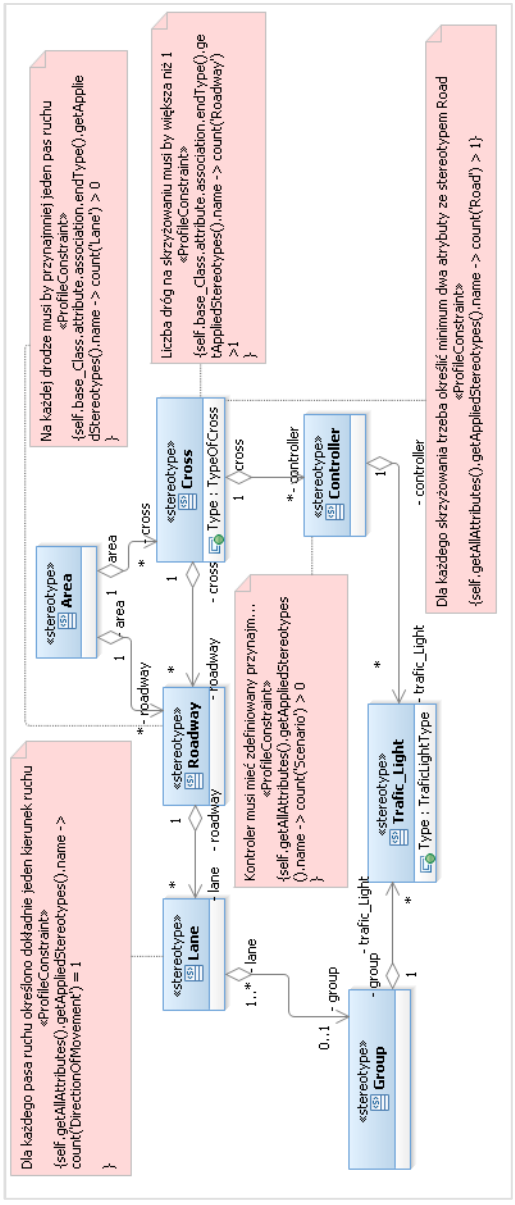
W modelowaniu systemów ruchu możliwe jest wykorzystanie języków dziedzinowych, które są dedykowane do budowy systemów kontroli i nadzorowania ruchem. Główną korzyścią płynącą z zastosowania języków dziedzinowych, jest ich użyteczność w identyfikowaniu istotnych cech i wymagań występujących na różnych poziomach abstrakcji. Dzięki precyzyjnemu wyrażaniu ograniczeń nałożonych na konstrukcje językowe, możliwa jest także kontrola poprawności elementów składowych oraz walidacja tworzonych modeli. W przypadku zastosowania języków dziedzinowych do reprezentacji modeli, możliwe jest utworzenie odrębnych języków dla każdego z nich – konceptualnego, logicznego i fizycznego. Dobór odpowiedniego języka dziedzinowego, wpływa natomiast na proces walidacji modelu konceptualnego – jego czytelność oraz operowanie pojęciami bezpośrednio związanymi z obszarem zastosowania, ułatwia ekspertowi dziedzinowemu dokonanie przeglądu. Niewątpliwą zaletą języków dziedzinowych jest również automatyzacja procesu transformacji modeli. Szczegóły koncepcji budowy i zastosowań języków dziedzinowych DSL można znaleźć w pracy [Fow12]. W pracy [DS13] został po raz pierwszy przez autorów zaproponowany język dziedzinowy TransML będący dialektem języka UML. Jest on zaimplementowany w postaci profilu języka UML, którego kluczowe elementy przedstawiono na rysunku 6.1. Profil ten jest adresowany do inżynierów sterowania ruchem skupiając się na kluczowych cechach i abstrakcjach z obszaru inżynierii ruchu drogowego.

Z punktu widzenia IT, profil ten stanowi reprezentację metamodelu języka dziedzinowego TransML, którego stereotypy jako nowe pojęcia języka są mapowane na metaklasy języka UML, a ograniczenia definiowane są w postaci typów enumeratywnych i wyrażeń w języku OCL.

Ponieważ budowany język dziedzinowy bazuje na semiotyce, udostępnia on graficzną reprezentację wypowiedzi, w których z każdym pojęciem wiążemy piktogram, który stanowi jego metaforę, tzn. powinien przez eksperta dziedzinowego być bezpośrednio zrozumiały. Znaczenie i piktograficzną reprezentację wybranych pojęć języka TransML przedstawiono w tabeli 6.1.

Opracowywane modele dziedziny, wyrażone w sposób obiektowy za pomocą pojęć z profilu, określają sformalizowany model pojęciowy i mogą być wykorzystane do automatycznej weryfikacji i walidacji modeli dziedziny przez środowisko CAD [DS13]. Takie podejście otwiera również drogę do automa-

tycznych transformacji modelu dziedzinowego do innych poziomów abstrakcji (przykładowy wynik transformacji typu model do modelu stanowi przekształcenie topologii drogi z rysunku 6.3 na model drogi na rysunku 6.2.a).



Rysunek 6.1. Kluczowe elementy profilu języka TransML [DS13].

Przedstawiona przez nas w niniejszym rozdziale propozycja języka TransML, wg Martina Fowlera [Fow12], jest językiem dziedzinowym typu „Internal DSL" będący dialektem języka UML, który został przez nas zaimplementowany w postaci profilu języka UML. Ponieważ TransML jest adresowany dla inżynierów sterowania ruchem, odwołuje się on do koncepcji dziedziny pionowej (vertical domain) [DHHT08], tzn. skupia się na kluczowych cechach i abstrakcjach z obszaru inżynierii ruchu drogowego.

Tabela 6.1. Semantyka wybranych pojęć języka TransML.

Lp	Nazwa pojęcia	Piktogram	Znaczenie pojęcia
1	Obszar (Area)		nazwany rejon grupujący drogi i skrzyżowania
2	Skrzyżowanie (Cross)		zasadnicze pojęcie języka - określa odcinek drogi na którym możliwa jest zmiana kierunku ruchu pojazdu, węzeł drogowy
3	Scenariusz (Scenario)		opis zachowania, który określa przebieg sterowania światłami w ruchu ulicznym dla danego skrzyżowania, a obsługiwany jest przez konkretny kontroler
4	Kkontroler (Controller)		urządzenie, które realizuje ustalony scenariusz sterowania światłami w ruchu ulicznym
5	Światła uliczne (Trafic light)		obiekt sterowania, który przedstawia uczestnikom ruchu decyzje określone w scenariuszu za pomocą sygnałów świetlnych
6	Pas ruchu (Lane)		określa odcinek drogi po którym mogą poruszać się uczestnicy ruchu w jednym kierunku.
7	Typ świateł ulicznych (Trafic light type)		światła z jedną, dwoma, trzema, ..., lampami
8	Detektor (Detector)		inteligentny czujnik umożliwiający zarówno wykrycie specyficznych obiektów w obszarze skrzyżowania jak i np. ich zaliczenie.
9	Stan pasa ruchu (State of lane)		Zakładamy, że pas ruchu może przebywać w jednym z trzech stanów: sprawny odcinek drogi, częściowo sprawny odcinek drogi, wyłączony odcinek drogi.
10	incydent na drodze		zdarzenie na drodze, które prowadzi do zmiany jej stanu np.: wypadek drogowy, roboty drogowe, … Zdarzenie zawsze posiada min. dwa atrybuty, tj. czas i miejsce – określone przez dystans

W naszej propozycji zasadniczymi użytkownikami modeli specyficznej dziedziny, której dotyczy rozdział, obok zespołu IT, będą inżynierowie sterowania ruchem ulicznym, jako interesariusze biznesowi (ang. *business stakeholders*), którzy muszą wyrazić i zrozumieć problem, który należy rozwiązać.

Warto zauważyć, że poszukiwane rozwiązanie ogranicza swój zakres jedynie do pojęć i zasad specyficznej dziedziny, opisanej w języku DSL, który z założenia ogranicza sposób budowy rozwiązania tak, aby unikać błędów projektowych.

Model dziedzinowy pozwala także na bardziej efektywną - od modeli ogólnego przeznaczenia - komunikację, zapewniając lepsze zrozumienie dla budowanego rozwiązania przez wszystkich jego uczestników. Może więc on zapewnić optymalny poziom szczegółowości komunikacji między tymi, którzy pracują w tej dziedzinie.

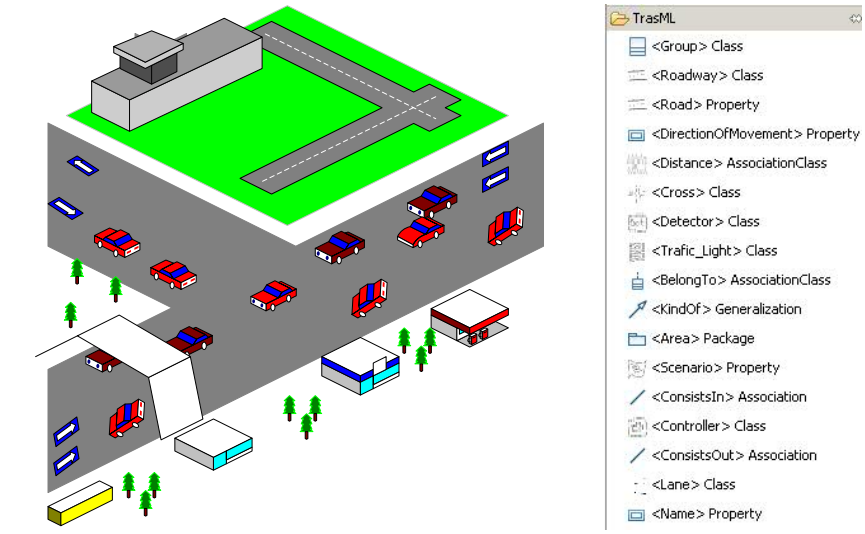
6.2 Proces projektowania

Budowę modelu sterowania ruchem dla sieci dróg w środowisku TransCAD rozpoczynamy od utworzenia topologii dróg, tj. umieszczenia elementów stanowiących opisy dróg występujących w ustalonym obszarze, w postaci piktogramów języka TransML (rysunek 6.2b), na diagramie klas. Diagram ten powinien być umieszczony w pakiecie z nazwą obszaru (na przykład skrzyżowanie ulicy Rosoła z Doliną Służewiecką w Warszawie). Inżynier ruchu tworzący topologię dróg wybiera elementy z palety i umieszcza je na diagramie klas. Elementami tworzącymi topologię dróg w zdefiniowanym obszarze, są: skrzyżowanie (Cross), droga (Roadway), pas jezdni (Lane), kierunek ruchu (Direction of movement).

W rozdziale założono, że zakres proponowanego podejścia do modelowania elementów inżynierii ruchu drogowego ogranicza się do projektowania dróg (w tym skrzyżowań drogowych), należało więc uwzględnić w nim dwa etapy prac:

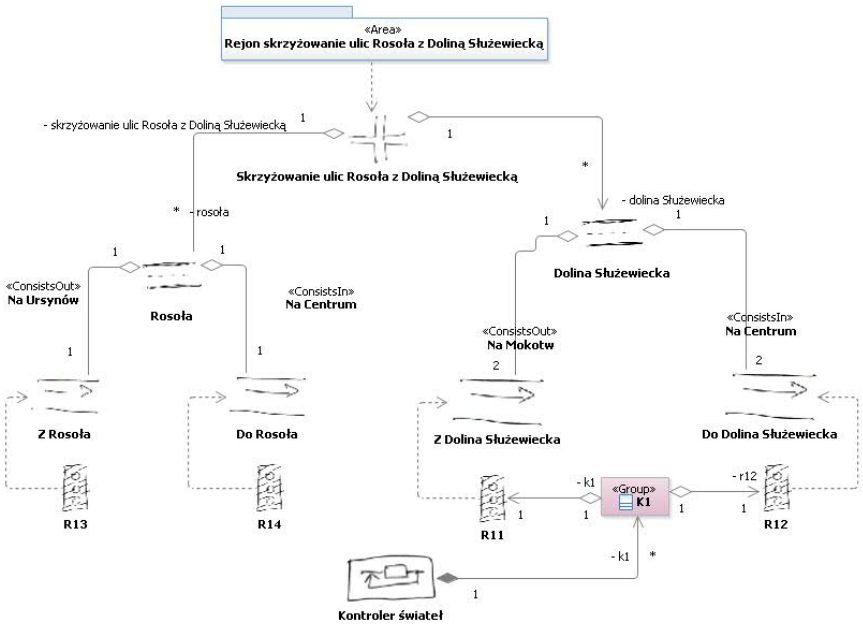
1. badanie i analizę ruchu oraz przedstawienie koncepcji wariantów rozwiązania,
2. projekt techniczny wybranego wariantu drogi.

Obecnie zakładamy, że proponowane podejście będzie wykorzystywane w pierwszym z powyższych zakresów (pomijamy procesy wspomagania projektów technicznych).



a) b)

Rysunek 6.2. a) Model drogi stanowiący wynik transformacji jej topologii; b) Paleta profilu języka TransML.



Rysunek 6.3. Topologia rejonu skrzyżowania ulic Rosoła z Doliną Służewiecką w Warszawie.

W podejściu tym inżynier ruchu, wyposażony w komputerowe środowisko projektowania sterowania ruchem ulicznym TransCAD, opracuje a następnie dokona badania wariantów rozwiązania dla drogi będącej przedmiotem projektu. Wynikiem analizy wariantów rozwiązania będzie jego decyzja, którą stanowił będzie symulacyjnie zweryfikowany model drogi o wysokiej przepustowości ruchu (z uwzględnieniem segregacji czasowej ruchu za pomocą sygnalizacji świetlnej).

Uproszczony scenariusz procesu modelowania sprowadza się do wykonania następujący zadań:

1. buduj topologię drogi;
   - buduj skrzyżowania dróg jednopoziomowych (skrzyżowania zwykłe, skanalizowane, z wyspą centralną, o ruchu okrężnym, z koleją);
   - buduj ronda (ronda duże, ronda małe, mini ronda, skrzyżowania z wyspą centralną (kołową, eliptyczną, rombową);
   - buduj odcinek drogi;
2. dokonaj weryfikacji i walidacji projektu topologii drogi;
3. przypisz odległości (i pozostałe wymagane atrybuty modelu);
4. rozmieść elementy inżynierii ruchu drogowego:
   - wprowadź segregację czasową ruchu za pomocą sygnalizacji świetlnej. Na rysunku 6.3 przedstawiono topologię skrzyżowania nad którą prace jeszcze trwają, ponieważ sygnalizatory R13 i R14 nie zostały zgrupowane i podłączone do kontrolera świateł – o czym powiadomi inżyniera ruchu mechanizm walidacji (po jego uruchomieniu);
   - –znaki drogowe;
5. dokonaj symulacyjnego badania ruchu drogowego:
   - określ punkty kolizji;
   - wyznacz powierzchnię kolizji;
   - określ przepustowości;
6. raportuj o przeprowadzonych badaniach.

Etapy procesu projektowego przedstawiono na rysunku 6.4.

Rysunek 6.4. Etapy procesu projektowego.

6.3 Środowisko projektowania systemów sterowania ruchem

Budowane przez nas środowisko stanowi narzędzie typu CAD (Computer Aided Design). Posiada ono własne repozytorium projektu oraz zbiór edytorów graficznych, które poprzez piktograficzne metafory, reprezentowane jako paleta pozwoli na budowanie modeli w zaproponowanym przez nas języku dziedzinowym - TransML.

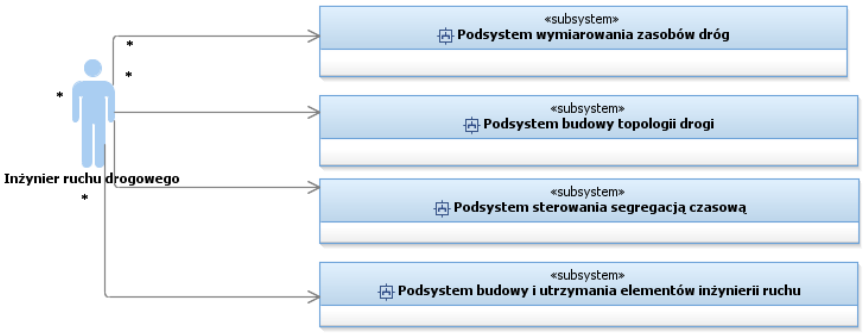
Pojęcia tego języka zostały zdefiniowane w postaci słownika i metamodelu dla wycinka rzeczywistości dotyczącego opisu elementów składających się na topologię drogi, w tym strukturę skrzyżowań dróg oraz możliwości odwzorowania zagadnień dotyczących inżynierii ruchu drogowego (w tym użytych znaków drogowych i sygnalizacji świetlnej).

Najciekawszym elementem budowanego środowiska TransCAD, w naszej ocenie, jest możliwość definiowania reguł - w postaci wyrażeń napisanych w języku OCL, jako ograniczeń modelu, które pozwolą inżynierowi ruchu w procesach projektowania wyeliminować błędy, które mogą być wychwycone przez automat (opisany w języku OCL).

6.3.1 Wymagania na system

W rozdziale prezentujemy jedynie kluczowe wymagania, które pozwolą zbudować obraz obszarów budowanego środowiska, a nie szczegółowe wymagania (ze względu na ograniczenia publikacyjne).

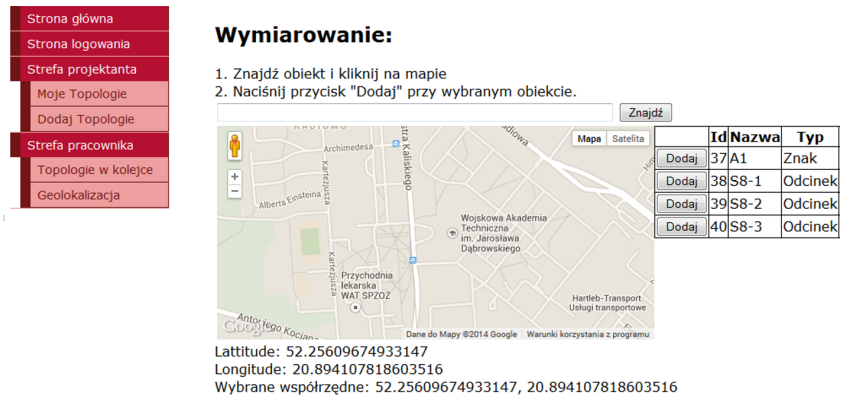
Zakres zaprezentowanych wymagań ograniczymy jedynie do specyfikacji funkcjonalności systemu, poprzez określenie: użytkowników systemu (aktorów) i jego usług (przypadków użycia). Wstępnie, usługi budowanego środowiska CAD zostały zgrupowane w cztery podsystemy (rysunek 6.5):



Rysunek 6.5. Model funkcjonalności z punktu widzenia inżyniera ruchu - szkic.

– Podsystem budowy topologii drogi: buduj drogę, usuń drogę, buduj skrzyżowanie, usuń skrzyżowanie;

– Podsystem wymiarowania zasobów dróg: przypisz odległość między początkiem a końcem, usuń wskazany wymiar, lokalizuj światła, lokalizuj detektor. Funkcjonalność ta jest realizowana przez wywołanie usług lokalizacyjnych Google Maps (rysunek 6.6), lub poprzez bezpośrednie wprowadzanie danych lokalizacyjnych.



Rysunek 6.6. Wymiarowanie modelu z użyciem usługi lokalizacyjnej Google Maps.

– Podsystem sterowania segregacją czasową: buduj diagram fazowy, buduj listę sygnałów drogowych, buduj macierz bezpieczeństwa, buduj diagram programu sygnalizacji, buduj diagram przejść międzyfazowych, buduj tablicę obliczeń przepustowości;

– Podsystem budowy i utrzymania elementów inżynierii ruchu: maluj pas ruchu, zetrzyj pas ruchu, przypisz kierunek ruchu do pasa ruchu na wskazanej drodze, wstaw znak drogowy, usuń znak drogowy, wstaw sygnalizator świetlny, usuń sygnalizator świetlny;

6.3.2 Zakres wspomagania środowiska TransCAD

Środowisko TransCAD w naszym projekcie stanowi kompleksowe rozwiązanie w zakresie komputerowo wspomaganego projektowania modeli ruchu drogowego.

Środowisko to wspiera następujące procesy:

1) Proces budowy środowiska CAD – szkielet tego środowiska stanowi produkt, IBM Rational Software Architect (IBM RSA), który rozszerzono o autorski dodatek (plugin) - w postaci profilu języka TransML (rysunek 6.1)

oraz pakiet IBM RSA Simulation Toolkit. Zbudowane przez nas środowisko jest otwarte na nowe pojęcia języka DSL – TransML, a profile dla kolejnych jego wydań będą budowane w procesach automatycznej generacji kodu.

2) Proces budowy modeli – Inżynier ruchu drogowego buduje swoje modele w środowisku TransCAD i zapisuje je w repozytorium IBM RSA;

3) Proces przeprowadzania eksperymentów (w tym budowy różnych wariantów rozwiązań). Dzięki pakietowi IBM RSA Simulation Toolkit opracowane modele zachowania mogą być symulowane (tak jak modele UML);

4) Proces oceny uzyskanych wyników.

W zaproponowanym narzędziu symulacja komputerowa została wykorzystana do badania opracowanych modeli topologii dróg i modeli ruchu drogowego na tych drogach. Modele te zostały opracowane w obiektowym języku modelowania TransML, bazującym na języku UML i standardach z nim związanych, a wspieranych przez grupę OMG, tj. fUML, Alf, OCL [OMG13].

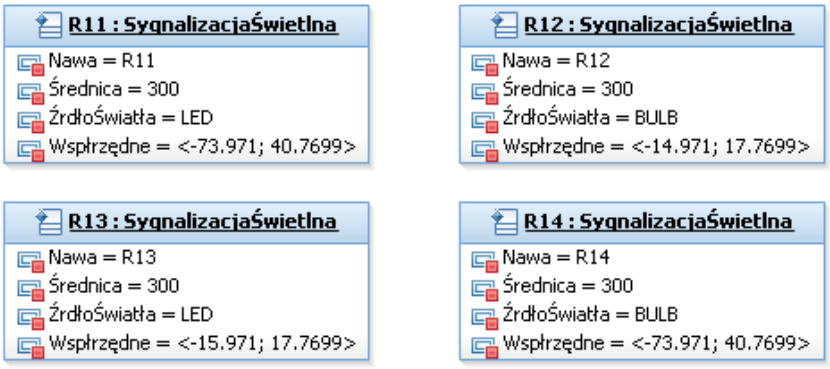
Podczas symulacji TransCAD odwzorowuje powiązane z topologią drogi zachowanie jej użytkowników (opisane przez scenariusz), kolejno przechodząc przez elementy topologii drogi wg. kolejności ustalonej w scenariuszu. Sterowanie procesem symulacji odbywa się poprzez zdarzenia, z poziomu kodu napisanego w języku Alf bądź ręcznie (np. przy blokach decyzyjnych).

W zbudowanym środowisku TransCAD (podobnie jak w IBM RSA) symulacja może być przeprowadzania w trybie wykonawczym (execute), bądź w trybie debugowania (debug). W pierwszym trybie inżynier ruchu drogowego jest tylko obserwatorem symulacji, natomiast w drugim trybie ma możliwość ingerowania w wykonanie (np. poprzez zmianę w modelach), podgląd i zmianę stanu zmiennych, wybór ścieżek decyzyjnych, itp.

Dodatkowo środowisko TransCAD umożliwia zebranie następujących wyników symulacji: historii komunikatów przesyłanych między obiektami, śladów przepływu komunikatów, historii zapisów na konsoli operatora symulacji. Możliwości środowiska można znacznie rozszerzyć stosując język Alf lub UAL do specyfikacji zachowania konkretnych elementów modelu, w celu np. symulacji interakcyjnej pracy, wykorzystania plików, czy stosowania złożonych struktur danych.

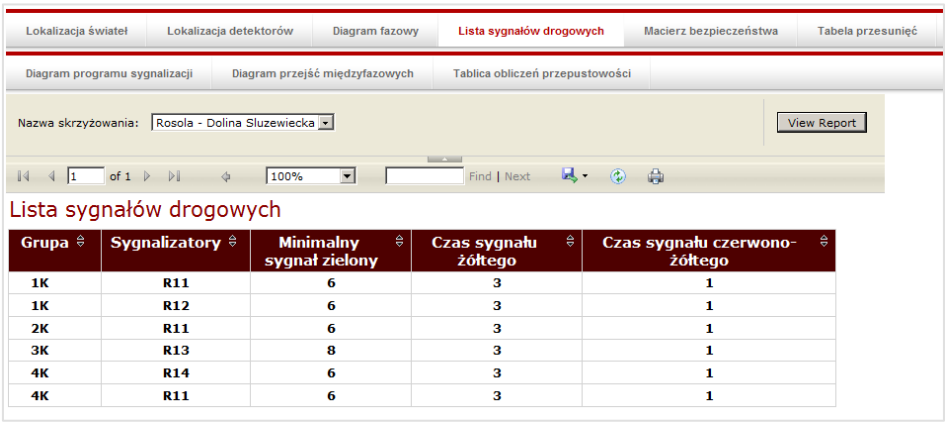
6.3.3 Przykład wykorzystania środowiska TransCAD

W rozdziale 2 przedstawiono topologię dróg dla rejonu skrzyżowania ulic Rosoła i Dolina Służewiecka. W rozdziale tym, dla tego przykładu opracowano wybrane charakterystyki.



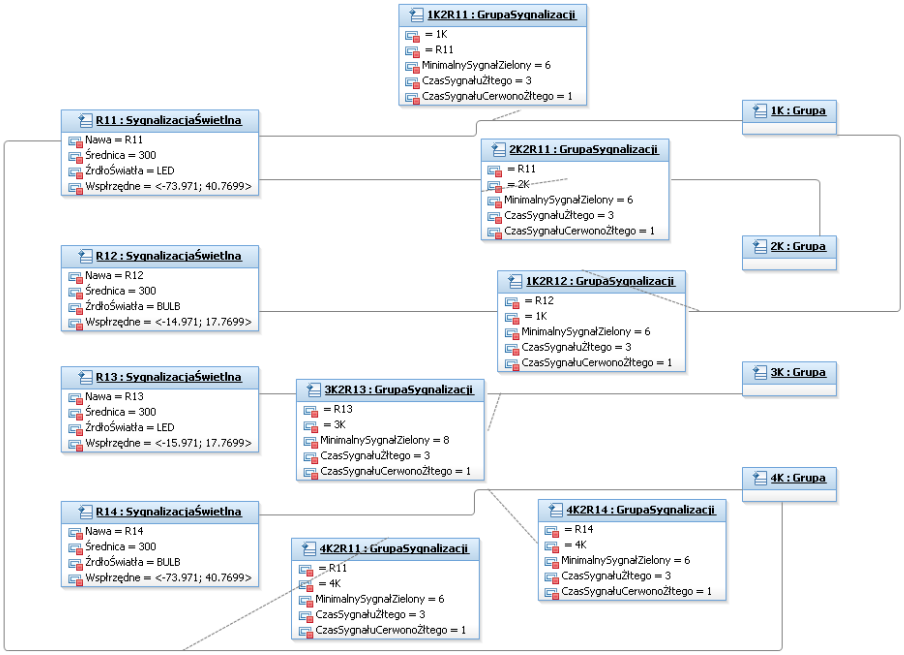
Rysunek 6.7. Model listy sygnałów drogowych na skrzyżowaniu ulic Rosoła i Dolina Służewiecka.

