POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY
INSTYTUT STEROWANIA I ELEKTRONIKI PRZEMYSŁOWEJ

PRACA DYPLOMOWA MAGISTERSKA
na kierunku Informatyka
specjalność: Inżynieria oprogramowania

Mateusz Mróz
Nr albumu: 221192

Rok akad.: 2013/2014
Warszawa, 14.01.2014

Projektowanie architektury korporacyjnej z zastosowaniem języka SoaML

1. Charakterystyka architektury korporacyjnej.
2. Przegląd języków do projektowania systemów informatycznych o architekturze SOA.
3. Przegląd metod projektowania architektury korporacyjnej.
4. Opracowanie metody projektowania systemów informatycznych o architekturze SOA.
5. Weryfikacja koncepcji na przykładzie systemu bankowego.
6. Podsumowanie i wnioski.

Kierujący pracą:

Dr inż. Włodzimierz Dąbrowski

Kierownik Zakładu Sterowania ISEP

Dr hab. Inż. Bartłomiej Beliczyński, Prof. PW

Termin złożenia pracy: 15.09.2014

(Podpis)

Warszawa, dnia 15.09.2014r.

Politechnika Warszawska
Wydział Elektryczny

OŚWIADCZENIE

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa magisterska pt. "Projektowanie architektury korporacyjnej z zastosowaniem języka SoaML":

- została napisana przeze mnie samodzielnie,
- nie narusza niczyich praw autorskich,
- nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam, że przedłożona do obrony praca dyplomowa nie była wcześniej podstawą postępowania związanego z uzyskaniem dyplomu lub tytułu zawodowego w uczelni wyższej. Jestem świadom, że praca zawiera również rezultaty stanowiące własności intelektualne Politechniki Warszawskiej, które nie mogą być udostępniane innym osobom i instytucjom bez zgody Władz Wydziału Elektrycznego.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Mateusz Mróz.....

Streszczenie pracy magisterskiej

Projektowanie architektury korporacyjnej z zastosowaniem języka SoaML

Celem pracy jest zaproponowanie metody projektowania architektury korporacyjnej z wykorzystaniem języka SoaML oraz przedstawienie jej zastosowania w praktyce. Autor analizuje tematykę oraz możliwości wykorzystania SOA jako konkretnej techniki konstrukcyjnej prowadzącej do wdrożenia architektury korporacyjnej w organizacji.

Pierwszy rozdział stanowi wprowadzenie do tematyki architektury korporacyjnej. Przedstawione zostały główne pojęcia z nią związane oraz korzyści, które mogą wynikać z jej wdrożenia w organizacji. Opisano również różne podejścia do jej budowy. W tej części pracy pojawia się także definicja SOA. Zaprezentowano ponadto możliwości adaptacji SOA w architekturze korporacyjnej.

W kolejnym rozdziale dokonano przeglądu języków do projektowania systemów informatycznych o architekturze SOA. Przedstawione zostały opisy języków modelowania: SoaML, ArchiMate oraz innych języków wspierających niektóre fazy projektowania (BPMN, BPEL, UML oraz IDEF).

Trzeci rozdział pracy zawiera przegląd metod projektowania rozwiązań w architekturze usługowej. Pokazano takie metody jak: RUP4SOA, SOMA, Papazoglou oraz Thomasa Erl'a. Każdy z opisów zawiera wyszczególnienie wszystkich faz oraz rozważania dotyczące wad i zalet ich stosowania.

W następnym rozdziale przedłożono opracowanie autorskiej metody projektowania architektury korporacyjnej z wykorzystaniem języka SoaML - „MatSOA". Szczegółowo ukazano jej fazy oraz przedstawiono możliwe sposoby stosowania. Scharakteryzowano produkt końcowy jako efekt wykorzystania tej metody.

Celem piątego rozdziału była weryfikacja opracowanej wcześniej metody. Na przykładzie wymyślonej sytuacji biznesowej w banku zaprezentowano próbę wdrożenia autorskiej koncepcji. Ukazano pełen proces przygotowywania systemu informatycznego. Opisano wykorzystywane oprogramowanie oraz środowisko wdrożenia. W tej części także przedstawiono wyniki testu i dokonano ich analizy. Poddano również rozważaniom relację pomiędzy celem systemu, a zaprojektowaną koncepcją.

Ostatni rozdział stanowi podsumowanie całej pracy. Dokonano porównania utworzonej metody z innymi obecnie już występującymi. Końcowa część zawiera wnioski na podstawie analiz wykonanych w pracy, rozważania dotyczące rozwoju metod projektowania architektury korporacyjnej oraz opis przemyśleń dotyczących przyszłości tworzenia systemów w oparciu o paradygmaty SOA.

Słowa kluczowe: architektura korporacyjna, SoaML, SOA

The Summary of Diploma Project

The enterprise architecture designing with the use of SoaML language

The purpose of this thesis is to develop the new enterprise architecture designing method with the use of SoaML language, and to present its applications. This work analyzes the issues and opportunities of using SOA as a specific construction technique leading to the implementation of the enterprise architecture within an organization.

The first chapter introduces the topic of the enterprise architecture. It demonstrates the main concepts associated with this term, and the advantages of the implementation of the enterprise architecture in the organization. Moreover, this chapter describes different approaches to building the enterprise architecture. It also defines the SOA term, and presents their opportunities of adoption in the enterprise architecture.

The next chapter provides an overview of the languages to design information systems with the use of SOA. This part of work contains several descriptions of the modelling languages such as: SoaML, ArchiMate and other languages that support different designing phases (BPMN, BPEL, UML and IDEF).

The third chapter constitutes an overview of the designing methods of the services architecture. There are following descriptions: RUP4SOA, SOMA, Papazoglou and Thomas Erl. All descriptions contain specifications of all their phases, as well as, consider the advantages and disadvantages of their use.

The next chapter presents author's method „MatSOA" of designing the enterprise architecture with the use of SoaML language. All phases are shown in detail, with consideration of possible uses and characteristics of the final result.

The purpose of fifth chapter is to verify the developed method. It describes the implementation of information system for an imaginary business situation in a bank. This part shows the whole process of the information system preparation. There is also a description of used software, a deployment environment and the analysis of performed tests.

The last chapter sums up the whole thesis. It demonstrates a comparison of the developed method with other methods which already exist and summary of carried-out analysis. There are also prospects for the future of the enterprise architecture designing methods, and the development of information systems based on the SOA paradigms.

Keywords: enterprise architecture, SoaML, SOA

Spis treści

Rozdział 1

Wstęp

W ostatnich latach coraz więcej organizacji decyduje się na wdrożenia systemów informatycznych. Ta tendencja związana jest przede wszystkim z usprawnianiem procesów biznesowych, ze wzrostem wydajności pracy, identyfikacją oraz minimalizacją błędów lub nieefektywnych działań w organizacjach.

Zróżnicowane potrzeby organizacji doprowadziły do powstania wielu rodzajów systemów informatycznych: CRM (ang. Customer Relationship Management) - wspomagające zarządzanie relacjami z klientami, ERP (ang. Enterprise Resource Planning) – wspomagające zarządzanie i planowanie zasobów przedsiębiorstwa czy też BPM (ang. Business Process Management) – wspomagające zarządzanie procesami biznesowymi.

Wprowadzanie systemów informatycznych do organizacji może wywoływać różnorodne efekty: techniczne (bezpośrednio związane z zastosowaniem techniki komputerowej - zwiększenie szybkości i dokładności przetwarzania danych, wzrost szczegółowości oraz poprawa bezpieczeństwa dla informacji), ekonomiczne (związane pośrednio ze wzrostem efektywności i szybkości podejmowania decyzji), organizacyjne (związane przede wszystkim z usprawnieniami struktury organizacyjnej i procesów zachodzących w przedsiębiorstwie - podniesienie sprawności obiegu dokumentów, eliminacja zbędnej pracy administracyjnej, poprawa koordynacji zadań oraz eliminacja błędów), socjopsychologiczne (związane z rozszerzeniem zakresu komunikacji pomiędzy pracownikami, usprawnieniem systemu ocen pracowniczych oraz polepszeniem kultury organizacyjnej) [3].

Utworzenie systemu informatycznego dla dużych firm lub instytucji nie stanowi łatwego zadania. Banki, uczelnie lub urzędy charakteryzują się często stosunkowo złożoną strukturą organizacyjną oraz panują w nich skomplikowane procesy wewnętrzne. Powstaje wówczas problem z formalnym opisem danej jednostki. Architektura korporacyjna pozwala opisać komponenty i struktury korporacji. Definiuje również wzajemne powiązania i relacje między poszczególnymi jej komponentami.

Odpowiednia definicja oraz wprowadzenie „ładu korporacyjnego” w dużych

organizacjach są niebywale istotne dla wdrożenia funkcjonalnego systemu informatycznego. Bardzo istotnym elementem jest wówczas zastosowanie odpowiedniej metody, która pozwoli na uporządkowane podejście do projektowania tego rodzaju systemów.

1.1 Cel pracy

Podstawowym celem pracy jest zaproponowanie metody projektowania systemów informatycznych opierającej się na wykorzystaniu języka SoaML. Dotychczas utworzono już podobne metody, jednakże żadna nie była w stanie w pełni sprostać wszystkim wymaganiom stawianym przez problematykę architektury korporacyjnej.

Projekt metody powstał w oparciu o wybranie najmocniejszych stron oraz pominięcie wad istniejących metod. Autor przystąpił do wdrożenia innowacyjnych elementów, które będą miały na celu usprawnienie procesu projektowania architektury korporacyjnej.

Zaprojektowana metoda została poddana weryfikacji i testom na podstawie utworzenia niewielkiego systemu bankowego.

1.2 Pozostałe założenia dla pracy

Oprócz założenia głównego dla pracy (utworzenie metody projektowania architektury korporacyjnej) przyjęto kilka innych, które powinna realizować:

- zdefiniowanie architektury korporacyjnej oraz wyjaśnienie terminów z nią powiązanych,
- przegląd języków wykorzystywanych do projektowania architektury korporacyjnej,
- przegląd obecnych metod wykorzystywanych do projektowania architektury korporacyjnej,
- weryfikacja koncepcji autorskiej metody przy wdrożeniu przykładowego systemu bankowego.

Rozdział 2

Charakterystyka architektury korporacyjnej

2.1 Wstęp

Żyjemy w coraz szybciej zmieniającym się świecie. Dynamika ta powoduje, że współczesne organizacje muszą charakteryzować się coraz większą elastycznością. Jednocześnie poziom złożoności panujących w nich procesów biznesowych jak również skala działania organizacji znacznie rosną [24].

Do innych problemów zalicza się również niejednorodność procesów biznesowych w poszczególnych jednostkach. Może wywoływać to trudności w stosowaniu wspólnych procedur postępowania w ramach całej organizacji. Rozwój i utrzymywanie zbyt wielu platform, gdy systemy tworzone były w różnych technologiach, może prowadzić również do trudności we współdziałaniu tychże systemów.

Rozdział będzie zawierał definicję terminu „architektura korporacyjna" oraz wyjaśnienie najważniejszych elementów z nim związanych. Podejmowane będą również rozważania nad tematyką „architektury zorientowanej na usługi" oraz nad sposobem zaadaptowania jej do architektury korporacyjnej.

2.2 Definicje głównych pojęć związanych z architekturą korporacyjną

Projektowanie architektury korporacyjnej (ang. Enterprise Architecture) wymaga doskonałej znajomości terminów z nią powiązanych. W tej sekcji zostanie dokonany przegląd oraz próba zdefiniowania głównych pojęć odnoszących się do architektury korporacyjnej.

Słowo „architektura" w nawiązaniu do systemów stanowi zbiór podstawowych

decyzji związanych z określonym rozwiązaniem programistycznym - zaprojektowanym z reguły tak, aby móc spełnić wszelkie atrybuty jakościowe (wymagania architektoniczne) danego projektu. Architektura tego typu zawiera przede wszystkim atrybuty, zdefiniowane współdziałania (zachowania i interakcje) umożliwiające spełnienie atrybutów jakościowych oraz główne komponenty. Może być wyrażona na więcej niż jednym poziomie abstrakcji, gdzie liczba poziomów zależy od rozmiaru i złożoności projektu.

Twórcy standardu uzupełniają powyższą definicję o następujące komentarze:

- Nazwa „architektura"nawiązuje do zestawu podstawowych (a nie wszystkich) właściwości systemu (rozpatrywanego jako całość), które określają jego formę i funkcje oraz związane z nim wartości, koszt i ryzyko.
- Architektura systemu odróżniana jest od jego opisu i nie musi być udokumentowana. Opis architektury stanowi próbę wyrażenia pewnej koncepcji systemu, tak aby móc podzielić się nią z innymi. Architektura jest pewnym abstraktem, natomiast opis jest próbą jego uchwycenia. W definicji wprowadzono również dwa rozłączne podejścia do architektury - rozumienie jej jako pewnej koncepcji systemu, która istnieje w ludzkim umyśle oraz jako percepcji właściwości systemu.
- Architektura opisywana jest zawsze w pewnym kontekście. Zrozumienie podstawowych właściwości systemu wiąże się z odniesieniem do jego otoczenia, środowiska i interesariuszy.

Kolejnym kluczowym terminem jest „korporacja"(ang. Enterprise). Pojęcie korporacja rozpatrywane jest najczęściej w kontekście całej jednostki organizacyjnej. Przyjmowane są różne definicje:

- Zbiór aktywności z aktorami w określonej dziedzinie, których łączy wspólny cel.
- Zorganizowany zbiór zasobów, które uczestniczą w wykonywaniu określonych procesów.
- System istniejący w celu realizacji jednej lub wielu misji w określonym środowisku.
- Zbiór organizacji posiadających wspólny zbiór celów lub wspólne raportowanie finansowe.

Wyraz korporacja wywodzi się z łacińskiego słowa „corporation"oznaczającego związek lub połączenie części. Nie istnieje obowiązująca definicja dla tego terminu jako organizacji gospodarczej.

Według jednej z definicji jest to duża organizacja zarządzana przez grupę specjalistów (kadrę zarządzającą), silnie oddziałująca na otoczenie zewnętrzne, posiadająca wyrazistą kulturę organizacyjną. Ma złożoną strukturę wewnętrzną, działającą na podstawie strategii długookresowych.

W przypadku organizacji gospodarczych za korporację może być uznawane przedsiębiorstwo lub jego część (np. zakład). W przypadku jednostek administracji rządowej korporację może stanowić cała administracja rządowa (wszystkie jej jednostki), resort lub jeden z jego fragmentów.

Współcześnie uważa się, że korporacją - w rozumieniu architektury korporacyjnej - może być również tak zwane rozszerzone przedsiębiorstwo („extended enterprise"). Ta kategoria przedsiębiorstwa definiowana jest jako zbiór jednostek prawnych, które są związane zintegrowanym łańcuchem wartości dodanej, dzięki czemu następuje zwiększenie wartości dostarczonej dla klientów [24].

2.3 Wprowadzenie do architektury korporacyjnej

Architektura korporacyjna jest spojrzeniem na całość organizacji. Stanowi opis struktury i funkcji komponentów takich jak: zasoby danych, strategia, procesy biznesowe, jednostki organizacyjne, systemy informatyczne oraz infrastruktura teleinformatyczna. Opisuje stan obecny i docelowy oraz proces przejścia między tymi stanami w organizacji [23].

Mimo, że architektura korporacyjna jest kojarzona głównie z technologią informatyczną, to nawiązuje przede wszystkim do metod optymalizacji procesów. Uwzględnia również opis architektury biznesowej, metod zarządzania efektywnością oraz organizacyjne uporządkowanie procesów [31].

Architekturze korporacyjnej przypisuje się kilka znaczeń: atrybutowe, rzeczowe oraz czynnościowe.

W ujęciu atrybutowym uważana jest za zbiór właściwości danej organizacji (oraz relacji między nimi) koniecznych do zapewnienia realizacji jej celów. Architektura korporacyjna stanowi nieodłączną właściwość każdej organizacji. Jej jakość może być rozpatrywana w odniesieniu do efektywności realizacji istniejących celów strategicznych analizowanej organizacji [23].

W ujęciu rzeczowym architektura korporacyjna definiowana jest jako formalna reprezentacja właściwości danej organizacji. W znaczeniu tym rozpatruje się ją również jako misję całej organizacji. Brane są pod uwagę informacje i zasoby techniczne niezbędne do realizacji zakładanych celów oraz proces przejścia związany z wdrażaniem nowych rozwiązań technicznych w nawiązaniu do zmian strategicznych w organizacji [23].

Spojrzenie czynnościowe zakłada definicję architektury korporacyjnej jako zadań i umiejętności zarządzania tym, co jest opisane w definicji atrybutowej [31].

Podejście prezentowane w książce „A Practical Guide to Federal Enterprise Architecture"uwzględnia aspekt transformacji organizacji w obszarze objętym architekturą korporacyjną. Stanowi to typową cechę złożonych systemów adaptacyjnych [23].

A. Goikoetxea definiuje architekturę korporacyjną (EA) jako ośmioelemntową strukturę, składającą się z następujących zbiorów:

$$EA : \{R, B, A, D, S, T, C, M\}, \tag{2.1}$$

gdzie:

- $R = (r_1, r_2, ..., r_n)$ - zbiór zawierający wszystkie wymagania systemowe - funkcjonalne i niefunkcjonalne, procesy, działania oraz reguły biznesowe.
- $B = (b_1, b_2, ..., b_p)$ - zbiór procesów biznesowych realizowanych w ramach organizacji. Metadane na temat wejść i wyjść dla każdego procesu zawarte są w zbiorze C.
- $A = (a_1, a_2, ..., a_q)$ - zbiór aplikacji biznesowych, funkcjonujących w ramach organizacji, które udostępniają usługi służące do implementacji procesów biznesowych. Metadane na temat relacji między tymi systemami przechowywane są w zbiorze C.
- $D = (d_1, d_2, ..., d_g)$ - zbiór danych, które konstytuują zasoby informacyjne organizacji. Metadane na temat organizacji tych danych, więzów integralności, reguł zarządzania przechowywane są w zbiorze C.
- $S = (s_1, s_2, ..., s_k)$ - zbiór systemów oprogramowania funkcjonujących w ramach organizacji umożliwiających realizację usług biznesowych przez aplikacje biznesowe. Metadane na temat wejść i wyjść poszczególnych aplikacji, interfejsy oraz standardy wytwarzania tych systemów przechowywane są w zbiorze C.
- $T - (t_1, t_2, ..., t_w)$ - zbiór komponentów technicznych organizacji - zarówno programowych (na przykład systemy operacyjne), jak i sprzętowych - niezbędnych do funkcjonowania systemów oprogramowania ze zbioru A.
- $C = (c_1, c_2, ..., c_h)$ - zbiór ograniczeń, które występują podczas tworzenia architektury korporacyjnej (takich jak np. standardy dotyczące oprogramowania i rozwiązań sprzętowych), reguł biznesowych oraz metadanych ze zbiorów R, B, A, D, S, T.

- $M = (m_1, m_2, ..., m_i)$ - zbiór miar, które charakteryzują architekturę korporacyjną. Używane są podczas tworzenia i utrzymywania architektury korporacyjnej w organizacji (dotyczą wymagań ilościowych: kosztów oraz wyników badań na temat efektywności działania organizacji).

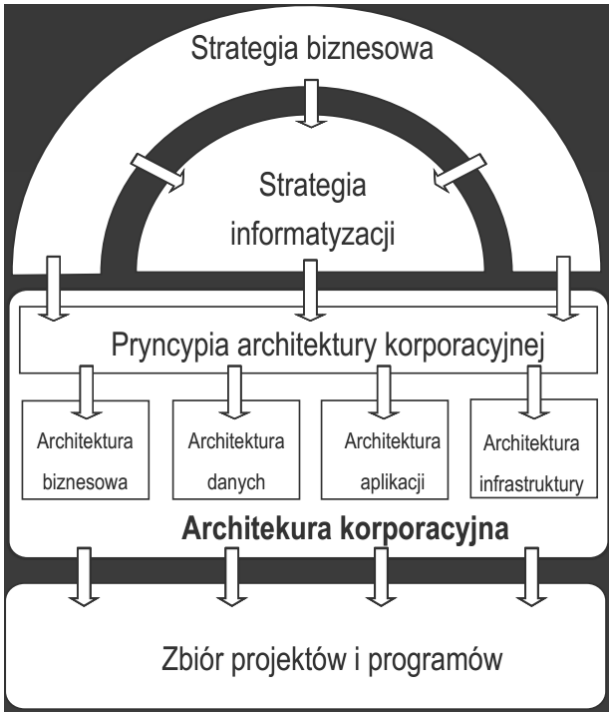
The Open Group w swoim opracowaniu dotyczącym TOGAF (The Open Group Architecture Framework) wskazuje, że architektura korporacyjna składa się z następujących elementów.

- Pryncypia architektury korporacyjnej (Enterprise Architecture Principles) - zbiór trwałych zasad opartych na strategii rozwoju organizacji, które stanowią reprezentacje całościowych potrzeb organizacji w zakresie tworzenia rozwiązań informatycznych.
- Domeny architektoniczne:
  - Architektura biznesowa (ang. Business Architecture) - opisuje strategię biznesową i sposoby zarządzania organizacją, jej strukturę organizacyjną oraz główne procesy biznesowe, a także relacje pomiędzy tymi elementami.
  - Architektura danych (ang. Data Architecture) - opisuje główne typy i źródła danych niezbędnych do funkcjonowania organizacji.
  - Architektura aplikacji (ang. Applications Architecture) - opisuje poszczególne aplikacje, ich rozlokowanie, wzajemne współdziałanie oraz relacje między tymi aplikacjami, a głównymi procesami biznesowymi w organizacji.
  - Architektura techniczna (ang. Technology Architecture) - opisuje komponenty infrastruktury technicznej, która stanowi podstawę funkcjonowania aplikacji (obejmuje ona m.in. systemy operacyjne, systemy zarządzania bazami danych, serwery aplikacyjne, sprzęt komputerowy oraz infrastrukturę komunikacyjną).

W ostatnim czasie coraz bardziej popularnym terminem staje się „architektura korporacyjna IT". W porównaniu do „zwykłej architektury korporacyjnej"wykazuje pewną odmienność.

Jednym z elementów, które różnicują te dwa pojęcia jest cel opracowania architektury. Zwykła architektura korporacyjna nakierowana jest na zwiększenie efektywności działania w kontekście całej organizacji. Architektura korporacyjna IT kładzie zaś nacisk na zwiększenie efektywności działania IT w kontekście biznesowych potrzeb organizacji.

Architektura korporacyjna stanowi szersze pojęcie pod względem zakresu opracowania architektury. Pojęcie to obejmuje organizację jako całość, natomiast architektura korporacyjna IT skupia się jedynie na obszarze IT danej jednostki.



Rysunek 2.1: Umiejscowienie architektury w korporacji [1].

Istotnym elementem, o którym należy wspomnieć, prowadząc rozważania na temat architektury korporacyjnej, jest pojęcie „ładu architektonicznego ”(ang. Enterprise architecture governance).

Ład architektoniczny stanowi przede wszystkim mechanizm dostarczający struktury, za pomocą której jest ustanawiany zbiór celów tworzenia architektury korporacyjnej oraz środków, poprzez które możliwe jest ich osiąganie i monitorowanie wydajności realizacji.

Jego podstawowym celem w odniesieniu do systemów informatycznych jest zapewnienie zgodności podejmowanych decyzji z zakresu IT z opracowaną architekturą korporacyjną.

Dobrze wprowadzony ład architektoniczny uważa się, że zapewni niezbędną dla organizacji elastyczność architektury korporacyjnej.

