POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY
Instytut Sterowania i Elektroniki Przemysłowej

PRACA DYPLOMOWA MAGISTERSKA

dyscyplina naukowa: INFORMATYKA

Mateusz Mróz
Nr imm.: 221192
Rok akad.: 2013/2014
Warszawa, 24 marca 2014

Temat pracy dyplomowej:

Projektowanie architektury korporacyjnej z zastosowaniem języka SoaML

Kierujący pracą:	**Kierownik Zakładu**
Dr inż. Włodzimierz Dąbrowski	*Prof. nzw. dr hab. inż. Andrzej Dzieliński*

Termin wykonania: 01.06.2014

Warszawa, dnia 01.06.2014r.

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

Praca magisterska rozpoczyna się od krótkiego

Słowa kluczowe: architektura korporacyjna, SoaML, SOA

Summary of diploma project

The Recognition of Parkinson's Disease on the Basis of Voice Using Neural Networks

Summary of this diploma...

Przedmowa

Tematem pracy było...

Spis treści

1 Wstęp **4**
- 1.1 Cel pracy . . . 5
- 1.2 Pozostałe założenia dla pracy . . . 5

2 Charakterystyka architektury korporacyjnej **6**
- 2.1 Wstęp . . . 6
- 2.2 Definicje głównych pojęć związanych z architekturą korporacyjną . . . 6
- 2.3 Wprowadzenie do architektury korporacyjnej . . . 8
- 2.4 Korzyści z wdrożenia architektury korporacyjnej . . . 12
- 2.5 Podejście do budowy architektury korporacyjnej . . . 13
- 2.6 SOA . . . 14
  - 2.6.1 Czym jest SOA? . . . 14
  - 2.6.2 Historia SOA . . . 15
  - 2.6.3 Budowa SOA . . . 15
  - 2.6.4 Usługa sieciowa w SOA . . . 17
  - 2.6.5 Warstwy architektury SOA . . . 18
  - 2.6.6 Podstawowe zasady SOA . . . 19
  - 2.6.7 Korporacyjna szyna usług - ESB . . . 22
- 2.7 Adaptacja SOA w architekturze korporacyjnej . . . 22

3 Przegląd języków do projektowania systemów informatycznych o architekturze SOA **24**
- 3.1 Wstęp . . . 24
- 3.2 SoaML . . . 25
  - 3.2.1 Czym jest SoaML? . . . 25
  - 3.2.2 Modelowanie z wykorzystaniem SoaML . . . 27
- 3.3 ArchiMate . . . 29
  - 3.3.1 Czym jest ArchiMate? . . . 29
  - 3.3.2 Modelowanie z wykorzystaniem ArchiMate . . . 30
- 3.4 Inne języki . . . 32

3.4.1 BPMN . . . 32
3.4.2 BPEL . . . 33
3.4.3 UML . . . 33
3.4.4 IDEF . . . 34

4 Metodyki projektowania rozwiązań w architekturze usługowej 35
4.1 Wstęp . . . 35
4.2 RUP4SOA . . . 35
4.2.1 Czym jest RUP? . . . 35
4.2.2 Wykorzystanie RUP w projektowaniu SOA . . . 36
4.2.3 Zalety i wady RUP4SOA . . . 37
4.3 SOMA . . . 38
4.3.1 Czym jest SOMA? . . . 38
4.3.2 Fazy metodyki SOMA . . . 38
4.3.3 Produkty SOMA . . . 41
4.3.4 Zalety i wady SOMA . . . 41
4.4 Metoda Papazoglou . . . 42
4.4.1 Service-Oriented Design and Development Methodology by Papazoglou . . . 42
4.4.2 Fazy metodyki . . . 43
4.4.3 Zalety i wady metody Papazoglou . . . 45
4.5 Metoda Thomasa Erl'a . . . 46
4.5.1 Opis metody . . . 46
4.5.2 Zalety i wady . . . 46

5 Opracowanie metody projektowania systemów informatycznych o architekturze SOA 48
5.1 Wstęp . . . 48
5.2 Metodyka MatSOA . . . 48

6 Weryfikacja koncepcji na przykładzie systemu bankowego 50
6.1 Projektowanie systemu bankowego w oparciu o utworzoną koncepcję . . . 50
6.1.1 Opis ogólny . . . 50
6.1.2 Analiza systemu . . . 50
6.1.3 Implementacja . . . 50
6.1.4 Proces wdrożenia . . . 50
6.1.5 Testy . . . 50
6.2 Cel systemu w odniesieniu do zaprojektowanej koncepcji . . . 50