Na rysunku 6.7 przedstawiono model listy sygnałów drogowych na skrzyżowaniu ulic Rosoła i dolina Służewiecka, a na rysunku 6.8 odpowiadającą mu listę sygnałów drogowych opracowaną w narzędziu DynaSignal [ADKM11].



Rysunek 6.8. Lista sygnałów drogowych.

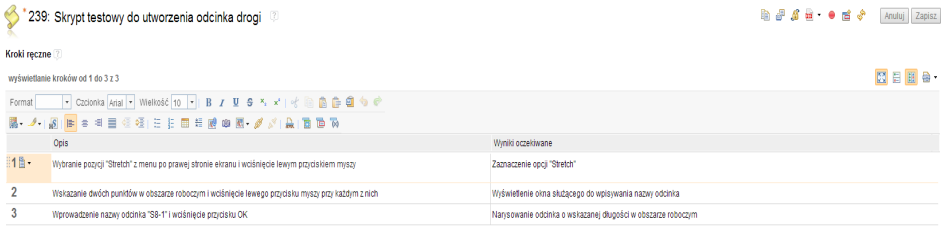
Rysunek 6.9 przedstawia fragment modelu dziedziny odpowiadający liście sygnałów drogowych z rysunku 6.8 opracowany w narzędziu TransCAD z wykorzystaniem języka dziedzinowego TransML.



Rysunek 6.9. Fragment modelu dziedziny w narzędziu TransCAD dla skrzyżowania Rosoła Dolina Służewiecka.

Ciekawą własnością budowanych modeli jest możliwość selektywnego grupowania dowolnych elementów w tzw. warstwy, z których możemy komponować dowolny interesujący nas widok.

Systematyczne badanie narzędzia TransCAD przeprowadzono z wykorzystaniem środowiska Rational Jazz Team Server Quality Management (rysunek 6.10).



Rysunek 6.10. Skrypt testowy dla przypadku: utworzenie odcinka drogi.

6.4 Wnioski

Rozdział prezentuje możliwości wykorzystania języka TransML zdefiniowanego w pracy [DS13] w narzędziu TransCAD do modelowania i projektowania systemów kontroli ruchu. W pracy zaprezentowano odniesienie proponowanego języka modelowania do wyników badań osiągniętych w pracy [ADKM11]. Zastosowane podejście daje szerokie możliwości projektantom systemów sterowania ruchem zarówno w modelowaniu jak i projektowaniu tego typu systemów. Zastosowanie języka dziedzinowego TransML w środowisku projektowym TransCAD umożliwia wykorzystanie elementów piktogramowych języka, definiowanie reguł dziedzinowych w formie odpowiednich ograniczeń (wyrażanych w języku OCL) oraz umożliwia przeprowadzanie symulacji rozwiązań.

W rozdziale zaprezentowano odniesienie do rezultatów osiągniętych we wcześniejszych badaniach nad strukturami danych w modelowaniu sterowania ruchem pokazując w jaki sposób można powiązać oba podejścia uzyskując poszerzenie możliwości projektanta lub analityka systemów sterowania ruchem.

Ramę technologiczną dla budowanego narzędzia CAD, w naszej propozycji, stanowią: CASE - IBM RSA ver. 9.0 z pakietem IBM RSA Simulation Toolkit. Rama ta wymagała od autorów budowy jej rozszerzeń w postaci plugin'ów, z których na obecnym etapie prac najistotniejszym jest dodatek w postaci oprogramowania palety narzędziowej dla opracowanego języka dziedzinowego TransML i związanego z nim systemu weryfikacji i walidacji modeli.

Opisany język może zostać rozszerzony o nowe cechy, a w szczególności pojęcie perspektywy projektowej, które pozwoli na nanoszenie na zdefiniowaną sieć dróg, informacji np.: o zagrożeniach dla różnych służb i obywateli, którzy mogliby wykorzystywać te informacje poprzez serwisy web i aplikacje typu gov.

Opisana koncepcja języka TransML oraz środowiska TransCAD może być rozwijana w wielu kierunkach i według nas stanowi interesujący kierunek badań nad konceptualnym i narzędziowym wsparciem procesów zarzadzania i sterowania ruchem w złożonych systemach drogowych.

Bibliografia

[ADKM11] Amborski K., Dąbrowski W., Kowalczuk P., Markowski K., Logistyka 3/2011, Zastosowanie baz danych w sterowaniu ruchem ulicznym, ISSN 1231-5478

[DKAK11] Dąbrowski W., Kowalczuk P., Amborski K., Kruczkowski P., Struktury GUI do baz danych w zastosowaniu do sterowania ruchem ulicznym w: Computer Systems aided science industry and transport, Transport Committee of the Polish

Academy of Science, 2011.

[DHHT08] Dubielewicz I., Hnatkowska B., Huzar Z., Tuzinkiewicz L. (2008). Języki dziedzinowe w modelowaniu danych w kontekście MDA, [w:] S. Kozielski, B. Małysiak, P. Kasprowski, D. Mrozek (red.), Bazy danych. Rozwój metod i technologii. Architektura, metody formalne i zaawansowana analiza danych, tom 1. Warszawa: Wydawnictwa Komunikacji i Łączności.

[DS13] Dąbrowski W., Stasiak A. Środowisko projektowania systemów sterowania ruchem, TTS Technika Transportu Szynowego, 10/2013, ISSN 1232-3829, pp. 1432-1435, 10, 10(235)/2013

[Fow12] Fowler M, Domain Specific Languages, Pearson Education Inc, 2012

[OMG13] Concrete Syntax for a UML Action Language: Action Language for Foundational UML (ALF), OMG, marzec 2013

Rozdział 7

Vero: framework dla wysokowydajnych aplikacji webowych w PHP 5.4+

W rozdziale przedstawiono proces wyboru konkurencyjnych platform programistycznych (framework) Open Source, tworzonych zgodnie z najnowszymi trendami oraz standardami w środowisku PHP 5.4+. Do badań wytypowano projekty Symfony 2 oraz Zend 2. Przedstawiono tytułowy framework Vero, następnie porównano sposób implementacji trzech aplikacji, wykonujących te same działania, ale zbudowanych w oparciu o różne frameworki. Opisano badania wydajności na trzech konfiguracjach sprzętowych przy włączonym i wyłączonym rozszerzeniu OPcache. Stwierdzono, że jest możliwe zbudowanie nowoczesnego i wydajnego frameworka nawet w środowiskach o małych zasobach obliczeniowych.

Wraz z widocznym w ostatnich latach rozwojem samego języka PHP, jak i całego środowiska, pojawiło się wiele narzędzi oraz bibliotek korzystających z nowości wprowadzanych w tym języku. Na ich podstawie budowane są kompletne frameworki, których głównym celem jest zmniejszenie wymaganego nakładu pracy programisty przy implementacji typowych funkcjonalności aplikacji webowych. Za takie funkcjonalności autorzy uważają: wyświetlanie pojedynczych danych i listingów z bazy danych oraz tworzenie i edycję danych w bazie (obsługa formularzy).

Obecnie istnieje na rynku wiele frameworków aplikacji webowych dla języka PHP. Część z nich rozwijanych jest od wielu lat, co jest powodem występowania zaszłości niezbędnych w celu utrzymania ich kompatybilności wstecznej. Znany jest autorom przypadek problemów, które wystąpiły podczas korzystania z frameworka Symfony 2. Po jego instalacji na oferowanym przez jednego z polskich dostawców niskobudżetowym serwerze współdzielonym (*Virtual Private Server*, VPS) czasy odpowiedzi, nawet przy generowaniu statycznych stron, często przekraczały jedną sekundę, a przy „zimnym starcie" dochodziły do kilku sekund. Problem taki występował też na zbliżonych parametrami serwerach alternatywnych firm.

Główną różnicą w konfiguracji VPS względem maszyn deweloperskich, gdzie problem nie występował, był brak włączonego rozszerzenia *OPcache* dla PHP. *OPcache* przechowuje w pamięci wstępnie sparsowane i przetworzone pliki PHP (ang. *precompiled script bytecode*), dzięki czemu aplikacje działają szybciej. W przeciwieństwie do starszego rozwiązania APC (*Alternative PHP Cache*), *OPcache* jest domyślnie wbudowane i włączone w PHP 5.5 oraz jest oficjalnie tworzone i wspierane przez firmę Zend.

Opisane wyżej kwestie skłoniły autorów do zbadania przyczyn takiego stanu rzeczy oraz próby implementacji rozwiązania, które pomimo wykorzystania nowoczesnych komponentów i trendów zachowa wystarczającą wydajność nawet na najsłabszych serwerach.

7.1 Standaryzacja projektów PHP

W celu ustandaryzowania projektów tworzonych z wykorzystaniem języka PHP, w 2009 roku została zawiązana nieoficjalna organizacja o nazwie *PHP Framework Interoperability Group* (PHP-FIG). Na zasadach demokratycznych wydaje ona dokumenty będące rekomendacjami (*PHP Standard Recommendation* – PSR). Stosowanie PSR przez programistów zwiększa prawdopodobieństwo zakończenia projektu sukcesem. Kod staje się bardziej czytelny i łatwiejszy w użyciu. PSR pozwala również na wykorzystanie uniwersalnych i lepiej przetestowanych mechanizmów ładowania bibliotek. Według autorów, PHP-FIG jest obecnie jedyną grupą w środowisku PHP, która została w jakikolwiek sposób sformalizowana.

Dotychczas zostało wydanych pięć rekomendacji:

- **PSR-0**: dotyczy konwencji nazewnictwa plików, klas i przestrzeni nazw, dzięki czemu kompatybilny kod może zostać dołączony za pomocą standardowego autoloadera.
- **PSR-1**: określa podstawowy format zapisu plików, w tym kodowanie znaków, konwencję tagów otwierających kod PHP oraz podstawy formatowania kodu, czyli sposób zapisu klamer i nawiasów.
- **PSR-2**: określa precyzyjne zasady formatowani kodu (*Coding Style Guide*), w tym, sposób tworzenia wcięć, stawiania pustych linii oraz konwencji nazewnictwa klas i zmiennych.
- **PSR-3**: definiuje interfejs (API) loggera systemowego; do rekomendacji dołączona jest prosta biblioteka z przykładowym kodem PHP.
- **PSR-4**: definiuje sposoby specyfikowania ścieżek dostępu do automatycznie ładowanych klas; jest uzupełnieniem PSR-0.

W celu poprawy interoperacyjności szczególnie ważna jest rekomendacja PSR-0, dzięki której możliwe stało się bezproblemowe łączenie różnych bibliotek i narzędzi w ramach jednego projektu z wykorzystaniem wspólnego autoloadera. [PG14]

7.2 Istniejące rozwiązania

Autorzy zidentyfikowali 22 projekty frameworków aplikacji webowych dla PHP, dostępnych na różnych licencjach Open Source. [CW13, Roz10, PF12]

W pierwszej kolejności odrzucone zostały te projekty, które wymagają wersji PHP niższej niż 5.3, co oznacza, że nie wykorzystują przestrzeni nazw. Odrzucone zostały również projekty, które wymagają PHP 5.3 (lub nowszego), jednak nie używają przestrzeni nazw lub nie spełniają PSR-0. W kolejnym kroku zostały wyeliminowane frameworki, które były na wczesnym etapie rozwoju.

Do badań zostały zakwalifikowane dwa frameworki: Symfony w wersji 2 oraz Zend, również w wersji 2. Obie implementacje mają następujące wspólne cechy:

- zgodne są z PSR-0 (również kod aplikacji),
- zbudowane są ze stosunkowo niezależnych komponentów,
- występują w stabilnej wersji,
- mają rozbudowaną dokumentację w postaci podręcznika oraz wykazu API,
- mają dużą społeczność użytkowników.

7.3 Specyfikacja wymagań dla frameworka Vero

Na podstawie obserwacji rozwoju i najnowszych trendów w środowisku programistów skupionych wokół języka PHP oraz na podstawie własnych doświadczeń wyróżniono następujące potrzeby i założenia, jakie powinien spełniać framework *Vero*:

1. Zgodność z architektonicznym wzorcem projektowym *Model-Widok-Kontroler* (MVC), który jest powszechnie implementowanym rozwiązaniem zarówno dla aplikacji webowych jak i desktopowych, w wielu różnych językach programowania. Wzorzec ten pozwala na przejrzyste modelowanie rzeczywistości oraz procesów biznesowych.
2. Podobnie jak wytypowane rozwiązania konkurencyjne, aplikacja budowana w *Vero* powinna wykorzystywać wzorzec projektowy wstrzykiwania zależności (DI). Dzięki temu można uniknąć stanu globalnego aplikacji, co z kolei przekłada się na łatwiejsze tworzenie testów jednostkowych poszczególnych komponentów oraz większą przejrzystość powiązań pomiędzy nimi. Komponent obsługujący wstrzykiwanie zależności powinien być maksymalnie wydajny, ponieważ operacje pobierania obiektów z kontenera DI wykonywane są bardzo często w wielu miejscach. Z tego też powodu w *Vero* konfiguracja kontenera DI powinna odbywać się poprzez kod PHP oraz implementację odpowiednich klas abstrakcyjnych i interfejsów. Jest to rozwiązanie zupełnie inne niż w *Symfony* czy *Zendzie*, gdzie konfiguracja odbywa się przez odpowiednie pliki konfiguracyjne.
3. Kod frameworka powinien być zgodny ze wszystkimi opublikowanymi do tej pory rekomendacjami PSR, natomiast na kodzie aplikacji *Vero* powinien wymuszać zgodność z PSR-0. Wymuszenie zgodności z pozostałymi

PSR w kodzie aplikacji nie jest możliwe, ponieważ dotyczą one formatowania kodu wewnątrz plików, a nie zasad organizacji samych plików.

4. Kod samego frameworka powinien funkcjonować jako zbiór komponentów (mniejszych bibliotek) tworzących wspólną bibliotekę. Dopiero przykładowa aplikacja, tzw. *sandbox*, powinna zawierać propozycję organizacji plików źródłowych i statycznych, kopie innych bibliotek i bibliotekę komponentów *Vero*.
5. Doświadczenie pokazuje, że najbardziej żmudną czynnością w podejściu klasycznym jest komunikacja z bazą danych. W celu przyspieszenia tego procesu framework powinien oferować mechanizm ORM (*Object-Relational Model*). W przypadku PHP może to być biblioteka *Propel* lub *Doctrine*. Ze względu na swoje właściwości oraz lepsze wsparcie społeczności, *Vero* powinien domyślnie wykorzystywać bibliotekę *Doctrine*.
6. Ponieważ natywna składania PHP nie jest wygodnym sposobem implementacji warstwy widoku aplikacji, framework powinien mieć zewnętrzny system szablonów. Spośród wielu narzędzi sugeruje się użycie biblioteki *Twig*, która jest zgodna z PSR-0 oraz, jak wynika z doświadczenia autorów, ma zadowalającą wydajność. [Ret11]
7. *Vero* powinien oferować komponent obsługi list stronicowanych i sortowanych. Podczas wykorzystania tego komponentu wszystkie możliwe opcje konfiguracyjne powinny przyjmować wartości domyślne tak, aby programista musiał w najprostszym przypadku zdefiniować jedynie nazwę encji, która ma być wyświetlana na listingu. Rozwiązanie takie wzorowane jest na podejściu frameworka *Ruby On Rails*.
8. Dostępne powinny być również współpracujące ze sobą komponenty walidacji i obsługi formularzy, które pozwolą na centralną konfigurację kodu HTML, implementację warunków walidacji oraz logiki wszystkich formularzy.
9. Komponent routingu, konieczny w każdym frameworku aplikacji webowych, powinien pozwalać na dopasowanie dowolnej ścieżki adresu URL do akcji wraz ze wstępnym parsowaniem argumentów. Konfiguracja routingu powinna odbywać się zwięźle i czytelnie, jednak struktura klas musi pozwalać na rozszerzenie mechanizmu w przyszłości.

7.4 Vero na tle konkurencyjnych femeworków

Podobnie jak konkurencyjne frameworki dla PHP 5.3 (lub nowszego), czyli wykorzystujące przestrzenie nazw (*Symfony 2*, *Zend 2*), *Vero* funkcjonuje jako zbiór luźno powiązanych komponentów. Taka budowa charakteryzuje się tym, iż żaden z komponentów nie posiada stanu globalnego. Oznacza to unikanie metod statycznych, które mają wpływ na wszystkie instancje danej klasy utworzone w ramach jednego żądania HTTP. Projekt taki pozwala na łatwiejsze two-

rzenie testów jednostkowych kodu oraz bardziej przejrzyste powiązania pomiędzy komponentami.

Tabela 7.1. Komponenty frameworka *Vero*.

Nazwa	Wartość
ACL	Access Control List – zarządzanie uprawnieniami
Application	Aplikacja niezależna od środowiska, podstawowy kontroler wzorca MVC
Cache	Mechanizm przechowywania danych tymczasowych dla poprawy wydajności. Nakładka na Doctrine\Common\Cache
Config	Obsługa plików konfiguracyjnych (JSON, PHP)
Debug	Helpery obsługi debugowania
Dependency Injection	Podstawowa implementacja wzorca wstrzykiwania zależności
I18n	Obsługa internacjonalizacji i tłumaczeń
Loader	Implementacja autoloadera zgodna z PSR-0
Log	Implementacja loggera systemowego zgodna z PSR-3
Routing	Obsługa adresów URL
UI	Komponenty obsługi formularzy oraz listingów
Validate	Sprawdzanie poprawności danych wprowadzanych przez użytkowników
Vendor	Klasy pomostowe do bibliotek *Twig* oraz *Doctrine*
View	Warstwa widoku wzorca MVC
Web	Komponenty/klasy związane z aplikacjami webowymi: żądanie i odpowiedź HTTP, sesja, autoryzacja użytkowników, kontroler współpracujący z routerem.

Aktualnie na *Vero* składają się komponenty wymienione w tabeli 7.1. Ponadto powstały podstawowe testy jednostkowe dla następujących komponentów:

- ACL,
- Loader,
- Routing,
- Validate,
- Web\Request (dla klasy, która przechowuje stan pojedynczego żądania HTTP).

W tabeli 7.2. przedstawiono porównanie wybranych elementów i właściwości poszczególnych rozwiązań. Frameworki *Zend* i *Symfony* mają mniejszą zgodność z rekomendacjami PSR, ponieważ założenia i pierwsze ich implemen-

tacje powstawały w czasie, gdy zaprezentowana i przegłosowana była tylko rekomendacja PSR-0. *Symfony* ma zgodność z PSR-3 dzięki wykorzystaniu w standardowej dystrybucji biblioteki *Monolog*.

Tabela 7.2. Porównanie wybranych elementów frameworków. Pogrubieniem została zaznaczona opcja zalecana lub proponowana jako pierwsza w oficjalnej dokumentacji.

	Symfony	**Zend**	**Vero**
Pliki konfiguracyjne	**YAML**, XML, PHP	PHP	**PHP**, JSON
Zgodność z PSR	PSR-0, PSR-3, PSR-4	PSR-0, PSR-4	PSR-0, PSR-1, PSR-2, PSR-3, PSR-4
System szablonów	***Twig***, PHP	PHP	*Twig*
Baza danych	***Doctrine***, *Propel*	Zend\Db	*Doctrine*
Format konfiguracji *Doctrina*	**Adnotacje**, YAML, XML	*Nie dotyczy*	**XML**, Adnotacje, YAML
Konfiguracja routingu	**YAML**, XML, PHP	PHP	XML
Mechanizm modułów	Tak (bundle)	Tak	Nie
Architektoniczny wzorzec projektowy	MVC	MVC	MVC
Wstrzykiwanie zależności (DI)	Tak	Tak	Tak
Konfiguracja DI	Pliki konfig. (**YAML**, XML, PHP) lub kod PHP	Pliki konfiguracyjne (PHP)	Kod PHP

7.5 Badania porównawcze frameworków

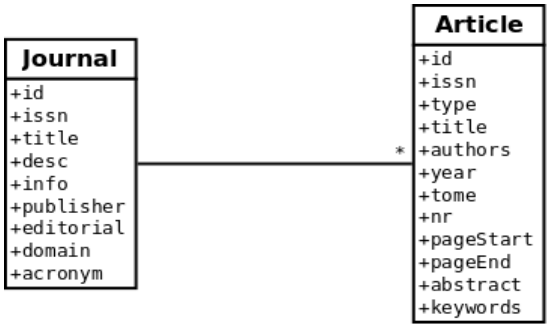
7.5.1 Opis wzorcowej aplikacji

Należy pamiętać, że sama kolekcja komponentów frameworka nie stanowi jeszcze podstawy do zbudowania własnej aplikacji. Dlatego też *Vero*, podobnie jak *Zend* czy *Symfony*, ma przykładową aplikację (*sandbox*) i w takiej postaci może być dystrybuowany. *Sandbox* składa się z propozycji struktury podstawowych plików i katalogów, konfiguracji kontenera dla wzorca wstrzykiwania zależności, kodu inicjalizującego autoloader i aplikację oraz przykładowej akcji

„Hello World”. Dystrybucja powinna zawierać również informację o zewnętrznych bibliotekach (np. w postaci pliku konfiguracyjnego dla narzędzia *Composer*) lub kod samych bibliotek. [VA14]

Ponieważ znane autorom benchamarki służyły głównie do testów stron „Hello World”, do realizacji badań zaimplementowana została autorska aplikacja, która w każdym z frameworków (*Symfony, Zend, Vero*) oferowała dokładnie te same funkcjonalności. Na każdym etapie implementacja przebiegała zgodnie z oficjalną dokumentacją w postaci podręcznika. [Pot13, GS13] Jeśli dokumentacja pozwalała na wybór jednego z kilku rozwiązań, wybierana była droga prezentowana jako zalecana lub jako pierwsza przedstawiona w dokumentacji.

Aplikacja łączy się z bazą danych, w której przechowywane są dwie encje, przedstawione na rysunku 7.1.



Rysunek 7.1. Encje wzorcowej aplikacji.

W bazie danych, która została wykorzystana podczas badań znajdowały się 4973 rekordy w tabeli *Article* oraz 11 rekordów w tabeli *Journal*.

Aplikacja ma następujące funkcjonalności:

- *index*: wyświetlanie statycznej strony HTML (bez połączenia z bazą danych),
- *empty*: wyświetlenie pustej strony (bez wykorzystania systemu szablonów),
- *journals*: wyświetlenie listy wszystkich czasopism,
- *articles*: wyświetlenie stronicowanej listy artykułów,
- *article*: wyświetlenie szczegółów danego artykułu,
- *edit*: wyświetlenie, obsługa oraz walidacja formularza edycji danego artykułu.

7.5.2 Stanowiska badawcze

W celu automatyzacji badania wydajności każdej z zaimplementowanych aplikacji, utworzono skrypt w języku PHP działający w środowisku konsolowym systemu Linux. Skrypt ten z wykorzystaniem narzędzia konsolowego *Siege* odpytuje kolejno każdy z zaimplementowanych adresów URL w każdej z aplikacji. Autorzy wybrali *Siedge*, gdyż w przeciwieństwie do popularnego *Apache Benchmark* pozwala na przetestowanie listy URL w jednym wywołaniu.

Jeśli dany adres przyjmuje dodatkowe parametry, skrypt tworzy listę adresów URL z losowym parametrem z prawidłowego zakresu. Takie adresy dotyczą akcji wyświetlającej dany artykuł i formularz edycji artykułu (losowe ID artykułu) oraz akcji wyświetlającej stronicowany listing (losowy numer strony). Wygenerowana lista jest następnie przekazywana do narzędzia *Siege*, które w takim przypadku odpytuje losowe adresy z listy.

Narzędzie *Siege* uruchamiane było z następującymi parametrami:

`-b -c 4 -t 5s`,

co oznacza kolejno:

— `b`: tryb benchmarku, odpytywanie bez przerw,
— `c 4`: odpytywanie jednocześnie przez cztery klienty,
— `t 5s`: badanie dla danego adresu URL prowadzone przez 5 sekund.

Dzięki opcji `c`, symulującej odpytywanie przez kilka klientów jednocześnie, podczas badania zaobserwować można było pełne wykorzystanie wszystkich rdzeni procesora, gdzie zrównoleglenie operacji wykonywane było przez serwer WWW *Apache*. Doświadczalnie stwierdzono też, że dalsze zwiększanie wartości tego parametru nie poprawia wyników, a jedynie pogarsza w niewielkim stopniu dla każdej z implementacji. Należy nadmienić, że nie chodziło w badaniu o postawienie serwera WWW w stan krytyczny, lecz stworzenie frameworkom takich samych warunków pracy.

Dzięki skryptowi każde badanie zostało wykonane wielokrotnie, a wynikiem była średnia liczba obsłużonych w ciągu sekundy żądań HTTP przez każdą z zaimplementowanych akcji kontrolera. Wyniki nie uwzględniają wpływu łączy sieciowych, gdyż eksperymenty były prowadzone z wykorzystaniem *localhosta*.

Badania zostały uruchomione w trzech środowiskach, które są typowe dla rozwiązań niskobudżetowych:

Localhost:

— laptop Dell Vostro 3300,
— procesor Intel Core i5-2410M @ 2.30GHz (2 rdzenie z hyper-threading / 4 rdzenie wirtualne),
— 4 GB pamięci RAM,

— Ubuntu Linux 13.10 Desktop,
— PHP 5.5.3, Apache 2.4.6, MariaDB 10.0.11.

Biuro:

— HP ProLiant MicroServer:
— procesor AMD Turion II Neo N40L @ 1.50 GHz (2 rdzenie),
— 4 GB pamięci RAM,
— Ubuntu Linux 12.04 Server (64 bits),
— PHP 5.5.10, Apache 2.4.6, MariaDB 10.0.10.