2.4 Korzyści z wdrożenia architektury korporacyjnej

Budowy architektury korporacyjnej nie powinno rozpatrywać się jedynie jako działania z obszaru informatyki, lecz jako całościowe przedsięwzięcie z pogranicza

zarządzania, jak i informatyki. Opracowania architektury korporacyjnej nie należy rozważać jako cel sam w sobie, ale stan pośredni uznawany za narzędzie pomocnicze do wykonania określonych działań wewnątrz organizacji. Takie podejście może przynieść organizacji szereg korzyści:

- lepsze dopasowanie realizowanych rozwiązań informatycznych do potrzeb strategicznych danej organizacji,
- zapewnienie większej interoperacyjności systemów informatycznych,
- wykorzystywanie tych samych komponentów może spowodować znaczne obniżenie kosztów działań w zakresie wprowadzania architektury korporacyjnej,
- możliwość szybszego podejmowania spójnych decyzji w zakresie tworzenia systemów informatycznych,
- efektywniejsze koordynowanie z perspektywy działań długoterminowych modyfikacji i rozbudowy poszczególnych systemów informatycznych,
- ułatwienie zarządzania finansami przeznaczonymi na cele IT,
- ułatwienie wdrożenia SOA,
- automatyzacja, optymalizacja i przejrzystość procesów biznesowych - zdefiniowana architektura korporacyjna pozwala zidentyfikować wszystkie procesy, gdzie możliwa jest automatyzacja oraz wyeliminować dublujące się czynności pomiędzy procesami,
- ograniczenie ryzyka operacyjnego - wdrażając architekturą korporacyjną, można znacznie zmniejszyć ryzyko operacyjne, zarówno poprzez automatyzację procesów, jak i lepszą kontrolę operacji.

Nie każda organizacja musi stosować sformalizowane podejście do zarządzania architekturą korporacyjną. Dużo zależy od złożoności organizacji - rozpatrywanej w ujęciu systemowym. Istotna jest również zmienność organizacji wynikająca na przykład ze zmienności otoczenia. Im organizacja jest bardziej złożona i im bardziej podlega zmianom, tym stosowanie architektury korporacyjnej jest bardziej uzasadnione [23].

2.5 Podejście do budowy architektury korporacyjnej

Budowa architektury korporacyjnej nie stanowi łatwego zadania. Bardzo istotnym czynnikiem wpływającym na sukces jej wdrożenia w danej organizacji jest odpowiedni projekt i podjęcie właściwych decyzji architektonicznych.

Projektując architekturę korporacyjną, należy rozważyć dwa główne problemy. Pierwszy dotyczy celu jej opracowywania. Przede wszystkim czy architektura korporacyjna ma mieć zastosowanie jako narzędzie wspierające określone przedsięwzięcie transformacyjne, czy też być na stałe wbudowana w ramy organizacji. Drugi problem związany jest z określeniem zakresu prac nad architekturą korporacyjną. Należy określić poszczególne wymiary: horyzont czasowy, zakres geograficzny organizacji oraz poziom szczegółowości w poszczególnych domenach.

Opracowując architekturę korporacyjną, można skorzystać z jednej dwóch ścieżek:

- wychodząca od architektury dla stanu bazowego - polega na opracowaniu najpierw architektury dla stanu bazowego „jak jest"(ang. As-is) bezpośrednio w czterech domenach architektonicznych: aplikacji, danych, biznesowej i technicznej. W kolejnym kroku analizowane są luki we wszystkich wymienionych domenach oraz przygotowywany jest plan przejścia (ang. Roadmap) pomiędzy stanem bazowym i docelowym. Do zalet tego podejścia można zaliczyć prostotę od strony zarządczej (można zidentyfikować i dobrze opisać poszczególne etapy prac). Wadę stanowi natomiast ryzyko zbyt dużego skupienia się na architekturze bazowej i wykonywania analiz na zbyt wysokim poziomie szczegółowości.
- wychodząca od architektury biznesowej - opiera się na wykonaniu czterech kroków. W trzech pierwszych dla stanu bazowego „jak jest", jak i dla stanu docelowego „jak będzie"(ang. To-be) opracowywane są kolejno architektury: biznesowa, danych i aplikacji oraz techniczna. W każdym z wymienionych kroków jest dokonywana również analiza luk. Mając opracowane architektury dla poszczególnych domen przygotowuje się zbiorczą analizę luk oraz plan przejścia pomiędzy stanem bazowym i docelowym. Zaletą tej ścieżki jest dostarczenie na najwcześniejszym możliwym etapie kluczowego elementu architektury korporacyjnej - architektury biznesowej. Stanowi to dobrą podstawę dla opracowywania kolejnych rodzajów architektur. Do wad tego podejścia zaliczyć można ryzyko z częstym występowaniem niskiego poziomu rozwinięcia części biznesowej organizacji [23].

2.6 SOA

2.6.1 Czym jest SOA?

Trudno o jednoznaczną definicję SOA (ang. Service Oriented Architecture – architektura zorientowana na usługi). Definicja SOA jest subiektywna, zależna od punktu widzenia. Z perspektywy biznesowej (odbiorcy usług) rozumieć ją można

jako zestaw usług wspierających realizację procesów biznesowych. Odnosząc się do SOA z perspektywy IT widzimy ją jako infrastrukturę potrzebną do dostarczenia tych usług.

Organizacja W3C (World Wide Web Consortium) podjęła próbę zdefiniowania SOA. Według niej SOA to zbiór komponentów, które mogą być wywoływane, i których interfejsy mogą być publikowane i wykrywane (ang. *A set of components which can be invoked, and whose interface descriptions can be published and discovered*).

Z kolei firma IBM definiuje SOA jako podejście do budowania systemów rozproszonych dostarczających funkcjonalność aplikacji w postaci usług, które mogą być udostępniane aplikacjom zewnętrznym lub innym usługom [11].

Istnieje również manifest SOA, który głosi, że orientacja na usługi kształtuje punkt widzenia na to co chcemy wykonać, a SOA stanowi typ architektury, który jest wynikiem takiej orientacji. (ang. *Service orientation is a paradigim that frames what you do. Service-oriented architecture (SOA) is a type of architecture that results from applying service orientation* [4]).

SOA sama w sobie nie jest jednakże żadną konkretną architekturą. Nie można jej traktować jako produkt lub tylko zbiór określonych rozwiązań technologicznych. SOA to przede wszystkim sposób myślenia – strategia, której naturalna realizacja jest reprezentowana przez usługi [30, 6]. SOA stanowi pewnego rodzaju paradygmat, który prowadzi do określonej architektury [30]. Reprezentuje przesunięcie w inżynierii oprogramowania i podnosi poziom abstrakcji, grupując wspólne działanie procesów biznesowych i wystawiając to jako usługę [28].

Mianem usługi w SOA określa się zbiór funkcjonalności pewnej aplikacji, który jest udostępniany jako interfejs [5]. Organizacja OASIS (Organization for the Advancement of Structured Information Standards) definiuje usługę w SOA jako mechanizm udostępniający jedną lub więcej funkcji, do których dostęp jest zapewniany przez zalecany interfejs i wykonywany zgodnie z ograniczeniami i politykami określanymi przez opis usługi (ang. *A mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description*). Operacje, zdefiniowane w interfejsie, dostarczają funkcji biznesowych poprzez wykorzystanie obiektów biznesowych. Ponadto te usługi są dostępne poprzez sieć.

2.6.2 Historia SOA

Przez ostatnie dekady tworzenie oprogramowania opierało się na kilku różnorodnych metodach. Jednym z ich podstawowych celów było radzenie sobie z coraz bardziej złożonym procesem wytwarzania oprogramowania. W ramach walki ze złożonością opracowano gruboziarniste konstrukcje takie jak funkcje, klasy i komponenty. Można myśleć o nich jak o „czarnych skrzynkach"(ang. Black boxes).

Te „czarne skrzynki"ukrywają swoje implementacje poprzez dostarczanie kontrolowanego dostępu do ich funkcjonalności poprzez interfejs. W pewnym momencie dostrzeżono, że ilość czarnych skrzynek jest zbyt duża. Problem ten rozwiązano, grupując poszczególne „czarne skrzynki" w komponenty [26].

Określenia „SOA" lub „Architektura zorientowana na usługi"zostały po raz pierwszy wykorzystane w pracy naukowej analityka z firmy Garnter Yefim V. Natis w dniu 12 kwietnia 1996 roku. Wcześniej w 1994 roku Alexander Pasik stworzył istotne podstawy do zdefiniowania SOA.

Przełomowym momentem w historii powstawania SOA było utworzenie usług typu Web Service przez firmę Microsoft w 2000 roku [14].

2.6.3 Budowa SOA

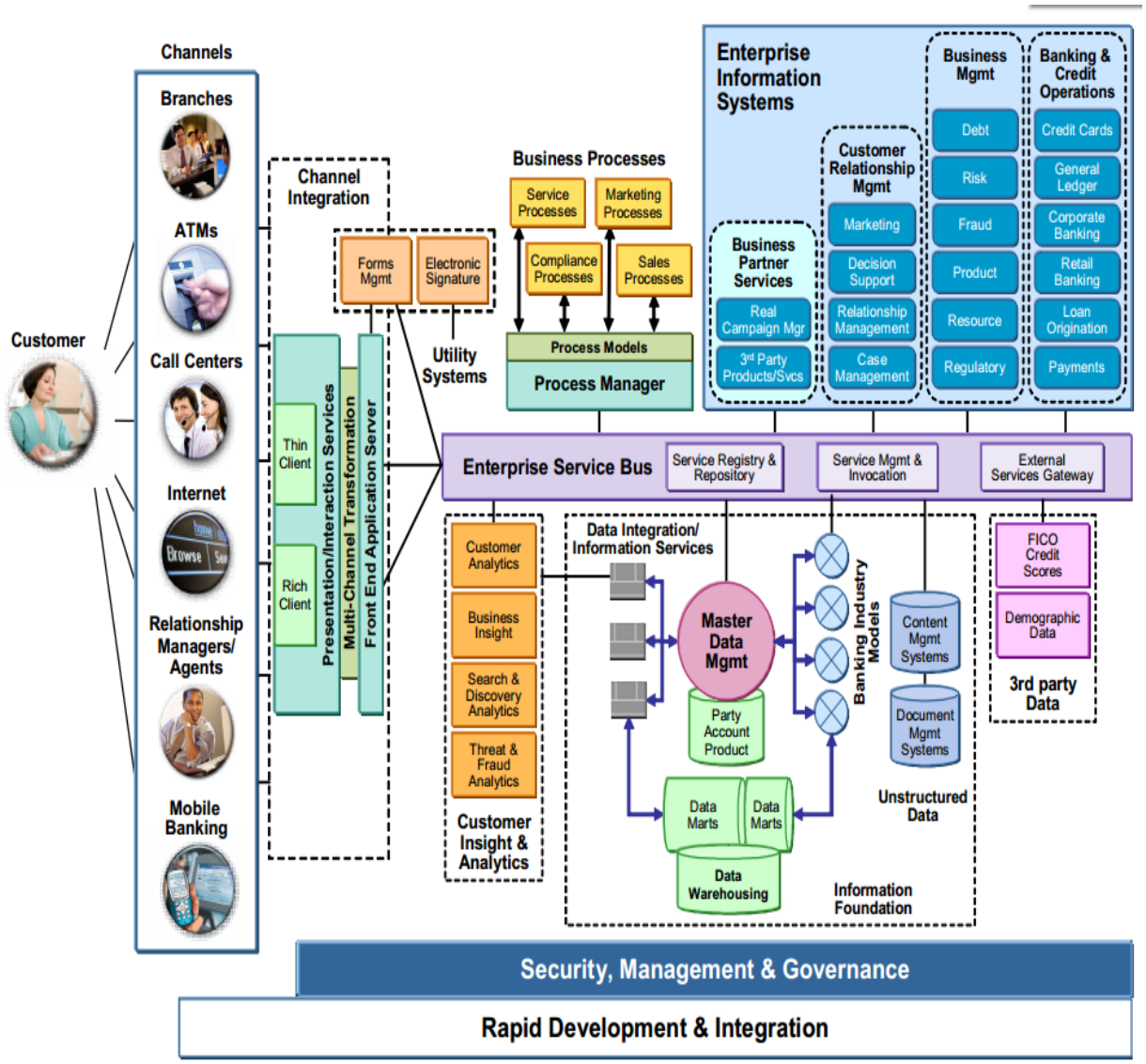
SOA wprowadza nowe podejście do budowy i zarządzania infrastrukturą informatyczną. Adaptacja SOA wiąże się z reguły ze zmianami w wielu obszarach funkcjonowania firmy. Ściśle wiąże się z panującymi w niej procesami biznesowymi i wymaga odpowiedniego dostosowania.

Systemy opierające się na SOA składają się z zestawu aplikacji biznesowych (rys. 2.2) pogrupowanych w niezależne komponenty, które komunikują się ze sobą, wymieniając usługi [6].

Komponenty te można traktować jako tzw. „czarne skrzynki". Klient korzystający z usługi otrzymuje jedynie interfejs. Implementacja udostępnionych metod nie jest dla niego istotna. Rozwiązania stosowane w SOA pomagają zapanować nad złożonością systemu. Podstawowa budowa SOA opiera się z reguły na szynie ESB (ang. Enterprsie Service Bus) integrującej poszczególne usługi. ESB jest efektywnym środkiem komunikacji w SOA. Pomaga uwolnić się od sieci powiązań typu „każdy z każdym" (ang. Point to point). Odpowiada za przesyłanie komunikatów do odpowiednich komponentów. Rozbudowa systemu polega na dołączaniu nowych usług do szyny integracyjnej.

Komunikaty, zanim zostaną wysłane do punktu docelowego, często poddawane są transformacjom i odpowiedniemu dostosowaniu za pomocą mediacji (ang. Mediations) na ESB. Przetworzony komunikat odpowiada formie jakiej oczekuje dostawca usługi.

Rejestr SOA (ang. SOA registry) stanowi centralny punkt informacyjny o sposobach dostępu, definicjach, regułach, bezpieczeństwie i innych danych wymaganych do wykorzystania usług udostępnionych w danym środowisku SOA. Zawiera informacje, gdzie poszczególne komponenty SOA są umieszczone. Na jego podstawie ESB potrafi prawidłowo przekierowywać żądanie usługi i ewentualną odpowiedź, a aplikacje i usługi korzystające z usług składowych potrafią skonstruować jej prawidłowe wywołanie.



Rysunek 2.2: Przykładowa architektura SOA [15].

Istotnym elementem w systemach typu SOA jest repozytorium (ang. SOA repository). Stanowi ono centralny „magazyn" dla elementów składowych usług takich jak: kod źródłowy, zestawy instalacyjne, specyfikacja itp. Repozytorium usług jest tworzone i wykorzystywane głównie na etapie projektowania usług.

2.6.4 Usługa sieciowa w SOA

Usługa sieciowa w SOA (ang. Web Service) stanowi usługę świadczoną przez sieć – na przykład Internet. Składnik oprogramowania niezależny od platformy sprzętowej oraz implementacji. Może być zdefiniowana za pomocą języka opisu usług – na przykład bazującym na XML WSDL (ang. Web Service Definition Language), publikowana i wyszukiwana w rejestrze (np. UDDI) oraz wywoływana zdalnie przez opisujący ją interfejs. Z reguły Web Service opiera się na konstrukcji uwzględniającej trzy typy komponentów (rys. 2.3): klienta usługi (ang. Service requestor), dostawcę usługi (ang. Service provider) oraz rejestr usług (ang. Service registry) lub broker usług (ang. Service broker). Transport danych w przypadku Web Service zazwyczaj opiera się na HTTP lub HTTPS z wykorzystaniem SOAP.

Każdy z komponentów jest odpowiedzialny za pełnienie swojej roli:

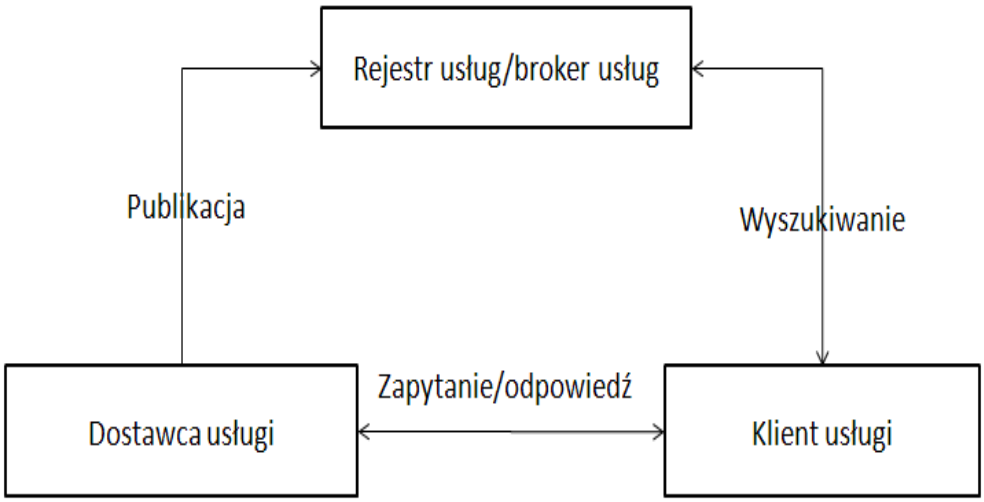
- dostawca usługi tworzy i wdraża Web Service. Publikuje również dostępność opisanego za pomocą WSDL Web Service'u w rejestrze usług (lub brokerze usług),
- rejestr usług lub broker usług odpowiada za rejestrację i kategoryzację opublikowanych usług. Dostarcza również możliwość ich wyszukiwania,
- klienci usług używają rejestru usług (lub brokera usług) do wykrycia Web Service'u opisanego za pomocą WSDL, a następnie wykonują zapytanie do dostawcy usługi. Dostawca usługi udziela odpowiedzi klientowi.

2.6.5 Warstwy architektury SOA

W SOA można wyszczególnić dziewięć warstw architektury. Każda z nich składa się z modelu fizycznego oraz logicznego. Model fizyczny opisuje sposób realizacji logiki za pomocą dostępnych technologii oraz produktów, natomiast logiczny zawiera wszystkie architektoniczne bloki, warunki logiczne, decyzje oraz inne - stanowi model konceptualny z dobranym poziomem abstrakcji.

Poszczególne warstwy w SOA to:

- Warstwa aplikacji - w jej skład wchodzą wszystkie aplikacje działające w środowisku IT i mające na celu wspieranie działania przedsiębiorstwa. Przykładami mogą być: aplikacje Java Enterprise lub .NET, systemy transakcyjne, bazy danych oraz systemy ERP i CRM.



Rysunek 2.3: Usługa sieciowa w SOA (Web Service).

- Warstwa komponentów - obejmuje składniki oprogramowania realizujące konkretne operacje usług. Odpowiada za fizyczną implementację usługi oraz jej opis. Dostarcza elementy zgodne z SCA (ang. *Service Component Architecture*) oraz specyfikacją SDO (ang. *Service Data Objects*).
- Warstwa usług - stanowi zbiór wszystkich usług zdefiniowanych za pomocą kontraktu pomiędzy dostawcą, a odbiorcą o sposobie dostarczenia usługi.
- Warstwa procesów biznesowych - w tej fazie tworzona jest kompozycja wielu usług realizujących konkretne biznesowe przypadki użycia. Proces biznesowy składa się z przepływów realizujących dany cel biznesowy organizacji i zagregowanych usług.
- Warstwa prezentacji - stanowi często wspólny interfejs dostępu do usług. Odpowiada za dostarczanie funkcji biznesowych dla klientów końcowych.
- Warstwa integracji - kluczowy element architektury SOA. Jej podstawową funkcją jest zlecanie klientom usług i dostarczanie ich przez dostawców. Przepływ operacji jaki jest wykonywany w tej warstwie to: odbiór zlecenia od klienta, dostarczenie go do dystrybutora, a następnie dostarczenie do klienta.
- Warstwa jakościowa - obejmuje monitorowanie procesów biznesowych odnoszących się do podstawowych wskaźników wydajności danej organizacji. Pełni rolę obserwatora dla pozostałych warstw i informuje o zdarzeniach niezgodnych z zakładanymi.
- Warstwa business intelligence - stanowi podstawę do budowania rozwiązań Business Intelligence umożliwiających modelowanie danych analitycz-

nych i eksploracje. W tej warstwie włączane są kluczowe zagadnienia związane z obiegiem informacji w przedsiębiorstwach.

- Warstwa zarządzania - odnosi się do zarządzania wszystkimi elementami cyklu życia SOA oraz warstw architektury (ze szczególnym naciskiem na warstwę jakościową). Wykorzystuje kluczowe wskaźniki wydajności lub efektywności (KPI) do egzekwowania wymagań jakościowych i funkcjonalnych wyspecyfikowanych przez przedsiębiorstwo [11].

2.6.6 Podstawowe zasady SOA

Systemy SOA mogą być bardzo różnorodne. SOA nie narzuca konkretnych technologii, a jej realizacja może odbywać się na wiele sposobów. Komunikacja między komponentami w SOA odbywa się głównie z wykorzystaniem WS (Web Service). Może jednak wykorzystywać też inne kanały komunikacji: HTTP, HTTPS, LDAP, FTP, IMAP, JMS oraz RMI.

Budowa systemu SOA może wykorzystywać szynę integracyjną ESB lub broker (można również łączyć wiele szyn ESB ze sobą lub łączyć broker'y z ESB). Mimo tej dużej dowolności istnieje zestaw zasad, na których powinien opierać się każdy system SOA.

- luźne powiązania - powiązania odnoszą się do połączeń lub relacji między poszczególnymi elementami. Termin „luźne powiązania" (ang. Loose Coupling) stanowi jeden z fundamentów SOA [30]. Odwołuje się do sposobu w jaki komponenty SOA współpracują ze sobą. Zasada „luźnych powiązań" promuje niezależną konstrukcję i ewolucję usług. Każdy z komponentów może pracować autonomicznie, wykonując określone czynności. Pracując razem, wymieniają między sobą komunikaty i mogą realizować to co jest zwykle możliwe przez duże, monolityczne aplikacje. „Luźne powiązania" pozwalają również na łatwą dekompozycję komponentów i wykorzystywanie ich do innych celów.

- interoperacyjność (ang. Interoperability) - opiera się na współpracy pomiędzy systemami. Zapewnienie interoperacyjności jest kolejną bardzo istotną zasadą, o której należy pamiętać, tworząc systemy SOA. W miarę rozwoju poszczególnych systemów (np. poprzez dodawanie nowych komponentów) problem integracji staje się coraz trudniejszy do rozwiązania. Należy już na etapie projektowania ograniczać do minimum wybór protokołów, które dany system będzie obsługiwał [30].

- kompozycyjność - zasada kompozycyjności (ang. Composability) jest związana z zachowaniem odpowiednich relacji między komponentami. Systemy,

które podążają za tą zasadą, cechują się możliwością łączenia swoich komponentów w różne kombinacje w celu spełnienia postawionych wymagań. Umiejętność efektywnego komponowania usług z już istniejących elementów jest kluczowym wymogiem do osiągnięcia niektórych z podstawowych celów przy tworzeniu systemów opierających się na architekturze zorientowanej na usługi.