7 Podsumowanie i wnioski **51**

Bibliografia **52**

Rozdział 1

Wstęp

W ostatnich latach zarysowuje się coraz większa potrzeba wdrażania systemów informatycznych. Ta tendencja związana jest przede wszystkim z usprawnianiem procesów biznesowych, ze wzrostem wydajności pracy, identyfikacją oraz minimalizacją błędów lub nieefektywnych działań w firmach. Występuje wiele ich rodzajów: CRM (ang. *Customer Relationship Managment*) - wspomagające zarządzanie relacjami z klientami, ERP (ang. *Enterprise Resource Planning*) – wspomagające zarządzanie i planowanie zasobami firmy czy też BPM (ang. *Business Process Managment*) – 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 i instytucji nie stanowi trywialnego 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 – niezbędnym do utworzenia funkcjonalnego systemu informatycznego. Bardzo istotnym elementem jest wówczas stosowanie odpowiedniej metodyki, która pozwoli na uporządkowane podejście do projektowania

tego rodzaju systemów.

1.1 Cel pracy

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

Projekt metodyki powstanie w oparciu o wybranie najmocniejszych stron oraz pominięcie wad istniejących metodyk. Podjęta zostanie również próba wdrożenia innowacyjnych elementów, które będą miały na celu usprawnienie procesu projektowania architektury korporacyjnej.

Zaprojektowana metodyka zostanie również poddana weryfikacji i testom przy tworzeniu niewielkiego systemu bankowego.

1.2 Pozostałe założenia dla pracy

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

- zdefiniowanie architektury korporacyjnej oraz wyjaśnienie terminów z nią powiązanych,
- omówienie obecnych metodyk wykorzystywanych do projektowania architektury korporacyjnej,
- omówienie języków 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 ich skala działania znacznie rosną.[22]

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

W rozdziale zostanie zdefiniowany termin „architektura korporacyjna" oraz wyjaśnione najważniejsze elementy z nim związane. Podjęta będzie również tematyka „architektury zorientowanej na usługi" oraz to jak ją zaadaptować do architektury korporacyjnej.

2.2 Definicje głównych pojęć związanych z architekturą korporacyjną

Projektując architekturę korporacyjną (ang. *Enterprise Architecture*) niezbędna jest doskonała znajomość terminów z nią powiązanych. W sekcji tej zostanie dokonany przegląd głównych pojęć odnoszących się do architektury korporacyjnej oraz podjęta próba ich zdefiniowania.

Słowo „architektura"w ujęciu do oprogramowania stanowi zbiór podstawowych decyzji związanych z oprogramowaniem lub rozwiązaniem programistycznym - zaprojektowanym z reguły tak, aby sprostać wszelkim atrybutom jakościowym (wymaganiom architektonicznym) danego projektu. Architektura tego typu zawiera z reguły główne komponenty, atrybuty oraz zdefiniowane współdziałania (zachowania i interakcje) umożliwiające spełnianie atrybutów jakościowych. Architektura może być wyrażona na więcej niż jednym poziomie abstrakcji, gdzie liczba poziomów zależy od rozmiaru i złożoności projektu. Termin ten posiada również powszechnie akceptowany opis w standardzie IEEE, który określa ją jako „zbiór podstawowych koncepcji lub właściwości systemu w danym środowisku zawarte w jego elementach, relacjach oraz zasadach projektowania i rozwoju"(IEEE 42010). [20]

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 uchwycenia tego abstraktu.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 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 jego 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.[22]

2.3 Wprowadzenie do architektury korporacyjnej

Architektura korporacyjna (ang. *enterprise architecture*) 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.[21]

Mimo, że architektura korporacyjna jest kojarzona głównie z technologią informatyczną, to przede wszystkim nawiązuje do metod optymalizacji procesów, wskazując architekturę biznesową, metody zarządzania efektywnością oraz organizacyjne uporządkowanie procesów[27]

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.[21]

W ujęciu rzeczowym architektura korporacyjna definiowana jest jako for-

malna reprezentacja właściwości danej organizacji. W znaczeniu tym rozpatruje się ją również jako misję całej organizacji - 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.[21]

Spojrzenie czynnościowe zakłada definicję architektury korporacyjnej jako zadań i umiejętności zarządzania tym, co jest opisane w definicji atrybutowej.[27]

Podejście prezentowane w *A Practical Guide to Federal Enterprise Architecture* uwzględnia aspekt transformacji organizacji w obszarze objętym architekturą korporacyjną, który jest immanentną cechą złożonych systemów adaptacyjnych.[21]