Virtual Private Server (OVH.PL):

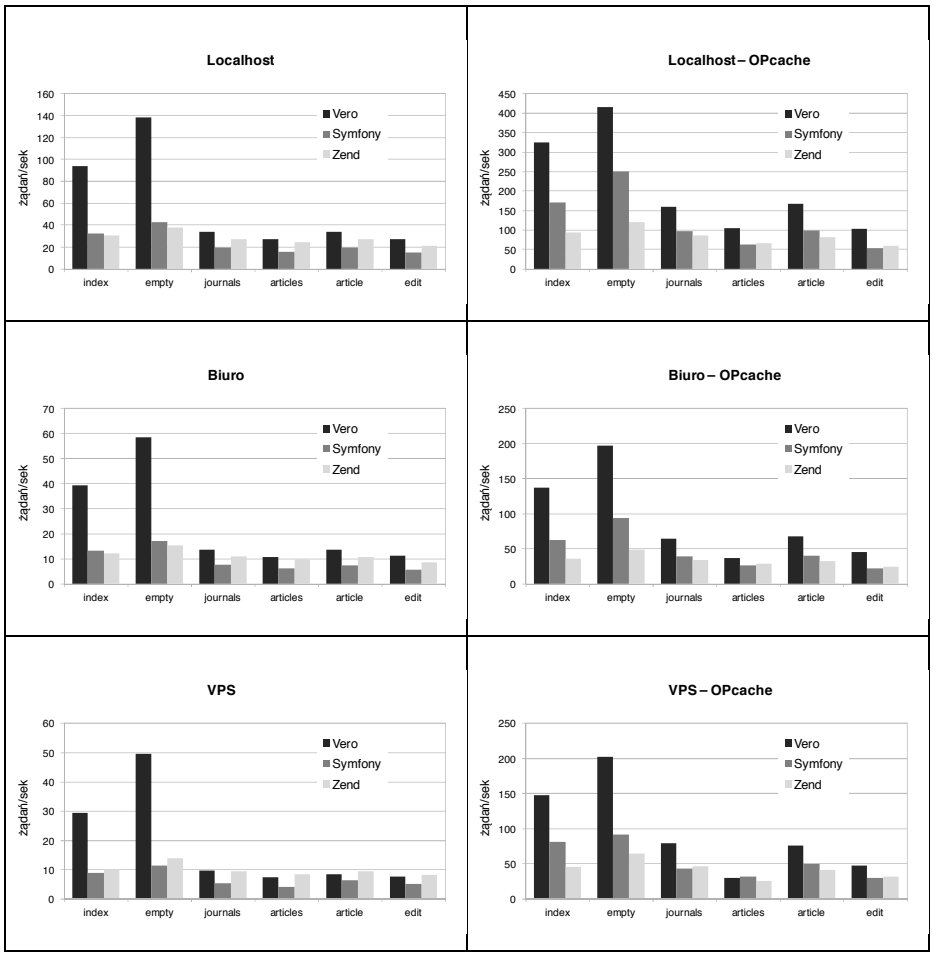
— VPS 2013 Classic 3,
— 1 vCPU,
— 2 GB pamięci RAM,
— Ubuntu Linux 12.04 Server (64 bits),
— PHP 5.5.9, Apache 2.4.7, MariaDB 10.0.9.

We wszystkich konfiguracjach badanie zostało przeprowadzone przy włączonym i wyłączonym rozszerzeniu *OPcache*, ponieważ prekompilacja jest silnie rekomendowana dla frameworka *Symfony* i biblioteki *Doctrine*.

Drugim zagadnieniem jest obliczenie liczby wszystkich plików z kodem PHP, które zostały dołączone i wykorzystane w danym przebiegu skryptu. Możliwe jest to dzięki dostępności funkcji get_included_files() dostępnej w standardowym API parsera PHP. Funkcja ta zwraca tablicę zawierającą listę wszystkich załadowanych plików z kodem źródłowym. Wywołanie tej funkcji następuje po zakończeniu wykonywania skryptu, a wynik jest zapisywany do pliku logu, skąd można wydobyć tę informację do dalszych obliczeń.

7.6 Analiza wyników badań

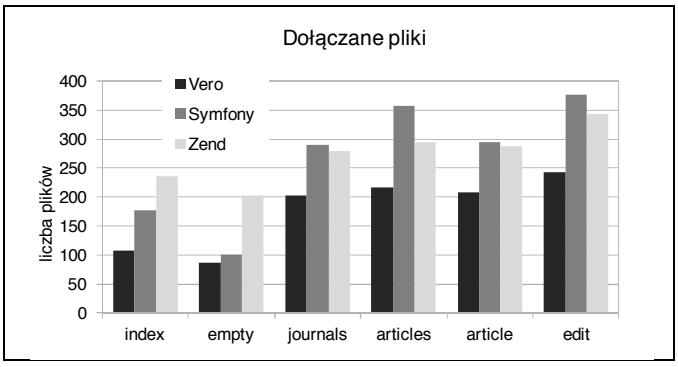
Rysunek 7.2. pokazuje liczbę obsłużonych żądań w ciągu jednej sekundy na różnych konfiguracjach. Wykresy w lewej kolumnie prezentują wydajność poszczególnych konfiguracji z wyłączonym rozszerzeniem *OPcache*. Wykresy w prawej kolumnie pokazują jak duży wpływ ma rozszerzenie *OPcache* na szybkość wykonywania aplikacji. Pomimo zachowania proporcji pomiędzy poszczególnymi frameworkami zauważyć można trzykrotne, a nawet czterokrotne zwiększenie liczby obsłużonych żądań w przypadku, gdy rozszerzenie *OPcache* jest włączone.



Rysunek 7.2. Wyniki badań wydajnościowych

Można też zauważyć, że w większości przypadków framework *Symfony* pozostaje szybszy, niż Zend, dopóki nie jest używana biblioteka *Doctrine* (akcje niewykorzystujące połączenia z bazą danych). Implementacja *Vero* pozostaje najszybsza w każdym przypadku, jednak na szczególną uwagę zasługuje duża wydajność w przypadku akcji wyświetlającej pustą stronę. Dzieje się tak dlatego, iż w *Vero* każda możliwa operacja wykonywana jest zgodnie z wzorcem leniwego inicjowania. Oznacza to, iż w akcjach niekorzystających z komponentów, np. bazy danych, sesji czy internacjonalizacji, komponenty te nie są ani ładowane, ani inicjalizowane. W podstawowym przebiegu skryptu ładowany jest tylko komponent ogólnej konfiguracji systemu i routingu.

Pewnym zaskoczeniem jest najmniejsza wydajność (w przypadku każdego frameworka) akcji generującej formularz. Z poziomu przeglądarki/klienta HTTP formularz, to statyczny kod HTML z wstawionymi w pewnych miejscach danymi z jednej encji z bazy danych, więc nie powinien on być znacznie mniej wydajny od akcji article. Jednak nowoczesne frameworki (również *Vero*), przechowują opis struktury formularza w postaci hierarchii obiektów, komponentów i helperów, przez co samo wygenerowanie ostatecznego kodu HTML wymaga pewnych zasobów. Jest to nieunikniony koszt wygody programisty, który może dzięki takiemu rozwiązaniu łatwo tworzyć i modyfikować globalnie wszystkie formularze w aplikacji.



Rysunek 7.3. Liczba dołączanych plików

Rysunek 7.3. przedstawia liczbę dołączanych plików z kodem źródłowym podczas obsługi jednego żądania HTTP. Wyniki te są niezależne od stanowiska badawczego, jednak można zauważyć prawidłowość, iż liczba dołączanych plików powiązana jest z ogólną wydajnością. Oczywiście taka zależność nie może być prawidłowa we wszystkich przypadkach, ponieważ w niektórych akcjach wykonywanych przez aplikacje webowe dokonywane mogą być obliczenia lub operacje wymagające znacznie większych zasobów pamięciowych i czasowych, inaczej mówiąc o większej złożoności obliczeniowej.

7.7 Podsumowanie

Przeprowadzone badania pokazują, że możliwe jest zbudowanie frameworka w języku PHP, który jednocześnie jest zgodny z najnowszymi trendami w środowisku oraz pozwala na zachowanie wystarczającej wydajności nawet na najsłabszych serwerach webowych.

Konkurencyjnymi projektami Open Source, spełniającymi podobne założenia biznesowe, są w tej chwili projekty *Symfony* w wersji 2 oraz *Zend Framework*, również w wersji 2.

Zbudowany i opisany framework *Vero* jest zgodny z rekomendacjami PSR-0..4, wykorzystuje bibliotekę *Doctrine* jako narzędzie ORM oraz system szablonów *Twig* w warstwie prezentacji. Sprawia to, że budowanie aplikacji w oparciu o taki zestaw narzędzi jest równie proste i szybkie, jak w konkurencyjnym frameworku *Symfony*.

Dzięki rezygnacji z mechanizmu bundli (*Symfony*) i modułów (*Zend*), wykorzystywaniu wzorca wstrzykiwania zależności i leniwego ładowania wszystkich komponentów, *Vero* pozostał najszybszym rozwiązaniem spośród trzech porównywanych frameworków. Uzyskana końcowa wydajność dla typowych operacji aplikacji webowej, mierzona jako liczba obsłużonych żądań na sekundę, jest nawet trzykrotnie większa niż dla frameworka *Zend*.

W porównaniu wydajności rozwiązań zastosowanych w *Symfony* i *Zend* lepiej wypada framework *Symfony* dla akcji, w których nie jest używane połączenie z bazą danych, za pomocą biblioteki *Doctrine*. *Zend* wydajniejszy jest dopiero w akcjach korzystających z bazy danych. *Vero*, dzięki zaoszczędzonym zasobom w innych miejscach, mimo użycia *Doctrina*, pozostaje wydajniejszy niż *Zend*.

7.7.1 Rekomendacje

Jak pokazują wykonane badania, bardzo duże znaczenie dla czasu wykonywania aplikacji webowych ma ładowanie plików z dysku. Proces ten można przyspieszyć dzięki tzw. akceleratorom (np. *OPcache*). Jego aktywowanie na serwerze zalecane jest przez dokumentację wielu nowych projektów tworzonych w środowisku PHP. Włączenie takiego rozszerzenia jest konieczne, z powodu dużej abstrakcji kodu narzędzi i bibliotek, które zdejmują z programisty aplikacji wiele powtarzalnej pracy.

Dzięki wykorzystaniu biblioteki ORM, wygodnego systemu szablonów oraz zwięzłego API frameworka, programiści mogą się skupić na odpowiednim dopracowaniu funkcjonalności widocznych dla użytkownika końcowego. Nie muszą się natomiast zajmować takimi zagadnieniami jak kwestie bezpieczeństwa, pisanie zapytań SQL czy tworzenie kodu HTML do obsługi formularzy.

Takie założenia przekładają się oczywiście na spadek wydajności, jednak przykład frameworka *Vero* pokazuje, że można użyć wymienionych udogodnień zachowując wysoką wydajność w innych aspektach, co całościowo przekłada się na zadowalającą wydajność.

Framework *Vero* udostępniony został jako wolne oprogramowanie (*Open Source*) [VF14]. W oparciu o niego zbudowany został komercyjny system zarządzania treścią *VeroCMS*, który spełnia takie same założenia względem wydajności i interoperacyjności [VC14]. *VeroCMS* już kilkakrotnie znalazł zastosowanie komercyjne. Obecnie realizowany jest przy jego pomocy projekt dofinansowany przez UE w programie 8.1. [LP14]

7.7.2 Kierunki rozwoju

Pomimo tego, że framework *Vero* jest już w pełni funkcjonalny dla pewnej grupy projektów, wymaga dużo pracy organizacyjnej. Przede wszystkim więcej komponentów powinno zostać pokrytych testami jednostkowymi. Równocześnie należałoby tworzyć dokumentację w postaci podręcznika opisującego sposób użycia poszczególnych funkcjonalności z punku widzenia programisty aplikacji.

W kwestii dalszej optymalizacji wydajności samego frameworka można:

- zwiększyć wykorzystanie mechanizmów cache, szczególnie jeśli dostępne jest rozszerzenie *OPcache*,
- korzystając z narzędzia *Xdebug* oraz jego funkcjonalności profilowania wyodrębnić procedury pochłaniające najwięcej czasu i zasobów, a następnie dokonać ich dodatkowej optymalizacji,
- w przypadku, gdy optymalizacja samego kodu PHP nie przyniesie już dalszych efektów, część kluczowych funkcjonalności zaimplementować jako rozszerzenie dla parsera PHP, napisane w języku C, podobnie jak ma to miejsce w przypadku biblioteki *Twig*.

Bibliografia

[CW13] Comparison of web application frameworks: PHP. *Wikipedia.org*, 2013. http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks #PHP

[Ret11] Rethans D. Twig extension. *Derickrethans.nl*, 2011. http://derickrethans.nl/twig-extension.html

[Pot13] Potencier F. i inni. Symfony: The Book. *Symfony.com*, 2013. http://symfony.com/doc/current/book/index.html

[Roz10] Rozpara G. Frameworki PHP: przegląd pięciu najpopularniejszych narzędzi dla programistów WWW. *Webhosting.pl*, 2010. http://webhosting.pl/Frameworki.PHP.przeglad.pieciu.najpopularniejszych. narzedzi.dla.programistow.WWW

[GS13] Getting Started with Zend Framework 2. *Zend.com*, 2013. http://framework.zend.com/manual/2.2/en/user-guide/overview.html

[LP14] Literacka Platforma Wydawnicza – serwis webowy. *Sunbook.pl*, 2014.

http://sunbook.pl/
[PF12] PHP Frameworks. *Phpframeworks.com*, 2012. http://www.phpframeworks.com/
[PG14] PHP Framework Interop Group. PHP Standard Recommendation. *Php-fig.org*, 2014. http://www.php-fig.org/psr/
[VA14] "Hello World" application for Vero Framework. *Github.com*, 2014 https://github.com/minchal/vero-hello-app
[VC14] VeroCMS – serwis demonstracyjny. *Si2.pl*, 2014. http://demo.si2.pl/
[VF14] Vero – Web Application Framework for PHP 5.4+. *Github.com*, 2014 https://github.com/minchal/vero

CZĘŚĆ III
JAKOŚĆ OPROGRAMOWANIA

Rozdział 8

Symulacyjne badanie jakości oprogramowania w zwinnym procesie produkcji

W niniejszym rozdziale poruszono tematykę badania wpływu odpowiedniego doboru personelu do zespołów wytwarzających oprogramowanie w oparciu o podejście zwinne na jakość powstającego produktu. W tym celu opracowano model symulacyjny, który - czerpiąc z osiągnięć nauk humanistycznych i inżynierii oprogramowania - obejmuje swoim zakresem obszary: zarządzania przepływem pracy, detekcją i naprawą błędów oraz budową zespołów w projekcie zarządzanym wg metodyki Scrum. Zbudowany model pozwala m.in. zbadać jak dopasowanie cech osobowości poszczególnych członków zespołu do różnych ról i procesów wpływa na jakość produktu w którego tworzenie są oni zaangażowani. Przeprowadzone przy jego użyciu eksperymenty symulacyjne dają ciekawe spojrzenie na aspekty dotyczące zarządzania zasobami ludzkimi i ich wpływu na przebieg projektu oraz jakość otrzymanego w rezultacie oprogramowania. Wykorzystanie zaproponowanego modelu w praktyce może przyczynić się do lepszego zarządzania projektami prowadzonymi w zgodzie z filozofią Agile poprzez dobór kadry charakteryzującej się odpowiednimi zdolnościami zarówno technicznymi jaki interpersonalnymi.

Współczesne projekty informatyczne w coraz większej części prowadzone są w oparciu o metodyki zwinne. Pojęcie zwinnego procesu produkcji programowania zostało zaproponowane w 2001 r. w Agile Manifesto. Ich zastosowanie ma z definicji zapewnić szybkie dostarczanie klientowi kolejnych wersji funkcjonującego oprogramowania charakteryzującego się określoną funkcjonalnością i jakością.

Generalnie metodyka (zwana zwinną) oparta jest na zdyscyplinowanym zarządzaniu projektem, które zakłada częste inspekcje wymagań i rozwiązań wraz z procesami adaptacji (zarówno specyfikacji jak i oprogramowania). Metodyka ta najczęściej znajduje zastosowanie w małych zespołach programistycznych, w których nie występuje problem komunikacji, przez co nie trzeba tworzyć rozbudowanej dokumentacji kodu. Kolejne etapy wytwarzania oprogramowania zamknięte są w iteracjach, w których za każdym razem przeprowadza się testowanie wytworzonego kodu, zebranie wymagań, planowanie rozwiązań itd. Metoda nastawiona jest na szybkie wytwarzanie oprogramowania wysokiej jakości.

Celem niniejszego rozdziału jest przedstawienie zaproponowanego modelu symulacyjnego, który obejmuje swoim zakresem obszary: zarządzania przepływem pracy, detekcją i naprawą błędów oraz budową zespołów w projekcie zarządzanym wg metodyki Scrum oraz systemu symulacyjnego, a następnie zbadanie przy jego użyciu jak dopasowanie cech osobowości poszczególnych członków zespołu do różnych ról i procesów wpływa na jakość produktu,

w którego tworzenie są oni zaangażowani. Dodatkowym celem jest jego adaptacja do badania jakości oprogramowania w zwinnym procesie produkcji.

Model koncepcyjny systemu symulacyjnego oparty został na modelu matematycznym zapisanym w postaci programu komputerowego, w skład którego wchodzą moduły oprogramowania umożliwiające odtwarzanie w przestrzeni wirtualnej i pomiar konkretnych metryk jakości oprogramowania w zwinnym procesie produkcji.

Przeprowadzone przy jego użyciu eksperymenty symulacyjne dają interesujące spojrzenie na aspekty dotyczące zarządzania zasobami ludzkimi i ich wpływu na przebieg projektu oraz jakość otrzymanego w rezultacie oprogramowania. Wykorzystanie zaproponowanego modelu w praktyce może przyczynić się do lepszego zarządzania projektami prowadzonymi w zgodzie z filozofią Agile poprzez dobór kadry charakteryzującej się odpowiednimi zdolnościami zarówno technicznymi jak i interpersonalnymi.

8.1 Jakość oprogramowania

W inżynierii oprogramowania używa się kilku różnych definicji jakości. Według Kana [KA02] najpopularniejsze z nich to: zaproponowana przez Crosbiego [CR79] ”zgodność z wymaganiami”, zaproponowana przez Gryna [GR01] ”przydatność do użytku” oraz zasugerowany przez samego Kana [KA02] ”brak defektów w produkcie”. Warto również wspomnieć o normie ISO/IEC 9126-1:2001, która definiuje model jakości oprogramowania. Na model ten składają się następujące charakterystyki [JK04]:

- funkcjonalność - dopasowanie do jawnie lub niejawnie wyrażonych potrzeb klienta;
- niezawodność - zdolność do poprawnej pracy przez wymagany czas w określonych warunkach;
- użyteczność - zrozumiałość i łatwość użytkowania;
- efektywność - wydajność działania i stopień zużywania zasobów;
- konserwowalność - koszt wykonywania modyfikacji w systemie;
- przenośność - zdolność systemu do zaadoptowania w innym środowisku.

Najbardziej wyczerpującą i wszechstronną definicję jakości oprogramowania zdecydowanie zawiera norma ISO/IEC 9126-1:2001. Jednak z uwagi na tematykę niniejszego rozdziału, czyli stosowanie modeli symulacyjnych w zapewnianiu jakości oprogramowania, zdecydowanie najwygodniejsza będzie następująca definicja: „jakość oprogramowania jest to ogół cech i/lub właściwości

produktu programowego oraz jego procesu wytwórczego, które wpływają na zdolności spełniania przez nich stwierdzonych i przewidywanych wymagań oraz oczekiwań jakościowych klienta".

Jakość oprogramowania stanowi zagadnienie wieloaspektowe, związane z tym, że jego twórcy muszą sprostać jawnym i ukrytym potrzebom klientów, oczekiwaniom komitetu sterującego, partnerów i audytorów. Ponadto, produkt programowy musi być konkurencyjny i przede wszystkim opłacalny, stworzony w terminie i z uwzględnieniem ograniczeń budżetowych.

Jakość oprogramowania jest jednym z fundamentów podejścia zwinnego. Modele jakości, specjalne techniki tworzenia kodu, wielopoziomowe testowanie oraz standaryzacja rozwiązań to tylko niektóre techniki jej osiągnięcia. Zaangażowanie zdopingowanego zespołu, poczucie własności i równocześnie współdzielenie kodu jest dodatkowym czynnikiem wzrostu jego jakości, w tym i podatności na modyfikacje.

8.2 Przegląd działań zorientowanych na jakość oprogramowania

Inżynieria oprogramowania kładzie niebagatelny nacisk na działania mające zapewnić wysoką jakość produktu, jakim jest oprogramowanie i/lub system informatyczny [FL06, KA02]. Działania te ujawniają się w wielu obszarach procesu wytwarzania oprogramowania, m.in. w:

- testowaniu,
- inspekcjach oprogramowania,
- modelach przyrostu niezawodności,
- bezpieczeństwie systemów informatycznych,
- modelach i metrykach oprogramowania.

Testowanie jest najobszerniejszym działaniem związanym z zapewnianiem wysokiej jakości systemu informatycznego pod względem ilości angażowanych zasobów. Zdarza się, że koszty testowania przekraczają nawet 50% całkowitych kosztów wytworzenia systemu informatycznego [MS01].

Wraz z pojawieniem się zwinnych metodyk wytwarzania oprogramowania, zaproponowano nowe podejście do testowania oprogramowania. W podejściu tym odwrócono kolejność aktywności w procesie wytwarzania oprogramowania. W programowaniu sterowanym przez testy (ang. *test driven development*) zaczynać należy od przygotowania testów, następnie wykonuje się implementacje, a dopiero na końcu należy się zastanowić nad architekturą (modelem) systemu, którą tworzy się w ramach faktoryzacji.

W badaniu jakości oprogramowania z wykorzystaniem podejścia symulacyjnego bardzo duży nacisk kładzie się na adekwatność modelu symulacyjnego,

który stanowi próbę całościowego podejścia do symulacji procesów wytwórczych w projektach prowadzonych w oparciu o metodykę zwinną, np. Scrum.

Inspekcja oprogramowania, często nazywana również przeglądem formalnym, to statyczna analiza kodu lub dokumentacji projektowej, która ma na celu wykrycie, identyfikację i usuniecie defektów oraz formalne potwierdzenie jakości produktu. Inspekcja jest zazwyczaj realizowana w postaci trwającego maksymalnie kilka godzin spotkania, na którym autor dokumentu lub fragmentu programu relacjonuje wykonane przez siebie modyfikacje. Reszta osób biorących udział w spotkaniu ma natomiast za zadanie zrozumieć i zweryfikować ich poprawność.

Modele przyrostu niezawodności to modele, które na podstawie historii odnajdywania defektów w procesie testowania szacują liczbę defektów, które jeszcze nie zostały odnalezione. Zazwyczaj modele te opierają się na rozkładzie wykładniczym [KA02]. Do najpopularniejszych modeli przyrostu niezawodności należą: Model Jelinskiego-Morandy, Model Littlewooda, Model niedoskonałego debugowania Goela-Okumoto, Niejednorodny model procesu Poissona Goela-Okumoto oraz Logarytmiczny model Poissona czasu wykonania Musy-Okumoto[KA02]. Rozszerzeniem tych modeli jest bazujący na rozkładzie Weibulla model Rayleigha. Model Rayleigha, w odróżnieniu od klasycznych modeli przyrostu niezawodności, wykorzystuje informacje nie tylko z fazy testów, ale z całego procesu wytwarzania oprogramowania. Dzięki temu można szacować liczbę defektów ukrytych w systemie na wcześniejszych etapach prac oraz uzyskać precyzyjniejsze oszacowanie w fazie testów.

Istnieje grupa systemów informatycznych, w których bezpieczeństwo odgrywa krytyczna role. Są to systemy, których awaria lub nieprawidłowe działanie może doprowadzić do czyjejś śmierci, uszczerbku na zdrowiu lub poważnych strat finansowych. Wyróżniamy dwa rodzaje analiz bezpieczeństwa: probabilistyczne oraz deterministyczne. Stosowane są m.in. następujące metody analiz [CR79]:

- diagramy przyczynowo- skutkowe,
- analiza rodzajów i skutków uszkodzeń,
- analiza drzew zdarzeń,
- analiza drzew niezdatności.

8.3 Zwinny proces produkcji oprogramowania

Zwinny proces produkcji oprogramowania stanowi zatem odpowiedź na:

- zbyt restrykcyjne procesy,

— „niepotrzebne" tracenie sił i środków,
— ograniczenie elastyczności w działaniach,
— zbyt długotrwałe oczekiwanie na rezultat projektu przy zmieniających się wymaganiach.

Zwinny proces produkcji oprogramowania oparty jest na zdyscyplinowanym zarządzaniu projektem, które zakłada częste inspekcje wymagań i rozwiązań wraz z procesami adaptacji (zarówno specyfikacji jak i oprogramowania) [AB02]. Kolejne etapy wytwarzania oprogramowania zamknięte są w iteracjach, w których za każdym razem przeprowadza się testowanie wytworzonego kodu, zebranie wymagań, planowanie rozwiązań itd. Skład zespołów jest zazwyczaj wielofunkcyjny oraz samozarządzalny, bez zastosowania jakiejkolwiek hierarchii organizacyjnej. Członkowie zespołu biorą odpowiedzialność za zadania postawione w każdej iteracji i sami decydują jak osiągnąć postawione cele [AB02]. Metoda nastawiona jest na szybkie wytwarzanie oprogramowania wysokiej jakości i bezpośrednią komunikację pomiędzy członkami zespołu, minimalizującą potrzebę tworzenia dokumentacji. Zwinność w procesach wytwarzania oprogramowania związana jest z:

— chęcią zmiany podejścia do procesu wytwórczego,
— odrzuceniem pewnych elementów formalizmu metodyk tradycyjnych,
— chęcią wykorzystania najlepszych praktyk pracy zespołowej, zwiększających jego efektywność i jakość rezultatu.

Ponadto zwinność w procesach wytwarzania oprogramowania:

— dynamizuje proces wytwarzania oprogramowania,
— wykorzystuje efekt synergii wynikający ze ścisłej współpracy z klientem w trakcie całego projektu,
— umożliwia dynamiczne sterowanie przyrostowym wytwarzaniem oprogramowania przy pomocy zmieniających się wymagań klienta.