- reużywalność - każda tworzona usługa powinna zachowywać zasadę reużywalności (ang. Reusability). Opiera się na takim projektowaniu usług, aby była możliwość wielokrotnego jej wykorzystania do tworzenia kolejnych usług [30].
- kontraktowość usług - wszystkie usługi powinny mieć zdefiniowany kontrakt (ang. Service contract), który zawierany jest każdorazowo pomiędzy usługą, a jej konsumentem. Wyrażane są przez niego cele i możliwości danej usługi. W kontraktach znajdują się również opisy informacji oferowanych i oczekiwanych przez usługi [29].
- enkapsulacja - zasada enkapsulacji (ang. Encapsulation) zakłada, że kontrakty usług mogą zawierać jedynie niezbędne informacje i mogą udostępniać jedynie te informacje, które są w nich zdefiniowane. Zasada ta podkreśla potrzebę ukrycia przez usługi tak wielu informacji jak to tylko możliwe.
- autonomiczność usług - autonomiczność usługi (ang. Service autonomy) jest kolejnym paradygmatem projektowania systemów typu SOA. Termin ten odwołuje się do usług o podwyższonej niezależności wobec środowisk wykonawczych. Usługa powinna mieć możliwość podmiany środowiska wykonawczego z lekkiego prototypowego (ang. Lightweight prototype) do pełnowymiarowego (ang. Full-blown), w którym są już uruchomione inne usługi odwołujące do niej. Zgodnie z zasadą autonomii każda z usług może być wdrażana, wersjonowana i zarządzana niezależnie od innych [29].
- wykrywalność usług - zasada wykrywalności usług (ang. Services discoverability) polega na tym, że usługi powinny być opisywane za pomocą metadanych w takich sposób, aby były efektywnie wyszukiwane, przetwarzane i interpretowane zarówno w czasie projektowania jak i wykonywania [29].
- spójność i ziarnistość usług - interfejsy usług w systemach SOA powinny być tak zaprojektowane, aby wiązały tylko określony zbiór wymagań biznesowych [30]. Należy zadbać o optymalną ziarnistość interfejsów dla obsługiwanych typów i rozmiarów danych (ziarnistość danych wejściowych i wyjściowych), wartości biznesowej oraz funkcjonalności (domyślna i parametryzowana ziarnistość funkcjonalności).
- bezstanowość usług - zasada bezstanowości usług (ang. Services statelessness) odwołuje się do minimalizacji użycia zasobów i ograniczania się do

przechowywania, i przetwarzania tylko absolutnie niezbędnych informacji. Usługa nie może być w stanie przetrzymywać informacji o wcześniejszych żądaniach klienta. Każda z informacji powinna być odizolowana od innych. Skuteczne stosowanie zasady bezstanowości może znacząco zwiększyć wydajność rozwiązania oraz zmniejszyć współbieżne działanie usług [30].

- enkapsulacja usług - enkapsulacja (ang. Encapsulation) stanowi jedną z podstawowych zasad poprawnego projektowania systemów typu SOA. Zapewnienie odpowiedniej hermetyzacji dla usług sprowadza się do ukrywania szczegółów konfiguracyjnych oraz implementacyjnych danej usługi.

2.6.7 Korporacyjna szyna usług - ESB

ESB (ang. Enterprise Service Bus) stanowi abstrakcyjną warstwę wymiany komunikatów. W systemach informatycznych wprowadza znaczną elastyczność, ponieważ pozwala na dynamiczne przyłączanie i odłączanie usług [11].

Głównym celem korporacyjnej szyny usług jest dostarczanie pewnego rodzaju wirtualizacji dla korporacyjnych zasobów, pozwalając na rozwijanie logiki biznesowej niezależnie od infrastruktury lub sieci oraz bez potrzeby pisania dodatkowego kodu. Zasoby w ESB są modelowane jako usługi, które oferują jedną lub więcej operacji biznesowych [9].

Dobra praktyka projektowania architektury systemu zapewnia, że wszystkie połączenia między aplikacjami będą realizowane z wykorzystaniem szyny [11]. Pełne sprostanie różnorodności wzorców integracji, jakie oferuje SOA, wymaga utrzymywania przez ESB trzech głównych paradygmatów integracji aplikacji korporacyjnych:

- Architektura zorientowana na usługi - rozproszone aplikacje zbudowane są z ziarnistych usług wieleokrotnego użytku z dobrze zdefiniowanymi, opublikowanymi i zgodnymi ze standardami interfejsami.
- Architektura zorientowana na komunikaty (ang. Message-driven architecture) - aplikacje wysyłają komunikaty między sobą.
- Architektura zorientowana na zdarzenia (ang. Event-driven architecture) - aplikacje generują i konsumują usługi niezależnie od pozostałych [9].

Stosowanie szyny integracyjnej wiąże się z wieloma zaletami. Przede wszystkim powoduje zmniejszenie kosztów dzięki szybkiemu i elastycznemu rozwiązaniu integracji eliminującym połączenia typu „Point-to-point". ESB umożliwia również rozwój obecnych środowisk bez wpływu na obecne, co pozwala na łatwe rozszerzanie systemu o nowe funkcjonalności [9].

2.7 Adaptacja SOA w architekturze korporacyjnej

Termin „Enterprise SOA"zyskał bardzo dużą popularność w ostatnich czasach. Wywodzi się z bardzo szerokiego zakresu. Posiada duże możliwości do łączenia elementów technologii z biznesem, jak i specyfikowania operacji poszczególnej usługi oferowanej pomiędzy systemami. Stanowi to również jedną z największych zalet SOA [7].

Architektura korporacyjna powinna dążyć do zapewnienia możliwie jak największej interoperacyjności tworzonych rozwiązań informatycznych i jak najłatwiejszej ich integracji. Problemem jest jednak zacieranie się świata biznesu i IT w kontekście architektury korporacyjnej.

W rozwiązaniu tych problemów pomaga SOA, która opiera się na koncepcji usług, kontraktów, procesów oraz orkiestracji (ang. Orchestration). Wszystkie z wymienionych elementów są powiązane z domeną biznesową. Pozwalają na kompleksowe opisywanie krytycznych dla biznesu funkcjonalności.

Problem ze względu na swój zakres oraz liczbę elementów branych pod uwagę do jego rozwiązania jest stosunkowo złożony. Konieczne jest dlatego oparcie się na odpowiedniej metodyce. W pracy zostanie przedstawione jak zaadaptować SOA w architekturze korporacyjnej z wykorzystaniem SoaML z zastosowaniem opracowanej optymalnej metody.

Rozdział 3

Przegląd języków do projektowania systemów informatycznych o architekturze SOA

3.1 Wstęp

Typy modeli tworzonych w ramach architektury korporacyjnej można podzielić na kilka grup:

- Medium komunikacyjne między interesariuszami, którzy reprezentują zarówno dział informatyki, jak i inne jednostki organizacji.
- Forma ukazania i dokumentowania pewnych decyzji - zarówno na poziomie technicznym w ramach architektury infrastruktury technicznej (na przykład w zakresie dotyczącym serwerów oraz sieci komputerowych), jak i biznesowym w ramach architektury biznesowej (odnośnie przebiegu procesów biznesowych).
- Abstrakcja organizacji, opisująca strukturę i działanie poszczególnych elementów składowych (zarówno na płaszczyźnie technicznej, jak i biznesowej). Taki opis pozwala na planowanie rozwoju organizacji, wspomaga jej transformację.

Opracowana architektura powinna być zrozumiała dla odbiorcy, tak aby na jej podstawie można było podjąć decyzje związane z funkcjonowaniem organizacji lub wdrożyć określone rozwiązania informatyczne. Istotne jest całościowe rozważenie organizacji - zarówno pod kątem pełnionych funkcji jak i jej struktury [23].

W niniejszym rozdziale zostaną omówione języki wykorzystywane do modelowania architektury korporacyjnej. Przy wykorzystaniu SoaML lub ArchiMate można w pełni zamodelować architekturę korporacyjną. Inne jęzki jak: BPEL, BPMN, czy UML, mogą wspierać proces projektowania jedynie w poszczególnych domenach.

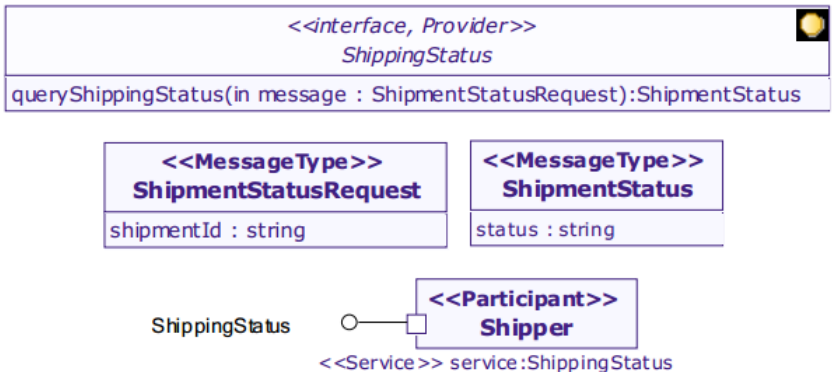
3.2 SoaML

3.2.1 Czym jest SoaML?

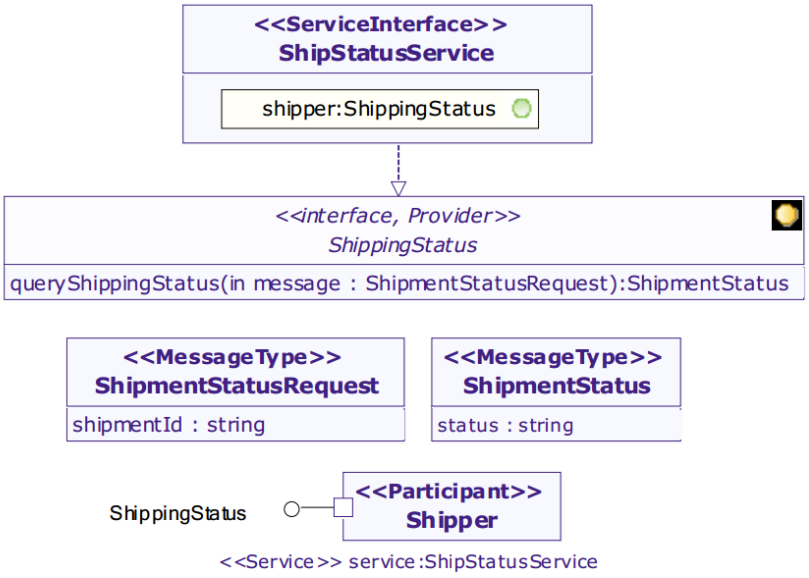
SoaML (The Service oriented architecture Modeling Language) stanowi język wyspecyfikowany przez OMG (Object Managment Group) w 2009 roku, który definiuje profil UML i metamodel dla projektowania usług w architekturze zorientowanej usługowo. Podstawowym celem SoaML jest modelowanie i projektowanie usług, tak aby wspierać dewelopment oparty na podejściu sterowanym modelami z perspektywy biznesowej i IT.

SoaML definiuje trzy różne podejścia dla specyfikowania usług:

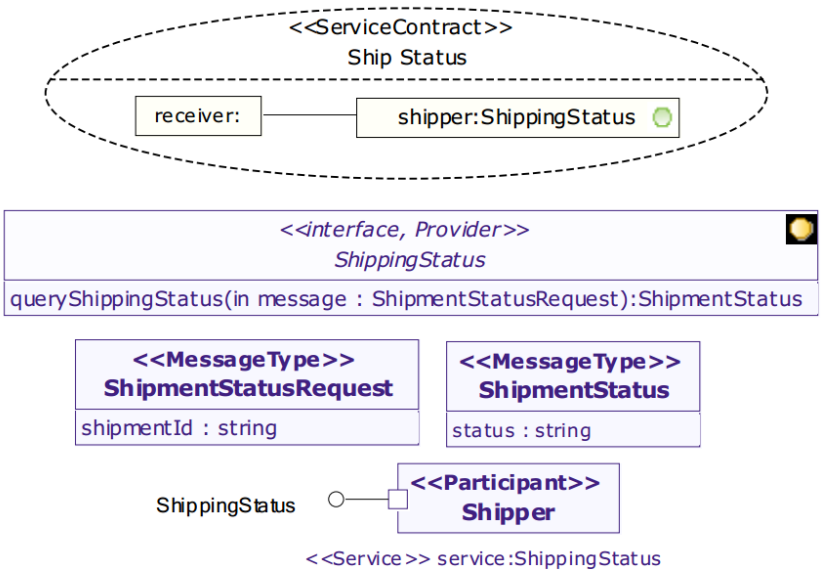
- Proste interfejsy (ang. Simple interfaces) - prosty interfejs, który opiera się na jednokierunkowej interakcji dostarczanej przez uczestnika interakcji jako interfejs UML (rys. 3.1). Uczestnik otrzymuje operacje na swój port i może dostarczyć rezultaty dla wywołującego zapytanie. Ten rodzaj jednokierunkowej interakcji może być stosowany dla anonimowych inicjatorów interakcji. Uczestnik interakcji nie ma informacji o inicjatorze ani o porządku usług. Jednokierunkowe usługi są najczęściej określane jako „RPC style Web Services". Prosty interfejs wykorzystywany jako port usługi może opcjonalnie stanowić element *«ServiceInterface»* [18].
- Interfejsy usług (ang. Service interfaces) - podejście bazujące na interfejsach usług skupia się na dwu i większej ilości interakcji między usługami, wymagając jednocześnie wyspecyfikowania zbioru powiązanych ze sobą interfejsów w ramach specyfikacji jednej usługi (rys. 3.2). Opiera się na komponentach i zezwala na połączenia między nimi z wykorzystaniem portów. Porty powinny specyfikować pożądane i dostarczane interfejsy.
- Kontrakty usług (ang. Contract interfaces) - podejście w oparciu o kontrakty usług definiuje specyfikacje usług, które definiują role każdego uczestnika usługi (takich jak dostawców i klientów) oraz jakie interfejsy powinni implementować do realizacji ich ról w usłudze (rys. 3.3). Te interfejsy z kolei są typami portów uczestników, które zobowiązują ich do realizacji ról w danym kontrakcie usługi. Podejście to rozszerza elementy współpracy (ang. Collaboration) UML do modelowania części strukturalnej interakcji usług.



Rysunek 3.1: Specyfikacja usługi „ShipStatus" wykorzystując podejście „prostego interfejsu". Zawiera prosty interfejs dostawcy usługi, jego operacje, typy wiadomości i odpowiedni port uczestnika interakcji [8].



Rysunek 3.2: Specyfikacja usługi „ShipStatusService" wykorzystując podejście oparte na interfejsach usług [8].



Rysunek 3.3: Specyfikacja usługi „ShipStatus" wykorzystując podejście oparte na „kontrakcie usługi" [8].

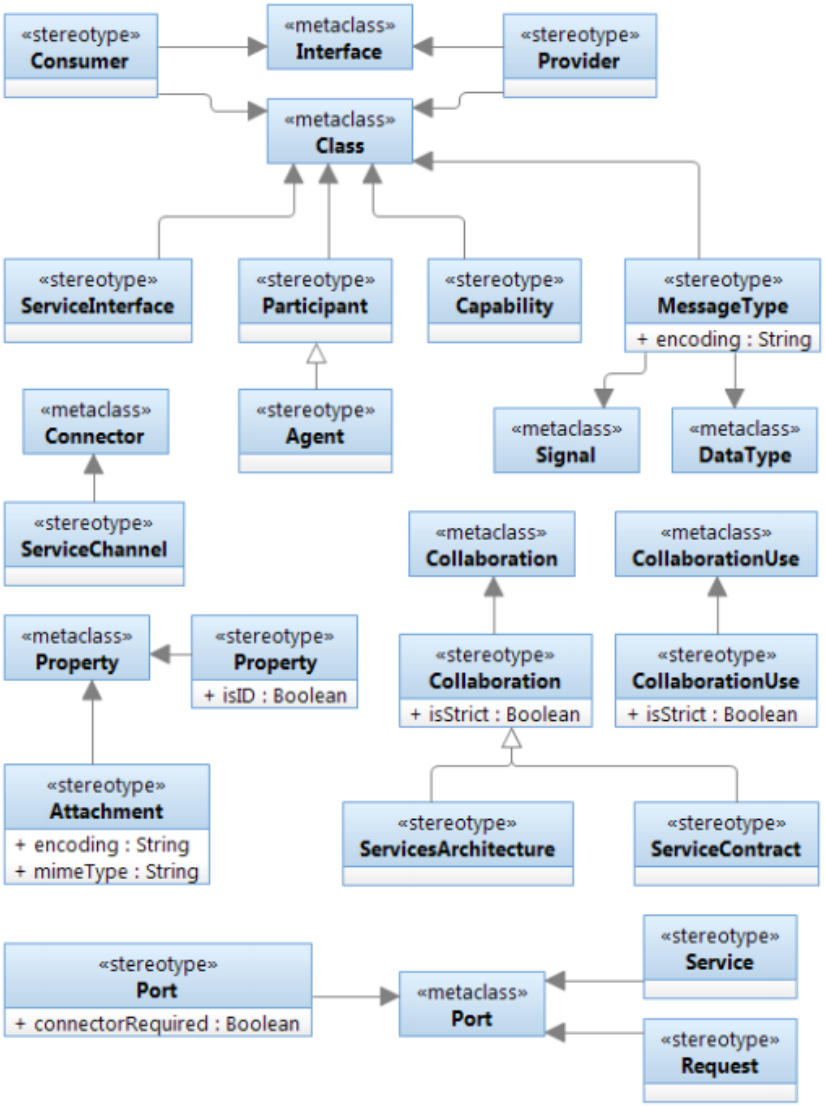
Wszystkie podejścia charakteryzują się wykorzystywaniem różnych elementów UML. Podstawowa różnica pomiędzy podejściem bazującym na kontrakcie i interfejsie polega na zdefiniowaniu interakcji pomiędzy uczestnikami niezależnie od nich w *«ServiceContract»*). Element ten określa obowiązki wszystkich uczestników interakcji lub indywidualnie dla każdego z uczestników i zapytań.

3.2.2 Modelowanie z wykorzystaniem SoaML

SoaML wspiera modelowanie wymagań dla SOA w oparciu o specyfikowanie usług systemowych, indywidualnych usług dla interfejsów oraz specyfikację implementacji usług. Metamodel SoaML 3.4 rozszerza metamodel UML dla bezpośredniego modelowania usług w rozproszonych środowiskach [8].

SoaML charakteryzuje się dodatkowymi abstrakcjami [11] i rozszerza UML w sześciu głównych obszarach:

- Uczestnicy realizujący kontrakty (ang. Participants) - na diagramach SoaML oznaczeni są stereotypem *«Paricipant»*.
- Interfejsy usług (ang. Service interfaces) - wykorzystywane do opisywania operacji dostarczanych i wymaganych do pełnej funkcjonalności usługi. Interfejs usługi może być wykorzystywany jako protokół dla portu usługi lub odpytywany port. Oznaczone są stereotypami *«ServiceInterface»*.



Rysunek 3.4: Podstawowe rozszerzenia UML zdefiniowane jako stereotypy dla SoaML [8].

- Kontrakty usług (ang. Service contracts) - wykorzystywane do opisywania wzorców interakcji między encjami usług. Umowa o świadczenie usług jest używana do modelowania porozumienia między dwiema lub większą ilością stron. Każda z usług z kontraktu posiada interfejs, który reprezentuje ją jako dostawcę bądź konsumenta. Oznaczone są stereotypami *«ServiceContract»*
- Usługi architektur (ang. Services architecture) - wyższy element widoku prezentujący zbiór elementów SOA. Oznaczany stereotypem *«ServicesArchitecture»*.
- Dane usług (ang. Service data) - wykorzystywane do opisywania komunikatów dla usług i ich załączników. Element „Typ wiadomości"(ang. Message Type) jest używany do specyfikowania informacji wymienianych pomiędzy konsumentami usług i dostawcami. Określa rodzaj komunikatu. Oznaczany jest na diagramach jako stereotyp *«MessageType»*. Załączniki są oznaczane jako stereotyp *«Attachment»*.
- Funkcjonalność (ang. Capabilities) - kolejny zdefiniowany element SoaML. Na diagramach klas określa funkcjonalność lub zasoby jakie oferują uczestnicy architektury. Przedstawiany jest w postaci komponentu ze stereotypem *«Capabilitiy»* [8].

3.3 ArchiMate

3.3.1 Czym jest ArchiMate?

ArchiMate stanowi otwarty i niezależny język modelowania dla architektur korporacyjnych. Zapewnia instrumenty, aby umożliwić architektom przedsiębiorstwa opis, analizę i wizualizację relacji między domenami biznesu w jednoznaczny sposób. Pozwala ponadto na opisywanie procesów biznesowych, struktur organizacyjnych, przepływów informacji, systemów informatycznych oraz technicznej infrastruktury.

ArchiMate to techniczny standard The Open Group. Jest wspierany przez różne narzędzia informatyczne do modelowania architektury korporacyjnej: Enterprise Architect, BizzDesign Architect, ARIS lub Microsoft Visio [32].

ArchiMate bazuje na dwóch podstawowych założeniach:

- podejściu warstwowym - związanym z warstwą biznesową, aplikacji, danych i techniczną,
- podejściu zorientowanym na usługi - usługi zewnętrzne są udostępniane przez warstwę niższą i wykorzystywane przez warstwę wyższą (na przykład warstwa aplikacji udostępnia usługi wykorzystywane w ramach warstwy bizne-

sowej). ArchiMate wprowadza ponadto pojęcie usług wewnętrznych, które są wykorzystywane w ramach określonej warstwy.

Dobra komunikacja pomiędzy warstwami: biznesową, aplikacyjną i technologiczną stanowi jedną z największych zalet języka ArchiMate.

W warstwie biznesowej (ang. Business layer) zawarte są usługi świadczone klientom zewnętrznym oraz produkty. Są realizowane przez przez procesy biznesowe i związanych z nimi aktorów biznesowych.

Warstwa aplikacji (ang. Application layer) wspiera warstwę biznesową poprzez usługi aplikacyjne, które są realizowane przez poszczególne komponenty aplikacyjne.

Warstwa technologiczna (ang. Technology layer) odzwierciedla usługi infrastrukturalne (na przykład usługi komunikacyjne) niezbędne do działania aplikacji.

Od wersji 2.0 możliwości języka ArchiMate zostały rozbudowane poprzez zdefiniowanie części centralnej oraz rozszerzeń związanych z implementacją i konfiguracją [23].

3.3.2 Modelowanie z wykorzystaniem ArchiMate

Jednym z najbardziej popularnych podejść do modelowania architektury korporacyjnej z wykorzystaniem ArchiMate jest oparcie się na metodzie top-down.

W początkowym etapie powinno zdefiniować się warstwę biznesową. Zalecane jest dlatego utworzenie w pierwszej kolejności „Business Model Canvas". Stanowi on szablon dla dokumentowania nowych lub istniejących modeli biznesowych. W efekcie zostaje utworzony diagram złożony z bloków opisujących różne istotne elementy organizacji. Stanowi on szkic strategii, który ma zostać wdrożony w ramach struktur, procesów i organizacji. Składają się na niego takie elementy jak: segmenty klientów, propozycje wartości, kanały, relacje z klientami, strumienie przychodów, kluczowe zasoby, działania, partnerzy oraz struktura kosztów.

Następnie powinno zidentyfikować się przepływy pomiędzy różnymi encjami „Business Model Canvas"oraz dokonać mapowania ogólnego modelu biznesu jako usług powiązanych. W ArchiMate, na najwyższym poziomie abstrakcji, organizacja i odpowiadający jej model biznesowy mogą być reprezentowane jako pojedyncza usługa biznesowa (ang. Business Service). Segmenty klientów i kluczowi partnerzy mogą być reprezentowani jako encje aktorów biznesowych (ang. Business actors). Każdy z nich powinien mieć przypisaną co najmniej jedną rolę biznesową (ang. Business role) oraz połączenie z usługą organizacji przez biznesowe relacje interfejsu (ang. Business interface relationships).

W kolejnym kroku rozwijane są szczegóły ogólnego modelu biznesowego. Następuje przejście z „Business Model Canvas"na „Enterprise Canvas".

3.4 Inne języki

3.4.1 BPMN

BPMN (ang. Business Process Modelling Notation) stanowi standard modelowania procesów biznesowych opracowany przez Business Process Management Initiative (BPMI), a następnie rozwijany przez Object Management Group (OMG).

Nie jest to język, który pozwoli na pełne zamodelowanie SOA lub architektury korporacyjnej. Można go wykorzystać jednak do modelowania fragmentów dotyczących procesów biznesowych.

Definiuje diagramy procesów biznesowych - zaprojektowane tak, żeby z jednej strony były łatwe do zrozumienia, a z drugiej pozwalały na modelowanie również złożonych procesów. Składa się ze skończonego i jednoznacznie zdefiniowanego zbioru elementów graficznych. Pozwalają na tworzenie łatwo zrozumiałych modeli procesów biznesowych.