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

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

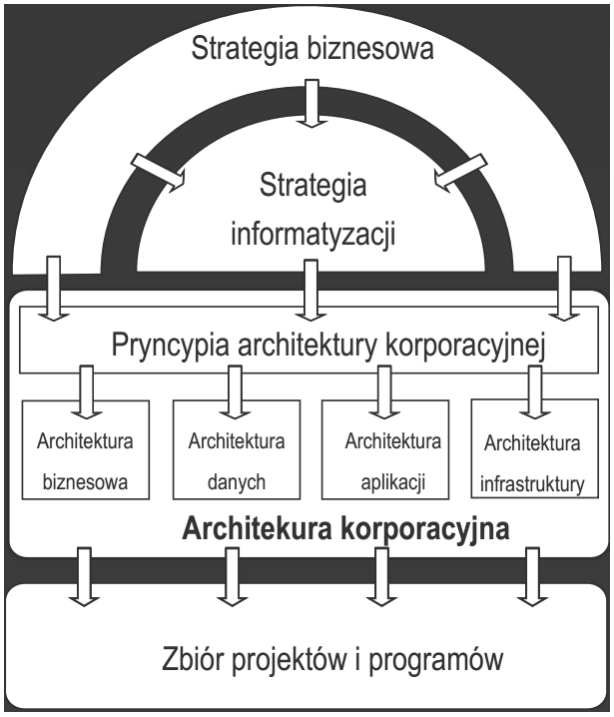
gdzie:

- $R = (r_1, r_2, ..., r_n)$ - 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 (dotyczące 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 (*Business Architecture*) - opisuje strategię biznesową i sposoby zarządzania organizacją, jej strukturą organizacyjną oraz główne procesy biznesowe, a także relacje pomiędzy tymi elementami.
  - Architektura danych (*Data Architecture*) - opisuje główne typy i źródła danych niezbędnych do funkcjonowania organizacji.
  - Architektura aplikacji (*Applications Architecture*) - opisuje poszczególne aplikacje, ich rozlokowanie, wzajemne współdziałanie oraz relacje między tymi aplikacjami, a głównymi procesami biznesowymi organizacji.
  - Architektura techniczna (*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).



Rysunek 2.1: Umiejscowienie architektury w korporacji. [1]

W ostatnim czasie coraz bardziej popularnym terminem staje się „architektura korporacyjna IT". Wykazuje ona pewne różnice w porównaniu do „zwykłej architektury korporacyjnej".

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.

Istotnym elementem, o którym należy wspomnieć prowadząc rozważania na temat architektury korporacyjnej jest pojęcie „ładu architektonicznego (ang. *enterprise architecture governance*)". Stanowi on przede wszystkim mechanizm dostarczający struktury, za pomocą której jest ustanawiany zbiór celów tworzenia architektury korporacyjnej oraz środków, poprzez które możliwe osiąganie 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 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, ale jako całościowe przedsięwzięcie z pogranicza zarządzania, jak i informatyki. Opracowanie architektury korporacyjnej nie należy rozpatrywać jako cel sam w sobie, ale jako stan pośredni postrzegany jako 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 dziań 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, 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.[21]

2.5 Podejście do budowy architektury korporacyjnej

Budowa architektury korporacyjnej nie stanowi zadania trywialnego. Bardzo istotnym czynnikiem wpływającym na sukces jej wdrożenia do danej organizacji jest odpowiednie projektowanie i podejmowanie odpowiednich 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") bezpośrednio w czterech domenach architektonicznych: aplikacji, danych, biznesowej i technicznej. W kolejnym kroku analizowane są luki we wszystkich wymienionych domenach oraz przygotowany 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 wykonania analiz na zbyt dużym 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"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 przygotowywana jest zbiorcza analiza luk oraz plan przejścia (ang. *roadmap*) pomiędzy stanem bazowym i docelowym. Zaletą tej ścieżki jest dostarczenie na

najwcześniejszym etapie kluczowego elementu architektury korporacyjnej - architektury biznesowej. Stanowi to dobrą podstawę dla opracowywania kolejnych architektur. Do wad podejścia zaliczyć można ryzyko z częstym występowaniem niskiego poziomu części biznesowej organizacji.[21]

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. [26, 7] SOA stanowi pewnego rodzaju paradygmat, który prowadzi do określonej architektury. [26] SOA reprezentuje przesunięcie w inżynierii oprogramowania i podnosi poziom abstrakcji, grupując wspólne działanie procesów biznesowych i wystawiając to jako usługę. [25]

Mianem usługi w SOA określa się zbiór funkcjonalności pewnej aplikacji, który jest udostępniany jako interfejs. [5] Organizacja OASIS 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 operując na obiektach biznesowych. Ponadto owe 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 tych 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 komponeenty.[23]

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.