Istotnym elementem podejścia zwinnego, silnie odróżniającego go od tradycyjnych metodyk realizacji projektów, jest sposób traktowania zespołu i organizacji jego pracy. Przyjęte zasady stałej współpracy wszystkich osób zaangażowanych w projekt oraz zwiększenia swobody pracy zespołu programistów całkowicie zmieniają strukturę organizacyjną, techniki pracy i zarządzania projektem. Zasady współpracy osób w czasie projektu zawierają bowiem następujące postulaty:

— samodzielnej organizacji zespołu,
— współodpowiedzialności całego zespołu za rezultat pracy,
— silnego zmotywowania (samo-motywowania) członków,

— zaspokojenia potrzeb członków zespołu,
— obdarzenia zespołu dużym zaufaniem.

8.4 Metryki oprogramowania

Metryki oprogramowania to miary pewnej właściwości oprogramowania lub jego specyfikacji. Norma IEEE 1061-1998 [IE98] definiuje metrykę jako ”funkcje odwzorowującą jednostkę oprogramowania w wartość liczbową. Ta wyliczona wartość jest interpretowana jako stopień spełnienia pewnej własności jakości jednostki oprogramowania.”. Metryki oprogramowania pełnią istotną rolę w niniejszym rozdziale. W związku z tym ich rola została dokładniej omówiona.

W działaniach projakościowych stosuje się bardzo różne metryki oprogramowania. Ze względu na rodzaj atrybutów (parametrów oprogramowania), jakie przy pomocy metryki są mierzone, możemy podzielić metryki na następujące typy:

— miary zewnętrzne (norma ISO/IEC 9126-2:2003) - zestaw metryk zewnętrznych służących do pomiaru charakterystyk określających zachowanie się systemu. Miary mogą być dedykowane różnym fazom cyklu życia oprogramowania w wersji wykonywalnej, np. fazie testowania lub później implementacji;
— miary wewnętrzne (norma ISO/IEC 9126-3:2003) - zewnętrzne metryki jakości służące do pomiaru charakterystyk oprogramowania będącego w postaci niewykonywanej. Może być to np. faza projektowana lub wczesnej implementacji;
— miary jakości użytkowej (norma ISO/IEC 9126-4:2004) - zbiór metryk jakości, który może zostać użyty do wygenerowania raportu technicznego zawierającego rozszerzalną listę atrybutów.

Z punktu widzenia celu niniejszej rozdziału najbardziej adekwatny będzie podział metryk z uwagi na typ artefaktu jaki opisują tj.:

— **metryki produktu** - które odzwierciedlają atrybut produktu dla stanu w jakim ten produkt znajdował się w danej chwili. W związku z tym są to metryki, które nic nie mówią o zmienności atrybutów w czasie. Przykładami takich metryk są liczba linii kodu oraz złożoność cyklometryczna McCabea.
— **metryki procesu** - obejmujące metryki uwzględniające zmienność atrybutu w czasie. Przykładem takiej metryki jest liczba defektów odna-

lezionych w zadanym okresie czy tez liczba modyfikacji wykonanych przez programistów w zadanym przedziale czasu.

- **metryki zespołu** wytwórczego oprogramowania - które odzwierciedlają charakterystyki zespołu wytwórczego wpływające na jakość wytwarzanego oprogramowani, np. produktywność zespołu, produktywność pojedynczego członka zespołu o określonej roli, liczba popełnionych błędów, odsetek błędnie zrealizowanych zadań, itp.
- **Metryki projektu.** Są to metryki, które odzwierciedlają podstawowe parametry projektu takie jak: zakres funkcjonalny (zbiór wymagań), czas realizacji projektu, budżet projektu (koszt), jakość projektu, ryzyko projektu, socjotechnika projektu.

Zbudowany przez autorów model symulacyjny wykorzystuje metryki produktu jako stałą bazę niezależną od typu przeprowadzanego badania symulacyjnego. W niektórych eksperymentach symulacyjnych baza ta jest rozszerzana o metryki procesu, zespołu oraz projektu w celu zweryfikowania wytypowanych charakterystyk zespołu i całego projektu.

8.5 Model symulacyjny badania jakości oprogramowania wytwarzanego w zwinnym procesie produkcji

Proces tworzenia modelu symulacyjnego obejmował następujące etapy:

- określenie symulowanego systemu,
- sformułowanie modelu,
- przygotowanie danych,
- zaprogramowanie modelu,
- ocena adekwatności modelu,
- planowanie eksperymentów symulacyjnych,
- wprowadzenie eksperymentów,
- interpretacja uzyskanych wyników.

Autorzy zbudowali model symulacyjny, który stanowi próbę całościowego podejścia do symulacji procesów wytwórczych w projektach prowadzonych w oparciu o metodykę Scrum. Stanowi on własną propozycję autorów i uwzględnia oprócz elementów skupionych wokół aspektów związanych przebiegiem zwinnych procesów produkcyjnych również:

- aspekt jakości tworzonego oprogramowania,
- aspekt jakości zespołu wytwórczego.
- aspekt jakości podstawowych parametrów projektu, prowadzonego zgodnie z zaleceniami metodyki zwinnej.

8.5.1 Sformułowanie modelu

Model symulacyjny zbudowany został na bazie założeń metodyki Scrum, AgileEVM oraz teorii małych grup (*ang. small group theory*) z wykorzystaniem środowiska symulacyjnego Vensim®.

Model symulacyjny bazuje na modelu matematycznym zapisanym w postaci pliku wejściowego pakietu symulacyjnego Vensim®. W jego skład wchodzi układ 121 równań różniczkowo-algebraicznych umożliwiający odtwarzanie w przestrzeni wirtualnej działań obejmujących:

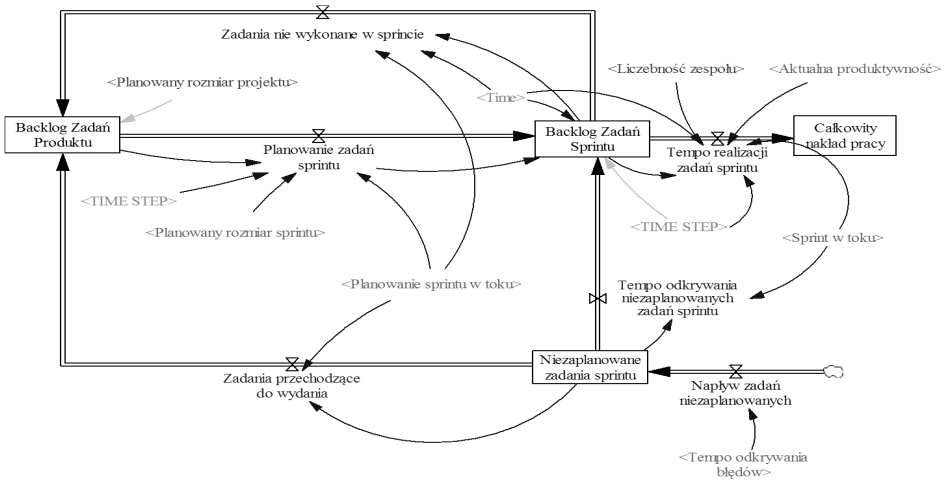
- przebieg procesu Scrum;
- planowanie i przepływ sprintu;
- analizę i ocenę jakości produktu programowego,
- badanie wybranych parametrów (charakterystyk) zespołu wytwórczego;
- badanie i ocenę procesu wytwórczego oprogramowania według ustalonych i/lub zdefiniowanych metryk,
- AgileEVM (wartość wypracowana);
- zarządzanie czasem symulacji.

W niniejszym rozdziale badanie symulacyjne jakości oprogramowania sprowadza się do odtwarzania przebiegu procesu produkcji oprogramowania w oparciu o metodykę zwinną Scrum (patrz Rysunek 8.1). Równocześnie w procesie symulacji są zbierane różnego typu charakterystyki zespołu wytwórczego (patrz Rysunek 8.2) oraz jakości procesu wytwórczego (patrz Rysunek 8.3) poprzez wykorzystanie odpowiednich funkcji środowiska symulacyjnego Vensim®.

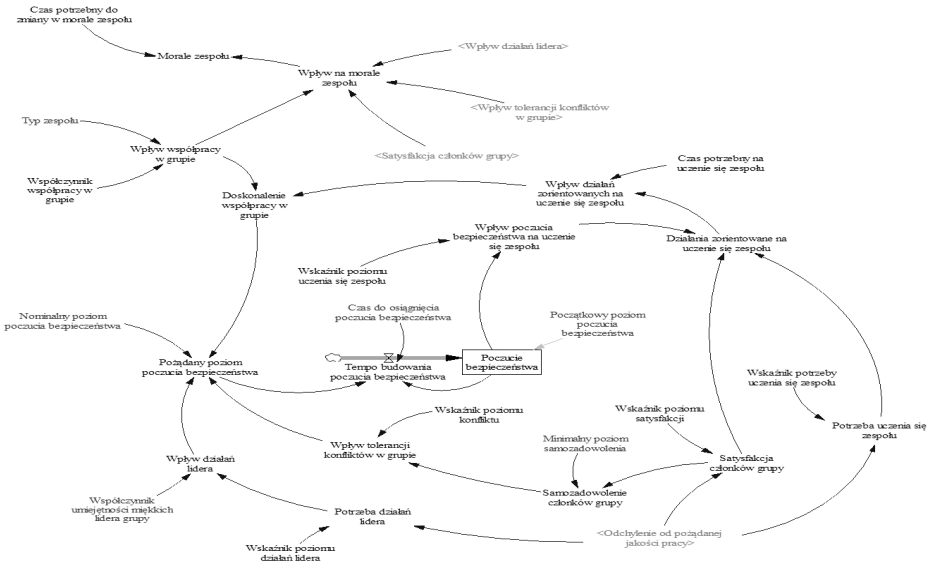
Wybrane fragmenty i szczegóły modelu, istotne z punktu widzenia rozważań podjętych w ramach niniejszego rozdziału, odzwierciedlają poniższe rysunki (Rysunek 8.1, Rysunek 8.2, Rysunek 8.3).

Zakres charakterystyk mających wpływ na dobór odpowiedniego personelu do zespołów wytwarzających oprogramowanie w oparciu o podejście zwinne w kontekście jakości powstającego produktu odzwierciedla poniższy rysunek (Rys. 8.2).

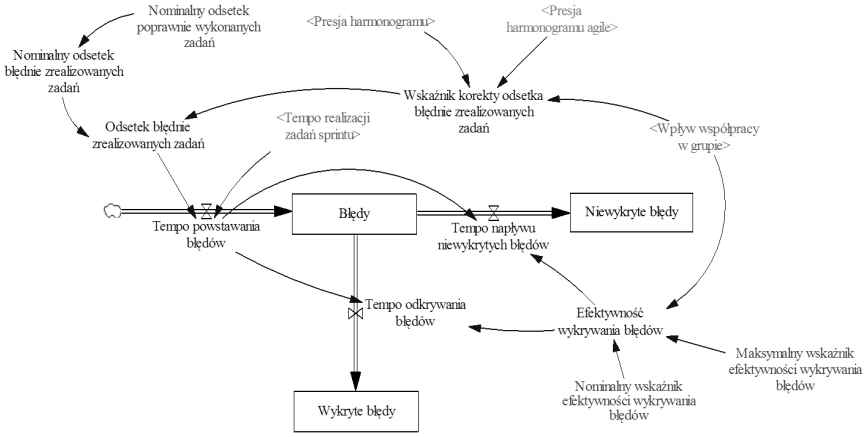
Zbiór metryk jakości oprogramowania badanych w procesie symulacji w kontekście możliwych błędów oprogramowania przedstawia poniższy rysunek (Rysunek 8.3).



Rysunek 8.1. Symulacyjny model przebiegu procesu produkcji oprogramowania w podejściu Scrum.



Rysunek 8.2. Zakres charakterystyk zespołu wytwórczego wpływających na jakość wytwarzanego oprogramowania.



Rysunek 8.3. Zakres metryk jakości oprogramowania wytwarzanego w podejściu zwinnym.

8.5.2 Ocena adekwatności modelu

W celu oceny prawdziwości zbudowanego modelu i jego zdolności do odwzorowywania badanej rzeczywistości w wybranym zakresie dokonano sformalizowanego badania adekwatności przyjętego modelu. Problematyka ta szczegółowo została opisana w [PS12]. Najogólniej rzecz ujmując polegała ona na opracowaniu testów pozwalających stwierdzić poprawność: strukturalną i behawioralną modelu symulacyjnego oraz jego odpowiedzi na zmiany reguł decyzyjnych.

Wielu autorów podnosi kwestię istotności budowania zaufania do modelu symulacyjnego przez przeprowadzenie szerokiego zakresu testów w wymiarach:

- struktury modelu - testy obejmujące m.in. weryfikację: struktury modelu, parametrów modelu, warunków ekstremalnych, granic modelu, zgodności wymiarów parametrów;
- zachowania modelu - testy obejmujące m.in. poprawność reprodukowania zachowań obiektu rzeczywistego (wraz z anomaliami), analizę wrażliwości na zmiany parametrów, zgodność statystyczną wartości na wyjściach modelu z wartościami pochodzącymi z obserwacji systemu (obiektu) rzeczywistego;
- stabilności reakcji na zmiany reguł decyzyjnych - testy obejmujące m.in. wrażliwość reguł decyzyjnych na zmianę parametrów.

Autorzy ci marginalizują równocześnie kwestię statystycznego dopasowania danych generowanych na wyjściu modelu do danych historycznych zarejestrowanych w wyniku obserwacji systemów (obiektów) w rzeczywistości. Podkreślają oni, że każdy zbiór danych wynikowych można zawsze dopasować do wymaganego poziomu dokładności, dlatego modele powinny być oceniane pod kątem ich adekwatności głównie według innych kryteriów niż dopasowanie statystyczne. W opinii autorów niniejszego rozdziału nie można do końca zgodzić się z takim stanowiskiem, z kilku niżej wymienionych powodów:

- — większość odbiorców spodziewa się istnienia formalnej metody oceny prawdziwości dostarczonych im modeli symulacyjnych;
- — pomimo, że diagnoza statystyczna dopasowania modelu dynamiki systemowej do danych uzyskiwanych z obserwacji systemów (obiektów) rzeczywistych nie jest kryterium wystarczającym do stwierdzenia, że opracowany model jest poprawny, to jej rola w procesie budowania zaufania do tegoż modelu jest nieoceniona, gdyż pozwala na częściowe sformalizowanie procesu oceny jego adekwatności;
- — dobrze zbudowany model dynamiki systemowej musi poprawnie odwzorowywać historyczne (przeszłe) zachowania systemu (obiektu) rzeczywistego na bazie dostępnych danych historycznych oraz umożliwiać równie prawidłową predykcję jego przyszłych zachowań na bazie założonych parametrów.

Ocenę adekwatności zaproponowanego modelu symulacyjnego dokonano przez zastosowanie metod statystycznych do oceny jego poprawności behawioralnej. Problem autorzy sprowadzili do opracowaniu reguły decyzyjnej, która na podstawie wyników eksperymentów przeprowadzonych na bazie modelu pozwoliła wyznaczyć stopień jego "prawidłowości" na podstawie dopasowania generowanych przy jego użyciu wartości do posiadanych danych historycznych [PS13b]. Do oceny błędu dopasowania modelu symulacyjnego dynamiki systemowej wykorzystano statystyki niezgodności Theila. Podejście to zasługuje na szczególną uwagę pośród całego zbioru testów służących temu celowi, ponieważ pozwala poznać nie tylko całkowitą wielkość błędu, lecz również umożliwia wskazanie przyczyny jego powstawania. Konstrukcja tych statystyk opiera się na zaobserwowanych w praktyce budowy modeli symulacyjnych przyczynach powstawania ich niedopasowania do charakterystyki obiektów rzeczywistych, z których jedna to słabość opracowanego modelu (błąd systematyczny), zaś druga to przypadkowość występująca w danych historycznych (błąd losowy).

Oceny adekwatności modelu symulacyjnego dokonano na podstawie następujących wielkości (8.1), (8.2) i (8.3)

$$U^M = \frac{(\bar{S}-\bar{A})^2}{\sigma^2}, \tag{8.1}$$

$$U^S = \frac{(S_S-A_A)^2}{\sigma^2}, \tag{8.2}$$

$$U^C = \frac{2(1-r)S_S A_A}{\sigma^2}, \tag{8.3}$$

za pomocą funkcji decyzyjnej określonej następująco (8.4):

$$d(U^M, U^S, U^C) = \begin{cases} h^0, jeżeli\ U^M \in \langle 0,\ 0.1\rangle \wedge U^S \in \langle 0,\ 0.2\rangle \wedge U^C \in \langle 0.8, 1\rangle \\ h^1, jeżeli\ U^M \notin \langle 0,\ 0.1\rangle \vee U^S \notin \langle 0,\ 0.2\rangle \vee U^C \notin \langle 0.8, 1\rangle \end{cases} \tag{8.4}$$

gdzie:

h^0 - nie ma podstaw do odrzucenia hipotezy o adekwatności opracowanego modelu symulacyjnego do procesów zachodzących w rzeczywistości;

h^1 - uzyskane wyniki zaprzeczają prawdziwości hipotezy o adekwatności opracowanego modelu symulacyjnego do procesów zachodzących w rzeczywistości,

przy czym:

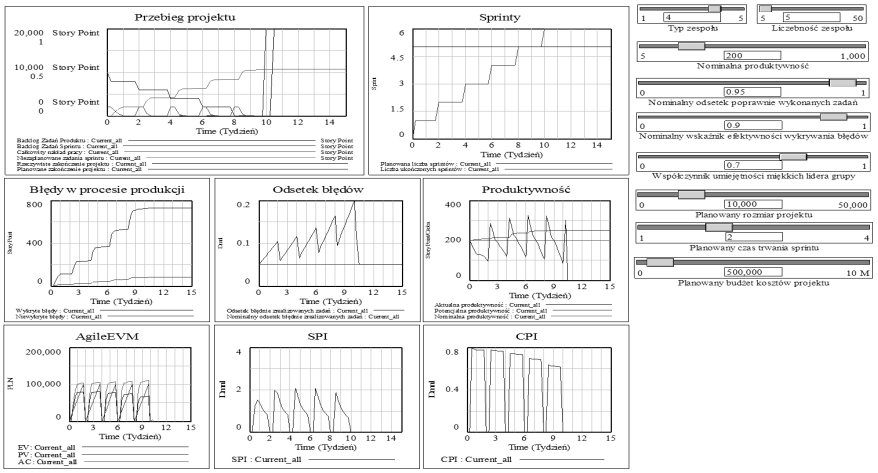
U^M - mierzy udział błędu systematycznego w wartości błędu średniokwadratowego,

U^S - mierzy udział błędu spowodowanego przez niezgodności wariancji w wartości błędu średniokwadratowego,

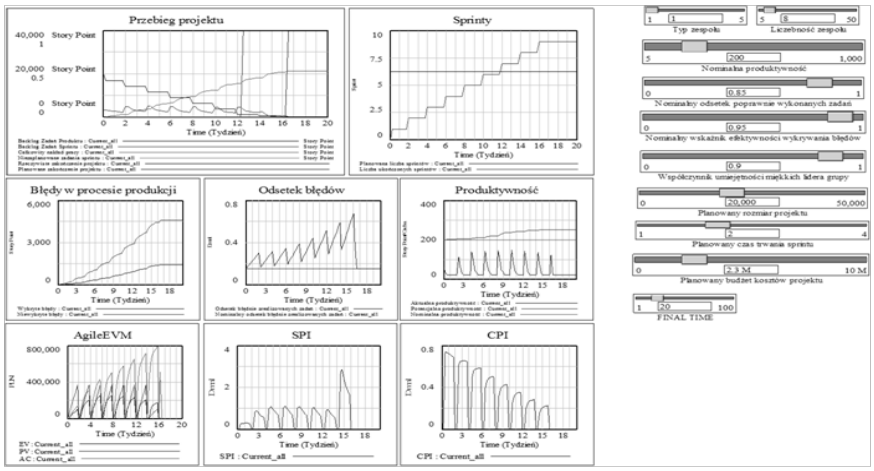
U^C - mierzy udział błędu spowodowanego przez niezgodność kowariancji w wartości błędu średniokwadratowego.

8.5.3 Prowadzenie eksperymentów symulacyjnych i uzyskane wyniki

W oparciu o zbudowany przez autorów model przeprowadzono eksperymenty symulacyjne dla dwóch projektów. Główne parametry modelu i uzyskane wyniki symulacji przedstawiają kolejne rysunki (Rysunek 8.4, Rysunek 8.5).



Rysunek 8.4. Wyniki symulacji dla projektu 1 – charakterystyki badanych metryk z zakresu: procesu wytwórczego, zespołu i parametrów projektu.



Rysunek 8.5. Wyniki symulacji dla projektu 2 - charakterystyki badanych metryk z zakresu: procesu wytwórczego, zespołu i parametrów projektu.

8.5.3.1 Schemat prowadzenia eksperymentów

Zaproponowany schemat zbudowany jest w oparciu o trzy rodzaje prowadzenia symulacji: graficzny, wstępny i statystyczny.

Pierwszy z nich, symulacja graficzna, oferuje prowadzenie symulacji z prezentacją graficzną, tak więc cały proces jest pod kontrolą użytkownika. Dzięki niemu można również określić, w którym miejscu występują tzw. "wąskie gar-

dła", które ujemnie wpływają na wydajność zaprojektowanej struktury. Można też określić moment czasowy występowania przestojów zespołu spowodowanych np. brakiem zaplanowanych zadań, bądź zbyt wolnym działaniem kontroli jakości. W każdym momencie można też obejrzeć dane dotyczące poszczególnych stanowisk/ról pracy, takie jak wydajność, produktywność, czasy przestojów zespołu lub poszczególnych członków zespołu wytwórczego, czy ilość wykonanych do tej pory zadań, lub odsetek niewykonanych zadań.

Drugi rodzaj przeprowadzania symulacji to prowadzenie kilku eksperymentów bez prezentacji graficznej określony mianem symulacji wstępnej. Po każdym przebiegu dane dotyczące zarówno całego procesu wytwórczego jak i poszczególnych jego elementów (sprintów) są możliwe do przejrzenia. Po wykonaniu wszystkich zaplanowanych przebiegów, dane średnie są prezentowane użytkownikowi. Kontrola nad symulacją nie jest tak duża jak w poprzednim podejściu, jednak cały proces odbywa się o wiele szybciej, a dane generowane po każdym przebiegu pozwalają ocenić poprawność pod względem ekonomicznym zaprojektowanej struktury procesu i/lub zespołu wytwórczego. Jest to najszybszy z zaprojektowanych, sposób na przeprowadzenie eksperymentu symulacyjnego. Jego celem jest szybkie przeprowadzenie 5 – 10 przebiegów symulacji, by na tej podstawie wyciągnąć wnioski co do sposobu działania zaproponowanej struktury. Tak mała ilość przebiegów pozwala jednak tylko na pobieżną ocenę przebiegu procesu wytwórczego.

Trzeci, ostatni rodzaj prowadzenia eksperymentu symulacyjnego jest zaprojektowany w celu zbierania danych statystycznych i dogłębnej oceny wydajności zaprojektowanego przebiegu procesu struktury zespołu wytwórczego. Jest to symulacja statystyczna. Podobnie jak w poprzednim podejściu, nie ma prezentacji graficznej podczas prowadzenia pojedynczej próby symulacyjnej. Nie są również podawane wyniki końcowe poszczególnych prób. W zamian za to czas potrzebny na przeprowadzenie jednej próby symulacji jest najmniejszy, a cały proces zbierania danych statystycznych nie wymaga udziału człowieka. Możliwe jest więc przeprowadzenie kilkuset prób, których wyniki zapisywane są do pliku. Ten proces trwa najdłużej, jednak daje pełne spektrum wyników, jakie możliwe są do osiągniecia przy użyciu zaprojektowanego schematu. Dane statystyczne, zebrane w wyniku jego działania można następnie przetwarzać np. w arkuszu kalkulacyjnym. Generowane są również różnego typu histogramy np. wydajności całego zespołu oraz czasów trwania poszczególnych sprintów. W oparciu o te rodzaje prowadzenia eksperymentów symulacyjnych opracowano sposób tworzenia, oceny i zbierania danych dotyczących zaprojektowanej struktury zespołu wytwórczego. Składa się on z następujących kroków:

a. zaprojektować strukturę zespołu wytwórczego;
b. przeprowadzić symulację wstępną;
c. dokonać oceny symulacji wstępnej [jeżeli niektóre z parametrów wyjściowych symulacji budzą wątpliwości, przeprowadzić symulację graficzną; jeżeli ocena zespołu wytwórczego, dokonana na podstawie kroków b) i c), wypadła pozytywnie, przeprowadzić symulację statystyczną. w przeciwnym razie powrócić do kroku a)];
d. dokonać analizy uzyskanych danych statystycznych.

Podstawą do prowadzenia analizy statystycznej są generowane przez system symulacyjny histogramy. Przedstawiają one rozkład wystąpień poszczególnych wartości w całym spektrum możliwych wyników - od wartości minimalnej do maksymalnej uzyskanych w danym eksperymencie. Na tej podstawie można skonstruować funkcję gęstości. Jednak aby miała ona kształt zbliżony do funkcji Gaussa, należy zaplanować dużą ilość powtórzeń podczas symulacji statystycznej. Powyższe rysunki (Rysunek 8.4, Rysunek 8.5) prezentują wyniki symulacji (wybrane wykresy) dla dwóch projektów realizowanych metodyką Scrum.

8.5.3.2 Walidacja modelu symulacyjnego

O ile weryfikacja jest zadaniem prostszym, gdyż polega na znajdowaniu i usuwaniu błędów logicznych i składniowych w strukturze modelu oraz opisujących go równaniach, o tyle walidacja jest zadaniem trudniejszym. Walidację należy rozpocząć już w pierwszych fazach modelowania, podczas tworzenia modelu koncepcyjnego. Następnie, w trakcie tworzenia modelu symulacyjnego, należy kontrolować postępy prac i zapewnić, aby uzyskiwane wyniki nie odbiegały od przyjętych założeń. Wreszcie w końcowej fazie cały model symulacyjny musi zostać przetestowany.

Wyniki wygenerowane przez symulator należy porównać z wynikami rzeczywistymi. Dopiero po sprawdzeniu i zaakceptowaniu stopnia zgodności obydwu z nich, model symulacyjny może zostać uznany za działający poprawnie i wdrożony do właściwej pracy.

Walidacja każdego symulatora może mieć inny charakter i przebiegać według odmiennej procedury. Zależne to jest od wielu czynników, między innymi od dostępności danych historycznych pochodzących z symulowanego systemu, możliwości wykorzystania wiedzy ekspertów, czy od założonego stopnia wiarygodności, jaki symulacja musi spełniać.