Elementarne typy obiektów BPMN stanowią:

- obiekty przepływu (ang. Flow objects) - stanowią je elementy odnoszące się do zdarzeń, czynności lub obiektów decyzyjnych,
- obiekty łączące (ang. Connecting objects) - do tej grupy zaliczyć można sekwencje elementów, komunikaty,
- tory przepływu (ang. Swimlanes),
- artefakty (ang. Artifact).

BPMN może być wykorzystywany do modelowania:

- procesów interakcji (globalnych) - wykorzystywane do reprezentacji interakcji między dwoma lub większą ilością obiektów biznesowych,
- abstrakcyjnych procesów (publicznych) - reprezentują interakcje między procesami prywatnymi a uczestnikami zewnętrznymi, albo dwoma procesami prywatnymi,
- prywatnych procesów biznesowych - procesy wewnętrzne, specyficzne dla konkretnej interakcji.

3.4.2 BPEL

BPEL (Business Process Execution Language) stanowi język pozwalający na definiowanie i wykonywanie procesów biznesowych. Został opracowany przez konsorcjum OASIS.

Podobnie jak BPMN umożliwia modelowanie jedynie fragmentu architektury korporacyjnej związanej z procesami biznesowymi. Jest postrzegany jako procesowe rozszerzenie standardów zdefiniowanych dla usług sieciowych, które pozwalają na współpracę między aplikacjami w środowiskach heterogenicznych. Opiera się na takich standardach jak SOAP, WSDL oraz Universal Description, Discovery and Integraton (UDDI).

W BPEL można opisywać procesy na dwa sposoby - abstrakcyjny lub wykonywalny.

Proces abstrakcyjny jest określony tylko częściowo. Musi być otwarcie zadeklarowany jako abstrakcyjny i nie powinien być wykonywalny. Ponadto udostępnia dwa mechanizmy umożliwiające ukrycie jego szczegółów: nieprzezroczyste (ang. Opaque tokens) oraz pominięcie (ang. Omission).

Z kolei proces wykonywalny w pełni specyfikuje opis, który można umieścić w środowisku wykonawczym. Musi być zdefiniowany wyłącznie na podstawie usług sieciowych oraz danych przedstawionych w języku XML. Konstrukcje procesu wykonywalnego powinny być dostępne dla procesu abstrakcyjnego.

Istnieją również rozszerzenia języka BPEL. Jednym z nich jest BPEL4WS, który nie uwzględnia człowieka w procesie biznesowym. Przeciwieństwo stanowi zaś BPEL4People, w którym człowiek stanowi integralną część procesu biznesowego. Innym przykładem jest oparty na języku Java BPELJ, który umożliwia zastąpienie usług sieciowych szeregiem niewielkich aplikacji. [11].

3.4.3 UML

UML (Unified Modeling Language) jest jednym z najpowszechniejszych graficznych języków wykorzystywanych do modelowania architektury korporacyjnej. Został stworzony w 1994 roku i jest rozwijany przez OMG (Object Management Group).

Przy wykorzystaniu elementów języka UML można wizualizować, specyfikować oraz budować systemy informatyczne. Stosowanie UML pozwala w graficzny sposób odwzorować zarówno behawioralne jak i strukturalne spojrzenie na system.

Modelowanie systemów z wykorzystaniem UML umożliwia jednoczesne dokumentowanie całości systemu w zrozumiały dla całego zespołu sposób [11]. Służy głównie do modelowania systemów informatycznych, w tym również uwzględniających aspekty architektoniczne.

UML znajduje coraz częściej zastosowanie w inżynierii systemów informatycznych, w modelowaniu procesów biznesowych oraz reprezentacji struktur organizacyjnych. W tym celu bardzo często wykorzystywane są możliwości jego rozszerzania - na przykład za pomocą profili. Istotę ich stosowania stanowi fakt, że dodajemy możliwość używania konkretnych znaczeń w stosunku do istniejących elementów modelu.

3.4.4 IDEF

Języki IDEF (Integrated DEFinition Methods) stanowią rodzinę języków wykorzystywaną do analizy biznesowej i modelowania. Pierwotnie wiązane były głównie z wojskowością. Obecnie IDEF ma bardzo wiele zastosowań i występuje w wielu odmianach.

IDEF0 są wykorzystywane do modelowania funkcji. Pozwalają na modelowanie decyzji, działań i czynności. Zarówno na poziomie systemu informatycznego, jak i całej organizacji. Celem opracowania IDEF0 było utworzenie języka pozwalającego na ułatwienie analizy systemów i organizacji, a także poprawienie komunikacji pomiędzy analitykami biznesowymi, i klientami.

IDEF3 pozwala na modelowanie procesów biznesowych. Przystosowany jest również do prezentowania behawioralnych aspektów.

IDEFIX został utworzony z kolei z przeznaczeniem do modelowania danych. Dostarcza semantycznych konstrukcji, które pozwalają na modelowanie konceptualnych schematów dla struktur danych używanych na poziomie całej organizacji.

Rozdział 4

Metody projektowania rozwiązań w architekturze usługowej

4.1 Wstęp

W projektowaniu rozwiązań w architekturze usługowej niezbędne jest posługiwanie się odpowiednimi metodami.

Na rynku istnieje obecnie wiele metod o różnej charakterystyce. Posiadają inne fazy, przebiegi oraz stopnie dojrzałości.

Wiele z metod wciąż jest słabo udokumentowana. Nie wszystkie posiadają również powszechnie dostępną specyfikację, a wykorzystanie sporej części z nich wiąże się z poniesieniem kosztów finansowych.

Niniejszy rozdział będzie stanowił przegląd metod projektowania rozwiązań w architekturze usługowej. Zostaną rozważone takie metody jak: RUP4SOA, SOMA, metoda Papazoglou oraz metoda Thomasa Erl'a.

4.2 RUP4SOA

4.2.1 Czym jest RUP?

RUP (ang. Rational Unified Process) stanowi proces wytwarzania oprogramowania oparty na iteracjach. Metoda została zdefiniowana przez grupę Rational Software (przejętą przez firmę IBM w 2003 roku).

RUP zapewnia zdyscyplinowane podejście do przydzielania zadań i obowiązków w ramach rozwoju organizacji. Celem podstawowym tej metody jest dostarczanie wysokiej jakości oprogramowania spełniającego potrzeby użytkowników koń-

cowych w zgodzie z harmonogramem i ramami budżetowymi [16]. Stanowi przede wszystkim bardzo duży zbiór praktyk, który może być dostosowywany i rozszerzany w celu jak najlepszego dopasowania się do danej organizacji. Charakterystyczny dla metody jest rozwój sterowany przypadkami użycia (ang. Use Case Driven Development) [10].

W RUP można wyróżnić poszczególne fazy:

- faza początkowa (ang. Inception phase) – obejmuje definicję problemu - zagadnienia biznesowego. Następuje w niej wstępne określenie wymagań, ryzyka, kosztów, harmonogramu, a także architektury systemu,
- faza opracowania (ang. Elaboration phase) – analiza dziedziny zagadnienia oraz ustalenie wymagań dla architektury systemu,
- faza konstrukcji (ang. Construction phase) – w tej fazie główny nacisk położony jest na budowę komponentów i innych funkcjonalności opracowywanego systemu,
- faza przekazania (ang. Transiation phase) – system jest przekazywany użytkownikowi oraz wdrażany. Szkoleni są pracownicy do obsługi systemu, następuje walidacja i końcowe sprawdzenie jakości [16].

4.2.2 Wykorzystanie RUP w projektowaniu SOA

RUP4SOA stanowi modyfikację metody RUP i dołączono do niej zadania i produkty potrzebne przy projektowaniu rozwiązań w architekturze usługowej. RUP4SOA stanowi komercyjny plug-in dla framework'a RUP. Rozszerza standardowy pakiet o zbiór dodatkowych artefaktów i właściwości. W odróżnieniu od klasycznego RUP metodę tą wyróżniają trzy dyscypliny związane z analizą i projektowaniem:

- analiza i projektowanie architektury SOA - przygotowanie architektury SOA zgodnie z wymaganiami,
- analiza i projektowanie kontraktów w architekturze SOA - analizie poddane są procesy integracyjne związane z dostarczaniem usług i realizacją kontraktów między organizacjami,
- analiza i projektowanie logiki usług SOA - identyfikacja usług w systemach informatycznych w jednostkach, w których wdrażana będzie architektura SOA.

Reszta dyscyplin jest analogiczna do metody RUP [11]. W RUP4SOA można wyróżnić poszczególne fazy:

- modelowanie biznesowe,

- specyfikacja wymagań,
- analiza i projektowanie architektury SOA,
- analiza i projektowanie kontraktow SOA,
- analiza i projektowanie logiki usług,
- implementacja,
- testowanie,
- wdrożenie,
- zarządzanie zmianą i konfiguracją,
- zarządzanie projektem,
- środowisko.

W fazie analizy i projektowania przygotowywana jest architektura SOA zgodnie z postawionymi wymaganiami i modelem biznesowym. Istotne jest, aby architektura była projektowana z zamysłem o możliwie jak najprostszym późniejszym wdrożeniu. Kolejną dyscyplinę stanowi analiza i projektowanie kontraktów w architekturze usługowej. Ta faza opiera się na analizie procesów integracyjnych związanych z dostarczaniem usług i wymianą kontraktów między organizacjami. Następny element metody RUP4SOA stanowi analiza i projektowanie logiki usług SOA, który powiązany jest z identyfikacją usług informatycznych w organizacjach.

Po fazach analizy i projektowania następują kolejno implementacja oraz testy. Utworzony produkt zostaje wdrożony na przygotowane środowisko. Równolegle trwają również prace związane z zarządzaniem projektem, konfiguracją oraz zmianami [11].

Podstwowym produktem wytworzonym przez „Architekta oprogramowania" w przypadku RUP4SOA jest model usług. Do podstawowych zadań „Architekta oprogramowania"zalicza się realizację fazy „identyfikacji usług".

Projektant w RUP4SOA odpowiada za przygotowanie projektu usługi, a jego odpowiedzialność jest związana z produktami: komunikat, usługa, kanał usługi, bramka usługi, współpraca usługi, partycja usługi, specyfikacja usługi oraz komponent usługi.

4.2.3 Zalety i wady RUP4SOA

RUP4SOA jest jedną z najpopularniejszych metod stosowanych do projektowania architektury usługowej. Największy nacisk kładzie na fazy związane z analizą i projektowaniem systemu informatycznego - rozszerzone o elementy związane z usługami sieciowymi. Fakt ten może powodować większą zgodność między wyobrażeniami klienta, a rzeczywistym produktem w formie systemu.

Jedną z jej największych zalet jest również opieranie się na metodyce RUP, która wykorzystuje doświadczenia i praktyki przyjęte przez organizacje na przestrzeni wielu lat. [13] Skutkiem tego dopracowano metodę w bardzo wysokim stopniu oraz pozbyto się sporej ilości błędów i niedociągnięć.

Do wad metody można zaliczyć fakt, że nie specyfikuje wielu elementów istotnych do budowy rozwiązań integracyjnych. Skupia się na projektowaniu pojedynczego systemu, który może być jedynie włączony do platformy integracyjnej [11]. Powoduje to znaczne utrudnienia związane ze skalowalnością tworzonych rozwiązań.

4.3 SOMA

4.3.1 Czym jest SOMA?

SOMA (ang. Service-Oriented Modeling and Architecture) stanowi kolejną metodę projektowania systemów w architekturze usługowej. Metoda cyklu rozwoju oprogramowania, która definiuje kluczowe techniki i opisuje poszczególne role w projekcie SOA. Swoim zakresem obejmuje również strukturę podziału pracy WBS (Work Breakdown Structure).

WBS obejmuje zadania związane z wejściowymi i wyjściowymi produktami pracy, normatywnymi wskazówkami do szczegółowej analizy, projektowaniem, implementacją i wdrażaniem usług oraz komponentów potrzebnych do budowy wydajnego, reużywalnego środowiska [11].

SOMA została utworzona przez firmę IBM. W swoim podejściu metoda wykorzystuje wiele elementów z Worldwide Project Management Method (jedna z metod zarządzania projektami) [1].

4.3.2 Fazy metody SOMA

SOMA bazuje na siedmiu podstawowych fazach:

- modelowanie biznesowe (ang. Business modeling and transformation),
- zarządzanie rozwiązaniem (ang. Solution management),
- identyfikacja (ang. Identification),
- specyfikacja (ang. Specification),
- realizacja (ang. Realization),
- implementacja (ang. Implementation),
- monitorowanie i zarządzanie (ang. Monitoring and management).

Poszczególne fazy nie następują po sobie w sposób liniowy. SOMA opiera się na iteracyjnym, przyrostowym podejściu do realizacji rozwiązań SOA. Powoduje to łagodzenie ryzyk projektowych oraz wynika z cyklu życia usług w modelu SOA. Zadania związane z budową rozwiązań informatycznych realizowane są w niej w podobny sposób bez względu na zakres projektu.

Można również mówić o fraktalnej charakterystyce metody. Każda kolejna iteracja (ang. Successive iteration) jest powiązana z pojęciem ewolucji usług. Opiera się nie tylko na ryzykach dotyczących implementacji, ale również na zależnościach związanych ze zbiorem usług, gdy zmieniają się w trakcie cyklu tworzenia systemu. W SOMA przypisywanie priorytetów w modelu usług odbywa się z wykorzystaniem diagramów zależności usług (ang. Service-dependency diagram). Na podstawie ryzyk związanych z architekturą rozwiązania wyłaniany jest podzbiór usług przewidywany do implementacji w kolejnej iteracji.

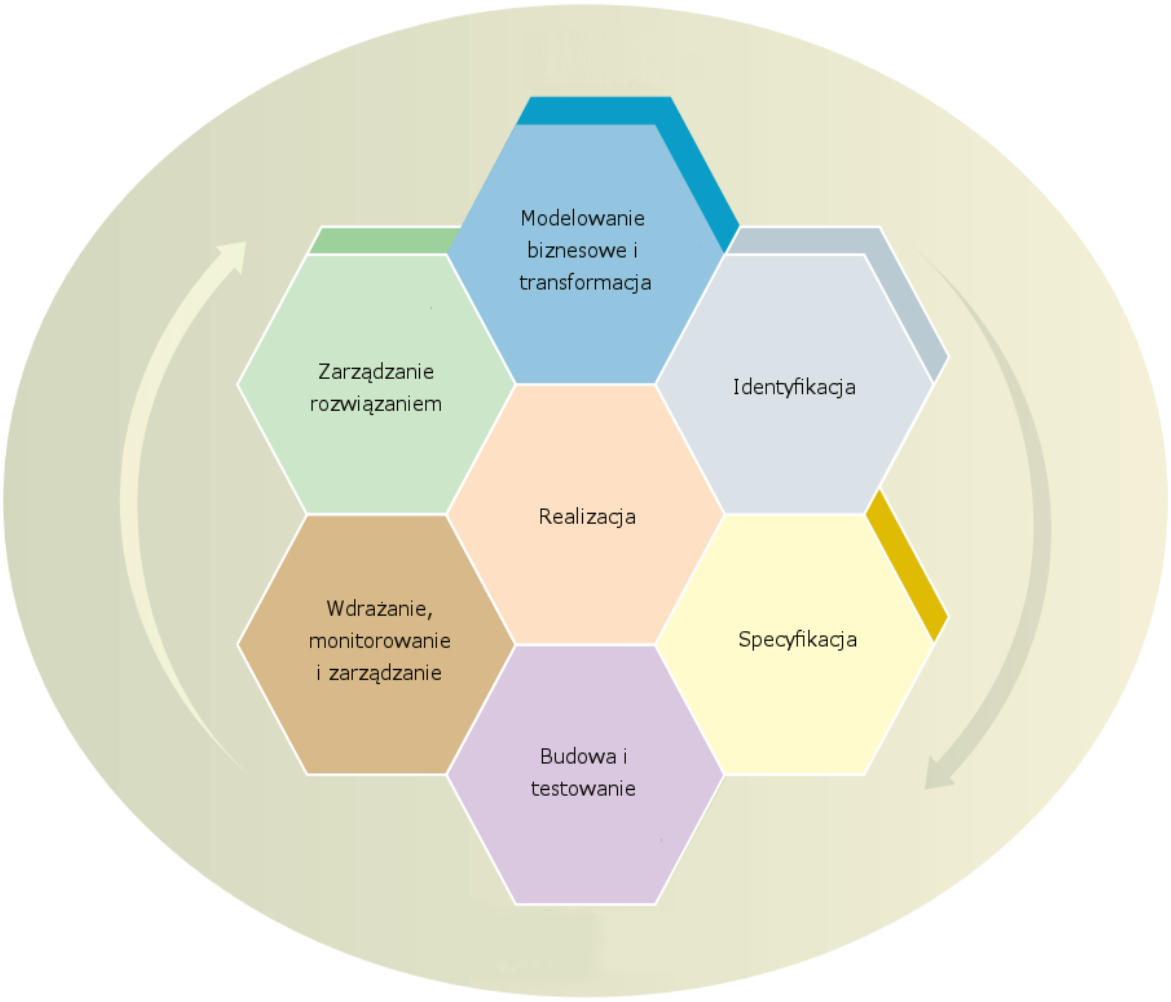
Charakterystyczną cechą dla SOMA jest również pojęcie przypadku usługowego (ang. Service case), który identyfikuje „reużywalny"(ang. Reuse) zbiór operacji usługi. Ważne jest, aby zidentyfikować taki zestaw usług, który łącznie pozwala na spełnienie założonych celów biznesowych [11].

W fazie modelowania biznesowego i transformacji działalność organizacji jest modelowana, symulowana i optymalizowana. Na tym etapie wyszczególnia się również główny obszar przekształcenia danej jednostki - podstawowy dla kolejnych projektów opierających się na pozostałych fazach SOMA. Modelowanie biznesowe i transformacja jest opcjonalnym elementem metody, ale ściśle zalecanym.

Realizacje SOA mają hybrydową naturę i z reguły obejmują wiele typów rozwiązań. W fazie zarządzania rozwiązaniem inicjowany jest projekt oraz wybierany odpowiedni scenariusz realizacji. Występuja dwa możliwe scenariusze: wytwarzania oprogramowania (ang. Custom development) oraz integracji pakietów aplikacji (ang. Package application integration). Na tym etapie tworzona jest również konfiguracja metody w celu dostosowania jej do potrzeb danego projektu. Ponadto zdefiniowaniu podlegają: zadania, role oraz produkty [1].

Faza identyfikacji związana jest z identyfikacją trzech elementarnych dla SOA konstrukcji: usług, komponentów oraz przepływów [11]. Do zadań tego etapu należy: dekompozycja obszarów biznesowych (Domain Decomposition), modelowanie usług w kontekście celów biznesowych (ang. Goal-Service Modeling) oraz analiza istniejących zasobów IT (ang. Existing Assets Analysis) [27]. Do produktów końcowych tej fazy zalicza się listę kandydujących usług do realizacji oraz powiązań między nimi [11].

Podczas specyfikacji usług projektowane jest rozwiązanie. Powstaje projekt podstawowy oraz niektóre elementy projektu szczegółowego. Analizowane są zasoby IT pod kątem zależności ze zidentyfikowanymi usługami w poprzedniej fazie: przepływami oraz zdarzeniami. Metoda dostarcza gotowe szablony, wzorce oraz



Rysunek 4.1: Fraktalne podejście do budowy systemów - fazy metody SOMA [1].

techniki wykorzystywane w celu określenia usług, przepływów i komponentów.

W skład fazy specyfikacji wchodzą następujące elementy:

- Identyfikacja zależności między usługami - w oparciu o szczegółowy przegląd danej usługi możliwe jest odkrycie jej zależności z innymi usługami lub aplikacjami. Mogą okazać się one niezbędne do dostarczenia niektórych funkcjonalności.
- Test papierka lakmusowego (ang. Service litmus test) - powiązany z podejmowaniem decyzji dotyczących ekspozycji.
- Identyfikacja kompozycji usług i przepływów - przegląd procesów biznesowych oraz obszarów funkcjonalnych. Umożliwia ustalenie kompozycji usług z innych usług i ich przepływów dostarczających pożądaną funkcjonalność biznesową.
- Identyfikacja wymagań niefunkcjonalnych - wymagania definiujące oczekiwany poziom usług.
- Definicja specyfikacji komunikatów - obejmuje identyfikację i specyfikację formatu oraz zawartości komunikatów wejściowych i wyjściowych dla usług.
- Dokumentacja decyzji dotyczących zarządzania stanem projektu.

Faza realizacji związana jest ze sprawdzaniem stabilności i możliwości wykonania zaprojektowanego rozwiązania poprzez budowę prototypów. Zadanie to powinno mieć miejsce w jak najwcześniejszym etapie projektu, aby zminimalizować niepowiedzenia i uniknąć niepotrzebnych ryzyk. Faza ta stanowi pewnego rodzaju formę technicznego studium wykonalności.

W skład fazy realizacji wchodzą następujące czynności:

- iteracyjna alokacja usług do komponentów,
- przypisanie komponentów w warstwach znajdujących się w architekturze aplikacji,
- identyfikacja i oszacowanie ograniczeń technologicznych, które mogą wpłynąć na wykonalność zadań.

W fazach implementacji i wdrożenia oraz monitorowania i zarządzania są konstruowane, generowane i łączone usługi, komponenty oraz przepływy. Dla istniejących komponentów tworzone są również poszczególne elementy opakowujące oraz odpowiednie mechanizmy. Wykonywane są również testy: integracyjne, jednostkowe, komponentów, przepływów oraz systemu dla usług [11, 1].

4.3.3 Produkty SOMA

Stosowanie SOMA przynosi rezultaty między innymi w postaci utworzonych modelów - usług i informacji.

Model usług zawiera zbiór informacji dotyczących aktualnie zidentyfikowanych usług w organizacji, które będą wykorzystane do realizacji założonych celów biznesowych oraz procesów. W modelu tym znajdują się również dane opisujące kwestie biznesowe, funkcjonalne, niefunkcjonalne oraz dotyczące technicznej realizacji. Wyszczególnione są również informacje o hierarchii złożoności, ekspozycji, zależności między usługami oraz wymagania związane z poziomem jakości.

Model informacji odpowiada za kontekst biznesowy. Zawiera elementy takie jak: model koncepcyjny, decyzje dotyczące realizacji oraz relacje i ekspozycje encji.

Oprócz wymienionych modeli metoda SOMA dostarcza również elementy: kontekst biznesowy, definicję procesów, identyfikację procesów, katalog reguł biznesowych oraz listę zdarzeń biznesowych.

4.3.4 Zalety i wady SOMA

Stosowanie metody SOMA może przynosić szereg korzyści. Model usług stanowiący końcowy rezultat etapu modelowania pozwala na łatwe i szybkie opracowanie projektu technicznego, implementację nowych usług oraz udostępnienie usług na podstawie już istniejących.

Ponadto SOMA zakłada wysoki stopień skalowalności systemów tworzonych zgodnie z jej regułami. Zapewnia elastyczność organizacji w związku ze zmianami biznesowymi oraz umożliwia redukcję kosztów wdrażania nowych usług.

Model usług powstały z zastosowania SOMA jest niezależny od platformy. Można wytwarzać go za pomocą wielu dostępnych na rynku narzędzi: IBM Rational Software Architect, Enterprise Architect, ARIS czy ModelioSoft.

SOMA doskonale wspiera identyfikację usług zgodnie z zasadą „top-down" w modelach procesów biznesowych i innych artefaktach analizy biznesowej. Specyfikacja usług również jest odpowiednio zdefiniowana. Jednakże definicja przejścia pomiędzy fazami identyfikacji i specyfikacji jest niejednoznaczna i często w projektach powstają z tego powodu różne nieścisłości (jak na przykład duplikaty usług) [35].

Do wad SOMA można zaliczyć fakt, że szczegółowy opis projektu technicznego usług biznesowych dostarcza tylko szczątkowych informacji dotyczących ich wdrożenia.