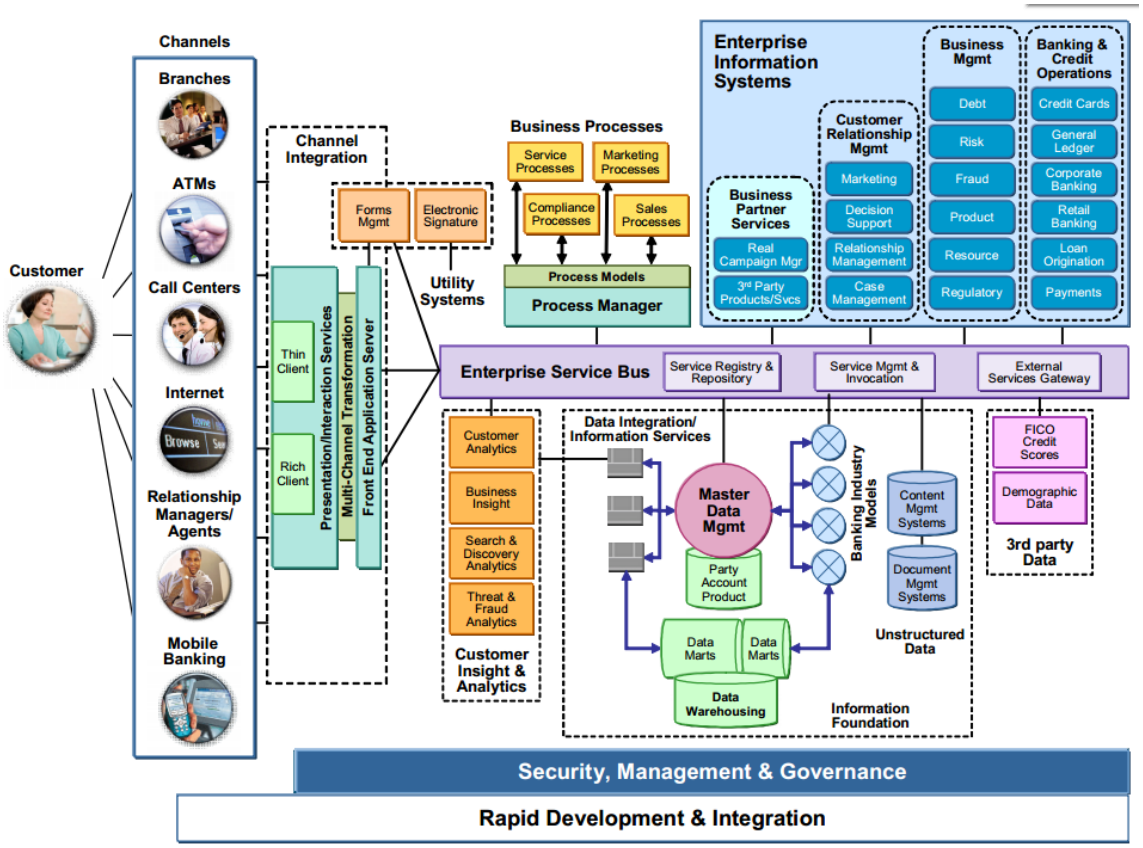
Przełomowym momentem w historii powstawania SOA było utworzenie usług typu *Web Service* przez firmę Microsoft w 2000 roku.[14]

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. [7, **?**].

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



Rysunek 2.2: Przykładowa architektura SOA.

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 „każdy z każdym". [?] 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 na temat sposobu dostępu, definicji, reguł, bezpieczeństwa 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.[?, ?]

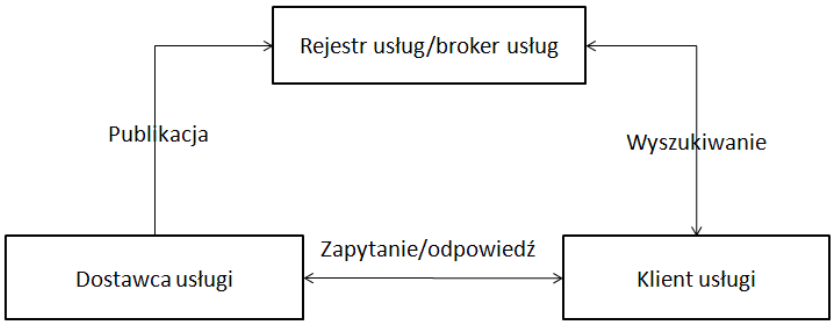
Istotnym elementem w systemach typu SOA jest repozytorium (ang. SOA repository). Stanowi on centralny „magazyn" dla elementów składowych usług takich jak: kod źródłowy, zestawy instalacyjne, specyfikacja itp. Repozytorium usług jest tworzone i wykorzystywane głównie na etapie projektowania usług. [SOAidntech]

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 zdefiniowany 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. [WSArchFunc]) : klienta usługi (ang. service requestor), dostawcę usługi (ang. service provider) oraz rejestr usług (service registry) lub broker usług (ang. service broker). Transport danych w przypadku Web Service z reguły opiera się na HTTP lub HTTPS z wykorzystaniem SOAP. [IBMWebServiceInfo] 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 usługu. 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.