W procesie walidacji opracowanego modelu symulacyjnego oparto się na danych rzeczywistych pochodzących z realizacji dwóch projektów realizowa-

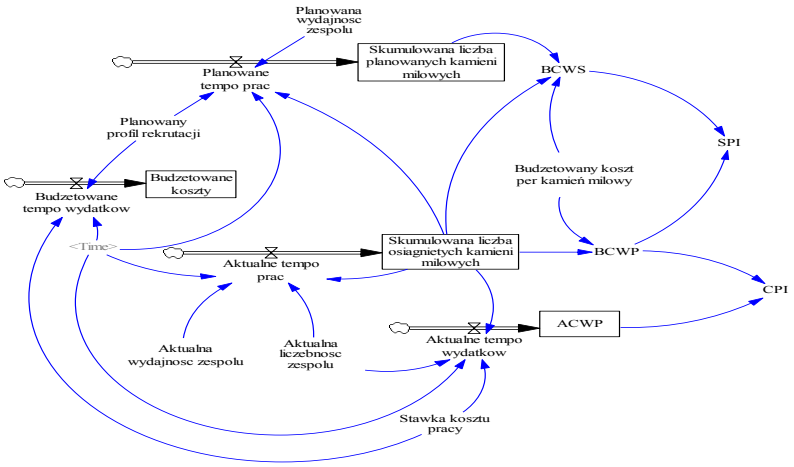
nych w Instytucie Systemów Informatycznych Wydziału Cybernetyki WAT. Walidacja przeprowadzona była w oparciu o następujące źródła:

- dane historyczne dotyczące podstawowych charakterystyk procesu wytwórczego i metryk oprogramowania, realizowanego zgodnie z wytycznymi i zaleceniami metodyki Scrum,
- dane dotyczące wydajności zespołów dla poszczególnych faz (sprintów) metodyki Scrum zapisane rejestrach systemu JIRA służącego do przydziału zadań w poszczególnych projektach,
- wartości rzeczywiste, zapisane w rejestrach: Jakości, Ryzyka oraz w Dziennikach Pracy poszczególnych Kierowników Projektów,
- wiedzę ekspercką.

8.5.3.3 Schemat procesu walidacji

Jako podstawę prac walidacyjnych przyjęto porównanie wartości wypracowanej "Earned Value". Nazwa "Earned Value" pochodzi od idei, że każdy produkt dostarczany przez projekt posiada planowany koszt, czyli swoją "wartość".

Model symulacyjny zbudowany zgodnie z założeniami metody Earned Value przy użyciu środowiska symulacyjnego Vensim® przedstawiono na rysunku poniżej (Rysunek 8.6).

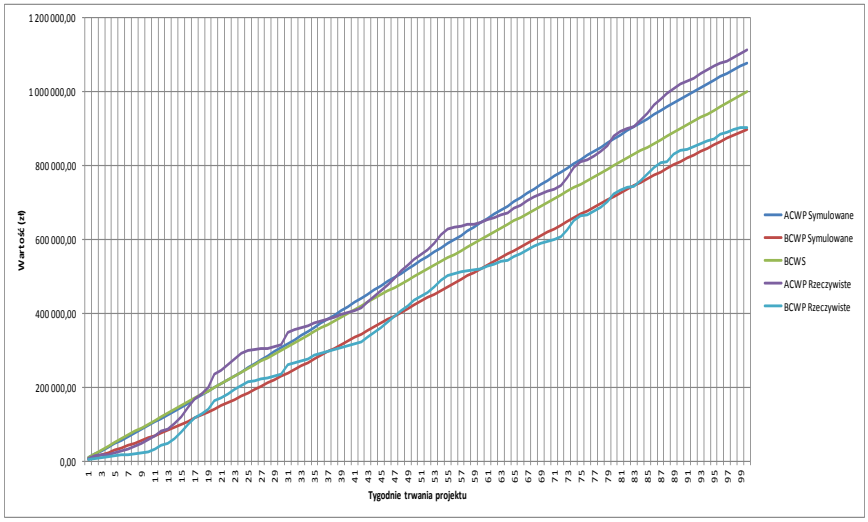


Rysunek 8.6. Model symulacyjny Earned Value.

Gdy produkt jest ukończony, jego "wartość" jest "zarobiona" w ramach projektu. Podejście takie, oparte na ciągłym monitorowaniu projektów na każdym etapie ich realizacji, pozwala wcześnie dostrzec ewentualne odchylenia harmonogramu oraz przekroczenia kosztów i na tej podstawie podjąć odpowiednie

działania, aby zminimalizować ryzyko niepowodzenia realizowanego przedsięwzięcia. Kontrola projektu obejmuje również wycenę wartości prac, produktów, usług itp. zrealizowanych do chwili kontroli. Pozwala to na porównanie trzech parametrów, tj. kosztów planowanych (BCWS), kosztów rzeczywistych (ACWP), wartości uzyskanej (BCWP) i określenia wskaźników odchyleń zawiązanych z realizacją projektu takich, jak planowany czas (harmonogram) - SPI - i budżet (koszt projektu) – CPI.

Na bazie skonstruowanego modelu został przeprowadzony eksperyment symulacyjny, którego wyniki posłużyły następnie jako materiał wejściowy do procesu walidacji a następnie do oceny jego adekwatności. Na poniższym wykresie (rysunek 8.7) zestawiono ze sobą wyniki symulacji oraz dane zebrane podczas realizacji rzeczywistego projektu IT, realizowanego metodyką Scrum.



Rysunek 8.7. Porównanie wyników symulacji z wartościami zaobserwowanymi w rzeczywistości.

Na ich podstawie dokonano obliczenia statystyk Theil'a, których wartości zaprezentowano w poniższej tabeli (Tabela 8.1).

Tabela 8.1. Uzyskane charakterystyki Theil'a.

Zmienna	$\mathbf{U^M}$	$\mathbf{U^S}$	$\mathbf{U^C}$
ACWP	0,03	0,13	0,84
BCWP	0,00	0,10	0,90

Duża wartość U^C i relatywnie małe wartości U^M i U^S pozwalają stwierdzić występowanie błędu niesystematycznego, który może być spowodowany dużą fluktuacją w zakresie parametrów wpływających na aktualne tempo prac zespołu. Na podstawie powyższych wyników oraz przyjętej postaci funkcji decyzyjnej (4) można stwierdzić, że nie ma podstaw do odrzucenia hipotezy o adekwatności opracowanego modelu symulacyjnego do procesów zachodzących w rzeczywistości.

8.6 Podsumowanie

W ramach prac będących tematem niniejszego rozdziału wykonano dużo zadań o naturze teoretycznej, typowo badawczej i inżynierskiej. Rozważania teoretyczne nie były głównym tematem, ale mimo to mają dość istotne znaczenie dla konstrukcji modelu symulacyjnego i schematu przeprowadzania eksperymentów symulacyjnych. W związku z tym w podsumowaniu wydzielono trzy części. Pierwsza koncentruje się na pojęciach: jakość oprogramowania, zwinny proces produkcji oprogramowania oraz modele i metryki oprogramowania. Druga cześć natomiast opisuje aspekt typowo badawczy: model symulacyjny, implementacja modelu, środowisko niezbędne do przeprowadzenia eksperymentów. Trzecia przestawia wnioski nasuwające się w wyniku oceny rezultatów uzyskanych przez autorów w drodze symulacji komputerowej.

8.6.1 Cześć teoretyczna

Problemy mierzenia jakości oprogramowania w inżynierii oprogramowania są pochodną problemów definiowania i pomiaru właściwości oprogramowania komputerowego jako produktu i określania właściwości procesu jego wytwarzania. Podstawowe problemy związane z mierzeniem jakości oprogramowania można sprowadzić do trzech głównych zagadnień, a mianowicie:

- dalszej standaryzacji i rozbudowy modeli jakości oprogramowania, w tym formalnych definicji podstawowych właściwości;
- konstrukcji efektywnych miar i metryk jakości oprogramowania dostosowanych do różnych narzędzi programistycznych i dziedzin ich wykorzystania;
- upowszechnienia metod symulacyjnych do badania jakości oprogramowania, wytwarzanego w oparciu o różne modele cyklu życia oprogramowania.

Nie ulega wątpliwości, że obecnie do oceny jakości procesów wytwórczych i jakości pracy zespołów przykłada się o wiele większą uwagę, niż do

oceny produktów programowych. Świadczy o tym chociażby fakt, że w Polsce wszystkie znaczące firmy wytwarzające oprogramowanie uzyskały certyfikat systemu zarządzania jakością ISO 90001, kilka zdecydowało się na ocenę CMM/CMMI, natomiast krótka norma dotycząca oceny jakości produktów programowych – ISO 9126, praktycznie nie jest stosowana, nie została nawet przetłumaczona na język polski. Sytuacja ta wydaje się być niezrozumiała, chociaż istnieją argumenty za takim postępowaniem, np. pojawienie się zwinnego procesu produkcji oprogramowania lub stosowanie symulacyjnych modeli jakości zespołu. Można jednak przypuszczać, że w niedalekiej przyszłości sytuacja musi ulec zmianie – w rzeczywistości dla odbiorcy oprogramowania wcale nie jest istotne, jak dojrzałe i ustabilizowane procesy realizuje wykonawca i/lub jakie stosuje symulacyjne modele jakości pracy zespołu, natomiast żywotne znaczenie ma jakość nabytego produktu programowego.

8.6.2 Cześć badawcza

Stworzenie modelu symulacyjnego do badania jakości oprogramowania jest zadaniem trudnym i wymagającym doświadczenia. Aktualnie nie ma żadnych ustalonych wzorców (norm i standardów technicznych) dotyczących tworzenia, ani tym bardziej walidacji tak złożonych modeli symulacyjnych. Zapewnienie adekwatności wyników generowanych przy użyciu symulatora do charakterystyk otrzymywanych z badań empirycznych jest chyba najtrudniejszą, lecz również najważniejszą rzeczą w całym procesie tworzenia symulatora, na co starano się zwrócić uwagę w niniejszym rozdziale.

Zaprezentowany model symulacyjny jest użytecznym narzędziem wspomagającym podejmowanie decyzji i dokonywanie oceny rzeczywistej jakości oprogramowania oraz produktywności zespołu wytwórczego w zwinnym procesie produkcji. Pozwala on na bardziej efektywną ocenę licznych parametrów projektów informatycznych, prowadzonych zgodnie z zaleceniami metodyk zwinnych. Ponadto daje kierownictwu firmy i kierownikowi projektu bardzo ważne narzędzie, a mianowicie dane, z którymi można porównać rzeczywistą wydajność zespołu wytwórczego.

Uzyskane wyniki pozwalają stwierdzić, że zbudowany model i przeprowadzone przy jego użyciu symulacje mogą być doskonałym narzędziem służącym do badania przebiegu projektów IT. Zbudowany model może również posłużyć do szkolenia kierowników projektu w zakresie prognozowania rozwoju zdarzeń w projekcie i oceny ich wpływu na jego powodzenie m.in. poprzez wykorzystanie analiz scenariuszowych typu "co-jeżeli?" opartych o zmianę kluczowych

parametrów modelu takich jak np.: liczebność zespołu, profil produktywności zespołu, profil rekrutacji itp.

8.6.3 Wnioski z oceny rezultatów

Wnioski nasuwające się w wyniku oceny rezultatów uzyskanych przez autorów w drodze symulacji komputerowej są następujące:

- typ zespołu realizującego projekt wraz z presją harmonogramu stanowią główne czynniki wpływające na jakość pracy wykonywanej w projekcie wyrażoną jako odsetek błędnie zrealizowanych zadań;
- im gorszy zespół tym wyższy odsetek błędów i niższa efektywność (%) ich wykrywania przez zespół (założono, że w projektach realizowanych zgodnie ze Scrum nie istnieje oddzielny zespół odpowiadający za testy, a role testerów pełnią cyklicznie kolejni członkowie zespołu wytwórczego);
- z kolei wyższy odsetek błędów wpływa negatywnie na satysfakcję zespołu, a przez to również na poczucie bezpieczeństwa i działania związane z samodoskonaleniem, co z kolei przekłada się morale zespołu i obniżenie jego produktywności.

Podziękowania

Prace badawcze, których wyniki przedstawiono w rozdziale są wspierane przez Narodowe Centrum Badań i Rozwoju (NCBiR) w ramach projektu realizowanego w ramach Przedsięwzięcia pilotażowego Wsparcie badań naukowych i prac rozwojowych w skali demonstracyjnej DEMONSTRATOR+ pt. ”Nowoczesny demonstrator symulatora dla operatorów pojazdów szynowych zwiększający efektywność i bezpieczeństwo ich działania”, na podstawie umowy nr UOD-DEM-1-501/001 z dnia 23.12.2013r.

Bibliografia

[AB02] Abrahamsson P. et al., *Agile software development methods, Review and analysis*, 2002.

[CR79] Crosby P. B., *Quality is free: The art of making quality certain,* McGraw Hill Custom Publishing, New York, USA, 1979.

[FL06] Flasiński M., *Zarządzanie projektami informatycznymi*, PWN, Warszawa, 2006.

[GR01] Gryna F. M., *Quality Planning and Analysis: From Product Development Through Use*, McGraw Hill Custom Publishing, New York, USA, 2001.

[IE98] *IEEE Std 1061-1998, Standard for a Software Quality Metrics Methodology*, IEEE, New York, 1998.

[IS01] *ISO/IEC 9126:2001 Software engineering - Product Quality, Part 1: Quality model*, ISO, Geneva, 2001.

[JK04] Jung H., Kim S., Chung C., *Measuring software product quality: A survey of ISO/IEC 9126,* IEEE Software, 2004, pp. 2188–92.

[KA02] Kan S., *Metrics and Models in Software Quality Engineering,* Addison-Wesley, Boston, 2002.

[MA08] Madachy R. J., *Software process dynamics*, Wiley, New Jersey, 2008.

[MS01] Myers G. J., Sandler C., *The Art of Software Testing,* John Wiley & Sons, 2004.

[PS12] Protasowicki T., Stanik J., *Ocena adekwatności modeli symulacyjnych dynamiki systemowej na przykładzie modelu Earned Value*, XIX Warsztaty Naukowe PTSK "Symulacja w Badaniach i Rozwoju", 2012.

[PS13a] Protasowicki T., Stanik J., *Modelowanie wartości Earned Value w zarządzaniu projektami z wykorzystaniem podejścia Agile. Opis dla potrzeb symulacji*, [w:] *Zarządzanie projektami i modelowanie procesów*, red. Szyjewski Z., red. Muszyńska K., PTI, Warszawa, 2013.

[PS13b] Protasowicki T., Stanik J., *Modelowanie wartości Earned Value w zarządzaniu projektami z wykorzystaniem podejścia Agile. Model* , [w:] *Zarządzanie projektami i modelowanie procesów*, red. Szyjewski Z., red. Muszyńska K., PTI, Warszawa, 2013.

[SK05] Skrobanek P., *Analiza zależności czasowych w drzewach niezdatności systemów związanych z bezpieczeństwem,* Praca doktorska, Politechnika Wrocławska, Wrocław, Polska, 2005.

[ST00] Sterman J. D., *Business Dynamics. System thinking and modeling for a complex world*, McGraw Hill, 2000.

Rozdział 9

Pomiar skutków wdrożenia narzędzi Business Intelligence w organizacji wsparcia informatycznego

W rozdziale przedstawiona została problematyka oceny zasadności oraz efektywności wdrożenia systemu klasy Business Intelligence w organizacji wsparcia informatycznego. Podmiot ten świadczy usługi IT dla klientów zewnętrznych, w szczególności – polegające na przechowywaniu i przetwarzaniu dużych ilości danych. Duża ilość realizowanych projektów i zgłaszanych w związku z nimi incydentów uniemożliwiały efektowne podejmowanie decyzji bez wsparcia dedykowanymi technologiami. Autorzy przedstawiają problemy, jakich badany podmiot doświadczył oraz opisują jakie narzędzie wybrano w celu poprawy sytuacji. Celem badania było zmierzenie i opisanie kluczowych procesów organizacji w oparciu o przygotowane zagregowane mierniki przed wdrożeniem systemu BI, a następnie rok po jego implementacji. Opracowany przez autorów model ocenowy pozwolił na ocenę kluczowych aspektów działalności firmy na przestrzeni dwóch lat. Tym samym pozwolił na uzyskanie odpowiedzi na pytanie osobom decyzyjnym, czy decyzja o wdrożeniu systemu Business Intelligence była słuszna, czy też nie.

Na przestrzeni lat znacznie wzrosła liczba danych potrzebna organizacjom do funkcjonowania. Znakomitym tego przykładem jest historia przenośnych nośników danych. Jeszcze na początku lat 90 dyskietka o pojemności niespełna 1,5 MB była wystarczająca do większości potrzeb; w najgorszym wypadku konieczne było wykorzystanie kilku dyskietek. Kilka lat później pojawiła się płyta CD, która na tamte czasy dysponowała niewyobrażalną pojemnością 700 MB. To jednak również okazało się za mało - kolejny skok technologiczny w postaci płyt DVD zwiększył dostępną pojemność przenośnych pamięci ponad sześciokrotnie. Dzisiejsze pamięci przenośne pendrive mają rozmiary mniejsze od pudełka zapałek, a pozwalają zapisać kolejne kilka (kilkadziesiąt) razy więcej danych niż na płycie DVD.

Wraz ze wzrostem dostępnej przestrzeni dyskowej zatracone zostały umiejętności gospodarowania danymi - po co ktoś miałby się przejmować trzema identycznymi kopiami tego samego pliku, skoro ma jeszcze dostępne setki gigabajtów wolnego miejsca? W przypadku domowego użytkownika nie stanowi to jeszcze tak wielkiego problemu, ale co się dzieje, gdy w ten sposób postępuje firma dysponująca setkami komputerów? Zapotrzebowanie na potrzebną pamięć rośnie błyskawicznie, a wraz z nimi koszty infrastruktury. Należy przy tym zauważyć, że mimo iż dane są gromadzone dla konkretnych celów, nierzadko bardzo trudno jest uzyskać z nich potrzebne informacje. Gdy przychodzi czas podjęcia strategicznych decyzji, konieczne staje się ręczna analiza wielu zbio-

rów danych (plików) i wyciąganie z nich wniosków. Takie mozolne przeszukiwanie rzadko kiedy daje wymierne efekty [SM98].

Zasadniczo proces podejmowania decyzji przebiega w kilku krokach: zdefiniowanie problemu, zbadanie wariantów wyboru, przewidzenie konsekwencji, wybór optymalnego wariantu. Już pierwszy z nich jest często problematyczny, zwłaszcza w opisywanym przypadku nadmiaru danych. Często okazuje się, że dane są rozproszone. Możliwa jest sytuacja, iż dane z jednej placówki są znacznie starsze od danych z innych, przez co niemożliwe jest podjęcie jednej wspólnej decyzji dla całej organizacji. Zdarzyć się może, że dane z kilku źródeł są redundantne, przez co manager dysponuje ich nadmiarem i ciężko jest z nich wybrać te faktycznie istotne. Wtedy szybko organizacja może doświadczyć tzw. efekt domina. Nadmiar danych komplikuje ich analizę, a mniej efektywna analiza negatywnie wpływa na tworzenie dostępnych wariantów wyboru. Bazując na błędnie przeanalizowanych, niepełnych lub zwyczajnie nieprawdziwych informacjach przewidzenie konsekwencji wyboru staje się co najmniej trudne. W takich okoliczność wariant, który wydaje się optymalny, w rzeczywistości wcale taki nie jest [Tar86].

W dzisiejszych czasach, kiedy znaczna część pracy została zinformatyzowana, paradoksalnie podejmowanie decyzji wcale nie jest prostsze niż w przeszłości. W większości organizacji, w tym także w organizacjach IT, procesowi podejmowania decyzji nieodłącznie towarzyszy szereg problemów [Gri98]:

- nadmiar bądź niedomiar danych,
- nieumiejętność interpretacji danych i przekształcania ich w informacje,
- nieaktualność danych (informacji),
- rozproszenie danych,
- wolny dostęp do danych,
- trudności z agregacją danych i informacji,
- trudności związane z ograniczaniem niepewności.

Istnieją technologie informatyczne, które wspierają proces podejmowania decyzji. Owo wsparcie uzyskiwane jest poprzez rozwiązywanie, bądź upraszczanie występujących przy tym procesie problemów. Pozwalają one na szybszy dostęp do danych, agregowanie ich i analizę. Ogromną ilość porozdzielanych informacji z różnych części firmy można połączyć w jednym miejscu, zachowując ich czytelność i nadając im uporządkowaną strukturę. Umożliwiają one także na ich analizę oraz stworzenie raportów, dzięki którym przy procesie podejmowania decyzji minimalizowane jest zjawisko niepewności, a co za tym idzie, zmniejszane jest ryzyko podjęcia błędnej decyzji, a zwiększa się prawdopodo-

bieństwo podjęcia tej właściwej. Narzędzia, o których mowa, określa się pojęciem Business Intelligence [Ros01].

Wdrożenie technologii Business Intelligence to znaczący koszt dla organizacji. Pojawia się jednak pytanie: czy można go uzasadnić w sposób ilościowy? Czy pod wdrożeniu systemu tej klasy istnieje możliwość oceny skutków tej decyzji? Doświadczenia autorów pokazują, że jest to problem wielu organizacji. Ocena sytuacji po wdrożeniu systemu ma zwykle naturę jakościową i opiera się na subiektywnych odczuciach użytkowników i może być skrajnie różna w zależności od ich stanowiska, zakresu wykonywanych czynności, czy zmian, które ich bezpośrednio dotknęły [Alt99].

Autorzy, dysponując bardzo szczegółowymi danymi ilościowymi zbieranymi w organizacji przez wdrożeniem oraz po implementacji Business Intelligence, dokonali ich analizy i uznali, że stanowią one bazę dla budowy modelu ocenowego zasadności tego wdrożenia. W rozdziale przedstawiono więc badaną organizację, opracowane dla niej wskaźniki pomiaru kluczowych aspektów działalności oraz zaprezentowano metodę ilościowej oceny efektywności wdrożenia.

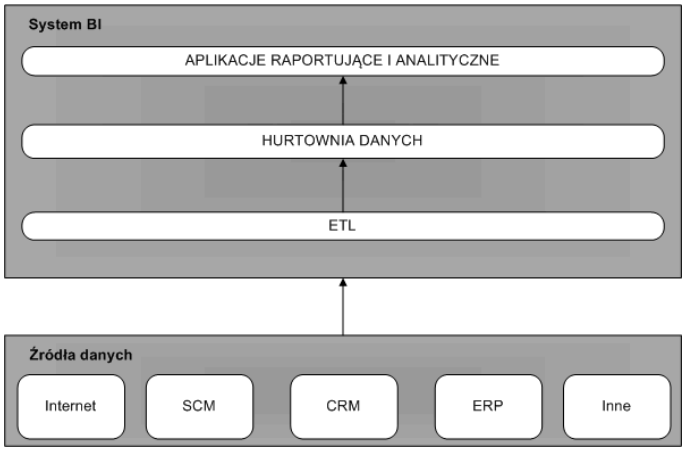
9.1 BI jako metoda wspierania procesu podejmowania decyzji

Business Intelligence, w skrócie nazywany BI, jest pojęciem definiowanym na bardzo wiele sposobów. W rozdziale przyjęto definicję stworzoną przez Gartner - firmę consultingową specjalizującą się w zagadnieniach strategicznego wykorzystania technologii [Jan08]. Gartner definiuje BI jako "wielowymiarowe pojęcie obejmujące aplikacje, infrastrukturę, narzędzia oraz metody pozwalające na dostęp oraz analizę informacji celem usprawnienia oraz zoptymalizowania procesu podejmowania decyzji i ogólnej działalności" [Gar12].

Typowa architektura systemu BI jest zbudowana z kilku elementów. Sercem takiego systemu są hurtownie danych, czyli bazy danych zorientowane na obsługę zapytań analitycznych. Dane do hurtowni trafiają z kilku źródeł: systemów ERP, CRM, SCM, Call Centers i innych przy pomocy procesów ETL (ang. *Extract, Transform, Load* - ekstrakcja, przekształcenie, ładowanie). Owe procesy integrują dane z różnych systemów w jeden spójny model, tzw. model wielowymiarowy. Następnie dane z hurtowni są zazwyczaj pobierane do silników przetwarzających zapytania analityczne (OLAP, ang. *Online Analitical Processing*), które pozwalają na szybką analizę na różnych płaszczyznach i przy różnych zestawach zapytań, czasami dość abstrakcyjnych.

Użytkownik następnie uzyskuje dostęp do wyników tych analiz poprzez narzędzia biznesowe i analityczne. Owe narzędzia prezentują informacje w po-

wiązanych ze sobą raportach i analizach - bardzo często dzieje się to poprzez tzw. *dashboardy* (kokpity menadżerskie). Oczywiście, te aplikacje mogą dostarczać informacji także w inny sposób, np. jako arkusze kalkulacyjne lub dokumenty pdf. Często możliwe jest także dostarczanie raportów na skrzynkę pocztową wybranych użytkowników, czy to w ramach standardowego monitoringu, czy w wypadku sytuacji alarmowej. Zazwyczaj systemy BI działają na platformach internetowych, pozwalając użytkownikom biznesowym na dostęp do raportów i analiz poprzez przeglądarkę internetową. Ma to na celu ułatwienie dostępu do informacji niezależnie od miejsca pobytu osoby, która usiłuje się do nich dostać. Na system BI składają się więc źródła danych, które poprzez obecność procesów ETL trafiają do hurtowni danych i w następnej kolejności do aplikacji raportujących i analitycznych. Schematycznie ideę BI można zobrazować jak na rys. 9.1.



Rysunek 9.1. Uproszczony schemat systemu BI.

System BI ułatwia podejmowanie decyzji, ponieważ ingeruje we wszystkie etapy tego procesu i upraszcza je [OS10]. Gromadzi on dane w jednym miejscu (eliminując problem wolnego dostępu do danych i ich rozproszenia, a także istnienia duplikatów), rozwiązuje problem nieaktualności informacji, pozwala na dokładniejszą analizę danych i zamianę ich na informacje. Oczywiście tylko system zawierający wszystkie podstawowe budulce, wyszczególnione na rysunku 9.2, może właściwie wspierać proces podejmowania decyzji. Pełne, kompleksowe systemy BI są oferowane przez takie korporacje, jak Microsoft, Oracle, IBM czy SAP. W parze z rozbudowaniem systemów oferowanych przez te firmy idzie wysoka cena. Większość małych i średnich firm nie ma ani środ-

ków, ani potrzeby stosowania tak drogich rozwiązań. Niemniej możliwe jest zbudowanie funkcjonalnego systemu BI, choćby niewielkiego, który będzie wspomagał proces podejmowania decyzji, z komponentów oferowanych przez mniejsze firmy. W badanej organizacji wsparcia IT wykorzystany został właśnie taki system, integrujący rozwiązania kilku różnych dostawców.