4.4 Metoda Papazoglou

4.4.1 Service-Oriented Design and Development Methodology by Papazoglou

Service-Oriented Design and Development by Papazoglou stanowi również metodę projektowania architektury systemów informatycznych zorientowanych na usługi. Metoda opiera się ponadto na kilku podstawowych założeniach.

- Zarządzanie całym cyklem życia usługi włączając: identyfikację, projektowanie, wdrożenie, wykrycie, stosowanie oraz utrzymywanie.
- Na etapie projektowania ustanowienie platformy i modelu programowania uwzględniającego połączenia, wdrażanie i zarządzanie usługami w obrębie określonego środowiska wykonawczego.
- Stosowanie najlepszych praktyk i narzędzi dla rozwiązań architektury, które są powtarzalne, przewidywalne oraz uwzględniają częste zmiany biznesowe.
- Dostarczanie wysokiej jakości wykonywalnych rozwiązań zorientowanych na usługi, które przestrzegają wymagań QoS (Quality of Service).

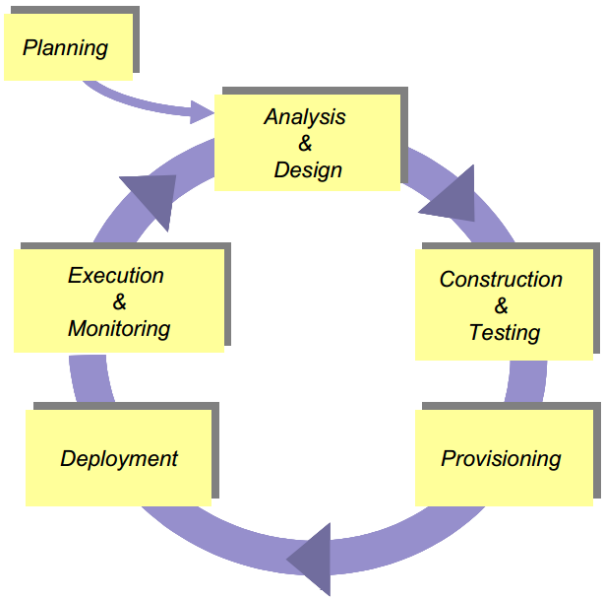
Fundamentem dla powyższych założeń jest to, że cele biznesowe i wymagania zawsze powinny prowadzić do niższego poziomu abstrakcji. Poszczególne fazy można rozpatrywać w następującej kolejności: projektowanie, implementacja, testowanie oraz transformacja procesów biznesowych w kompozytowe aplikacje. W ten sposób wymagania biznesowe mogą być śledzone przez cały cykl życia projektu - od celów biznesowych przez projektowanie oprogramowania oraz zasobów kodu, aż do kompozytowych aplikacji.

4.4.2 Fazy metody

Metoda utworzona przez Papazoglou składa się z elementów (rys. 4.2) bazujących na innych metodach takich jak: Rational Unified Process, Component-based development oraz modelowania procesów biznesowych zaproponowanego przez Paula Harmona w 2003 roku [19].

Podstawowym celem metody jest osiągnięcie jak najlepszej integracji usług oraz możliwie najwyższego poziomu ich interoperacyjności. Poszczególne fazy, które wchodzą w skład metody są wymienione poniżej.

- Faza planowania (ang. Planning phase) - określa prawdopodobieństwo wykonalności oraz naturę i zakres rozwiązań usługowych w kontekście danej organizacji. Istotne jest, żeby technologia usługowa pasowała jak najlepiej do obecnego charakteru organizacji. Kluczowe w tej fazie jest dlatego bardzo



Rysunek 4.2: Fazy metody zaproponowej przez Papazoglou [19].

dobre poznanie obecnego środowiska biznesowego. Czynności związane z fazą planowania obejmują: analizę potrzeb biznesowych jako mierzalne cele, przegląd obecnie wykorzystywanych technologii, ustalenie koncepcji wymagań dotyczących nowego środowiska i mapowanie ich na nowe lub dostępne implementacje.

- Faza analizy (ang. Analysis phase) - w trakcie tej fazy wcześniej zebrane wymagania poddawane są analizie. Postawione uprzednio cele biznesowe podlegają przeglądowi. Analitycy biznesowi tworzą kompletny model procesów typu *as-is* umożliwiając tym samym zapoznanie się z obecnie dostępnymi usługami i procesami biznesowymi. Po udostępnieniu tego modelu organizacja projektuje, symuluje i analizuje potencjalne zmiany do obecnego stanu systemu. Określa również wskaźnik rentowności ROI (Return On Investment) wynikający ze zmian w panujących procesach biznesowych. Produktem tej fazy jest model procesów biznesowych typu *to-be*, które będą realizowane przez rozwiązania SOA. Przyjmuje się, że w skład fazy wchodzą następujące czynności: identyfikacja procesów (ang. Process identification, określanie zakresu procesów (ang. Process scoping), analiza luk biznesowych (ang. Business gap analysis), oraz analiza realizacji procesów (ang. Process realization analysis).
- Faza projektowania (ang. Design phase) - projektowanie systemu zorientowanego usługowo wymaga od projektantów dostarczenia modeli i dobrze

zdefiniowanych interfejsów dla wszystkich ważnych komponentów usług. Projektowanie usług bazuje na podejściu wykonywania równolegle dwóch ścieżek: jednej do produkcji usług (ewentualnie z wcześniej utworzonych elementów) oraz drugiej związanej z asemblacją usług reużywalnych (wielokrotnego użytku). Projektowanie usług, podobnie jak analiza usług posiada swoje własne specjalne charakterystyki i techniki. Faza projektowania rozpoczyna się od rozważenia kilku istotnych problemów: zarządzanie usługami i ziarnistością komponentów (ang. Managing Service and Component Granularity), projektowanie reużywalności usług (ang. Designing for service reuse) i ich kompozytowości (ang. Designing for service composability).

- Faza specyfikacji usług (ang. Service specification phase) - składa się z trzech równie istotnych elementów: specyfikacji strukturalnej (ang. Structural specification), specyfikacji zachowań (ang. Behavioral specification) oraz specyfikacji polityk (ang. Policy specification). Podczas tej fazy interfejsy usług, które zostały wyszczególnione w fazie analizy są definiowane na podstawie kryteriów powiązań (ang. Coupling) i spójności (ang. Cohesion).
- Faza specyfikacji procesów biznesowych (ang. Business process specification phase) - dzieli się na trzy mniejsze etapy: opisywanie struktury procesów biznesowych (ang. Designing of the business process structure), opisywanie ról biznesowych (ang. Describing of business roles) oraz rozważanie zagadnień niefunkcjonalnych związanych z procesami biznesowymi (ang. Non-functional business process concerns).
- Faza konstrukcji usług (ang. Service construction phase) - obejmuje definicję interfejsów i opisy implementacji. Tworzone są usługi typu Web Service lub przekształcane istniejące.
- Faza testowania usług (ang. Testing phase) - testowanie usługi jest ogólnie scharakteryzowane jako walidacja mająca na celu sprawdzenie, że postawione wymagania zostały spełnione, a osiągnięte rezultaty są na akceptowalnym poziomie (zgodnie z obowiązującymi standardami założonymi w trakcie analizy, projektowania i implementacji).
- Faza wdrażania usług (ang. Deployment phase) - polega na publikacji interfejsu oraz definicji implementacji tej usługi. Jej zasięg obejmuje inne procesy, aplikacje oraz organizacje.
- Faza wykonania. (ang. Execution phase) - w jej zakres wchodzi upewnienie się czy wszyscy wyznaczeni uczestnicy są w stanie wykonywać daną usługę. W tej fazie „Web service'y"są już w pełni wdrożone i funkcjonalne. Podczas tego etapu usługobiorca może wynaleźć definicję usługi oraz wywołać wszystkie stowarzyszone z nią operacje.

- Faza monitorowania usług (ang. Monitoring phase) - monitorowanie poziomu usług jest zdyscyplinowanym podejściem ustalenia dopuszczalnych poziomów usług, które dotyczą celów biznesowych, procesów i kosztów. W tej fazie następuje wykonanie pomiarów, monitorowanie, raportowanie i poprawianie jakości usług i aplikacji dostarczanych przez rozwiązania zorientowane usługowo. [19]

4.4.3 Zalety i wady metody Papazoglou

Do zalet metody Papazoglou można zaliczyć przede wszystkim pokrycie pełnego cyklu wytwarzania usług (ang. Lifecycle coverage). Metoda Papazoglou doskonale wspiera rozwijanie usług już istniejących dzięki opieraniu fazy analizy na strategii *meet in the middle* oraz swojemu przyrostowemu charakterowi.

Metoda jest jednakże ciągle w fazie rozwoju i testów. Nie była wdrażana produkcyjnie. Do jej wad można zaliczyć również niepełną dokumentację. Polegając tylko na niej ciężko wykonać wdrożenie systemu zgodnie z założeniami metody Papazoglou. [21]

4.5 Metoda Thomasa Erl'a

4.5.1 Opis metody

Metoda Thomasa Erl'a stanowi pierwszą metodę SOA, która w pełni zasługiwała na komercyjne stosowanie. Stanowi przewodnik jak krok po kroku przejść dwie podstawowe fazy - analizy i projektowania. Opiera się na metodzie *top-down*.

Analiza zorientowana usługowo według Thomasa Erl'a powinna być rozważana w trzech etapach: definicji wymagań biznesowych (ang. Business definition requirements), identyfikacji w istniejących systemach elementów możliwych do automatyzacji (ang. Identify existing automation systems) oraz przygotowania modelu usług kandydujących (ang. Model candidate services).

W pierwszej fazie analizy rozpatrywane są cele i zadania organizacji oraz rozważane potencjalne zmiany do obecnych aplikacji . Sprawdzane są możliwości ich przekształcenia niezbędne do budowy systemu w oparciu o SOA. Analitycy biznesowi przygotowują model procesów biznesowych typu *as-is* przeznaczony do wglądu przez wszystkie podmioty biorące udział w wytwarzaniu usługi.

W kolejnym etapie identyfikowane są elementy procesów organizacji, które mogą zostać poddane automatyzacji. Rozpatrywany jest również utworzony wcześniej model *as-is* pod kątem utworzenia nowych lub modyfikacji istniejących procesów biznesowych. Efektem tego etapu jest przygotowanie modelu procesów biznesowych *to-be*, który będzie implementowany.

Ostatnią fazę analizy stanowi podproces modelowania usług, w którym są identyfikowane oraz uwzględniane na modelach „usługi kandydujące"(ang. Service candidates) [28]

Utworzone modele „usług kandydujących"są wykorzystywane w następnym etapie - projektu zorientowanego usługowo (ang. Service oriented design). „Usługi kandydujące"są wówczas szczegółowo specyfikowane i później realizowane jako „Web service'y".

4.5.2 Zalety i wady

Metoda projektowania SOA zaproponowana przez Thomasa Erl'a może być używana jedynie w połączeniu z innymi metodami. Dostarcza opisy dotyczące zorientowanych usługowo faz analizy i projektu, ale nie precyzuje tak istotnych elementów jak rozpoczęcie projektu SOA oraz jak przeprowadzić analizę biznesową organizacji. Ponadto metoda Thomasa Erl'a nie specyfikuje wystarczająco wykorzystania elementów obecnego systemu. Role podmiotów biorących udział w działaniu systemu nie są również spójnie określone.

Do pozytywnych cech metody można zaliczyć fakt, że wspiera najpopularniejsze standardy i technologie takie jak: BPM, WSDL, WS-BPEL oraz WS-*. Metoda jest również łatwa do zaadaptowania ze względu na dość prosty algorytm stosowania. Metoda Thomasa Erl'a opiera się również na manifeście Agile, który wspiera „zwinne tworzenie oprogramowania"[17, 21].

Rozdział 5

Opracowanie metody projektowania systemów informatycznych o architekturze SOA

5.1 Wstęp

W poprzednim rozdziale został dokonany przegląd istniejących metod projektowania systemów informatycznych o architekturze SOA.

Stopień złożoności biznesowej niektórych organizacji jest naprawdę wysoki. Bez zastosowania odpowiedniego, usystematyzowanego podejścia utworzenie sprawnie funkcjonującego systemu informatycznego stanowi istotny problem. Występuje wiele metod o różnej charakterystyce jednakże niemal każda wykazuje pewne niedociągnięcia.

Większość metod łączy opis sposobu przejścia od stanu początkowego (wyniknięcie potrzeby posiadania systemu informatycznego) do końcowego (utworzenie systemu informatycznego zgodnego z wymaganiami zebranymi w trakcie analizy). Każda z nich stanowi hierarchicznie uporządkowany zbiór elementów takich jak: fazy, aktywności, zadania oraz kroki. Opierają się również na wykorzystywaniu różnorodnych technik oraz zalecanych języków modelowania i notacji [17].

W rozdziale zostanie podjęta próba zdefiniowania nowej metody projektowania systemów informatycznych o architekturze SOA oraz porównanie jej charakterystyki z obecnie dostępnymi.

5.2 Metoda MatSOA

MatSOA stanowi autorską metodę projektowania architektury korporacyjnej. Obecne metody posiadają zarówno wiele wad jak i zalet. Różnią się podziałami na etapy, wykorzystywanymi technologiami oraz zakresem stosowalności. MatSOA stanowi połączenie najlepszych cech z już istniejących wzorców. Wprowadza także pewne własne elementy usprawniające projektowanie architektury korporacyjnej.

MatSOA w odróżnieniu od wielu występujących metod uwzględnia również analizę obecnych systemów (ang. Legacy systems). Cechę charakterystyczną dla MatSOA stanowi fakt, że do specyfikacji usług wykorzystuje język SoaML stanowiący rozszerzenie dla UML.

W celu dokładnego opisania metody zostaną przedstawione jej poszczególne fazy oraz powiązane notacje modelowania. Będą opisane również możliwe narzędzia, które wspierają proces projektowania w oparciu o MatSOA.

Metoda MatSOA składa się z sześciu głównych faz, których sekwencja została przedstawiona na rys. 5.1. Wszystkie fazy zawierają szereg aktywności, z których każda ma przypisaną rolę wykonania oraz określone artefakty wejściowe i wyjściowe. Istotne jest również, aby dostosowywać poszczególne fazy do specyfiki danego projektu.

Faza analizy organizacji oraz wykrywania usług ma równoległy przebieg do analizy obecnych systemów. Produkty wyjściowe powstałe w wyniku dwóch podejść analizy - *top-down* oraz *bottom-up* - są następnie konsolidowane. W kolejnym etapie projektowane są usługi, które z kolei są wykorzystywane do przygotowania procesów biznesowych.

W tej sekcji pracy zostaną omówione poszczególne fazy metody MatSOA.

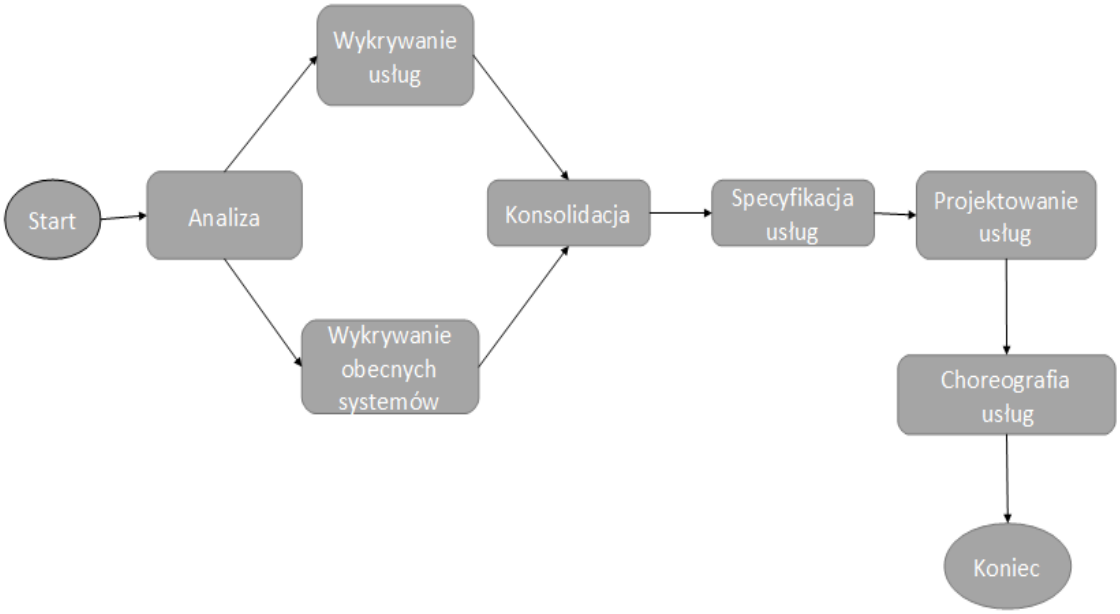
5.2.1 Analiza

W celu identyfikacji wymagań organizacji należy rozpatrzyć poszczególne aspekty z nią związane: strategie biznesowe, procesy biznesowe oraz domeny funkcjonalne. W wypadku MatSOA analiza organizacji ma przebieg kilku etapowy oraz powinna być przeprowadzana przez analityka biznesowego.

BMM - Motywacyjny Model Biznesowy

W pierwszym kroku analizy organizacji w przypadku metody MatSOA tworzony jest motywacyjny model biznesowy (BMM - Business Motivation Model). Stanowi on część opisu organizacji. Umożliwia między innymi uchwycenie i przedstawienie w prosty sposób planów biznesowych. Do podstawowych celów BMM zalicza się ponadto:

- identyfikację czynników, które powodują konieczność utworzenia planu,



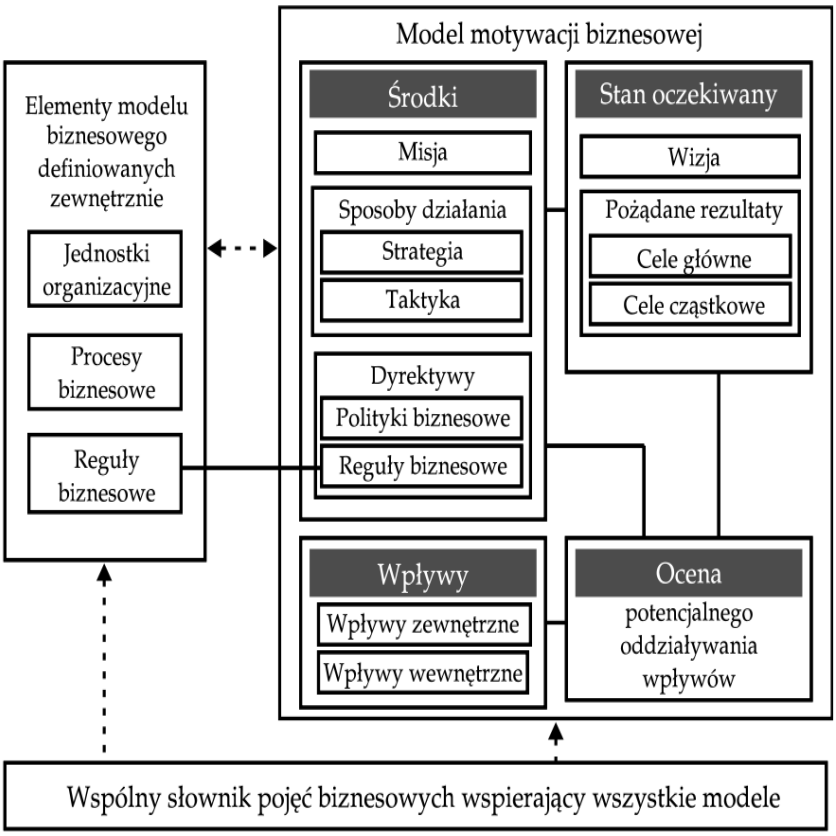
Rysunek 5.1: Fazy metody MatSOA.

- identyfikację oraz definicję elementów niezbędnych do wykorzystania w układanym planie,
- określenie, jak wymienione elementy są ze sobą powiązane.

BMM został stworzony przez BRG (Business Rules Group), a następnie w 2005 roku grupa OMG jako pierwsza poddała BMM formalnej specyfikacji. Utworzyła również schemat, na którym można opierać się przy tworzeniu BMM (rys. 5.2) [34].

Motywacyjny model biznesu stanowi element integrujący względem innych modeli (np. modeli procesów biznesowych lub modeli reguł biznesowych) i nie jest uwarunkowany (zależny) metodycznie od innych standardów. BMM stanowi pewnego rodzaju ramę koncepcyjną pozwalającą stworzyć repozytorium modeli wspólnie odwzorowujących model biznesowy danej organizacji. W BMM istotną rolę odgrywa wspólny słownik pojęć biznesowych zapewniający spójność semantyczną tworzonych modeli.

Do podstawowych elementów BMM zalicza się: stan oczekiwany (ang. Ends) - coś co dana organizacja chce osiągnąć, wizję (ang. Vision) - ogólne przedstawienie pożądanych osiągnięć, cele (ang. Goals) - wynikają z utworzonej wcześniej wizji, zadania (ang. Objectives) - wynikają z celów postawionych przez organizację, misję (ang. Mission), strategie (Strategies) i taktyki (Tactics) [34].



Rysunek 5.2: Schemat ogólny BMM utworzony przez organizację OMG [33].

W kolejnej fazie analizy organizacji na podstawie BMM są identyfikowane procesy biznesowe niezbędne do wspierania jej strategii działania. Ustalane są również kryteria (np. częstość zmian w procesach, obecny poziom możliwej integracji z istniejącymi systemami lub stopień, w którym procesy mogą być z góry określane i wydzielane) na podstawie, których wyszczególniany jest inny typ lub części procesów biznesowych, które powinny być wspierane przez systemy SOA.

Procesy biznesowe definiuje się jako ustrukturyzowany i zmierzony ciąg czynności, który jest zaprojektowany do osiągnięcia określonego wyniku [11]. Każda z czynności poddawana jest klasyfikacji zgodnie z tym czy może być wykonywana automatycznie (Business service), czy wymaga interakcji między użytkownikiem, a systemem (User interaction) oraz czy jej działanie ma charakter wyłącznie ręczny (Manual).

MatSOA zakłada identyfikowanie mierzalnych, udokumentowanych i powtarzalnych procesów. Modele procesów biznesowych powinny być budowane na różnych poziomach szczegółowości i mieć budowę hierarchiczną. Modele wysokiego poziomu określają przepływ pomiędzy poszczególnymi podmiotami (rolami), a w dalszej kolejności są dekomponowane na procesy wykonywane przez pojedyncze osoby.

Zalecaną notację modelowania stanowi BPMN (Business Process Modelling Notation). Określenie przepływu danych w procesie nie jest wymagane na tym etapie. Wykonywanie uproszczonych modeli procesów biznesowych, w których uwzględnione są jedynie aktywności ma na celu przede wszystkim utworzenie zarysu jak będzie wyglądała integracja z poszczególnymi systemami.

Równolegle do modelowania procesów biznesowych powstaje warstwowy model domen funkcjonalnych (ang. FD - Functional Domains). Proponowaną notacją dla FD są diagramy przypadków użycia UML. Każda aktywność modelu procesu biznesowego jest przypisywana do domeny funkcjonalnej. Modelowanie procesów biznesowych i FD powinno być wykonywane iteracyjnie. Połączenia aktywności procesów i domen funkcjonalnych pomaga później w zapewnieniu reużywalności usług.

Wymagania funkcjonalne

Po zidentyfikowaniu wymagań od klienta należy przeprowadzić ich analizę. Wykryte wymagania są na tym etapie uszczegóławiane oraz określane są ich priorytety oraz ryzyka. Odrzucane są również wymagania sprzeczne lub nieprawidłowe.

Analiza wymagań funkcjonalnych pozwala na zidentyfikowanie i opisanie pożądanego zachowania systemu. W przypadku rozbudowy systemów identyfikowane są nowe funkcjonalności, o które system ma zostać rozszerzony.

Analiza wymagań funkcjonalnych przedstawiana jest za pomocą diagramów przypadków użycia.