Rysunek 2.3: Usługa sieciowa w SOA (Web Service). [**?**]

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. Do przykładów można zaliczyć: aplikacje Java Enterprise lub .NET, systemy transakcyjne, bazy danych oraz systemy ERP i CRM.
- Warstwa komponentów - obejmuje składniki oprogramowania realizujące konkretne operacje usług Odpowiada za fizyczną implementację usługi oraz jej opis dostarczając 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 przypadku 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 zlecenie 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 obserwację pozostałych warstw i informuje o zdarzeniach niezgodnych z zakładanymi.
- Warstwa Business Intelligence - stanowi podstawę do budowania rozwiązań Business Intelligence umożliwiających modelowanie danych analitycznych i eksploracje. W tej warstwie włączane są kluczowe zagadnienia związane z obiegiem informacji w przedsiębiorstwach.
- Warstwa zarządzania - odnosi się do zarządzania wszystkimi elementami cyklu życiu SOA oraz wszystkich 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 mędzy komponentami w SOA może wykorzystywać różne kanały komunikacji: WS, HTTP, HTTPS, LDAP, FTP, IMAP, JMS, 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. [SOAterlprD] Termin „luźne powiązania" (ang. Loose Coupling) stanowi jeden z fundamentów SOA. [SOAsdj102009] 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. [SOAterlprD, SOAidntech]
- interoperacyjność - 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 oraz wybierać odpowiednie protokoły, które dany system będzie obsługiwał. [SOAsdj102009]
- composability - zasada composability jest związana z zachowaniem odpowiednich relacji między komponentami. Systemy, które podążają za ta zasadą cechują się możliwością łączenia swoich komponentów w różne kombinacje w celu spełnienia postawionych wymagań. Umiejętność efektywnego komponowania usług z już istniejących jest kluczowym wymogiem dla osiągnięcia niektórych z najbardziej podstawowych celów przy tworzeniu systemów opierających się na architekturze zorientowanej na usługi. [SOAterlprD]
- reużywalność - każda tworzona usługa powinna zachowywać zasadę reużywalności (ang. reusability). Opiera się ona na takim projektowaniu usług, aby była możliwość wielokrotnego jej wykorzystania do tworzenia kolejnych usług. [SOAsdj102009]
- kontraktowość usług - każda usługa powinna mieć zdefiniowany kontrakt (ang. service contract), który zawierany jest każdorazowo pomiędzy usługą, a jej konsumentem. Wyrażane są przez nie cele i możliwości danej uslugi. [SOAterlprD] W kontraktach znajdują się również opisy informacji oferowanych i oczekiwanych przez usługi. [SOAinfoq10]
- abstrakcyjność - zasada abstrakcji (ang. abstraction) 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 tylo możliwe. [SOAterlprD]

- autonomiczność usług - autonomia usługi (ang. service autonomy) jest kolejnym paradygmatem projektowania systemów typu SOA. Termin ten odwołuje się do usług o podwyższonej niezależności od środowisk wykonawczych. [SOAsrvautoWCSS] 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. [SOAinfoq10]

- wykrywalność usług - zasada wykrywalności usług (ang. services discoverability) polega na tym, że usługi powinny być opisywane za pomocą meta danych w takich sposób, aby były efektywnie wyszukiwane, przetwarzane i interpretowane zarówno w czasie projektowania jak i wykonywania. [SOAinfoq10, SOAsrvautoWCSS]

- spójność i ziarnistość usług - interfejsy usług w systemach SOA powinny być tak zaprojektowane, aby wiązały tylko określony zbiór wymagań biznesowych. [26] 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 funkcjnonalności). [SOAdefgranaaImp]

- 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. [SOAsrvautoWCSS] 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. [26]

- enkapsulacja - 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 systemie informatycznym 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* jest bardzo popularny 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.[6]

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 ilość 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 wykorzystaniem opracowanej optymalnej metodyki.

Rozdział 3

Przegląd języków do projektowania systemów informatycznych o architekturze SOA

3.1 Wstęp

Modele tworzone w ramach architektury korporacyjnej możemy traktować na kilka sposobów:

- 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 określone decyzje związane z funkcjonowaniem organizacji lub wdrożeniem określonych rozwiązań informatycznych. Należy wziąć również pod uwagę fakt, że potencjalnymi osobami często mogą

być różne osoby - analityk systemowy, projektant systemów, czy też osobą odpowiedzialną za komórkę informatyczną w danej organizacji).[21]

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.