9.2 Charakterystyka badanej firmy i jej główne problemy

Badanym podmiotem jest firma Avena Technologie, działająca w branży informatycznej i telekomunikacyjnej. Zajmuje się budową i administracją sieci miejskiej w Darłowie; obsługuje tam ok. 3 tysięcy odbiorców. Firma świadczy usługi outsourcingu IT: instalacja i administracja infrastruktury sieciowej, wdrożenia i administracja telefonią VoIP (ang. *voice over IP*), zdalna pomoc techniczna (ang. *helpdesk*). Ponadto Avena zajmuje się montażem i utrzymywaniem bezprzewodowego Internetu na terenie Gdańska i Gdyni wykorzystując Hot Spoty. Oprócz tego firma pracuje nad inteligentnym systemem monitorowania ruchu drogowego. Projekt wiąże się z montażem kamer na trójmiejskiej obwodnicy, przygotowaniem odpowiedniego oprogramowania umożliwiającego stałą obserwację ruchu na drogach, w szczególności takich zdarzeń jak: korki, wypadki, roboty drogowe, w celu udostępnienia tych informacji 24 h/dobę wszystkim użytkownikom Internetu.

Avena została założona 10 lat temu, w 2003 roku. Od tamtej pory dynamicznie się rozwija. Obecnie ma dwie siedziby w Polsce: w Gdyni i w Gdańsku. Firma świadczy zewnętrzne usługi informatyczne ponad 30 firmom w okolicach Trójmiasta [PF12]. Większość pracowników w obu siedzibach zajmuje się pracą w dziale helpdesk. Jest to pomoc techniczna realizowana główne w sposób zdalny, co wciąż jest pewnym wyróżnikiem dla firm z tej branży. Doświadczenia autorów wskazują, że niewiele firm w Polsce dostarcza taką usługę kompleksowo - popyt znacznie przewyższa podaż. Podstawowe zadania pracowników – określanych w dalszej części pracy mianem administratorów - to doraźna lub kompleksowa pomoc klientowi w problemach software'owych. Klienci oczekują od administratorów zarówno działań związanych z eliminacją nieprzewidzianych błędów systemu operacyjnego i oprogramowania roboczego, jak też stałego wsparcia w zakresie obsługi wdrożonych systemów (np. ich konfiguracji według wizji i potrzeb klienta czy zapewnienia bezpieczeństwa danych).

Właśnie ze względu na poważne obciążenie pracą w dziale helpdesk firma zmuszona była w jakiś sposób usprawnić zarządzanie. Przy tysiącach zgłoszeń serwisowych (ang. *Service Request,* w skrócie SR; skrót ten będzie używany

w dalszej części rozdziału) firma stopniowo traciła możliwość kontroli własnej działalności. Niemożliwe stało się gospodarowanie danymi klientów, monitorowanie efektywności pracowników, analizowanie informacji i łączenie ich w pewne wzorce. Mało tego, firma miała problem z określeniem ilości świadczonych usług - kierownictwo nie było w stanie określić, ile czasu zajmuje obsługa klienta, ile to kosztuje i w związku z tym czy współpraca z klientem się opłaca.

Od roku 2010, kiedy Avena zaczęła dostarczać usługi związane z helpdesk, zdolność kierownictwa do podejmowania racjonalnych decyzji znacznie spadła. Ze względu na charakter świadczonych usług i ogromną liczbę zgłoszeń, pojawiły się liczne problemy uniemożliwiające lub przynajmniej utrudniające podejmowanie decyzji:

- nadmiar danych w pewnych obszarach (np. tysiące rekordów związanych z SR) i niedomiar w innych (np. wiedza o aktualnym stanie infrastruktury IT u klientów),
- trudny i wolny dostęp do danych (np. klient wysuwa zapytanie o możliwość instalacji nowszego systemu operacyjnego na wszystkich swoich komputerach - ręczne sprawdzenie setki komputerów jest bardzo czasochłonne),
- rozproszenie danych (na setki komputerów u dziesiątków klientów) ,
- nieaktualność danych (bez automatycznej aktualizacji o stanie infrastruktury dane za każdym razem trzeba zdobywać od nowa)
- nieumiejętność monitorowania własnej działalności (którzy pracownicy obsługiwali danego klienta, ile czasu to zajęło, jak zmierzyć ich wydajność),
- problemy z agregacją danych i zamienianiem ich w konkretne informacje (np. wynikające z różnic w oprogramowaniu i sprzęcie u klientów).

Podejmowanie decyzji w takich warunkach wiązało się z operowaniem w stanie wysokiej niepewności, niezależnie od tego, czy problem leżał po stronie klienta, czy organizacji wsparcia. Kierownictwo uświadomiło sobie, że firma nie jest w stanie sprawnie funkcjonować bez wsparcia odpowiednich systemów informatycznych. Żeby poradzić sobie z wymienionymi powyżej problemami, firma zdecydowała się na wdrożenie systemu BI. Zrobiono to jednak nie mając przekonania, co do słuszności tej decyzji. Kierownictwo nie dysponowało żadnymi przesłankami uzasadniającymi planowany wydatek. Nie dokonano również oceny zmian, jakie zaszły w organizacji po wdrożeniu. To stało się celem badań autorów. W następnym rozdziale przedstawiono metodę, którą autorzy opracowali, by określić w sposób ilościowy zasadność wdrożenia BI w firmie Avena.

9.3 Pomiar skutków wdrożenia systemu BI

9.3.1 Założenia i kluczowe wskaźniki

Czy możliwe jest mierzenie skutków wdrożenia systemu BI? W tym celu konieczne jest stworzenie szeregu wskaźników, które posłużą do oceny i sprawdzeniu, jak zmieniły się ich wartości przed i po wdrożeniu systemu BI, najlepiej w analogicznych okresach. Wskaźniki te powinny pokrywać swoim zakresem większą część działalności firmy, a nie tylko jeden konkretny obszar. Wynika to z faktu, że system BI ma ogólnie wspierać procesy podejmowania decyzji we wszystkich aspektach działalności firmy, a zatem wskaźniki mają za zadanie w liczbowy sposób zobrazować możliwie wiele tychże aspektów.

Chociaż firmy różnią się między sobą, kilka wskaźników może być zastosowanych do większości z nich. Oczywiście, mogą one zostać dokładniej dopasowane do charakterystyki badanej organizacji, a kilka może powstać dokładnie dopasowanych do konkretnej organizacji. Przykładowe wskaźniki to [Lit12]:

- liczba obsłużonych klientów/dokonanych transakcji/sprzedanych towarów itp.
- liczba i wydajność pracowników
- rzeczywisty czas pracy pracowników
- czas od rozpoczęcia do zakończenia czynności
- stopień zadowolenia klientów.

Należy zauważyć, że część z tych wskaźników można w łatwy sposób określić liczbowo (np. liczba dokonanych transakcji), inne zaś mają charakter jakościowy (np. stopień zadowolenia klientów). Autorzy postawili jednak sobie za cel dokonać oceny ilościowej. Dlatego konieczne było zastosowanie pewnych uproszczeń celem ustalenia wartości wskaźników służących do oceny. Oczywiście, niezbędne było także zebranie odpowiednich danych, by możliwe było porównanie okresów przed i po wdrożeniu systemu BI. Naturalnie zdecydowanie prostsze okazało się to dla okresu "po wdrożeniu", ponieważ system BI pozwala na uzyskanie potrzebnych informacji.

Żeby ocenić zasadność wdrożenia systemu BI w firmie Avena, porównano wobec tego dwa okresy:

- okres od 01.04.2011 do 30.09.2011 (O1)
- rok później, czyli okres od 01.04.2012 do 30.09.2012 (O2).

Do opisania działalności firmy wybrano następujące cztery aspekty:

- liczba SR (ang. *Service Request* – zgłoszenie serwisowe, rozumiane jako problem, z którym zgłasza się użytkownik do organizacji wsparcia IT), który wymaga podjęcia określonych działań,

— liczba administratorów i ich efektywność,
— średni czas naprawy MTTR (ang. *Mean Time To Repair* - średni wymagany czas do naprawy uszkodzonego urządzenia; w tym przypadku to szacunkowy czas wymagany od administratora od otwarcia do zamknięcia zgłoszenia.),
— stopień zadowolenia klientów.

Każdy z tych aspektów został w trakcie badania opisany przez autorów ilościowo i do każdego z nich stworzone zostały 1-2 wskaźniki. Należy przy tym dodać, że system BI został po raz pierwszy uruchomiony pod koniec marca 2011, a więc tuż przed rozpoczęciem O1.

9.3.2 Porównanie liczby zgłoszeń serwisowych (SR)

Organizacja obsłużyła w ciągu sześciu miesięcy w O2 łącznie 4201 SR, czyli średnio 700 zgłoszeń miesięcznie. Między 1 kwietnia a 30 września 2012 było 130 dni roboczych, co daje trochę ponad 32 zgłoszenia dziennie.

$$AVG\,\frac{SR}{DZIEN} = \frac{4201}{130} = 32{,}32 \tag{9.1}$$

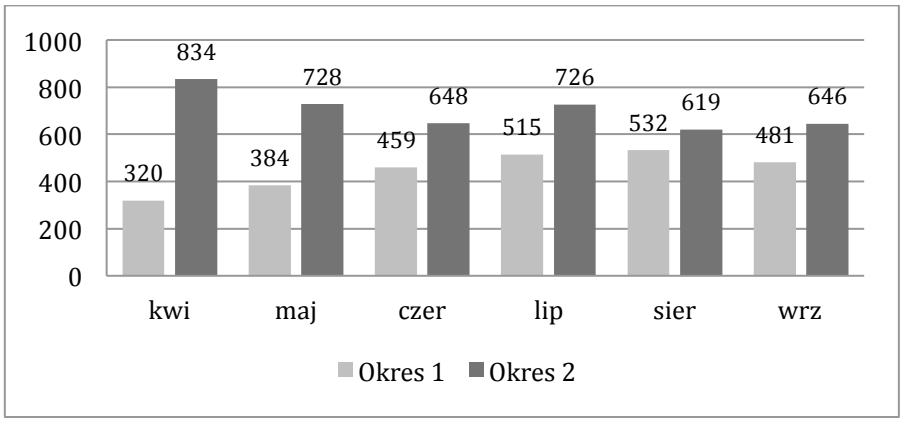
Avena w O1 zrealizowała mniej zgłoszeń roboczych niż w O2 - liczba rozwiązanych SR wyniosła 2691. Liczba dni roboczych w Okresie 1 wyniosła 131 dni. Pozwala to obliczyć średnią dzienną liczbę zgłoszeń.

$$AVG\,\frac{SR}{DZIEN} = \frac{2691}{13\,1} = 20{,}54 \tag{9.2}$$

Taka liczba zgłoszeń oznacza, że Avena obsłużyła w okresie O2 zdecydowanie więcej niż rok wcześniej - wzrost wyniósł aż 56,11%.

$$\frac{O2}{O1} * 100\% = \frac{4201}{2691} * 100\% = 156{,}11\% \tag{9.3}$$

Oczywiście, liczba zgłoszeń, a co za tym idzie - średnia liczba zgłoszeń - nie jest jedyną różnicą między badanymi okresami. Jedna wartość liczbowa dotycząca okresu sześciu miesięcy nie jest jeszcze miarodajna. Znacznie bardziej interesujący jest rozkład liczby zgłoszeń w poszczególnych miesiącach, który został zobrazowany poniżej na rysunku nr 9.2.



Rysunek 9.2. Liczba SR w O1 i O2.

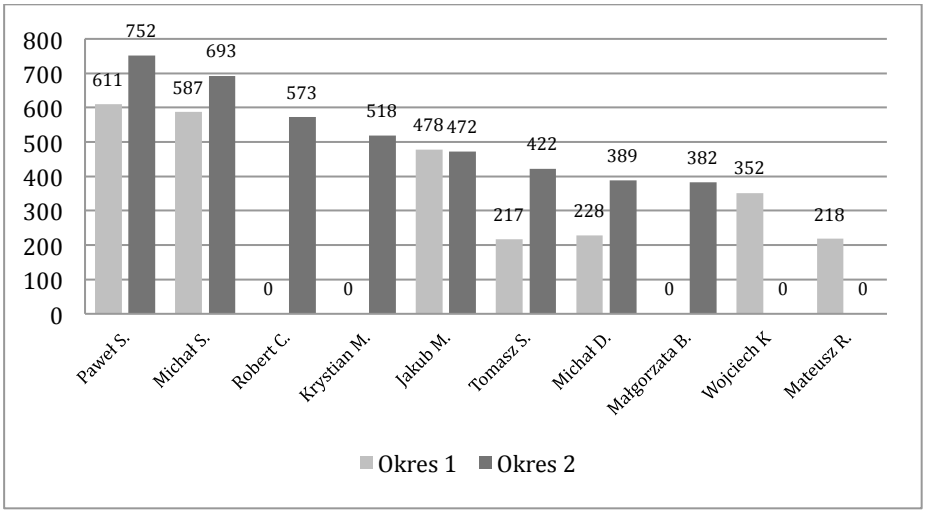
Na wykresie można dostrzec znaczącą różnicę między wartościami uzyskanymi w miesiącu kwietniu. W ciągu pierwszego miesiąca od wdrożenia systemu BI badana organizacja obsłużyła ponad dwukrotnie mniej SR niż rok później. W kolejnych miesiącach ta różnica zaczyna się zacierać. W sierpniu jest już relatywnie niewielka i wynosi tylko 16,35%. Widać, że firma potrzebowała czasu na oswojenie się z nowym systemem, ale z pewnością nie jest to jedyny powód. Na komputerach klientów musiały zostać zainstalowane programy-agenty wykorzystywane do pracy całości systemu, a konkretnie do zbierania danych. To oczywiście musiało potrwać. Podobnie klienci musieli się przyzwyczaić do nowego systemu zgłoszeń. Widać, że po kilku miesiącach różnice między O1 a O2 znacznie się zmniejszyły.

9.3.3 Porównanie liczby administratorów oraz ich efektywności

Na początku 2011r. do obsługi SR było przydzielonych 7 administratorów, a więc jednego mniej niż rok później. Pięć osób z tej grupy dalej pracuje w firmie. Do kwietnia 2012r. zatrudniono trzech nowych administratorów. Tam gdzie to było możliwe, zestawiono liczbę SR z obu okresów, w pozostałych wypadkach przedstawiono liczbę SR administratora tylko w jednym okresie. Liczbę zamkniętych SR przez administratorów w obu okresach przedstawia rysunek nr 9.3.

Z wykresu można odczytać, że w 4/5 przypadkach administratorzy, którzy zostali w firmie, w roku 2012 odpowiedzieli na więcej zgłoszeń niż rok wcześniej. W jednym wypadku jest odwrotnie, ale ta różnica jest bardzo nieznaczna

(jedynie 6 SR, czyli mniej więcej dzień pracy). Oczywiście nie można porównać jednego okresu do drugiego tylko na podstawie tych pięciu pracowników, którzy dalej pracują, ale trzeba spojrzeć szerzej. W tym celu zmierzono średni czas wykonywanych czynności (rzeczywiście zarejestrowany czas pracy) u wszystkich pracowników w obu okresach. Czas ten został uzyskany poprzez podzielenie sumy czasów wykonywanych czynności (system BI zapisuje te dane) poprzez liczbę roboczogodzin dostępnych w tych okresach.



Rysunek 9.3. Porównanie liczby SR zamkniętych przez administratorów.

Poprzez zestawienie danych dotyczących średniego dziennego czasu pracy w obu okresach (nie zamieszczone w niniejszym rozdziale ze względu na ograniczenia jego objętości) można było łatwo ustalić, którzy pracownicy się poprawiają, a którzy nie. Zauważono na przykład, że w O1 dwójka pracowników miała szczególnie niski dzienny czas pracy (trochę ponad 2 godziny dziennie). To właśnie te osoby zostały zwolnione. Można wysnuć wniosek, iż pracownicy spędzają na rozwiązywaniu problemów więcej czasu w O2 niż w O1. Nie jest to dziwne, zważywszy na to, że liczba SR się zwiększyła. Warto więc dołączyć do analizy tabelę, która zestawia kilka wskaźników jednocześnie: liczba zamkniętych SR, średni dzienny czas pracy, liczba SR/h, a także pokaże procentowy przyrost tych wartości. Zasadne jest sporządzić taką tabelę porównawczą tylko dla tych pracowników, którzy pracowali w obu badanych okresach.

Tabela 9.1. Wskaźnik efektywności Δ SR - Δ t.

Admin.	**Śr. t O1 [h]**	**Śr. t. O2 [h]**	**Δt**	**SR O1**	**SR O2**	**ΔSR**	**Efektywność Δ SR - Δ t**
Paweł S.	4,97	6,11	22,94%	611	752	23,08%	0,14%
Michał S.	5,18	5,60	8,11%	587	693	18,06%	9,95%
Jakub M.	4,34	5,77	32,95%	478	472	-1,26%	-34,20%
Tomasz S.	3,61	2,41	-33,24%	217	422	94,47%	127,71%
Michał D.	4,60	5,68	23,48%	228	389	70,61%	47,14%
			10,85%			40,99%	30,15%

Tabela 9.1 obrazuje, jak zmieniła się wydajność pracowników między dwoma okresami. Po pierwsze zestawia czasy potrzebne do zamknięcia SR i pokazuje, czy administrator pracuje więcej czy mniej. Wskaźnik Δt sam w sobie nie mówi jednakże, czy to dobrze czy źle - przecież pracownik może pracować mniej, a mieć większą wydajność. Żeby ową wydajność obliczyć, następne trzy kolumny porównują, jak zmieniła się liczba rozwiązanych problemów (ΔSR). Ostatnia, najważniejsza kolumna, to wskaźnik efektywności pracownika. Zestawia różnicę zmiany czasu potrzebnego do wykonania zadania i zmianę liczby zamkniętych SR. W ten można określić tempo wzrostu wydajności administratora. Jeżeli wartość w tej kolumnie jest równa 0, to znaczy, że wzrost średniego czasu pracy jest wprost proporcjonalny do wzrostu liczby SR. Jeżeli wartość jest dodatnia, to administrator wykorzystuje swój czas efektywniej, jeżeli ujemna - mniej efektywnie. W tabeli widać, że 4/5 administratorów rok po wdrożeniu BI zwiększyło swoją wydajność.

9.3.4 Porównanie średnich czasów naprawy (MTTR)

Parametr średniego czasu naprawy MTTR (ang. *mean time to repair*) oznacza średni potrzebny czas, jaki jest potrzebny do naprawy urządzenia w wyniku wystąpienia awarii. Czas ten jest często określany w umowie i pozwala ocenić, czy firma zajmująca się konserwacją (w tym wypadku oczywiście badana firma rozwiązuje problemy związane z IT) wywiązuje się z zobowiązań zgodnie

z oczekiwaniami (tzw. umowy SLA – *Service Level Agreement*). Tak też jest w tym wypadku - klienci oczekują, że ich problemy zostaną rozwiązane w określonym czasie. Jeżeli czas nie jest narzucony przez SLA, to manager przy przydzielaniu zdania decyduje jak duży jest MTTR do konkretnego zadania. Po rozwiązaniu problemu zgłoszenie musi być wpierw sprawdzone i dopiero wtedy zamknięte. Zbadano średni czas zamknięcia i sprawdzenia oraz MTTR w O1, żeby wykazać, czy firma wywiązuje się ze swoich zobowiązań w odpowiednim czasie. W O2 w 4/6 wypadkach przekraczała MTTR [Sta11].

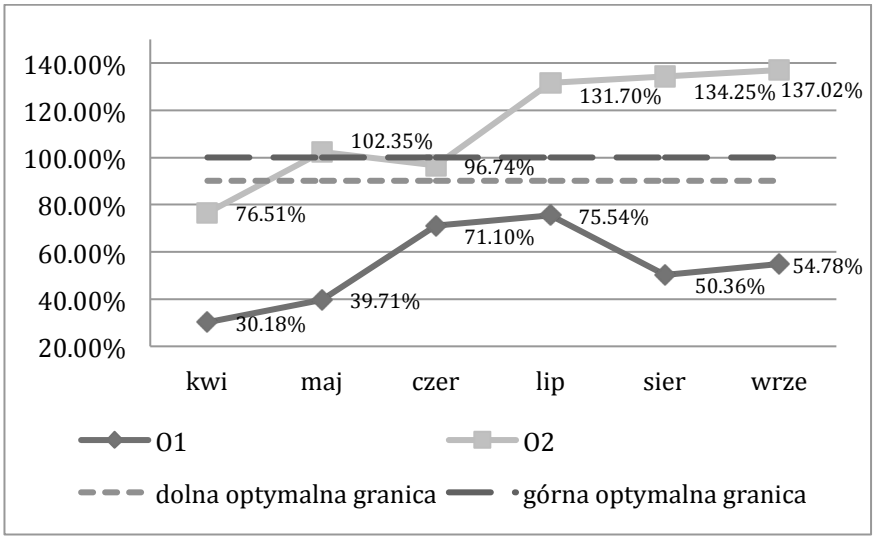
W O1 Avena w każdym jednym miesiącu zmieściła się w oczekiwanym czasie zamknięcia i sprawdzenia zadania. W kwietniu 2011 firma dysponowała olbrzymim zapasem czasu, wynoszącym ponad 1100 roboczogodzin. Kierownictwo zorientowało się, że sporo czasu zostało zmarnowane i stopniowo zmniejszało MTTR. Podobnie klienci doszli do wniosku, że czas, jaki przydzielili na wykonanie usługi jest za przeszacowany i go zredukowali. Niemniej w O1 w dalszym ciągu MTTR pozostał za duży, firma dysponowała nadmiarem czasu. W O2 jest odwrotnie: firma nie nadążała z pracą. Gdyby w O1 Avena podpisała SLA, bez problemu dotrzymałaby warunków umowy. W O2 to już jest niemożliwe.

W teoretycznie idealnej sytuacji, MTTR powinien być równy lub większy tzw. czasowi zamknięcia i sprawdzenia (CZiS, czyli realny czas kompletnego obsłużenia zgłoszenia). Oczywiście im mniejsza różnica tych czasów tym efektywniej czas pracy jest wykorzystywany. Czas potrzebny na zamknięcie i sprawdzenie powinien wynosić około 90-95% MTTR, żeby w razie nagłego wypadku dysponować pewnym zapasem czasowym. Bez uwzględnienia takiego zapasu czasowego można przyjąć górną granicę jako 100% (czyli, że czas zamknięcia i sprawdzenia SR jest dokładnie taki sam jak MTTR). W siedmiu badanych miesiącach wartość CZiS/MTTR okazała zbyt mała, w czterech za duża, a tylko w jednym jest odpowiednia (czerwiec 2012). Uzupełnieniem tabeli jest rysunek 9.4.

Wykres pokazuje wyraźnie, jak bardzo odbiegają wartości wskaźnika CZiS/MTTR od normy. Wartości wskaźnika dla obu okresów O1 i O2 powinny znajdować się między dolną a górną granicą - wtedy wartość wskaźnika byłaby optymalna. Można zaobserwować, że w obu okresach występują dwie tendencje: w O1 CZiS/MTTR był generalnie za niski, w O2 - generalnie za wysoki.

9.3.5 Porównanie obsługiwanych firm i stosunek otwartych SR do zamkniętych SR

Wzrost liczby SR w O2 w stosunku do O1 nie jest przypadkowy. Firma pomiędzy tymi okresami zdobyła nowych klientów, żadnego nie straciła. Liczba klientów w O1 wynosiła 29, w O2 36. Wśród pierwszej dziesiątki największych klientów 7 z nich zgłosiło ponad 100 zgłoszeń, pozostali - mniej. Co ciekawe, 8 największych klientów w O1 w O2 również znajduje się w pierwszej 10 najważniejszych klientów Aveny. Jest to o tyle ważne, że mimo pewnych niedoskonałości w pracy (o których wcześniej już pisano), klienci dalej decydują się na korzystanie z usług Aveny. Pokazuje to tabela 9.2. Przedstawia ona stosunek zgłoszeń otwartych do zamkniętych przez wszystkie firmy (klientów). Ponadto, pokazuje jak zmieniała się liczba zgłoszeń między badanymi okresami.



Rysunek 9.4. Wskaźnik CZiS/MTTR w okresach O1 i O2.

Tabela 9.2. Zmiana otwartych zgłoszeń każdego klienta.

Otwarte SR w O1	Otwarte SR w O2	Procentowa zmiana	
2691	4201	100,81%	Dla wszystkich klientów
2691	3759	99,32%	Bez uwzględnienia nowych klientów

Przeciętnie każda firma-klient zgłosiła o 100% więcej SR w O2 niż w O1 (co niekoniecznie znaczy, że całkowita liczba zgłoszeń między O1 i O2 również się podwoiła). Mimo, że ta wartość jest dla Aveny pozytywna, to autorzy wskazali kierownictwu ekstrema pojawiające się w danych szczegółowych. Pewne firmy niemalże przestały korzystać z usług Aveny (dlaczego? Wygasła umowa czy może klienci są niezadowoleni z usług?). Można powiedzieć, że zmiana liczby zgłoszeń przez każdą z firm jest wskaźnikiem zadowolenia z usług badanej organizacji. Według rozkładu normalnego, jeżeli firma X w jednym okresie zgłasza n SR, to w analogicznym okresie rok później powinna zgłosić mniej więcej tyle samo n (to założenie nie uwzględnia istotnych zmian, jakie mogą zajść w firmie - znaczny spadek wartości, masowe zwolnienia, budowa nowej siedziby i zatrudnienie nowych pracowników do pracy itp., które oczywiście mogłyby znacząco wpłynąć na liczbę zgłoszeń). Oczywiście, wzrost lub spadek liczby zgłoszeń może zależeć od zdarzeń czysto losowych. Prawdopodobieństwo wystąpienia takich sytuacji jest niestety niemożliwe do oszacowania, można jednak założyć, że istnieje taka sama szansa na wystąpienie zdarzeń losowych o negatywnym, jak i pozytywnym charakterze. Dlatego wpływ zdarzeń losowych został pominięty w tej analizie.