Wymagania niefunkcjonalne

W metodzie MatSOA na wymagania niefunkcjonalne składa się opis atrybutów jakościowych, jakim ma podlegać tworzony system, model domeny oraz wykrywanie dotychczasowych systemów. Zdarza się, że w niektórych metodach do wymagań niefunkcjonalnych zaliczana jest także analiza dotychczasowych systemów. W MatSOA ten etap jest zawarty w późniejszej fazie.

Model domeny przedstawia podmioty wchodzące w skład, w której działa dana organizacja lub przedsiębiorstwo. Przedstawia również relacje pomiędzy danymi podmiotami. Modelowany jest za pomocą diagramu przypadków użycia UML.

Wykrywanie obecnych systemów jest bardzo ważnym etapem analizy. Należy określić, które z systemów dotychczas występujących w organizacji będą zależne od wprowadzanych nowych funkcjonalności.

Do atrybutów jakościowych w przypadku MatSOA zalicza się przede wszystkim spełnianie podstawowych pryncypiów SOA. Są to głównie: luźne powiązania, autonomiczność, enkapsulacja oraz reużywalność usług.

5.2.2 Identyfikacja usług

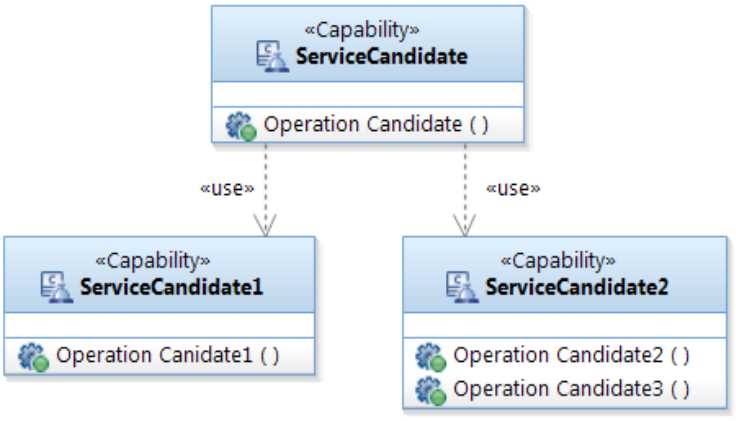
Identyfikacja usług stanowi kolejny etap metody MatSOA. W jej efekcie otrzymywany jest model kandydatów usług wykorzystywany w kolejnych fazach metody. Faza ta powinna się rozpoczynać od wykrycia kandydatów usług (ang. Service Candidates).

Owi kandydaci usług stanowią pewnego rodzaju propozycje dotyczące usług. W późniejszym etapie poddawane są specyfikacji pod warunkiem, że odpowiednia usługa nie występuje w dotychczasowym systemie.

Modele kandydatów usług są tworzone na podstawie wcześniej ustalonych wymagań funkcjonalnych w sposób zorientowany na usługi. Artefakty modelowane w fazie analizy wymagań są odpowiednio transformowane do kandydatów usług. Stanowi to wstępny etap do przygotowania modeli docelowych usług (rys. 5.3).

Do modelowania kandydatów usług wykorzystywany jest element języka SoaML - *«Capability»*. Każdy element *«Capability»* zawiera operacje. Można wykazywać zależności pomiędzy nimi za pomocą *«Usage Dependency»*. W tabeli (5.1) zostały zawarte mapowania kandydatów usług na poszczególne elementy języka SoaML.

Wykrywanie usług może mieć charakter iteracyjny. Już na tym etapie powinno się uwzględniać wymagania niefunkcjonalne związane z atrybutami jakościowymi. Należy przejrzeć czy wykryte operacje nie są wewnętrzne. Musimy również rozważyć szanse na reużywalność poszczególnych kandydatów usług.



Rysunek 5.3: Identyfikacja usług - metoda MatSOA.

Element kandydata usługi	**Element w języku SoaML**
Kandydat usługi	klasa UML ze stereotypem *«Capability*
Operacja kandydata usługi	Operacja wewnątrz elementu *«Capability»*
Zależność kandydata usługi	Usage Dependency - zależność pomiędzy poszczególnymi *«Capability»*

Tablica 5.1: Mapowanie kandydatów usług na elementy języka SoaML.

Analiza obecnych systemów i zasobów

Analiza istniejących systemów (ang. Legacy systems) oraz zasobów stanowi bardzo istotny etap metody MatSOA. Wykonywana jest równolegle do wykrywania nowych usług. Umiejętne przeprowadzenie tej fazy często może znacznie zredukować koszty wprowadzania nowych funkcjonalności w systemach organizacji. Na tym etapie identyfikowane są elementy takie jak: istniejące aplikacje, standardy branżowe, modele oraz pozostałe zasoby niezbędne do realizacji usług.

W założeniu metody MatSOA analityk systemowy powinien zidentyfikować istotne systemy. Następnie te rozpoznane systemy są analizowane pod kątem przydatnych funkcjonalności. Często się zdarza, że nie wszystkie elementy dotychczasowych systemów mogą być wykorzystane bezpośrednio. Jeśli jest to usługa SOA wprowadza się wówczas odpowiednie modyfikacje do obecnych systemów. W innych wypadkach należy rozważyć czy jest szansa wykorzystania ich poprzez opakowującą je usługę (ang. Wrapper). Sposobów na używanie istniejących zasobów jest wiele: API, dostępne bazy danych, komponenty oprogramowania, które mogą być opakowane oraz wywołanie komendy z wiersza poleceń.

Takie opakowane elementy są następnie modelowane jako kandydaci usług wraz z odpowiadającymi im operacjami. Istotne jest, żeby analityk wybierał tylko te usługi, które są powiązane z ustalonymi wymaganiami funkcjonalnymi.

Podobnie jak w przypadku etapu identyfikacji usług analizę obecnych systemów i zasobów można wykonywać w sposób iteracyjny. W każdym kroku powinno następować sprawdzanie czy utworzony model kandydatów usług spełnia wymagania niefunkcjonalne. Jeśli wykryte usługi naruszają istotnie takie zasady SOA jak „luźne powiązania"bądź „reużywalność"to wtedy rozważana jest ich modyfikacja.

5.2.3 Konsolidacja

Po wykryciu nowych usług metodą typu *top-down* i identyfikacji usług istniejących systemów typu *bottom-up* konieczna jest ich odpowiednia konsolidacja. Faza ta wykonywana jest przez architekta systemowego i może mieć charakter iteracyjny.

Operacje wywodzące się z systemów już istniejących mają pierwszeństwo. Istotne jest dlatego tutaj rozważenie czy operacje w wykrytych poszczególnych kandydatach usług na podstawie analizy wymagań funkcjonalnych nie występują w dotychczasowych systemach. W trakcie konsolidacji ponownie należy wziąć pod uwagę wymagania niefunkcjonalne i jeśli wyniknie konieczność wykonać odpowiednie modyfikacje w systemie dotychczasowym. Zmiana w obecnym systemie może zostać jedynie rozważona jeśli rozbieżność pomiędzy wymaganiami i aktualnym systemem jest zbyt duża.

W wyniku konsolidacji tworzony jest model kandydatów usług uwzględniający

rozszerzenie dotychczasowych elementów systemu o nowe funkcjonalności.

5.2.4 Specyfikacja usług

Po ustaleniu modeli kandydatów usług poszczególne usługi są specyfikowane. Określane są parametry wejściowe i wyjściowe oraz opisywane poszczególne kontrakty usługowe. Na tym etapie tworzony jest również model danych. Przebieg wywołań pomiędzy poszczególnymi interfejsami usług opisywany jest za pomocą diagramu sekwencji. Z kolei każdy interfejs usługi przedstawiany zostaje jako osobna „Linia życia"(ang. Lifeline).

5.2.5 Projektowanie usług

Po ustaleniu modelu operacji usług, poszczególne usługi mogą zostać zaprojektowane. Celem tej fazy jest na podstawie utworzonej wcześniej specyfikacji usług przygotowanie odpowiednich komponentów usług (ang. Service Components).

Do realizacji tych celów zostaną wykorzystane odpowiednie elementy języka SoaML (tab. 5.2). Istotnym elemenetem są uczestnicy usługi (*«Participant»*). Na podstawie wcześniej utworzonych modeli należy zdecydować, którzy uczestnicy będą wykorzystani i jak zostaną przyłączeni do funkcjonującego systemu. Element *«Participant»* zostanie wykorzystany do przedstawienia modelu komponentu, który będzie dostarczał (element *«Service»*) lub konsumował (element *«Request»*) poszczególne usługi.

Utworzone komponenty można łatwo integrować z pozostałymi częściami systemu.

5.2.6 Choreografia usług

Choreografia usług stanowi końcową fazę metody MatSOA. Tworzone są moduły SCA (ang. Service Component Architecture), w których opisywany jest sposób komunikacji w systemie między komponentami. Poszczególne wywołania usług poddawane są choreografii z wykorzystaniem WS-BPEL. Wywołania usług definiowane są jako aktywności, a w przypadku wystąpienia błędów wyrzucane są odpowiednie wyjątki. Jeśli zachodzi potrzeba transformacji komunikatów do wymaganej formy przez producenta lub konsumenta można utworzyć odpowiednie mediacje. Przed wywołaniem usługi komunikat zostaje przetworzony wówczas do akceptowalnej formy.

Przygotowane moduły SCA są wdrażane bezpośrednio na serwer gdzie definiowane są połączenia punktów końcowych do poszczególnych usług zewnętrznych.

Element projektu usługi	**Element w języku SoaML**
Interfejs usługi (ang. Service interface)	*«ServiceInterface»* - klasa UML ze stereotypem ServiceInterface
Dostarczana operacja (ang. Provided operaton)	Operacja wewnątrz interfejsu realizowana przez element *«ServiceInterface»*
Realizowana operacja (ang. Realized operation)	Operacja, która jest połączona z elementem *«ServiceInterface»* poprzez użycie Usage Dependency z UML
Komponent usługi (ang. Service component)	Usage Dependency - Uczestnik - komponent UML nazywany *«Participant»*
Dostarczana usługa (ang. Provided service)	Usługa - port UML ze stereotypem *«Service»*
Wymagana usługa (ang. Required service)	Zapytanie - port UML ze stereotypem *«Request»*
Wewnętrzne zachowanie (ang. Internal behaviour)	Aktywność UML, która jest dodawana jako *«OwnedBehaviour»* do elementu *«Participant»*
Port	Element *«Port»* wykorzystywany przy tworzeniu dostarczanej usługi
Kanał usług	Element *«ServiceChannel»* wykorzystywany przy tworzeniu połączeń pomiędzy konsumentami i producentami usług

Tablica 5.2: Projektowanie usług w odwzorowaniu na poszczególne elementy języka SoaML.

5.2.7 Końcowy produkt

Celem zastosowania metody MatSOA jest doprowadzenie do utworzenia systemu informatycznego opierającego swoje działanie na kooperacji usług. System utworzony zgodnie z paradygmatami MatSOA powinien charakteryzować się dużą ziarnistością i wysokim poziomem skalowalności.

Istotne jest, aby w trakcie identyfikacji oraz specyfikacji usług konfrontować modele z wymaganiami niefunkcjonalnymi z wyszczególnionymi atrybutami jakościowymi.

Rozdział 6

Weryfikacja koncepcji na przykładzie systemu bankowego

6.1 Wstęp

Systemy bankowe często charakteryzują się dużą złożonością i panuje w nich wiele różnorodnych procesów biznesowych. Odpowiednio zaprojektowany system jest więc niezbędny we współczesnej bankowości.

W niniejszym rozdziale metoda MatSOA zostanie poddana weryfikacji pod względem przydatności do tworzenia tego typu systemów. Systemy bankowe to dziedzina bardzo szeroka dlatego zostanie przedstawiony projekt tylko niewielkiej części wyobrażonego systemu. Niemniej jednak na tym przykładzie zostaną zaprezentowane wszystkie fazy autorskiej metody.

W oparciu o zasady Model Driven Development (MDD) zostanie również przedstawiony proces implementacji zgodnie z utworzonymi modelami. Wdrożenie systemu wymaga również istotnych rozważeń związanych z konfiguracjami środowiska, na którym będzie funkcjonować. Zostaną więc też przedstawione szczegóły techniczne dotyczące budowania systemu oraz jego instalacji na serwerach.

Na zakończenie rozdziału zostaną opisane również testy jakim zostanie poddany system po utworzeniu.

6.2 Projektowanie systemu bankowego w oparciu o utworzoną koncepcję

Przykład wykorzystany w pracy będzie oparty na wymyślonej przez autora koncepcji organizacji. Zostaną przedstawione i poddane rozważaniom jednakże tylko wybrane modele mające na celu jak najlepszą prezentację poszczególnych faz me-

tody MatSOA.

Modele w rozdziale zostały przygotowane z wykorzystaniem narzędzi Rational Software Architect w wersji 8.5 wraz z doinstalowanym dodatkiem do obsługi języka SoaML oraz WebSphere Integration Developer w wersji 7.0, w którym zdefiniowano połączenia pomiędzy komponentami i zaimplenetowano całość systemu.

6.2.1 Opis sytuacji biznesowej

FrostRes stanowi niewielki bank krajowy. Ze względu na krótki pobyt na rynku pełni tylko podstawową liczbę usług związanych przede wszystkim z utrzymywaniem kont klientów i wykonywaniem podstawowych operacji finansowych.

Bank posiada obecnie system informatyczny, jednakże nie jest on wystarczający na znaczną ilość klientów, która stale się zwiększa. Wymaga także modernizacji związanej z wprowadzonymi ustawami rządowymi.

W obecnej chwili system umożliwia zarządzanie kontami klientów (otwarcie i zamknięcie konta, pobranie informacji o koncie, pobranie historii operacji wykonywanych na koncie) oraz obsługę transakcji finansowych (przelewy krajowe oraz wpłata i wypłata pieniędzy).

Zarząd banku prognozuje wykonanie szeregu zmian w istniejącym systemie. Ze względu na fakt, że organizacja dopiero się rozwija i ma ograniczony budżet zdecydowano się jak na razie na wprowadzenie tylko jednej nowej usługi - przelewów międzynarodowych. Podstawowe wymaganie stanowi fakt, żeby kraje uczestniczące w transakcjach przelewów posiadały rachunki walutowe spełniające normy IBAN (International Bank Account Number). Oprócz tego ustalono, że system po modernizacji ma być w pełni zgodny z pryncypiami SOA.

Inna istotna zmiana związana jest z nową ustawą wprowadzoną przez rząd, która wprowadziła konieczność weryfikacji każdego przelewu powyżej 10000 złotych przez usługę udostępnioną przez Narodowy Bank Polski. W związku z tym konieczna będzie integracja z usługą weryfikacji przelewów ze strony Narodowego Banku Polskiego. W systemie wykorzystywane są dane o wysokim stopniu poufności dlatego wszelkie integracje z systemami zewnętrznymi muszą ścisło podlegać standardom bezpieczeństwa wymiany komunikatów w architekturze usługowej.

Ze względu na fakt, że w przyszłości planowane są kolejne modernizacje oraz rozszerzenia, system musi wykazywać się stosunkowo dużą skalowalnością. Istotne jest również, żeby stworzony system charakteryzował się wysokim poziomem ziarnistości oraz zapewniał jak największą reużywalność usług.

6.2.2 Analiza

Analiza zostanie przedstawiona w podziale na poszczególne etapy zgodne z metodą MatSOA. Rozpocznie się od opisania BMM, na podstawie którego zostanie

utworzony uproszczony model procesu biznesowego. W kolejnych krokach zostaną przedstawione rozważania dotyczące wymagań funkcjonalnych i niefunkcjonalnych.

Przygotowanie Business Motivation Model (BMM)

Budowa systemu dla FrostRes została rozpoczęta od przygotowania motywacyjnego modelu biznesu. Tworzenie BMM jest krokiem opcjonalnym metody MatSOA jednakże stanowi bardzo dobry wstęp dla analizy. W zależności od stopnia złożoności sytuacji biznesowej może wykazywać się różnym poziomem szczegółowości. Modernizacja banku FrostRes nie stanowi zbyt złożonego problemu biznesowego, dlatego tworzony BMM będzie zawierał tylko najbardziej podstawowe elementy.

Przygotowanie BMM przedstawionego na rys. 6.1 rozpoczęto od rozważenia środków (element *«Means»*). Następnie określono misję (element *«Mission»*), ustalono strategie (elementy *«Strategy»*) oraz ułożono odpowiednie taktyki (elementy *«Tactic»*) wdrażania biznesu. Na podstawie analizy powyższych elementów wyodrębniono proces biznesowy, który będzie realizował „Obsługę przelewu międzynarodowego"(element *«Business Process»*).

W BMM opisano również stan oczekiwany (element *«End»*) przez bank FrostRes. Dekompozycję rozpoczęto tym razem od określenia wizji (element *«Vision»*) powiązanej bezpośrednio z utworzonym wcześniej opisem misji. Następnie przedstawiono cele główne (element *«goal»*) oraz cząstkowe (element *«Objective»*).

W następnym kroku wykryty wcześniej proces „Obsługa przelewu międzynarodowego"(rys. 6.2) opisano z wykorzystaniem języka BPMN. Opiera się na usługach poddanych pewnego rodzaju „choreografii". Ma on charakter jedynie poglądowy. Na tym etapie nie jest zdefiniowany jeszcze przepływ danych pomiędzy poszczególnymi aktywnościami procesu. Tworzenie uproszczonej formy modelu procesu w początkowej fazie cyklu życia projektu SOA ma na celu jak najwcześniejsze wykrycie istotnych problemów związanych z logiką przebiegu wywoływania usług pomiędzy poszczególnymi podmiotami. Pozwala to także na ewentualne dostosowanie uczestnikom integracji wystawianych interfejsów. Poniżej został opisany przebieg procesu wraz z przepływem pomiędzy uczestniczącymi rolami.

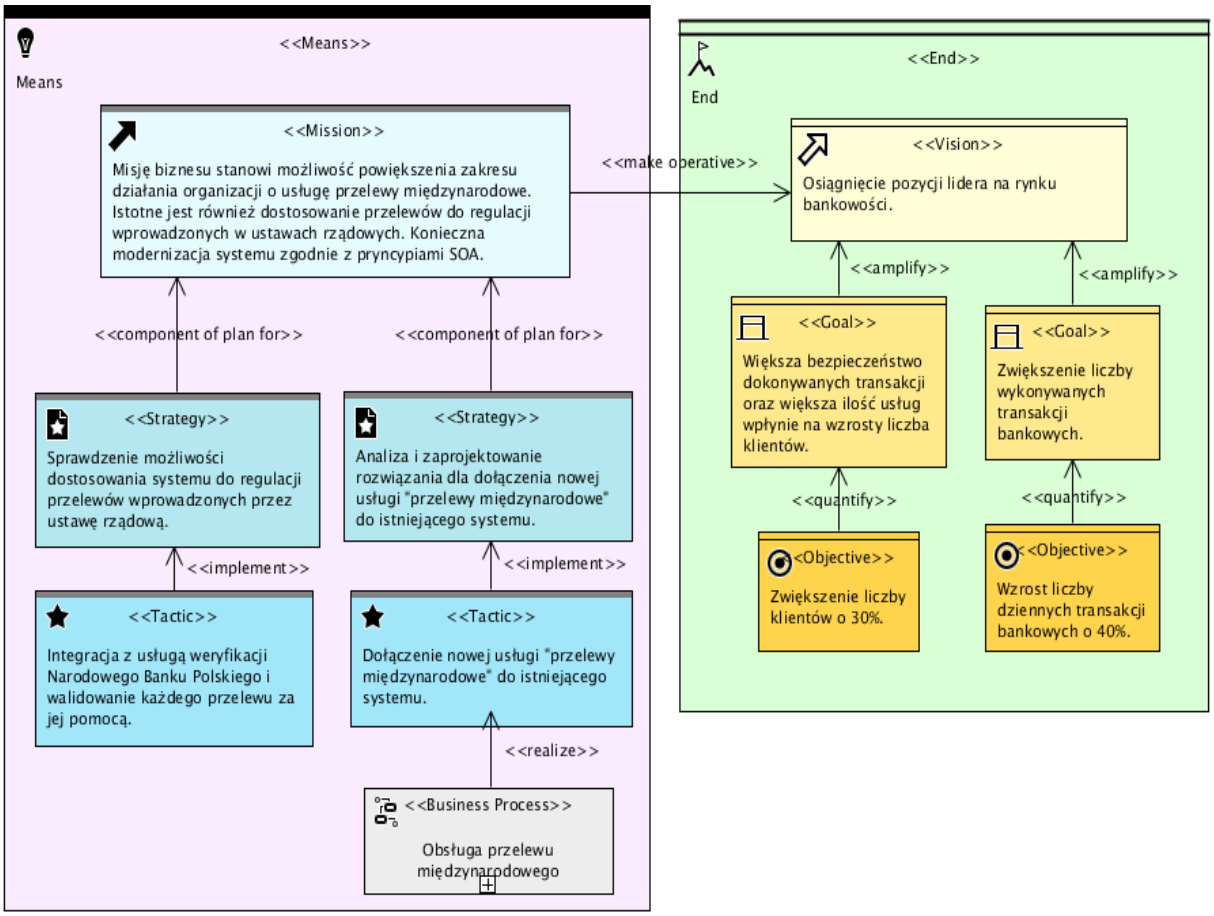
1. Klient tworzy zlecenie przelewu międzynarodowego.
2. Klient wysyła zlecenie przelewu międzynarodowego.
3. SZOB rejestruje zlecenie przelewu międzynarodowego.
4. SZOB weryfikuje zlecenie przelewu międzynarodowego.

[dane przelewu są poprawne; przelew międzynarodowy na kwotę wyższą niż 10000 złotych]

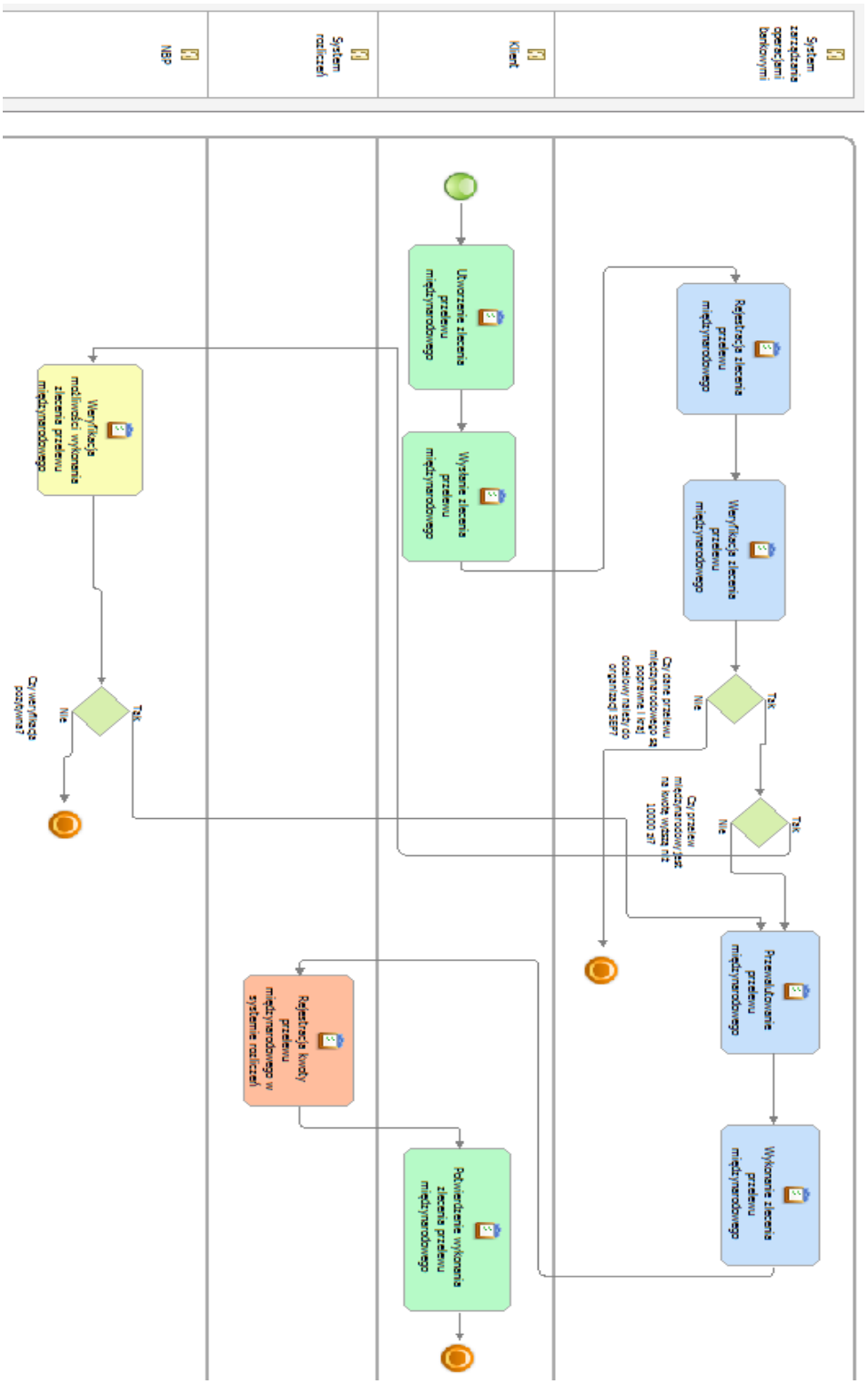
5.1 NBP weryfikuje możliwość wykonania przelewu.