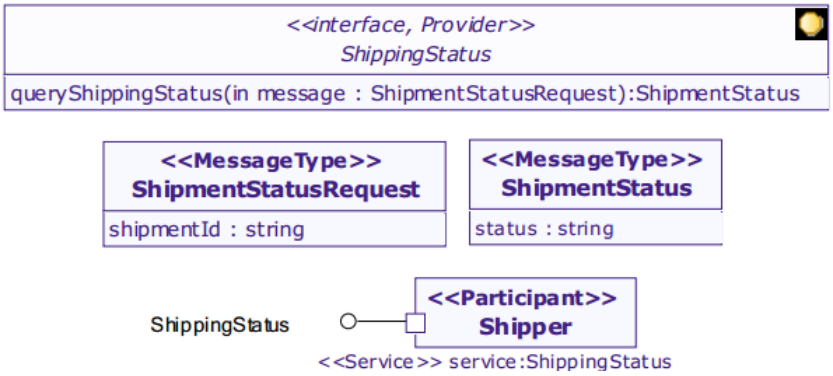
3.2 SoaML

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:

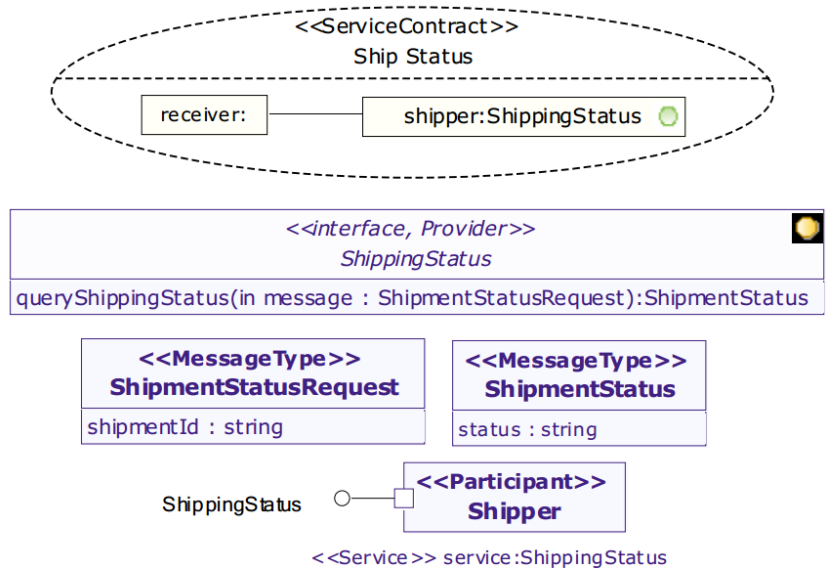
- Proste interfejsy (ang. *Simple interfaces*) - prosty interfejs, który opiera się na jednokierunkowej interakcji dostarczanej przez uczestnika interakcji jako interfejs UML. Uczestnik otrzymuje operacje na swój port i może dostarczyć rezultaty dla wywołującego zapytanie. Ten rodzaj jednokierunkowej 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*. [16]
- 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.
  TODO: Nie wiadomo czy dobrze przetłumaczone. Podejście to 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



Rysunek 3.1: Specyfikacja usługi *Ship Status* 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]

powinni implementować do realizacji ich ról w usłudze. 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.