9.4 Ocena efektywności wdrożenia BI

9.4.1 Wybór wskaźników do oceny

Z wywiadów przeprowadzonych przez autorów wynika, że kierownictwo Aveny na pewnym etapie rozwoju firmy stwierdziło, że nie jest w stanie dłużej podejmować trafnych decyzji. Dlatego zdecydowano się na wdrożenie BI na początku 2011 roku. Organizacja między badanymi okresami zmieniała się pod wpływem działań wynikających z wyciągniętych wniosków na podstawie danych zgromadzonych i przeanalizowanych przez system BI (przynajmniej w jakiejś części). Mimo, iż nieznane są poszczególne pojedyncze decyzje, można zmierzyć ich wpływ poprzez szereg parametrów. Dzięki wypracowanym wskaźnikom (opisanych wcześniej) można ocenić, czy wdrożenie BI było uzasadnione, czy firma na podstawie uzyskiwanych informacji podjęła decyzje, które zwiększyły efektywność firmy.

Do oceny autorzy wybrali osiem wskaźników, które można zaliczyć jednej do dwóch kategorii:

— kierunkowe - mówią o kierunku zachodzących zmian, najważniejsza w nich nie jest wartość, ale znak.

— wartościowe - w ich przypadku najważniejsza jest sama liczba, która powinna mieścić w określonym przedziale. Ten przedział został wyznaczony subiektywnie przez autorów.

Wszystkie te wskaźniki zostały zebrane w tabeli 9.3, wraz z wyliczonymi wartościami oraz sugerowanymi normami dla każdego.

Tabela 9.3. Wartości wskaźników oceny

Lp	Wskaźnik	Akronim wskaźnika	Typ wskaźnika	Wartość	Norma/komentarz
1	Zmiana liczby SR	ΔSR	kierunkowy	+ 56,11%	Brak górnego limitu
2	Zmiana średniego dziennego czasu pracy administratora	ΔAAT	kierunkowy	+ 1h	Brak górnego limitu
3	Wartość średniego dziennego czasu pracy administratora	AAT	wartościowy	4,9h	0-8
4	Efektywność administratorów (tych, którzy pracowali w obu okresach)	Δ SR - Δ t	kierunkowy	+ 30,15%	Brak górnego limitu,
5	Wartość stosunku czasu zamknięcia i sprawdzenia do MTTR	CZiS/MTTR	wartościowy	113,10%	90-100%
6	Zmiana liczby obsługiwanych klientów	ΔKlient	kierunkowy	+ 24,13%	Brak górnego limitu
7	Zmiana różnicy otwartych i zamkniętych zgłoszeń	ΔO-ΔZ	wartościowy	-19	0
8	Utrzymanie klienta	%SRC	wartościowy	45%	75-100%

Wartość każdego ze wskaźników należy pokrótce skomentować i uzasadnić, oceniając jej charakter (pozytywny/negatywny). Każda wartość została przez autorów oceniona w skali 1-5 (gdzie 5 to wynik bardzo dobry, 1 – mierny). W kolumnie "komentarz" przedstawione zostało krótkie uzasadnienie, dlaczego wskaźnik uzyskał właśnie taką ocenę. Ocena wdrożenia systemu BI i efektów działań przeprowadzanych Avenę będzie średnią arytmetyczną ocen poszczególnych wskaźników.

Tabela 9.4. Ocena wartości wskaźników.

Wskaźnik	Komentarz	Ocena
ΔSR	Wzrost liczby zgłoszeń aż o 56% w ciągu roku świadczy o znacznym tempie rozwoju firmy.	5
ΔAAT	Wzrost o 1h (albo o 25,64%) mówi o tym, że administratorzy faktycznie pracują o godzinę więcej w ciągu dnia. To zmiana w dobrym kierunku, o ile AAT nie zostanie przekroczony.	5
AAT	Średni dzienny czas pracy administratora wynosi 4,9h. Przy założeniu, że dzień pracy trwa 8 godzin, 4,9h stanowi tylko 61,25% dnia roboczego. Wartość wskaźnika jest stanowczo zbyt niska; administratorzy nie wykorzystują w pełni swojego czasu pracy.	2
Δ SR - Δ t	Wartość wskaźnika Δ SR - Δ t równa 30,15% oznacza, że administratorzy, którzy pracowali w firmie w O1 i O2 nieproporcjonalnie podnieśli swoją wydajność w stosunku do oczekiwanej aż o ponad 30%. To bardzo dużo.	5
CZiS/MTTR	Wielkość wskaźnika CZiS/MTTR wynosząca 113,10% wskazuje, że firma nie mieści się w wyznaczonych czasach obsługi SR.. Z drugiej strony, MTTR był w O1 wyraźnie nieoszacowany (zawyżony). Zaostrzenie MTTR to zmiana pozytywna, ale nie przy zwiększającej się liczbie SR. Avena nie jest w stanie dotrzymać rygorystycznych MTTR określonych w SLA.	2
ΔKlient	Avena przy zatrudnieniu tylko jednego dodatkowego pracownika obsłużyła w O2 24,13% więcej klientów, niż w O1. To spory wzrost.	5
ΔO-ΔZ	Firma zamknęła o 19 więcej zgłoszeń niż przyjęła w O2, co znaczy, że musiała zamykać i sprawdzać zaległe zgłoszenia. Z drugiej strony, wartość 38 w O1 świadczyła o zbyt wolnej pracy i nadmiernym rozłożeniu zadań w czasie. Kierunek zmian pozytywny, ale wartość wskaźnika w dalszym ciągu nieprawidłowa.	3
%SRC	Ponad połowa (55%) klientów zgłosiła w O2 mniej SR niż w O1. Może to wynikać z sytuacji panującej u klientów, albo z poziomem świadczonych usług. Choć większość firm korzysta z usług w Aveny w mniejszym stopniu, żaden z klientów nie zaprzestał współpracy. Konieczne jest zbadanie ekstremów i ustalenie przyczyn występującego zjawiska.	2
		Ocena: 3,63 (72,6%)

Średnia arytmetyczna z przyjętych ocen wyniosła 3,63 co stanowi 72,6% wartości maksymalnej (5). Ostatecznie efektywność zmian, jakie zaszły w Avenie w badanych okresach została wyrażona w procentach. Dla celów późniejszej interpretacji i porównań przyjęto następujące referencyjne przedziały ocen (w procentach):

- <0,50>: ocena niedostateczna. Wartości poniżej 50% oznaczają, że zmiana nie niesie ze sobą żadnych, nawet najmniej pozytywnych skutków.
- (50,65): ocena dostateczna. Zmiany są niewielkie, ale pozytywne.
- <65,80): ocena dobra. Zmiany są raczej pozytywne, aczkolwiek nie wszystkie rozwiązania są dostatecznie dopracowane.
- <80,100>: ocena bardzo dobra. Zmiany wyraźnie pozytywne, niewiele elementów można poprawić.

Na podstawie tej skali można powiedzieć, że wynik 72,6% jest na tyle dobry, że daje podstawy do uznania wdrożenia systemu BI opłacalnym. Jednocześnie zindentyfikowane zostały te obszary działalności firmy, które nie działają dostatecznie efektywnie. Na podstawie przeprowadzonych analiz i porównań oraz wiedzy autorów o firmie można wskazać kilka trafnych i nietrafnych decyzji podjętych przez kierownictwo. Możliwe jest także wskazanie decyzji podjętych z właściwych przyczyn, które ze względu na zmianę pewnych czynników okazały się nie do końca słuszne:

- Udało się wykryć, którzy pracownicy pracują niewydajnie. Zostali oni zwolnieni i zastąpieni innymi, z których każdy pracuje wydajniej od poprzedników. Decyzja trafna.
- Poprzez grupowanie incydentów udało się zdefiniować pewne powtarzające się problemy. W firmie została wprowadzona polityka tzw. SIPów. Jeżeli jakiś problem się powtarza, któryś z pracowników (często szczebla kierowniczego) otrzymuje polecenie stworzenia SIPa, czyli artykułu trafiającego do bazy wiedzy, opisującego krok po kroku rozwiązanie problemu. Efektem jest zwiększenie wydajności pracowników i możliwość obsługi większej liczby zgłoszeń przez administratora. Decyzja trafna.
- Zredukowany został MTTR. Wykryto, że MTTR w O1 były zawyżone, przez co pracownicy mieli za dużo czasu na wykonanie zadania, więc obniżono dopuszczalny czas zamknięcia i sprawdzenia SR. Ponieważ pracownicy dysponowali nadmiarem czasu, możliwe było podpisanie umów z większa liczbą klientów, co miało pozwolić na lepsze wykorzystanie czasu pracowników. Niestety, dotychczasowi klienci również stwierdzili, że problemy kierowane do Aveny można rozwiązywać

szybciej, więc obniżyli MTTR. W rezultacie Avena obsługuje za dużo klientów w stosunku do swoich możliwości i nie mieści się w MTTR. Decyzja o wewnętrznym obniżeniu MTTR była zatem słuszna w O1, ale ze względu na zmianę czynników zewnętrznych niesie ze sobą negatywne skutki w O2. Decyzja częściowo trafna.

Podsumowując, można powiedzieć, że ocena skuteczności wdrożenia systemu BI rzędu 72% jest uzasadniona. Firmie dzięki tej inwestycji udało się wykryć kilka problemów i poprzez podjęcie decyzji opartych na konkretnych informacjach zareagować na nie. Widać, że w firma od początku wdrożenia na początku O1 do końca O2 znacznie się rozwinęła. Jednakże bardzo szybki rozwój przyniósł ze sobą nowe problemy, którym firma musi teraz sprostać. Avena jeszcze nie jest w stanie wykorzystać systemu, którym dysponuje w 100%. Świadczy o tym fakt, że kierownictwo przegapiło kilka momentów, w których należało wykryć problem i jakoś na niego zareagować. Przykładowo, zatrudnienie jednego lub dwóch administratorów mogłoby rozwiązać problem przekraczania czasów MTTR, a co za tym idzie ograniczyłoby niezadowolenie klientów. Niemniej, mimo pewnych niedoskonałości w funkcjonowaniu organizacji, decyzje podejmowane przez kierownictwo Aveny w większości wypadków okazały się trafne.

9.5 Podsumowanie

W rozdziale przedstawiono metodę oceny wdrożenia technologii informatycznej – systemu klasy Business Intelligence - w organizacji IT. System wdrożono, jednak bez wcześniejszych analiz potencjalnych skutków tej zmiany. Autorzy dokonali takiej analizy poprzez porównanie danych dostępnych dla dwóch okresów. Dało to możliwość określenia, w których aspektach swojego działania firma zyskała, a w których straciła. Okazało się, że niewątpliwym sukcesem jest zwiększenie liczby obsługiwanych SR o ponad połowę, zwiększenie wydajności pracowników o 20-30% poprzez obserwacje, standaryzacje i tworzenie baz wiedzy. Avena obsługuje także 24% klientów więcej niż poprzednio. Te wyniki są efektem podejmowania trafnych decyzji, co stało się możliwe dzięki wykorzystaniu narzędzi Business Intelligence. Pojawiło się przy tym kilka zagrożeń: przekraczanie dopuszczalnych czasów MTTR czy podjęcie współpracy ze zbyt wielką liczbą klientów w odniesieniu do swojego potencjału

Należy podkreślić, że ocena decyzji o wdrożeniu systemu BI została podjęta w oparciu o założenie, że w badanej firmie nic, poza wdrożeniem samego BI, się nie zmieniło. Oczywiście, to założenie jest nie do końca słuszne - każda

organizacja zmienia się w czasie. Przeprowadzone analizy zostały w oparciu wyłącznie o konkretne wartości liczbowe pochodzące z odpowiednich tabel baz danych firmy. Jednakże nie wszystko, co zmienia się w firmie, znajduje w nich odwzorowanie. Dlatego przyjęto, że brane pod uwagę są tylko te dane, które znajdują swoje liczbowe odzwierciedlenie, a wszelkie zmiany w firmie są wynikiem wdrożenia systemu BI. Przy takim właśnie założeniu można powiedzieć, że zastosowanie systemu BI pozwoliło na ograniczenie problemów związanych z podejmowaniem decyzji i poskutkowało polepszeniem warunków podejmowania decyzji, co pozytywnie wpłynęło na pracę całej organizacji. Celem autorów jest obecnie weryfikacja działania modelu dla innych organizacji wsparcia, także dla innych technologii niż Business Intelligence.

Bibliografia

[Alt99] Alter S. Information Systems: a management perspective, 1999.

[Gri98] Griffin R. W. Podstawy Zarządzania Organizacjami, 1998.

[Gar12] Gartner - witryna, http://www.gartner.com [2012.11.05].

[Jan08] Januszewski A. Funkcjonalność informatycznych systemów zarządzania Tom 2 System Business Intelligence, 2008.

[Lit12] Litka M. Analiza decyzji o wdrożeniu systemu Business Intelligence w organizacji wsparcia IT -projekt modelu ocenowego. 2012.

[PF12] Panorama Firm, http://panoramafirm.pl/szukaj?k=Helpdesk &l=trójmiasto, [2012.11.16].

[SM98] Samuelson W.F., Marks S.G. Ekonomia menedżerska, 1998.

[Sta11] Stanley S. MTBF, MTTR, MTTF & FIT - Explanation of Terms. 2011.

[Tar86] Targalski J. Podejmowanie decyzji. *Organizacja i zarządzanie*, 1986.

[Ros01] Radosiński E.. Systemy informatyczne w dynamicznej analizie decyzyjnej, 2001.

[OS10] Orłowski C., Sitek T. Supporting Management Decisions with Intelligent Mechanisms of Obtaining and Processing Knowledge. *Knowledge-Based and Intelligent Information and Engineering Systems,* 2010.

CZĘŚĆ IV

UTRZYMANIE SYSTEMÓW INFORMATYCZNYCH

Rozdział 10

Pozyskiwanie struktury obiektowej z kodu zarządzanego przy wykorzystaniu metod inżynierii odwrotnej

Jedną z wielu cech oprogramowania napisanego w językach kompilowanych do kodu pośredniego jest możliwość skutecznego wykorzystania metod inżynierii odwrotnej do podglądu struktury oprogramowania a nawet uzyskania ekwiwalentu kodu źródłowego. Celem rozdziału jest porównanie metod inżynierii odwrotnej, które umożliwiają odtworzenie struktury obiektowej oprogramowania z kodu zarządzanego. Pozyskane dane posłużą do dalszej analizy oprogramowania pod względem ogólnie rozumianej jakości z uwzględnieniem dobrych praktyk oraz stosowania wzorców projektowych.

Dziedzina informatyki uznawana jest za jedną z najszybciej rozwijających się dziedzin w ostatnich latach. Informatyka łączy się z wieloma innymi dziedzinami działalności ludzkiej. W każdym obszarze zastosowań konieczne jest użycie odpowiedniej maszyny, którą może być potężny komputer, tablet, telefon, czy też mały pilot pełniący rolę breloczka przy kluczach. Każde z owych urządzeń byłoby nieużyteczne, gdyby nie został napisany dla nich odpowiedni program. Rzemiosło jakim jest programowanie, daje każdemu programiście nieskończenie wiele możliwości na realizację wymaganego programu, co w konsekwencji przekłada się na różną jakość kodu programu. Dysponując kodem źródłowym danego programu można dokonać jego interpretacji i do uzyskanych wyników zastosować jedną z metod oceny kodu programu. W sytuacji, gdy kod źródłowy jest niedostępny można wciąż próbować wyznaczyć jakość oprogramowania, dzięki zastosowaniu inżynierii odwrotnej.

10.1 Zastosowania inżynierii odwrotnej

Inżynieria odwrotna, nazywana zamiennie inżynierią wsteczną, posiada swoje zastosowanie w wielu sztukach inżynierskich. W ogólnym ujęciu polega na badaniu przedmiotu, którym może być element mechaniczny, układ elektroniczny czy też oprogramowanie, w celu poznania zasady działania oraz sposobu wykonania danego przedmiotu. Jednym z przykładów tego procesu jest powstanie kodeka „DivX ;-)” na podstawie innego kodeka MPEG 4 firmy Microsoft w 1999 roku.

Inżynieria odwrotna w oprogramowaniu dotyczy technik przekształcenia kodu maszynowego do kodu niskiego poziomu (assemblera), co nazywane jest dezassemblacją [Kas04] lub kodu pośredniego do kodu wysokiego poziomu, co

określane jest dekompilacją. Popularne zastosowania inżynierii odwrotnej w oprogramowaniu to między innymi: zaawansowane testowanie (debugowanie), poznanie sposobu działania kodu i algorytmów, poznanie struktury kodu, modyfikacja oprogramowania, pozyskanie wymagań funkcjonalnych oraz niefunkcjonalnych, przygotowanie dokumentacji opisującej architekturę systemu.

Należy zwrócić uwagę, iż niektóre z wymienionych zastosowań są powiązane z łamaniem prawa a szczególnie praw autorskich. Nie należy jednak negatywnie traktować inżynierii odwrotnej. Jest to narzędzie, więc wszystko zależy od osoby, która go używa oraz do jakich celów zostanie wykorzystane.

Żaden proces inżynierii odwrotnej nie gwarantuje pełnej skuteczności, ponieważ istnieją różne metody zabezpieczeń. Niektóre z nich to [Ros09]: sprawdzanie integralności danych programu, kompresja i szyfrowanie plików binarnych, a następnie dekompresja oraz deszyfrowanie po uruchomieniu, zaciemnianie kodu, zabezpieczanie przed debugerami.

Kod zarządzany jest częścią platformy programistycznej .NET należącej do firmy Microsoft. Platforma .NET jest wirtualnym środowiskiem wykonawczym wraz ze zbiorem podstawowych bibliotek. Efektem kompilacji programu napisanego w jednym z języków obsługiwanych przez .NET nie jest kod maszynowy, lecz kod pośredni, a dokładniej wspominany już kod zarządzany (ang. *managed code*). Przy pierwszym uruchomieniu wymagany fragment kodu zarządzanego konwertowany jest na kod maszynowy, zależny od konkretnej architektury. Przy ponownym wywołaniu jest wykorzystywany kod maszynowy, który wcześniej został już przekonwertowany.

Podstawowe elementy tej platformy są następujące [Tro09]:

- CLR (od ang. *Common Language Runtime*) jest to wspólne środowisko uruchomieniowe, które odpowiada za ładowanie i zarządzanie typami .NET, zarządza pamięcią i wątkami, sprawdza poziom zabezpieczeń itd.,
- CTS (od ang. *Common Type System*) jest to wspólny system typów, dodatkowo jest specyfikacją opisującą wszystkie możliwe typy danych oraz konstrukcje programistyczne obsługiwane przez środowisko uruchomieniowe, określa zasady komunikacji między typami,
- CLS (od ang. *Common Language Specification*) tak określana jest specyfikacja wspólnego języka, definiuje podzbiór wspólnych typów oraz konstrukcji programistycznych precyzujących zgodność z językami programowania obsługiwanymi przez .NET,

— CIL (od ang. C*ommon Intermediate Language*) jest to wspólny język pośredni, czyli inaczej język w jakim zapisane są pliki binarne platformy .NET.

Plik wykonywalny .NET oprócz kodu zarządzanego zawiera również nagłówek oraz obraz PE (od ang. *Portable Execution*) [HEW11], znany z natywnych aplikacji dla Windows. Dzięki czemu rozpoczyna się ładowanie środowiska uruchomieniowego .NET i wykonywanie przez nie żądanej aplikacji. Dodatkowo do zestawów .NET dołączane są metadane (tzw. manifest), które zawierają informacje o wymaganych modułach natywnych, innych zestawach .NET i ich wersjach.

10.1.1 Opis metod inżynierii odwrotnej dla kodu zarządzanego

Rozważania nad metodami inżynierii odwrotnej dla kodu zarządzanego należy rozpocząć od tego, iż firma Microsoft projektując platformę .NET przewidziała również taki proces i przygotowała podstawowe rozwiązanie przeznaczone do tego celu, czyli refleksję znajdującą się w przestrzeni nazw System.Reflection. Oprócz tego społeczność deweloperów oraz firmy komercyjne tworzą własne narzędzia dające bardzo duże możliwości. Same narzędzia nie będą podlegały dalszym rozważaniom a jedynie metody oraz rozwiązania z jakich korzystają w procesie inżynierii odwrotnej.

Wybór metod został z góry ograniczony do takich, które spełniają podstawowe kryteria porównania, natomiast sam opis tych metod został ograniczony do najważniejszych aspektów mających znaczenie właśnie w tym przypadku. Inne, bardzo szerokie obszary zastosowań oraz cechy wybranych metod nie będą omawiane.

10.1.2 Refleksja

Jak już zostało wspomniane, mechanizm refleksji jest dostarczany jako część platformy .NET, na równi z wieloma innymi przestrzeniami nazw, jak np. System.Console umożliwiającą operacje na konsoli.

Podstawowym celem refleksji jest dostarczanie informacji o zestawach .NET, niezależnie czy analizowany zestaw należy do frameworka .NET czy jest dziełem niezależnych twórców.

Podstawowe możliwości refleksji to:

— wczytywanie zestawów z różnych źródeł, czyli wewnętrzne zestawy .NET, biblioteki zewnętrzne, GAC,

— odczytywanie informacji zawartych w manifeście, czyli wersje, wymagane biblioteki itd.,
— odczytywanie informacji o strukturze analizowanego zestawu (również z dostępem do elementów prywatnych), tj. typach, metodach, właściwościach, polach,
— wykonywanie kodu programu.

Mechanizm refleksji jest wykorzystywany w różnych zastosowaniach, które nie są bezpośrednio związane z inżynierią odwrotną. Przykładem tego są mechanizmy wtyczek oraz rozszerzeń stosowanych w wielu programach. W środowiskach deweloperskich podpowiadanie możliwych do wyboru klas, metod oraz możliwość dodawania dodatkowych bibliotek i komponentów do projektu, to również zasługa mechanizmu refleksji. Innym popularnym pośrednim wykorzystaniem refleksji jest późne wiązanie.

Bardzo dużą zaletą refleksji jest wsparcie od firmy Microsoft, przez co zagwarantowana jest pełna zgodność ze środowiskiem .NET oraz rozwój wraz z kolejnymi wersjami frameworka .NET. Cechą, którą można uznać za wadę, są zabezpieczenia CAS (od ang. *Code Access Security*) nakładane na kod programu, do której refleksja musi się dostosować. Dodatkowo refleksja nie daje możliwości modyfikacji zestawów .NET.

Struktura przestrzeni nazw System.Reflection jest w pełni obiektowa oraz zgodna z konwencjami przyjętymi w całej platformie .NET. Jej wykorzystanie polega na załadowaniu żądanego zestawu .NET, co jest możliwe na wiele sposobów. Przykładowe wykorzystanie przedstawia listing 10.1, który zawiera kolejno wczytanie zestawu, odszukanie w nim konkretnego typu a następnie pobranie dodatkowych informacji o metodach, polach oraz właściwościach. Szczegółowe informacje dotyczące refleksji można znaleźć w dokumentacji Microsoft Developer Network [MSDN].

```
1.  Assembly assembly = Assembly.LoadFile(path);
2.  Type type = assembly.GetType("ConcreteType");
3.  string typeName = type.Name;
4.  MethodInfo mi = type.GetMethod("ConcreteMethod");
5.  string methodName = mi.Name;
6.  FieldInfo fi = type.GetField("ConcreteField");
7.  string fieldName = fi.Name;
8.  PropertyInfo pi = type.GetProperty("ConcreteProperty");
9.  string propertyName = pi.Name;
```

Listing 10.1. Przykładowy kod wykorzystujący mechanizm refleksji.

10.1.3 Projekt Mono.Cecil

Projekt Mono.Cecil jest częścią większego projektu o nazwie Mono [MON], dodatkowo Mono to open source. Mono.Cecil jest biblioteką dla platformy .NET umożliwiającą sprawdzanie oraz badanie zestawów .NET. Biblioteka tworzona jest przez niezależnych twórców, przez co istnieje ryzyko niekompatybilności z coraz nowszymi wersjami .NET.

W zakresie inżynierii odwrotnej zapewnia przynajmniej identyczne funkcje co mechanizm refleksji. Oprócz dodatkowych możliwości, jego twórcy zadbali, aby dane rozwiązanie było odporne na ograniczenia jakie narzucane są na refleksję.

Istotną cechą Mono.Cecil jest możliwość modyfikowania zestawów .NET. Owa możliwość zagwarantowała projektowi dużą popularność w zastosowaniach do szeroko rozumianego testowania zabezpieczeń oprogramowania. Narzędzia korzystające z tego projektu, jak np. ILSpy [ILS] dają możliwość podglądu ekwiwalentu badanego kodu programu z możliwością wyboru języka C#, VB bądź CIL.

Struktura Mono.Cecil zaprojektowana jest w sposób obiektowy z zachowaniem wielu podobieństw do mechanizmu refleksji. Listing 10.2 przedstawia kod wykorzystujący Mono.Cecil w analogiczny sposób, podobnie jak dla mechanizmu refleksji. Pierwsza różnica, zauważalna po przeanalizowaniu obu listingów, to odmienne przyrostki w nazwach głównych klas. Większe różnice zachodzą w elementach składowych owych klas. Refleksja umożliwia dostęp do pól badanego typu poprzez odpowiednie metody, natomiast Mono.Cecil umożliwia to poprzez właściwości. Podobne różnice występują również dla pozostałych elementów składowych badanego typu. Kolejne różnice zachodzą w zastosowanych typach danych. Przykładowo Mono.Cecil dla zmiennych lokalnych znajdujących się w ciele badanej metody, wykorzystuje po prostu klasę opisującą typy, czyli TypeReference. Jednakże refleksja dostarcza w tym miejscu zupełnie nową klasę LocalVariable.

10.1.4 Inne narzędzia

Wymienione wcześniej określenia, refleksja oraz Mono.Cecil, to dwa najważniejsze rozwiązania umożliwiające zastosowanie ich do pozyskiwania struktury obiektowej. Inne, ze względu na małą popularność, zakończony rozwój lub niespełnienie podstawowych kryteriów, nie mogą zostać do tego wykorzystane.