[pozytywna weryfikacja przelewu]

6.1.1 SZOB przewalutowuje przelew międzynarodowy.



Rysunek 6.1: Model BMM dla modernizacji banku FrostRes.



Rysunek 6.2: Proces obsługi przelewu międzynarodowego - diagram BPMN.

7.1.1 SZOB wykonuje zlecenie przelewu międzynarodowego.
8.1.1 SR rejestruje kwotę przelewu międzynarodowego w systemie rozliczeń.
9.1.1 System wyświetla potwierdzenie wykonania przelewu klientowi.
[dane przelewu są poprawne; przelew na kwotę niższą niż 10000 złotych]
5.2. SZOB przewalutowuje przelew międzynarodowy.
6.2. SZOB wykonuje zlecenie przelewu międzynarodowego.
7.2. SR rejestruje kwotę przelewu międzynarodowego w systemie rozliczeń.
8.2. System wyświetla potwierdzenie wykonania przelewu klientowi.
[dane przelewu nie są poprawne]
5.3. System wyświetla informację o niepozytywnej weryfikacji danych przelewu.
6.3. Zakończenie obsługi przelewu międzynarodowego.
[dane przelewu są poprawne; przelew międzynarodowy na kwotę wyższą niż 10000 złotych]
5.4. NBP Weryfikuje możliwość wykonania przelewu.
[niepozytywna weryfikacja przelewu]
6.5.1 System wyświetla informację o niepozytywnej weryfikacji danych przelewu.
6.5.2 Zakończenie obsługi przelewu międzynarodowego.

Po utworzeniu i opisaniu procesu biznesowego można przystąpić do jego dekompozycji.
1. Utworzenie zlecenia przelewu międzynarodowego.
2. Wysłanie zlecenia przelewu do banku.
3. Rejestracja zlecenia przelewu.
4. Weryfikacja zlecenia przelewu międzynarodowego.
5. Weryfikacja możliwości wykonania przelewu międzynarodowego.
6. Przewalutowanie przelewu międzynarodowego.
7. Wykonanie przelewu międzynarodowego.
8. Rejestracja kwoty przelewu międzynarodowego w systemie rozliczeń.

Wymagania funkcjonalne

Ze względu na to, że modyfikacje systemu dla banku FrostRes są niewielkie opis wymagań funkcjonalnych nie będzie złożony.

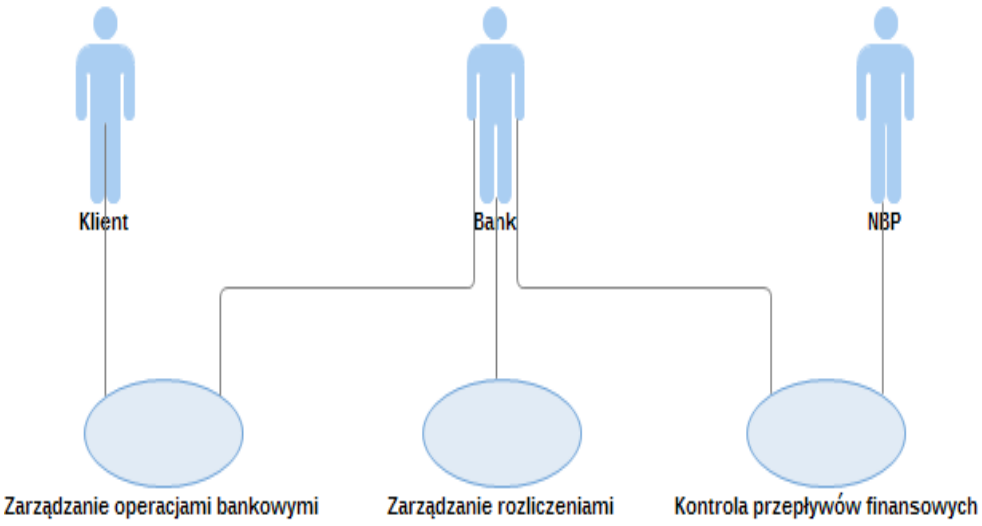
Podstawowym wymaganiem funkcjonalnym, które wchodzi w skład modyfikacji dotychczasowego systemu jest obsługa przelewów międzynarodowych. Zlecenie przelewu międzynarodowego powinno być tworzone z poziomu formularza internetowego poprzez pracownika banku.

Jeśli przelew wykona się poprawnie powinien być zwrócony odpowiedni komunikat świadczący o powodzeniu transakcji. W Przypadku błędów system powinien w odpowiedzi uwzględniać informację o błędach.

Wymagania niefunkcjonalne

Na wymagania niefunkcjonalne składają się: model domeny, wykrycie obecnych systemów oraz atrybuty jakościowe.

Model domeny został podzielony na poszczególnych aktorów oraz odpowiadające im obszary funkcjonalne. Tak jak przedstawia model z rys. 6.3 klient i bank będą przypisani do „Zarządzania operacjami bankowymi". Wynika to z faktu, że zarówno klient jak i bank będą mieli możliwość uruchomienia zlecenia przelewu międzynarodowego. Bank ponadto został przypisany do obszaru „Zarządzanie rozliczeniami". Będzie miał możliwość rejestracji faktu wykonania transakcji przelewu w systemie. Bank i NBP ponadto będą miały możliwość „Kontroli przepływów finansowych", która będzie polegać na wstępnej weryfikacji zlecenia przelewu ze strony banku, a później kolejnej w wykonaniu NBP.



Rysunek 6.3: Model domeny funkcjonalnej dla banku FrostRes.

Dotychczas w banku znajdowało się kilka podsystemów wchodzących w skład jednego większego systemu: zarządzania kontami klientów, obsługa transakcji finansowych oraz zarządzanie rozliczeniami. W przypadku banku FrostRes istotne dla wprowadzonych modyfikacji są: zarządzanie operacjami bankowymi oraz zarządzanie rozliczeniami. Wszystkie podsystemy są zgodne z pryncypiami SOA,

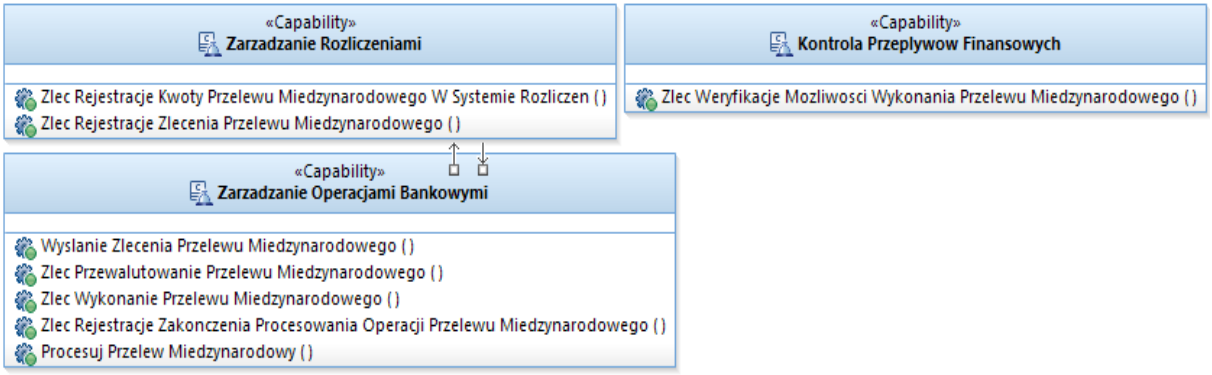
więc nie będzie konieczne tworzenie usług obudowujących. W późniejszym etapie metody MatSOA zostaną wykryte usługi obecnych systemów.

Do atrybutów jakościowych zalicza się przede wszystkim podstawowe zasady SOA. Stosowanie się do nich w przypadku banku FrostRes jest niebywale ważne, ponieważ w systemie planowane są częste zmiany. Musi charakteryzować się dlatego jak największą ziarnistością, reużywalnością oraz powinien być w jak najwyższym stopniu skalowalny.

6.2.3 Wykrywanie usług

W kolejnym etapie analizy wykonywane jest wykrywanie usług. Identyfikując usługi rozpatrujemy domeny funkcjonalne oraz zależności pomiędzy nimi kierując się utworzonym wcześniej, uproszczonym modelem procesu biznesowego.

W pierwszym kroku wykrywania usług tworzymy wstępny model kandydatów usług jak na rys. 6.4.

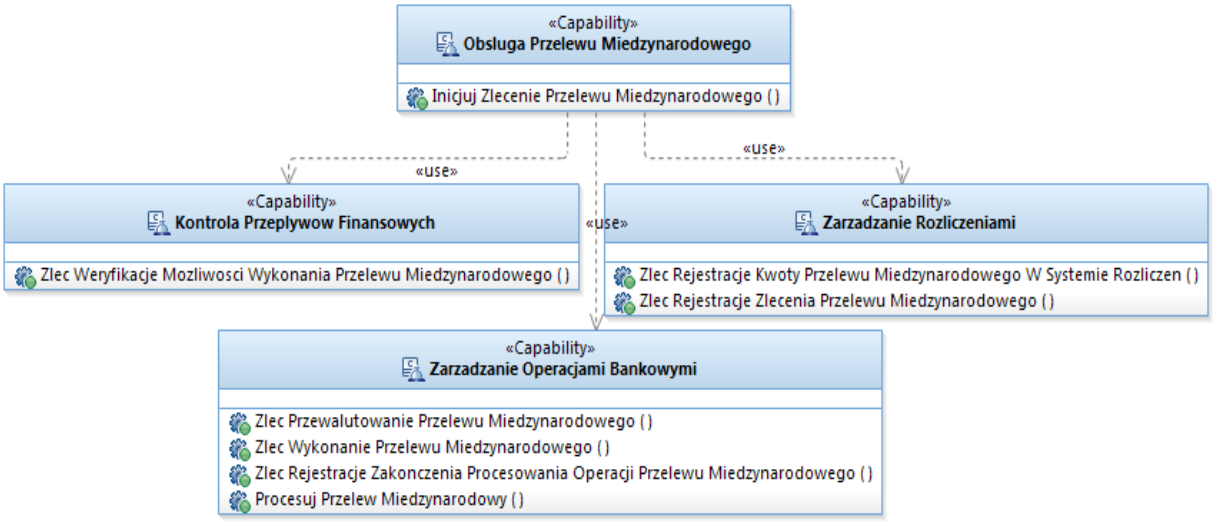


Rysunek 6.4: Wykrywanie usług - wersja inicjalna

Na powyższym diagramie kandydatów usług można zauważyć, że do poszczególnych domen (Zarządzanie Rozliczeniami, Kontrola Przepływów Finansowych, Zarządzanie Operacjami Bankowymi) zostały przypisane odpowiednie operacje.

Inicjalny diagram kandydatów należy rozważyć pod kątem refaktoryzacji w celu spełniania wymagań niefunkcjonalnych. Na diagramie można zauważyć, że operacja *Wysłanie Zlecenia Przelewu Miedzynarodowego* jest nadmiarowa (jest wewnętrzna i nie ma wpływu na przepływ pomiędzy poszczególnymi kandydatami usług). Ponadto na diagramie nie ma metody, która odpowiadałaby za uruchomienie obsługi przelewu międzynarodowego.

Na kolejnym diagramie (rys. 6.5) została dokonana refaktoryzacja inicjalnego modelu kandydatów usług.



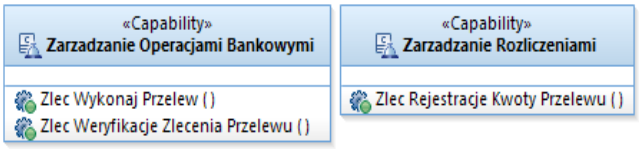
Rysunek 6.5: Wykrywanie usług - wersja po refaktoryzacji

Została usunięta metoda nadmiarowa *Wysłanie Zlecenia Przelewu Miedzynarodowego* oraz utworzony oddzielne Capability *Obsluga Przelewow Miedzynarodowych* z metodą *Inicjuj Zlecenie Przelewu Miedzynarodowego*.

6.2.4 Wykrywanie usług obecnych systemów

Na etapie analizy zostały wykryte dotychczasowe systemy, które występują w banku FrostRes: zarządzanie operacjami bankowymi oraz zarządzanie rozliczeniami. System banku dotychczas spełniał operacje związane z wykonywaniem przelewów krajowych oraz wpłatami i wypłatami pieniędzy z rachunka klienta.

Na diagramie (rys. 6.6) zostały przedstawione metody usług obecnych systemów już po wykonyanej refaktoryzacji.

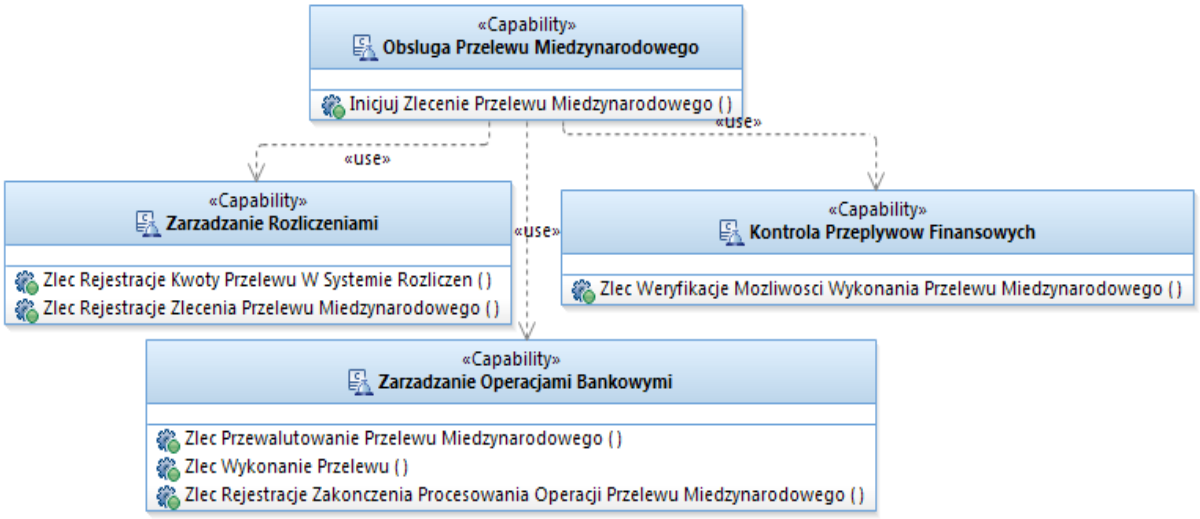


Rysunek 6.6: Wykrywanie usług obecnych systemów - wersja po refaktoryzacji

6.2.5 Konsolidacja usług

W poprzednich etapach wykryte usługi poddano konsolidacji (rys. 6.7). Zauważono, że z usług systemów dotychczas występujących można uogólnić niektóre

metody związane z przelewem międzynarodowym. W tym wypadku metodę *Zlec Wykonanie Przelewu Miedzynarodowego* można zastąpić operację *Zlec Wykonanie Przelewu*. Podobnie operację *Zlec Rejestracje Kwoty Przelewu Miedzynarodowego* zastąpiono metodą *Zlec Rejestracje Kwoty Przelewu*.



Rysunek 6.7: Konsolidacja usług

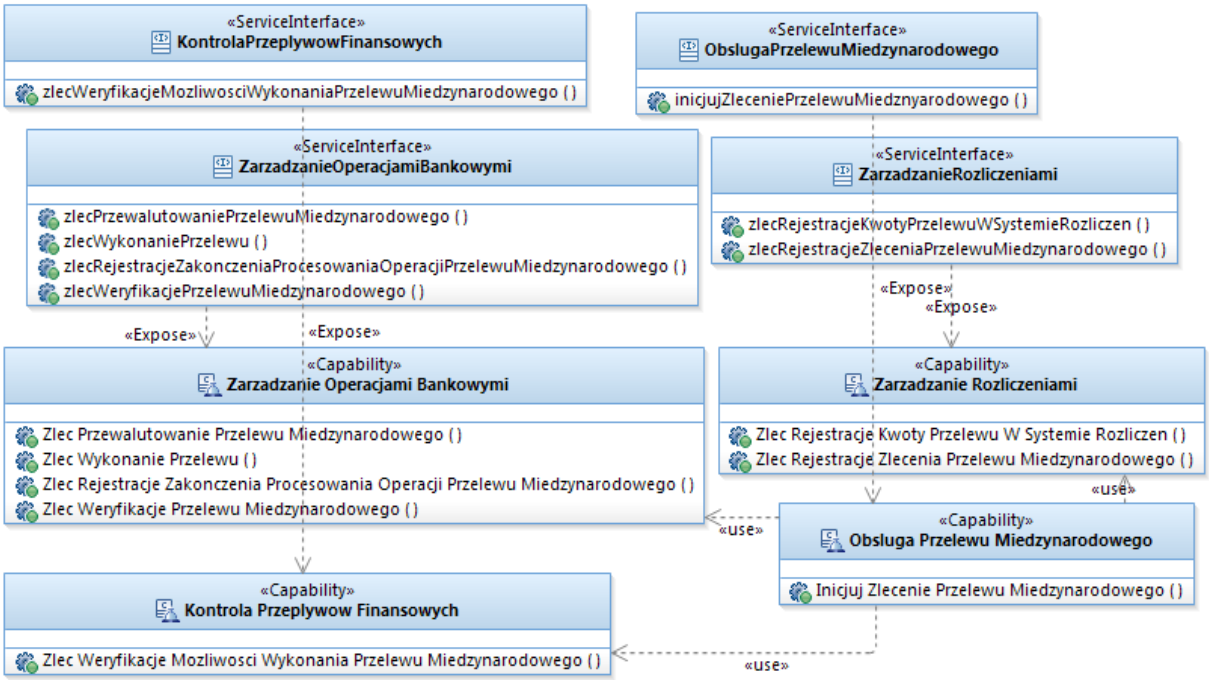
6.2.6 Specyfikacja usług

Przetworzone usługi w wyniku konsolidacji zgodnie z metodą MatSOA poddano w kolejnym etapie specyfikacji. W tej fazie przygotowano model danych oraz ustalono argumenty wejściowe i wyjściowe dla poszczególnych operacji interfejsów (rys. 6.8).

Wszystkie elementy *«Capability»* zostały wykorzystane do utworzenia elementów *«ServiceInterface»* przy pomocy połączenia typu *«Expose»*.

W przypadku specyfikacji usług dla banku FrostRes przyporządkowano do każdego elementu *«Capability»* jeden element *«ServiceInterface»*. Stosowanie tej praktyki nie jest zawsze konieczne. Często zdarza się możliwość przypisania więcej niż jednego elementu *«Capability»* do jednego elementu *«ServiceInterface»*.

W celu bardziej szczegółowego wyspecyfikowania usług banku FrostRes przygotowano również opisy kontraktów usług oraz modele sekwencji pomiędzy poszczególnymi podmiotami w procesie.



Rysunek 6.8: Specyfikacja usług

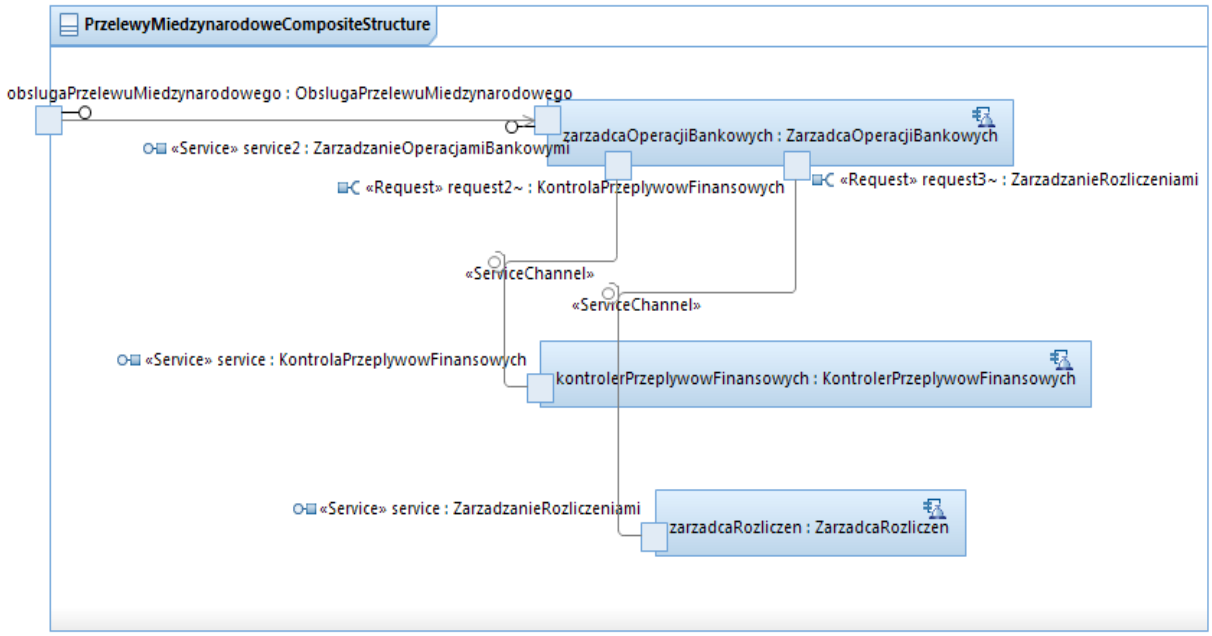
6.2.7 Projektowanie usług

Projektowanie usług rozpoczęto od utworzenia diagramu uczestników (ang. Participants diagram) biorących udział w usługach. Zaznaczono na nim interfejsy dostarczane oraz wywoływane. Wyodrębniono takich uczestników jak: *ZarzadcaOperacjiBankowych*, *KontroloerPrzeplywowFinansowych*, *ZarzadcaRozliczen*.

Na podstawie diagramu uczestników oraz specyfikacji usług przygotowano diagram komponentów dla przelewów międzynarodowych (rys. 6.9). Komponent posiada jeden interfejs zewnętrzny *ObslugaPrzelewuMiedzynarodowego*. Poszczególne interfejsy uczestników usługi (zarówno wywoływane jak i dostarczane) zostały połączone przy użyciu elementu *«ServiceChannel»*.

6.2.8 Choreografia usług

Choreografia usług w formie procesu biznesowego BPEL została utworzona w narzędziu WebSphere Integration Developer. Przygotowanie procesu rozpoczęto od utworzenia diagramu asemblacji gdzie zaznaczone są usługi importowane do procesu (ang. Inbound) oraz eksportowane (ang. Outbound).