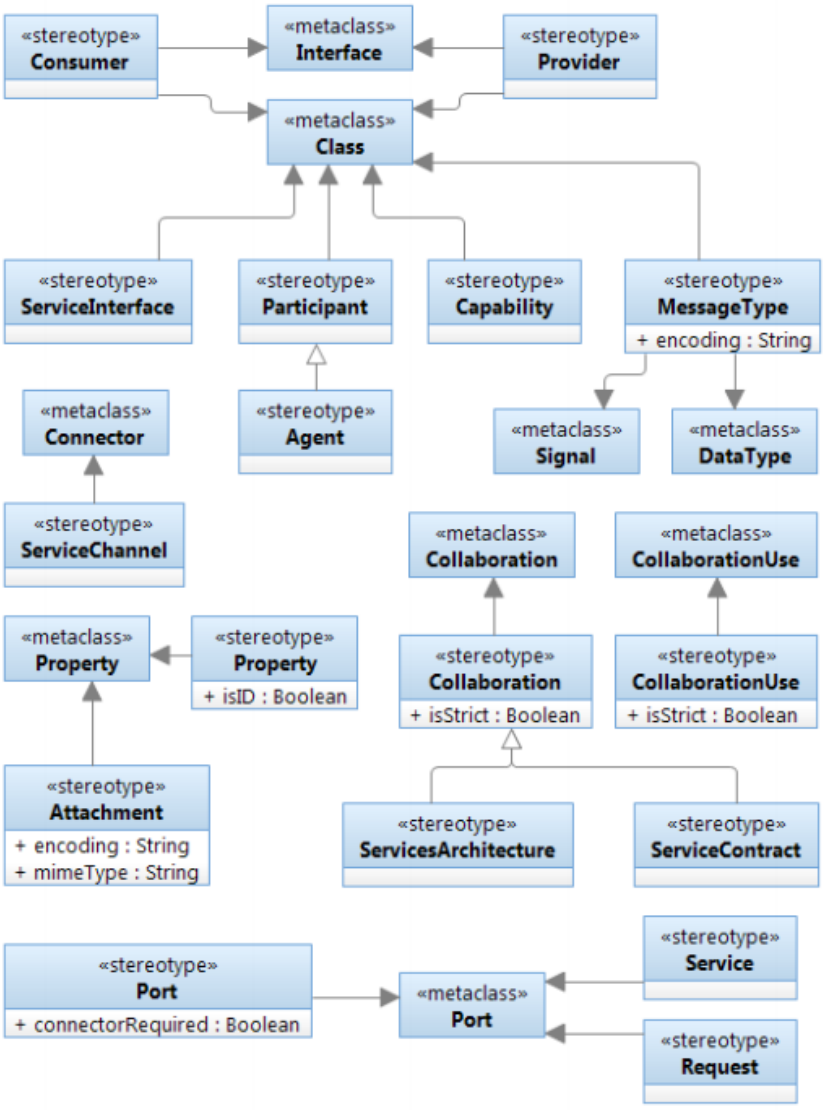
Rysunek 3.3: Specyfikacja usługi *Ship Status* wykorzystując podejście oparte na „kontrakcie usługi”.[8]

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[**?**] rozszerza metamodel UML dla bezpośredniego modelowania usług w rozproszonych środowiskach. [8]

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



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

- 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»*.
- 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 *capabilities.*[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

jak na przykład: Enterprise Architect, BizzDesign Architect, Archi, ARIS lub Microsoft Visio.[28]

ArchiMate bazuje na dwóch podstawowych założeniach:

- podejściu warstwowym - związane 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 biznesowej). ArchiMate wprowadza ponadto pojęcie usług wewnętrznych, które są wykorzystywane w ramach określonej warstwy.

Komunikacja pomiędzy warstwami: biznesową, aplikacyjną i technologiczną stanowi jedną z największych zalet ArchiMate.

W warstwie biznesowej (ang. *Business layer*) zawarte są usługi świadczone klientom zewnętrznym oraz produkty. Są one 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 motywacją, implementacją i konfiguracją.[21]

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 najpierw *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, organizacja i odpowiadający jej model biznesowy może być reprezentowany jak 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*) i być połączonym 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*.

- The Value Proposition (EC: value-proposition) is represented by one or more Product entities, with related Value entities.
- We have already mapped the Key Partner and Customer Segment entities as Business Actor / Business Role / Business Interface entities.
- All of the Enterprise Canvas cells are mapped to Business Function entities with appropriate names (default: same names as in Enterprise Canvas); these also represent in part the Key Activities cell of the Business Model Canvas.
- All of the Enterprise Canvas interface cells (customer-relations, customer-channels, value-return; supplier-relations, supplier-channels, value-outlay) also include Business Interaction entities (which may optionally replace the respective Business Function entity), each linked to a Business Interface entity, representing the channels and other interfaces to the overall service.
- Each of the six channels (Business Interface entities) links to one or more Business Object entities, which in turn link to the Business Interface of a Customer Segment or Key Partner; each Business Object may optionally be linked to a Meaning entity.
- Each Business Object entity attached to a customer-channel Business Interface should also be linked to one or more Product entities; the same may optionally apply to Business Object entities attached to the supplier-channel Business Interface.
- The Business Object entities associated with a supplier-channel and value-outlay, or customer-channel and value-return, should be linked by a Contract entity, representing the required relationship between the respective Enterprise Canvas main-channel and back-channel (e.g.

Business Model Canvas linkage between Value Proposition, Customer Channel. Customer Segment and Revenue Stream).

3.4 Inne języki

3.4.1 BPMN

BPMN (ang. *Business Process Modeling 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 fragmentu dotyczącego procesów biznesowych.