Warto wspomnieć o narzędziach, tzw. dekompilatorach, które również muszą wykorzystywać mechanizmy inżynierii odwrotnej. Narzędzia takie jak:

IlSpy, Telerik JustDecompile oraz DevExtras .NET CodeReflect wykorzystują Mono.Cecil. Jedynie w przypadku JetBrains dotPeak uzyskane informacje wykazały, iż wykorzystywane jest inne rozwiązanie.

```
1.  AssemblyDefinition assembly = AssemblyDefinition.ReadAssembly(path);
2.  ModuleDefinition mod = assembly.Modules.First();
3.  TypeDefinition type = mod.GetType("ConcreteType");
4.  string typeName = type.Name;
5.  MethodDefinition md = type.Methods.Where(q => q.Name == "Con-
    creteMethod").First();
6.  string methodName = md.Name;
7.  FieldDefinition fd = type.Fields.Where(q => q.Name == "Con-
    creteField").First();
8.  string fieldName = fd.Name;
9.  PropertyDefinition pd = type.Properties.Where(q => q.Name == "Con-
    creteProperty").First();
10. string propertyName = pd.Name;
```

Listing 10.2. Przykładowy kod wykorzystujący Mono.Cecil

10.2 Porównanie

10.2.1 Cel porównania

Podstawowym celem porównania jest wykazanie większej skuteczności jednej z metod w pozyskiwaniu struktury obiektowej z oprogramowania, tj. informacji o typach oraz ich elementach składowych, przy zachowaniu jak najkrótszego czasu przetwarzania. Pozyskane dane, w dalszych krokach, powinny zostać przekształcone do odpowiednej reprezentacji formalnej, np. grafów [SG13] bądź ontologii [DE05]. Następnie dane reprezentowane w sposób formalny mogą podlegać dalszemu zautomatyzowanemu przetwarzaniu oraz analizie. Wynik owej analizy jest uzależniony od kryteriów wprowadzanych dynamicznie. Planowane wykorzystanie całości to wyszukiwanie zaimplementowanych wzorców projektowych w kodzie programu a następnie ich weryfikacja oraz sprawdzenie poprawności. Wybranie odpowiedniej metody jest bardzo istotne, ponieważ to od niej będzie rozpoczynał się cały proces.

Inżynieria odwrotna może budzić niejednoznaczne odczucia związane z jej stosowaniem. W przedstawionym celu nie ma to jednak znaczenia, ponieważ analizie będzie podlegała struktura obiektowa a nie konkretne implementacje rozwiązań. Zaciemnianie kodu także nie jest przeszkodą w jego analizowaniu, ponieważ nie wpływa to na strukturę.

Dodatkowe kryteria przyjęte w porównaniu:

- skalowalność jest to możliwość analizowania zarówno małych jak i dużych zestawów .NET,
- przenośność w obrębie systemu Windows i platformy .NET, bez potrzeby instalowania dodatkowych składników,
- łatwość implementacji , czyli łatwe i intuicyjne API,
- stabilność inaczej obsługa sytuacji niestandardowych bez potrzeby przerywania całego procesu,
- zakres to dostęp do zawartości prywatnej oraz ciała metod.

10.2.2 Środowisko porównawcze

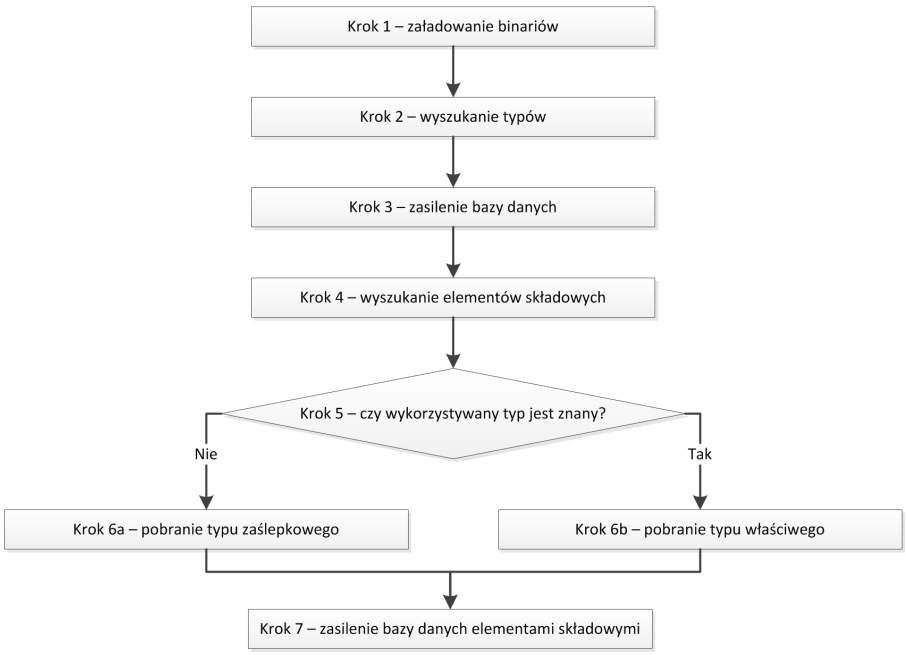
Na potrzeby przeprowadzenia eksperymentu została przygotowana aplikacja napisana w technologii .NET, której działanie polega na przeanalizowaniu wskazanych binariów a następnie zapisaniu uzyskanych informacji w bazie danych.

Schemat blokowy opisujący działanie aplikacji został przedstawiony na rysunku 10.1. Jest on taki sam niezależnie od wybranej metody inżynierii odwrotnej. Opis kolejnych kroków:

- krok 1 występuje po wskazaniu jednego lub kilku plików binarnych następuje załadowanie ich przez wybraną metodę inżynierii odwrotnej,
- krok 2 to przeanalizowanie zestawów oraz sklasyfikowanie typów,
- krok 3 to zasilenie bazy danych znalezionymi typami,
- krok 4 to ponowne przeanalizowanie zestawów oraz wyszukanie wszystkich metod, właściwości, pól i relacji,
- krok 5 to instrukcja warunkowa, tzn. czy dla każdego znalezionego elementu w kroku 4 znany jest typ tego elementu (np. typ zwracany przez metodę, typ przyjmowany w parametrze, typ pola itp.),
- krok 6a w tym przypadku typ nie jest znany, więc należy pobrać typ zaślepkowy (ang. *fake*), np. nieznana klasa, nieznany typ wyliczeniowy,
- krok 6b tu typ jest znany, więc wystarczy go odszukać w bazie danych,
- krok 7 to zasilenie bazy danych elementami występującymi w każdym typie.

Komentarza wymaga krok 2 i 4, a następnie krok 5 oraz 6. W pierwszym przypadku konieczne było rozdzielenie procesu analizy na dwa etapy, celem pierwszego etapu jest pobranie wszystkich typów, natomiast w drugim etapie następuje uzupełnienie tych typów o elementy składowe (metody, pola, właściwości, relacje). Głównym celem tworzenia nowych typów jest ich wykorzysta-

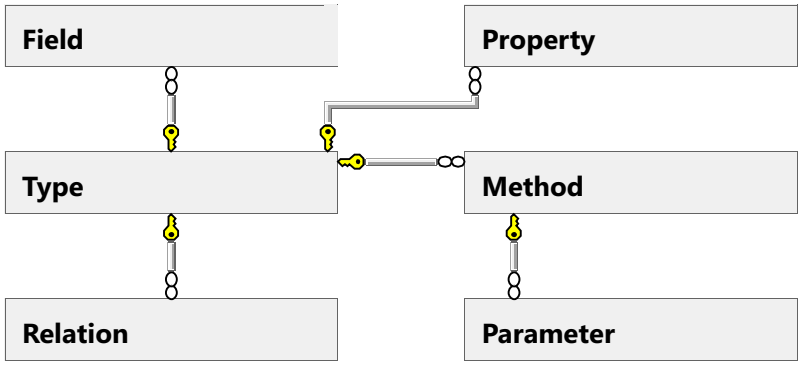
nie przez inne typy. Z tej przyczyny pojawiał się problem w wymienionym drugim etapie, gdy program podczas analizy natrafił na typ (np. jedno z pól było konkretnego typu) a ten typ nie został jeszcze przeanalizowany, przez co program nie mógł znaleźć go w bazie danych. W przypadku kroków 5 oraz 6 podział był konieczny ze względu na to, iż dla niektórych typów nie można uzyskać żadnej informacji. Wtedy pobierany jest typ zaślepkowy, oraz przewidziane zostały odpowiedniki: klas, interfejsów, typów wyliczeniowych, generycznych, bądź też wartościowych. Przypadek taki może wystąpić w wielu sytuacjach, np. jako zabezpieczenie przed inżynierią odwrotną lub też przy braku dostępu do powiązanych zestawów.



Rysunek 10.1. Schemat blokowy opisujący działanie aplikacji testowej

Rysunek 10.2. przedstawia uproszczony model bazy danych. Na rysunku nie zostały uwzględnione tabele KindOfType, AccessModifier, Modifier oraz KindOfRelation, które są słownikami i zawierają odpowiednio: rodzaj typu (array, class, enum, generic, interface, value); modyfikatory dostępu (public, private, internal, protected); modyfikatory (abstract, sealed, static, virtual, itd.), rodzaje relacji (association, generalization, realization, aggregation, composition). Tabela Type reprezentuje wszystkie przeanalizowane typy. Jej pierwsze rekordy są

predefiniowane oraz zawierają omawiane wcześniej typy zaślepkowe. Pozostałe tabele odpowiadają zgodnie z ich nazwami: Method- metody, Property- właściwości, Field- pola, Relation- relacje zachodzące pomiędzy typami. Dodatkowo, dla uproszczenia, na rysunku nie zostały uwzględnione relacje opisujące zwracany typ przez metody oraz typy danych pól, właściwości, parametrów.



Rysunek 10.2. Uproszczony model bazy danych.

Przygotowana baza danych przedstawia prosty model danych opisujący podstawowe cechy paradygmatu programowania obiektowego. Celem takiej struktury jest możliwość zarejestrowania przetworzonych elementów oraz wykonania prostych analiz. Struktura bazy danych została przygotowana na potrzeby wykonania porównania, jednakże w takiej postaci nie będzie miała zastosowania w dalszych działaniach.

10.2.3 Przebieg eksperymentu

Porównanie zostało wykonane na trzech różnych zestawach binariów stworzonych lub współtworzonych przez autora rozdziału, różniących się od siebie wielkością. Wybór owych zestawów nie jest przypadkowy. Podyktowane jest to dobrą znajomością danych zestawów przez autora rozdziału, dzięki czemu bardziej intuicyjne było weryfikowanie poprawności otrzymanej struktury w eksperymencie względem oryginalnych kodów źródłowych. Jak pokazały wyniki eksperymentu przy jednym z wybranych rozwiązań wystąpiło zdecydowanie więcej problemów niż przy drugim. Dobra znajomość struktury kodów źródłowych oraz występujących w nim nazw okazało się niezastąpioną cechą, która znacząco wpłynęła na szybkie ich rozwiązanie. Z przetwarzania zostały wyłączone zewnętrzne biblioteki, ponieważ nie są one przedmiotem analizy.

Dla każdego zestawu zostały wykonane następujące testy:

— pozyskanie pełnej struktury obiektowej z wykorzystaniem refleksji,

— pozyskanie pełnej struktury obiektowej z wykorzystaniem Mono.Cecil.

Każdy test został powtórzony dziesięciokrotnie i z każdego powtórzenia został zanotowany czas wykonywania.

10.2.4 Analiza wyników porównania

Tabela 10.1 przedstawia średni czas wykonywania oraz odchylenie standardowe dla czasów z całej serii powtórzeń i różnicę procentową pomiędzy przebadanymi metodami. Czas został wyrażony w sekundach. W przypadku zestawu nr 1 i nr 3 zauważalny jest zysk czasowy na rzecz Mono.Cecil. Wyniki w przypadku zestawu nr 2 są ze sobą porównywalne, jednakże zastanawiająca jest tak mała różnica. Aby uzyskać na to odpowiedź należy przeanalizować również tabelę 10.2. Niska wartość odchylenia standardowego świadczy o braku zdarzeń losowych, które mogłyby niekorzystnie wpłynąć na cały proces.

Tabela 10.1. Średni czas wykonywania analizy

Zestaw	**Refleksja**		**Mono.Cecil**		**Zysk dla Mono.Cecil**
	Czas	σ	Czas	σ	
Zestaw 1	1,88	0,02	0,42	0,02	348 %
Zestaw 2	26,88	0,33	27,25	0,18	-1,3 %
Zestaw 3	793,52	33,66	591,27	22,85	34 %

Tabela 10.2 przedstawia liczbowe zestawienie ilości znalezionych typów, pozostałych elementów składowych oraz maksymalnego wykorzystania pamięci RAM.

Po przeanalizowaniu danych z tabeli 10.2 dla zestawu nr 1 można zauważyć większą ilość znalezionych typów w przypadku Mono.Cecil. Jest to spowodowane poprzez założenia tej biblioteki, które wprowadzają dodatkową dekompozycję na moduły [CEC]. Cecha ta nie ma dużego znaczenia, więc łatwo można ją wyeliminować. Większe znaczenie mają różnice w ilości odnalezionych elementów składowych oraz relacji. Owa różnica wynika również z założeń projektowych obu omówionych metod inżynierii odwrotnej. Cechą refleksji jest pobieranie wszystkich elementów składowych badanego typu w tym odziedziczonych z klas nadrzędnych, natomiast Mono.Cecil pobiera jedynie elementy składowe występujące wyłącznie w badanym typie (pozostałe są dostępne po-

przez właściwość BaseType badanego typu). Ta cecha nie ma również dużego znaczenia, choć wprowadza dodatkową złożoność. Rozwiązanie przyjęte w Mono.Cecil jest bardziej intuicyjne dla osób korzystających z diagramów klas w notacji UML.

Tabela 10.2. Zestawienie liczbowe znalezionych elementów oraz wykorzystanie pamięci RAM

Badany zestaw / element		**Refleksja**	**Mono.Cecil**
Zestaw 1	Typy	6	7
	Pola	33	13
	Właściwości	12	4
	Metody	93	11
	Parametry metod	61	13
	Relacje	233	52
	Wykorzystanie RAM	53 MB	49 MB
Zestaw 2	Typy	48	158
	Pola	197	743
	Właściwości	262	382
	Metody	950	690
	Parametry metod	821	653
	Relacje	3031	3270
	Wykorzystanie RAM	78 MB	100 MB
Zestaw 3	Typy	512	1051
	Pola	1259	7404
	Właściwości	2667	2333
	Metody	13682	6069
	Parametry metod	11414	8214
	Relacje	40772	36287
	Wykorzystanie RAM	287 MB	228 MB

Bardzo istotne znaczenie mają różnice wynikające z przeanalizowania wyników dla zestawu nr 2 oraz 3. Przede wszystkim należy wyjaśnić znacznie większą ilość znalezionych typów przez Mono.Cecil. Znalezienie większej ilości typów nie wynika wyłącznie ze wspomnianej wcześniej dodatkowej dekompozycji na moduły. Różnica owa wynika z problemów w trakcie działania mechanizmu refleksji. Jak wykazało debugowanie kodu aplikacji przygotowanej na potrzeby tego testu, mechanizm refleksji nie był w stanie przeanalizować zestawów, w których były wykorzystywane typy spoza badanego zestawu. Mecha-

nizm refleksji zgłaszał wewnętrzny wyjątek mimo, iż brakujące typy znajdowały się w tym samym katalogu co katalog analizowany. Mono.Cecil działając na dokładnie tym samym katalogu oraz tych samych plikach binarnych nie wykazał żadnych problemów. Jest to znacząca wada mechanizmu refleksji, ponieważ nie zawsze mogą być dostępne wszystkie biblioteki powiązane z badanym zestawem. Problemu tego nie można rozwiązać bez modyfikacji kodu źródłowego i ponownej kompilacji całej biblioteki. Załadowanie zestawów za pomocą metody Assembly.ReflectionOnlyLoadFrom pomaga wyłacznie w niewielkim stopniu, ponieważ ilość znalezionych typów jest wciąż znacząco mniejsza niż w przypadku Mono.Cecil. Szczególnie widoczne jest to w przypadku zestawu nr 2, gdzie Mono.Cecil okazało się trzykrotnie skuteczniejsze, tj. zostało znalezionych oraz przeanalizowanych trzykrotnie więcej typów. W tej sytuacji różnica niecałych 0,4 sekundy na niekorzyść Mono.Cecil jest w pełni uzasadniona, ponieważ zostało wykonane trzykrotnie więcej przebiegów głównej pętli. Jednakże uzyskany czas prawie 10 minut dla zestawu nr 3 oraz Mono.Cecil jest dość duży. Czas potrzebny na skompilowanie tego zestawu nie przekracza jednej minuty, zatem nie będzie możliwe dostarczenie odpowiednich statystyk zaraz po kompilacji. Należy jeszcze pamiętać, iż po etapie pozyskania struktury konieczne będzie przygotowanie odpowiednich statystyk, co też zajmie odpowiednią ilość czasu. Zestaw nr 2 w stosunku do pozostałych zestawów wyróżnia bliskie współdziałanie ze sprzętem. Jest to aplikacja automatyzująca pracę sprzętowego sterownika podłączonego do portu USB, a to wymaga wykorzystania natywnych bibliotek. Charakter przetwarzanych danych wymusza na całej aplikacji wykorzystanie bardzo wielu typów prostych, w szczególności byte oraz tablic byte[]. Różnice implementacyjne mogą przyczyniać się do różnic w wynikach porównania.

W celu wykazania ostatecznej skuteczności w pozyskiwaniu struktury obiektowej, kody źródłowe badanych zestawów zostały przeanalizowane również poprzez aplikację SourceMonitor [SRM]. Wynik ilości znalezionych typów był jednoznaczny z uzyskanym przez Mono.Cecil. Jednakże wymagało to przygotowania odpowiedniego zapytania do bazy danych. Przykładowo SourceMonitor nie był w stanie zinterpretować typów wygenerowanych automatycznie przez ORM (od ang. *Object Relational Mapping*) Entity Framework. Jak wykazał dany test oraz szczegółowa analiza przeprowadzona ręcznie dla wielu z przebadanych typów, rzeczywisty rozmiar przebadanych zestawów (tj. ilość typów oraz ich elementów składowych) jest tożsamy z wynikami uzyskanymi przez Mono.Cecil.

W tabeli 10.2. także zaprezentowane zostało maksymalne wykorzystanie pamięci operacyjnej przez aplikację testującą. Różnice dla małych zestawów nie są znaczne. Dla większych zestawów wyniki są przewidywalne, wraz ze wzrostem ilości typów wzrasta czas przetwarzania oraz zapotrzebowanie na pamięć RAM. Potwierdza to test wykonany na projekcie MediaPortal 2 [MP2]. Zostały przebadane wszystkie zestawy .NET z aplikacji klienta, serwera oraz wszystkie pluginy. Aplikacja testowa przy wykorzystaniu Mono.Cecil przetworzyła 6543 typy w czasie 4 godzin i 25 minut. Wykorzystanie pamięci RAM wzrosło do 645 MB. Jest to wynik spełniający kryterium skalowalności.

Poza mierzalnymi aspektami, uwaga została zwrócona jeszcze na cechy bibliotek, oraz aspekty związane z ich wykorzystaniem. Zestawienie owych cech przedstawia tabela 10.3.

Tabela 10.3. Zestawienie pozostałych cech

Cecha	**Refleksja**	**Mono.Cecil**
Odczyt struktury obiektowej	Tak	Tak
Odczyt ciała metod	Zmienne lokalne	Zmienne lokalne oraz instrukcje
Możliwość edycji zestawów .NET	Nie	Tak
Interpretacja kodu CIL	Nie	Tak
Odczyt elementów prywatnych	Tak	Tak
Dostęp do kodu źródłowego biblioteki	Nie	Tak
Ilość zgłaszanych wyjątków w trakcie działania	5 / proces implementacji	0 / proces implementacji
Skuteczność pozyskiwania typów	59,7 %	100 %
Poziom trudności implementacji	Niski	Niski

Znaczącą wadą refleksji jest generowanie wyjątków wewnętrznych, które wymagają dodatkowej obsługi w implementacji. W trakcie procesu implementacji aplikacji testowej, w przypadku refleksji należało obsłużyć 5 różnych wyjątków, co nie obyło się bez debugowania kodu źródłowego. Mono.Cecil przy analogicznej implementacji nie generuje żadnych wyjątków. Kolejna wada to brak dostępu do kodów źródłowych, aby ewentualnie poprawić błędy bądź zmodyfikować bibliotekę do własnych potrzeb. Open source w przypadku Mono.Cecil jest zasadniczą zaletą pomimo mniej pewnego wsparcia niż w przypadku firmy Microsoft. Niewystarczająca skuteczność, która dla refleksji wy-

niosła średnio 59,7 % jest ostatecznym argumentem do wskazania zwycięzcy porównania.

10.2.5 Wnioski

Przeprowadzony test jednoznacznie wskazuje zwycięzcę porównania, którym jest Mono.Cecil. Mono.Cecil charakteryzuje się większą skutecznością pozyskiwania struktury obiektowej i pomimo wykonywania większej liczby operacji działa szybciej niż mechanizm refleksji. Jedyną cechą podważającą słuszność zwycięstwa Cechą Mono.Cecil jest niepewne wsparcie, w odróżnieniu do produktów firmy Microsoft.

Mechanizm refleksji ustępuje zwycięzcy w wielu kwestiach. Wspomniany wcześniej problem występujący przy braku dostępu do definicji typu oraz inne problemy opisane w [CDB] oraz w [CDP] dyskwalifikują wykorzystanie refleksji w planowanym przedsięwzięciu.

10.3 Podsumowanie

W pracy przedstawiono dwie najbardziej znaczące metody do stosowania inżynierii odwrotnej w kodzie zarządzanym. Następnie została pojęta próba praktycznego zweryfikowania skuteczności oraz wydajności owych metod. Przeprowadzony eksperyment został wykonany w kontekście dalszych prac badawczych związanych z statyczną analizą oprogramowania.

Wnioski wynikające z eksperymentu wskazują jednoznacznie, iż zwycięstwo należy się bibliotece Mono.Cecil. Korzyści wynikające z zastosowania tego rozwiązania przewyższają jego wady.

Bibliografia

[CEC] Dokumentacja dostępna pod adresem: http://www.mono-project.com/Cecil:FAQ

[CDB] Artykuł dostępny pod adresem: http://codebetter.com/patricksmacchia/2008/03/18/mono-cecil-vs-system-reflection/

[CDP] Artykuł dostępny pod adresem: http://www.codeproject.com/Articles/499960/Accessing-Assembly-Metadata-with-Reflection-or-Mon

[DE05] Dietrich J., Elgar C., *A formal description of design patterns using OWL*, Software Engineering Conference, Australia, 2005

[HEW11] Hewardt M., *Debugowanie .NET Zaawansowane techniki* diagnostyczne, s. 59, Helion, Gliwice, 2011

[ILS] Projekt dostępny pod adresem: http://ilspy.net

[Kas04] Kaspersky K., *Dezasemblowanie kodu*, s. 15, Wyd. RM, 2004

[MON] Projekt dostępny pod adresem: http://www.mono-project.com
[MP2] Projekt dostępny pod adresem: http://www.team-mediaportal.com
[MSDN] Dokumentacja dostępna pod adresem: http://msdn.microsoft.com/en-us/library/system.reflection(v=vs.110).aspx
[Ros09] Ross J., *Bezpieczne programowanie Aplikacje hakeroodporne*, Helion, Gliwice, 2009
[SG13] 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
[SRM] Program dostępny pod adresem: http://www.campwoodsw.com/sourcemonitor.html
[Tro09] Troelsen A., *Język C# i platforma .NET 3.5*, s. 37-53, PWN, Warszawa, 2009

Autorzy i afiliacje

WSTĘP
Lech Madeyski
Politechnika Wrocławska, Wydział Informatyki i Zarządzania, Instytut Informatyki
lech.madeyski@pwr.edu.pl
Mirosław Ochodek
Politechnika Poznańska, Wydział Informatyki
miroslaw.ochodek@cs.put.poznan.pl

ROZDZIAŁ 1
Artur Kasprzyk
AION Sp. z o.o.
artur.kasprzyk@aion.com.pl
Anita Walkowiak
Politechnika Wrocławska, Instytut Informatyki
anita.walkowiak@pwr.wroc.pl

ROZDZIAŁ 2
Tomasz Protasowicki
Wojskowa Akademia Techniczna, Wydział Cybernetyki
tprotasowicki@wat.edu.pl

ROZDZIAŁ 3
Romuald Hoffmann
Wojskowa Akademia Techniczna, Wydział Cybernetyki
rhoffmann@wat.edu.pl
Tomasz Protasowicki
Wojskowa Akademia Techniczna, Wydział Cybernetyki
tprotasowicki @wat.edu.pl

ROZDZIAŁ 4
Alicja Ciemniewska
Poznańskie Centrum Superkomputerowo – Sieciowe
alicja@man.poznan.pl

Paweł Kędziora
Poznańskie Centrum Superkomputerowo – Sieciowe
kedziora@man.poznan.pl
Marek Lewandowski
Poznańskie Centrum Superkomputerowo – Sieciowe
marekl@man.poznan.pl
Cezary Mazurek
Poznańskie Centrum Superkomputerowo – Sieciowe
mazurek@man.poznan.pl
Marcin Wolski
Poznańskie Centrum Superkomputerowo – Sieciowe
marcin.wolski@man.poznan.pl

ROZDZIAŁ 5
Walery Susłow
Politechnika Koszalińska, WEiI
walery.suslow@tu.koszalin.pl
Jacek Kowalczyk
Politechnika Koszalińska, ITiE
jacek.kowalczyk@tu.koszalin.pl
Marta Boińska
Uniwersytet Gdański, WNS
marta.boinska@ug.edu.pl
Janina Nowak
Uniwersytet Gdański, WNS
dokjn@ug.edu.pl
Michał Statkiewicz
Politechnika Koszalińska, WEiI
michal.statkiewicz@tu.koszalin.pl

ROZDZIAŁ 6
Andrzej Stasiak
Wojskowa Akademia Techniczna, Wydział Cybernetyki
astasiak@wat.edu.pl
Włodzimierz Dąbrowski
Politechnika Warszawska, Instytut Sterowania i Elektroniki Przemysłowej
w.dabrowski@ee.pw.edu.pl

ROZDZIAŁ 7
Michał Pawłowski
SI2 Sp. z o.o.
michal@pawlowski.be
Ziemowit Nowak
Politechnika Wrocławska, Instytut Informatyki
ziemowit.nowak@pwr.edu.pl

ROZDZIAŁ 8
Tomasz Protasowicki
Wojskowa Akademia Techniczna, Wydział Cybernetyki
tprotasowicki@wat.edu.pl
Jerzy Stanik
Wojskowa Akademia Techniczna, Wydział Cybernetyki
jstanik@wat.edu.pl

ROZDZIAŁ 9
Tomasz Sitek
Politechnika Gdańska, Wydział Zarządzania i Ekonomii
tsitek@zie.pg.gda.pl
Michał Litka
Avena Technologie
litka.michal@gmail.com

ROZDZIAŁ 10
Rafał Wojszczyk
Politechnika Koszalińska, Wydział Elektroniki i Informatyki
rafal.wojszczyk@tu.koszalin.pl