Rysunek 6.9: Diagram komponentów dla usługi przelew międzynarodowy

6.3 Proces wdrożenia

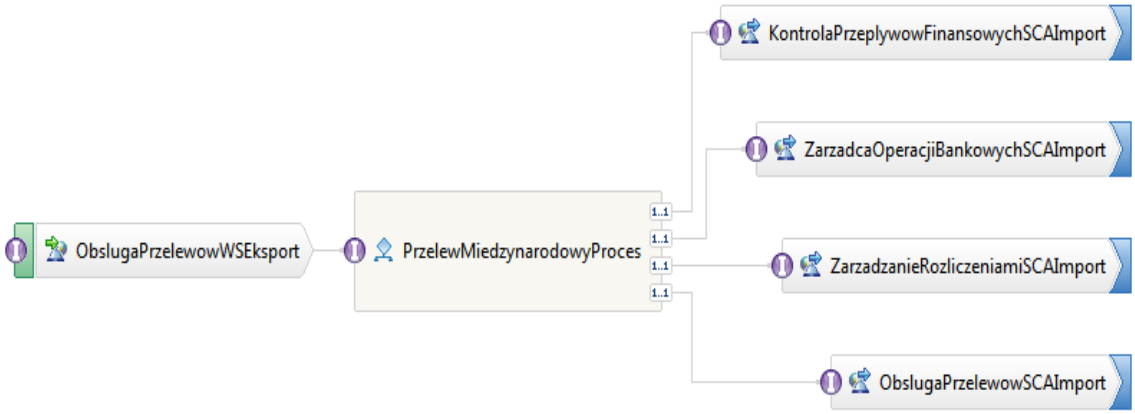
Utworzone modele usług zostały poddane transformacie UML do SOA w narzędziu Rational Software Architect. Wygenerowane zostały opisy poszczególnych usług przy wykorzystaniu WSDL oraz zdefiniowano obiekty danych za pomocą XSD (ang. XML Schema Definition).

Wygenerowane elementy zostały następnie zaimportowane do narzędzia WebSphere Integration Developer. Utworzono w nim projekt typu „Rozwiązanie integracji". W projekcie na diagramie asemblacji (ang. Assembly diagram) przedstawiono komunikację pomiędzy poszczególnymi komponentami (6.10).

6.4 Testy

W celu weryfikacji poprawnego funkcjonowania, system dla banku FrostRes został poddany testom obejmujących kilka istotnych obszarów.

Testy funkcjonalne rozpoczęto od sprawdzenia poszczególnych systemów czy udzielają pożądanych odpowiedzi. Owa weryfikacja opierało się na wykonaniu bezpośrednich zapytań SOAP do każdego z systemów przy wykorzystaniu narzędzia SoapUI. Po rozważeniu otrzymanych wyników stwierdzono, że systemy zewnętrzne udzielały oczekiwanych odpowiedzi.



Rysunek 6.10: Diagram asemblacji dla przelewu międzynarodowego

Po integracji z systemami zewnętrznymi wykonano test polegający na tworzeniu różnych zapytań (zmianie ulegały dane wejściowe) do usługi wystawianej przez system banku FrostRes związanej z przelewami międzynarodowymi. Stwierdzono, że wszystkie odpowiedzi są zgodne z założeniami.

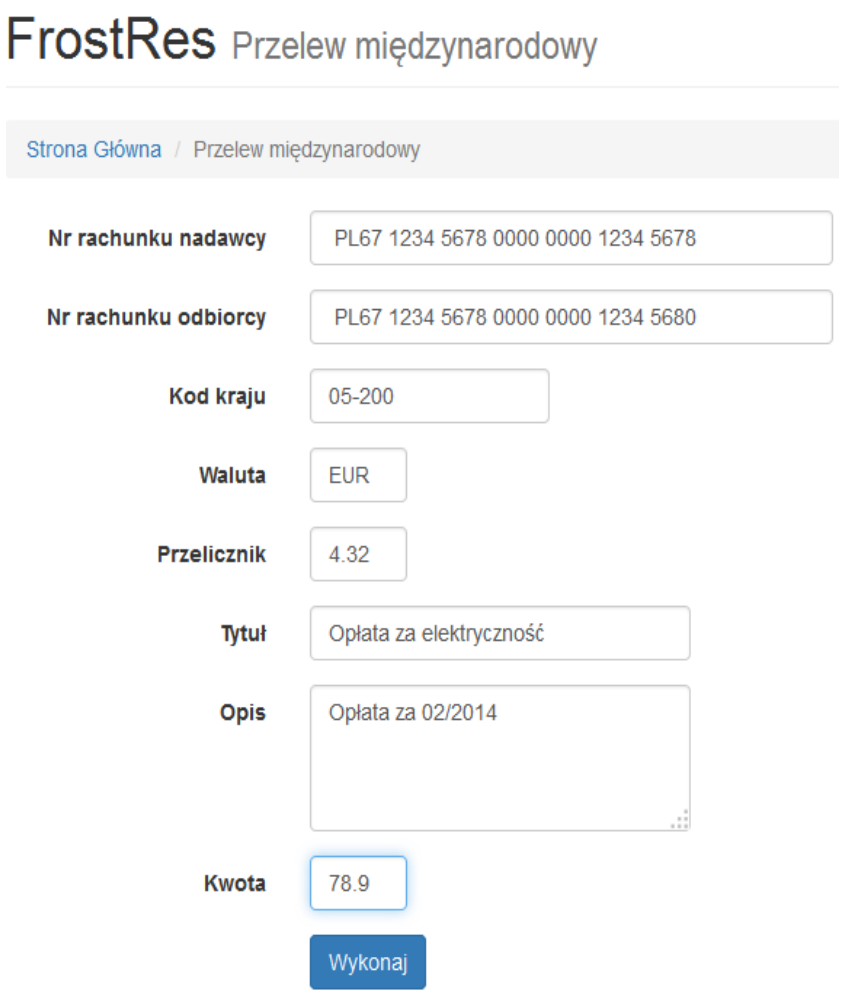
Nastąpiła również próba testów z poziomu interfejsu użytkownika. Wypełniano formularz i sprawdzano czy interakcja ze strony systemu będzie poprawna (rys. 6.11).

Testy niefunkcjonalne, którym został poddany system obejmowały głównie obszar związany z wydajnością systemu. Do przeprowadzenia testów wydajnościowych zastosowano narzędzie LoadUI.

Wyniki przeprowadzonych testów umieszczono w tab. 6.1. Poszczególne czasy odpowiedzi na zapytania związane ze zleceniem przelewu międzynarodowego okazały się bardzo krótkie (średni czas odpowiedzi wyniósł 424,22). Ponadto w ciągu 10 sekund bank jest w stanie obsłużyć aż blisko 63 przelewy międzynarodowe. Stanowi to o wysokim poziomie wydajności systemu. Należy wziąć jednak pod uwagę, że uruchomiony system integrował się z aplikacjami zaślepiającymi docelowe systemy. W przypadku integracji z prawdziwymi systemami różnice w czasach odpowiedzi mogłyby wynikać z faktu obsługi komponentów przez różne architektury fizyczne oraz opóźnień w połączeniach sieciowych.

Min. czas odp. [ms]	**Max czas odp. [ms]**	**Śr. czas odp. [ms]**	**Ilość odp. w ciągu 10 s.**
215	998	424,22	63

Tablica 6.1: Statystyki wydajnościowe systemu dla banku FrostRes.



Rysunek 6.11: Testy z poziomu interfejsu - formularz przelewu międzynarodowego

6.5 Cel systemu w odniesieniu do zaprojektowanej koncepcji

Zaprojektowana koncepcja umożliwiła utworzenie systemu zgodnego z paradygmatami SOA przy wykorzystaniu języka SoaML. Przykładowy system posiadał niezbyt skomplikowane wymagania funkcjonalne jednakże metodę MatSOA równie dobrze można by przełożyć na bardziej złożone systemy.

Cel systemu został w pełni osiągnięty. Przede wszystkim charakteryzuje się bardzo wysokim stopniem skalowalności, ponieważ można go łatwo rozbudowywać poprzez dołączanie nowych komponentów. Ponadto pomiędzy poszczególnymi elementami systemu występują luźne powiązania. Każdy z komponentów może działać niezależnie od pozostałych.

Inne zasady SOA takie jak reużywalność, interoperacyjność czy kontraktowość również zostały spełnione. Każdy komponent systemu może być wykorzystany w kontekście innego systemu. Zasada interoperacyjności została uwzględniona, ponieważ system banku FrostRes może z powodzeniem komunikować się z pozostałymi systemami (jak na przykład opisywany NBP). Ponadto pomiędzy usługami, a konsumentami widać wyraźnie zdefiniowane kontrakty (kontraktowość usług).

Rozdział 7

Podsumowanie i wnioski

W poprzednich rozdziałach dokonano próby zdefiniowania „architektury korporacyjnej"oraz omówiono najpopularniejsze związane z nią metody. Przedstawiono również autorską koncepcję prowadzącą do utworzenia architektury korporacyjnej w organizacji.

Rozdział ten stanowi podsumowanie wcześniejszych badań. Autorska metoda zostanie porównana do innych, obecnie występujących rozwiązań. Nastąpi również próba oceny przydatności oraz możliwości wykorzystania metody w realnych projektach systemów informatycznych.

Koniec rozdziału będzie obejmował rozważania dotyczące przyszłości tworzenia systemów informatycznych w oparciu o paradygmaty SOA.

7.1 MatSOA na tle innych metod projektowania architektury korporacyjnej

W rozdziale czwartym dokonano przeglądu kilku istniejących metod projektowania architektury korporacyjnej. Obecnie występuje o wiele więcej, jednakże powszechnie wykorzystywana w praktyce jest tylko ich niewielka część.

W tabeli (tab. 7.1) przedstawiono zestawienie najpopularniejszych metod projektowania architektury korporacyjnej. Wyszczególniono również parametry i cechy charakterystyczne, na podstawie których odbędzie się porównanie. Autor pracy dokonuje subiektywnej oceny, w jakim stopniu każda z metod spełnia poszczególne atrybuty.

Elementy, które wzięto pod uwagę w zestawieniu:

- Strategia dostarczania (ang. Delivery strategy) - występują trzy spotykane strategie dostarczania architektury korporacyjnej. Zależą one od stopnia złożoności domeny biznesowej oraz poziomu oparcia się na istniejących elemen-

tach organizacji. Strategia *top-down* oznacza tworzenie całości architektury korporacyjnej od początku. *Bottom-up* oznacza z kolei wykorzystywanie istniejących elementów organizacji do utworzenia architektury korporacyjnej. *Meet-in-the-middle* stanowi natomiast konsolidację obydwu podejść.

- Wykorzystanie w komercyjnych projektach - istotne jest potwierdzenie przydatności metody w praktyce. W oparciu o opinię przypadków (ang. Proof-of-concept) metoda może być również udoskonalana i poprawiana. Wiele przykładów zastosowań metody zakończonych sukcesem świadczy o jej wysokim poziomie dojrzałości.
- Stopień oparcia się na Agile - wiele istniejących metod wymaga ścisłego przestrzegania narzucanych przez nie schematów. Nie jest to odpowiednie podejście, ponieważ zawsze istnieje ryzyko wprowadzania rozległych zmian. Parametr ten określa poziom, w jakim metodyka jest w stanie przestrzegać zasad manifestu Agile.
- Wykorzystywane języki i techniki - metody korzystają z różnorodnych języków oraz technik w celu jak najlepszego przedstawienia zamysłów projektantów architektury korporacyjnej.
- Pokrycie typowych faz cyklu życia projektu - ta cecha oznacza, w jakim stopniu metoda pokrywa wszystkie podstawowe fazy życia projektu: planowanie, analizę i projektowanie, implementację, testowanie oraz wdrożenie.
- Prawnie zastrzeżony - nie wszystkie metody są powszechnie dostępne do użytkowania. Niektóre organizacje nie udostępniają wszystkich istotnych szczegółów specyfikacji, a niektóre udostępniają je tylko odpłatnie.
- Jakość specyfikacji - wiele z metod posiada niedokładne opisy, co znacznie utrudnia ich wdrażanie w poszczególnych projektach. W skali rosnącej 1-5 oceniono jakość oficjalnych specyfikacji poszczególnych metodyk.
- Poziom skalowalności, elastyczności oraz rozszerzalności - zbiór cech takich jak: skalowalność, elastyczność oraz rozszerzalność wskazuje, w jakim stopniu metoda jest w stanie dostosować się do różnego rodzaju sytuacji projektowych. Skalowalność wskazuje na zdolność metody do przystosowywania się do różnych rozmiarów aplikacji lub punktów krytycznych. Elastyczność należy postrzegać jako poziom możliwości konfiguracji metody w trakcie jej używania. Z kolei poziom rozszerzalności oznacza, jak duże są możliwości danej metody w przypadku rozbudowy o nowe elementy lub fazy.

Na podstawie tabeli 7.1 można zauważyć, że większość omawianych metod opiera się na strategii dostarczania typu *Meet-in-the-middle*. Jedynie metoda Thomasa Erl'a wykorzystuje podejście *Top-down*.

	MatSOA	**RUP4SOA**	**Papazoglou**	**T. Erl**	**SOMA**
Strategia dostarczania	Meet-in-the-middle	Meet-in-the-middle	Meet-in-the-middle	Top-down	Meet-in-the-middle
Wykorzystywano w komercyjnych projektach	N	T	N	T	T
Stopień oparcia się na Agile (skala 1-5 rosnąco)	5	4	3	3	4
Wykorzystywane języki i techniki	BPMN, SoaML, WSDL	BPMN, UML, BPEL, WSDL, WS-BPEL	CBD, BPM, BPMN, WSDL, BPEL, UML	BPM, WSDL, WS-BPEL, WS-* specifications	BPMN, UML, BPEL, WSDL, WS-BPEL
Pokrycie wszystkich faz cyklu życia projektu (ang. Lifecycle coverage)	T	T	T	N	T
Prawnie zastrzeżony	N	T	T	N	T
Jakość specyfikacji metodyki (skala 1-5 rosnąco)	5	4	2	3	4
Poziom skalowalności, elastyczności oraz rozszerzalności (skala 1-5 rosnąco)	5	4	2	2	4

Tablica 7.1: Mapowanie kandydatów usług na elementy języka SoaML.

Nie wszystkie rozpatrywane metody powszechnie wykorzystywano w komercyjnych projektach. Wiele metod (na przykład Papazoglou) opracowano dotychczas jedynie w teorii i nie weryfikowano ich przydatności w praktyce. Najpopularniejsze metody takie jak RUP4SOA oraz SOMA ze względu na swoją dojrzałość mają najwięcej potwierdzonych przypadków użycia w komercyjnych projektach.

Większość metod sugeruje podejście Agile w tworzeniu architektury zorientowanej usługowo. Autor pracy uznał, że najwięcej cech Agile miała MatSOA (ocena 5) oraz RUP4SOA i SOMA (oceny 4). Metody te charakteryzują się najbardziej elastycznym podejściem do stosowania schematów. Pozwalają w największym stopniu na dopasowywanie się do specyfiki lub sytuacji projektowej.

Analizowane metody wykorzystują różne języki i techniki do osiągnięcia swoich celów. Większość opiera się na BPMN oraz UML. Jedynie MatSOA do tworzenia modeli architektury zorientowanej usługowo wykorzystuje SoaML. Każda z metod do komunikacji między usługami wykorzystuje Web Service oraz używa plików WSDL.

Na podstawie zestawienia w tabeli 7.1 można zauważyć, że jedynie metoda Thomasa Erl'a nie pokrywa wszystkich standardowych faz cyklu życia projektu. Spełnia ona tylko fazy analizy i projektowania.

Z wybranych metod tylko metody MatSOA oraz Thomasa Erl'a nie są prawnie zastrzeżone. Reszta z metod (RUP4SOA, Papazoglou oraz SOMA) udostępnia prawa do swojej specyfikacji komercyjnie.

Nie wszystkie z metod są dobrze zdefiniowane i posiadają szczegółową specyfikację. Fakt ten stanowi istotny problem dla ich odpowiedniego używania w realnych projektach. Metoda Papazoglou została jak dotąd opisana jedynie w zwięzłym artykule naukowym. Dużo lepiej opisane są metody takie jak RUP4SOA, SOMA oraz MatSOA.

Metody poddane analizie cechują się różnym poziomem skalowalności, elastyczności oraz rozszerzalności. Autor ocenił, że tymi cechami najbardziej charakteryzują się metody MatSOA (ocena 5) oraz RUP4SOA i SOMA (oceny 4). Metody Papazoglou i Thomasa Erl'a wykazują konieczność trzymania się ściśle ustalonych założeń i zasad.

7.2 Rozwój metod projektowania architektury korporacyjnej

Mimo że termin „architektura korporacyjna"znany jest już od kilkunastu lat, to rzadko stosuje się go w obecnych podejściach architektonicznych. Problem wynika przede wszystkim z niskiego poziomu dojrzałości większości organizacji. Nie rozumieją potrzeby stosowania architektury korporacyjnej lub są nieprzygotowane

do jej przyjęcia [25].

Architektura korporacyjna stanowi jednak doskonały mechanizm prowadzący do transformacji organizacji w celu sprawnego wdrożenia przyjętej strategii biznesowej. Szereg korzyści takich jak: lepsze zrozumienie biznesu i potrzeb klienta, lepsze dopasowanie podejmowanych działań do wymagań biznesowych oraz zwiększenie interoperacyjności systemów, rodzi możliwości, że architekci coraz częściej będą opierać się na podejściu proponowanym przez architekturę korporacyjną [20]. Stan ten może powodować z kolei rozwój metod projektowania architektury korporacyjnej. Istnieje obecnie wiele metod, jednakże spora część z nich została opracowana wyłączenie w teorii i nie ma potwierdzenia w zastosowaniach praktycznych.

Według autora pracy rozwój metod może mieć również ścisły związek z ewolucją tworzenia systemów informatycznych w architekturze zorientowanej usługowo.

7.3 Przyszłość tworzenia systemów informatycznych w oparciu o paradygmaty SOA

Bardzo wiele współczesnych systemów informatycznych opiera się na paradygmatach SOA. Podstawową zaletą tego podejścia jest koncepcja kreowania rozwiązań w oparciu o łączenie niezależnych usług. Taka filozofia pozwala tworzyć łatwo skalowalne, wykazujące się dużą elastycznością systemy informatyczne.

Wymagania biznesowe w wielu organizacjach z biegiem lat ulegają zmianie. Istotne jest dlatego tworzenie systemów, które będą umożliwiały łatwą modyfikację lub rozbudowę o nowe funkcjonalności. Rozwiązania bazujące na paradygmatach SOA uwzględniają w doskonały sposób możliwość takich zmian. Uważa się więc, że tego typu systemy będą coraz bardziej popularne w przyszłości.

Bibliografia

[1] Arsanjani Ali, *IBM Systems Journal*, wyd. 47, nr 3, 2008.

[2] Autor nieznany, `http://www.archimate.nl/en/about_archimate/what_is_archimate.html`.

[3] Autor nieznany, `http://mfiles.pl/pl/index.php/Efekty_wdra%C5%BCania_informatycznych_system%C3%B3w_zarz%C4%85dzania`.

[4] Autor nieznany, `http://www.soa-manifesto.org/`.

[5] Bean J., *and Web Services Interface Design: Principles, Techniques, and Standard*, Morgan Kaufmann Publishers, 2010.

[6] Binildas A. Christudas, Malhar Balai, Caselli Vincenzo, *Service Oriented Architecture with Java: Using SOA and Web Services to Build Powerful Java Applications*, Packt Publishing, 2008.

[7] Casanave Cory, *Enterprise Service Oriented Architecture - using the OMG SoaML Standard*, Model Driven Solutions, 2009.

[8] Elvesæter Brian, Berre Arne-Jørgen, Sadovykh Andrey, *Specyfing services using the Service Oriented Architecture modeling language (SoaML): A baseline for specification of Cloud-based Services*, In CLOSER, 2011.

[9] Endrei Mark, Ang Jenny, Arsanjani Ali, Sook Chua, Comte Philippe, Krogdahl Pål, Luo Min, Newling Tony, *Patterns: Service-Oriented Architecture and Web Services*, IBM, 04/2004.

[10] Fowler Martin, `http://www.martinfowler.com/articles/newMethodology.html`.

[11] Górski Tomasz, *Platformy integracyjne. Zagadnienia wybrane*, Wydawnictwo naukowe PWN, 2012.

[12] Graves Tom, http://weblog.tetradian.com/2011/07/26/from-biz-model-to-ea/.

[13] Johnson Simon, *Tooling platforms and RESTful ramblings* https://www.ibm.com/developerworks/community/blogs/johnston/entry/rup_for_soa_and_soma?lang=en, 2006.

[14] Josuttis Nicolai, *SOA in Practice: The Art of Distributed System Design*, O'Reilly, 2007.

[15] Keen Martin, Kaushik Rashmi, *Case Study: SOA Banking Business Pattern*, IBM, 2010.

[16] Kruchten Philippe, *The Rational Unified Process: An Introduction*, Addison-Wesley Professional, 12/2003.

[17] Offermann Philipp, Udo Bub, *Empirical comparison of methods for information systems development according to SOA*, Deutsche Telekom Laboratories, 2009.

[18] OMG contributors, *Service oriented architecture Modeling Language (SoaML) Specification*,OMG, 03/2012.

[19] Papazoglou Michael, *Service-Oriented Design and Development Methodology*, INFOLAB, 2005.

[20] Proppe Mirosław, Suchorzewska Aleksandra, *Architektura korporacyjna w administracji publicznej*, https://www.kpmg.com/PL/pl/IssuesAndInsights/ArticlesPublications/Documents/Architektura%20Korporacyjna%20w%20administracji%20publicznej_secured.pdf.

[21] Ramollari Ervin , A Survey of Service Oriented Development Methodologies, 2007.

[22] Rotem-Gal-Oz Arnon, *Wzorce SOA*, Helion, 2013.

[23] Sobczak Andrzej, *Architektura korporacyjna. Aspekty praktyczne i wybrane zagadnienia teoretyczne*, Ośrodek Studiów nad Cyfrowym Państwem, 2013.

[24] Sobczak Andrzej, *Kodeks dobrych praktyk architektów korporacyjnych*, http://architekturakorporacyjna.pl/kodeks-dobrych-praktyk-architektow-korporacyjnych/1683/.

[25] Sobczak Andrzej, *Kodeks dobrych praktyk architektów korporacyjnych*, http://architekturakorporacyjna.pl/architektura-korporacyjna-w-roku-2015-prognoza/5508/.

[26] Stevens Michael, *Service Oriented Architecture Introduction*, http://architekturakorporacyjna.pl/kodeks-dobrych-praktyk-architektow-korporacyjnych/1683/, 2012.

[27] Suda Michał, Marcin Roszczyk, *Projektowanie Modeli Usług dla rozwiązań-typu SOA*, IBM, 2006.

[28] Svanidaite Sandra, *A Comparison of SOA Methodologies Analysis And Design Phases*, Institute of Mathematics and Informatics, 2012.

[29] Tilkov Stefan, *10 Principles of SOA*, http://www.infoq.com/articles/tilkov-10-soa-principles, 2007.

[30] Wasiukiewicz Radosław, *SOA, czyli Service Oriented Architecture*, Software Developer Journal, 10/2009.

[31] Supernak Szymon, *Wprowadzenie do modelowania architektur korporacyjnych w usługach publicznych*, Zeszyty Naukowe Nr 4, WWSI, 2010.

[32] The Open Group, http://www.infoq.com/articles/tilkov-10-soa-principles.

[33] OMG, *Business Motivation Model - specification*, 05/2010.

[34] Wesołowska Halina, http://analizait.pl/2013/business-motivation-model-po-coz-my-to-robimy, 05/2013.

[35] Zimmermann Olaf, Koehler Jan, Leymann Frank, *Architectural Decision Models as Micro-Methodology for Service-Oriented Analysis and Design*, Zurich Research Laboratory, 2007.