Definiuje diagramy nazywane diagramami procesu biznesowego - 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 budowanie zarówno modeli procesów zrozumiałe przez analityków biznesowych, jak również przez kadrę zarządzającą.

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 sekwnecje 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 pozwala na 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. *opague tokens*) oraz pominięcie (ang. *omission*).

Z kolei proces wykonywalny w pełni specyfikuje opis, który można umieścić w środowisku wykonawczym. Musi być w pełni 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 część procesu biznesowego. Innym przykładem jest jeszcze BPELJ, który umożliwia zastąpienie usług sieciowych małymi aplikacjami napisanymi na podstawie języka Java.[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 rozwijany jest w ramach 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 spojrzenia na system.

Modelowanie systemów z wykorzystaniem UML umożliwia jednoczesne dokumentowanie całości systemu w zrozumiały dla całego zespołu sposób.[11]

UML służy głównie do modelowania systemów informatycznych, w tym również uwzględniających aspekty architektoniczne.

UML znajduje coraz częściej zastosowanie w inżynierii systemów, do modelowania procesów biznesowych lub reprezentacji struktur organizacyjnych. W tym celu bardzo często wykorzystywane są możliwości jego rozszerzenia - na przykład za pomocą profili. Istotą ich stosowania jest 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 szerokie zastosowania 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 ich klientami.

IDEF3 pozwala na modelowanie procesów biznesowych. Przystosowany jest również do prezentowania behawioralnych aspektów planowanych lub działających systemów.

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

Rozdział 4

Metodyki projektowania rozwiązań w architekturze usługowej

4.1 Wstęp

4.2 RUP4SOA

4.2.1 Czym jest RUP?

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

RUP zapewnia zdyscyplinowane podejście do przydzielania zadań i obowiązków w ramach rozwoju organizacji. Celem podstawowym tej metodyki jest dostarczenie wysokiej jakości oprogramowania spełniającego potrzeby użytkowników końcowych w zgodzie z harmonogramem i ramami budżetowymi. [15] Stanowi przede wszystkim bardzo duży zbiór praktyk, który może być dostosowywany i rozszerzany w celu jak najlepszego dopasowania się do danej organizacji. Charaktertystyczne dla metody jest rozwój sterowany przypadkami użycia (ang. *Use Case Driven Development*).[10]

W RUP można wyróżnić poszczególne fazy:

- faza początkowa (ang. *inception*) – wstępne określenie wymagań, ryzyka, kosztu, harmonogramu, a także architektury systemu,
- faza opracowania (ang. *elaboration*) – ustalenie wymagań (większości przypadków użycia), architektury systemu oraz planu całego procesu wytwarzania systemu,

- faza konstrukcji (ang. *construction*) – tworzenie systemu (kolejnych komponentów), w trakcie następuje oddanie pierwszej (i być może dalszych) wersji użytkownikowi,
- faza przekazania (ang. *transiation*) – system jest przekazywany użytkownikowi, wdrażany, szkoleni są pracownicy obsługi systemu, następuje walidacja i końcowe sprawdzenie jakości.[15]

4.2.2 Wykorzystanie RUP w projektowaniu SOA

RUP4SOA stanowi modyfikację metodyki 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, metodyka ta opiera się na wyróżnieniu trzech dyscyplin związanych 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 metodyki 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 opracowywana architektura była projektowana z zamysłem o jak najprostszej realizacji i 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 metodyki 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. W następnym etapie 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 metodyk stosowanych do projektowania architektury usługowej. Największy nacisk kładzie na fazy związane z analizą i projektowaniem systemu informatycznego - rozszerzonym o elementy związane z usługami sieciowymi. Fakt ten może powodować większą zgodność między wyobrażnieniami 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]

Metodyka ta nie specyfikuje wielu elementów istotnych do budowy rozwiązań integracyjnych. Skupia się na projektowaniu pojedynczego systemu, który może być jedynie włączony do platformy integracyjnej. [11]

4.3 SOMA

4.3.1 Czym jest SOMA?

SOMA (ang. *Servie-Oriented Modeling and Architecture* stanowi kolejną metodykę projektowania systemów w architekturze usługowej. [11] Metoda cyklu rozwoju oprogramowania definiująca kluczowe techniki i opisująca poszczególne role w projekcie SOA oraz struktury podziału pracy (WBS - ang. *Work Breakdown Structure*). WBS obejmuje zadania związane z wejściowymi i wyjściowymi produktami pracy, z normatywnymi wskazówkami do szczegółowej analizy, projektowaniem, implementacją i wdrażaniem usług oraz komponentów potrzebnych do budowy wydajnego, reużywalnego środowiska.

SOMA została utworzona i jest rozwijana przez firmę IBM. W swoim podejściu metodyka wykorzystuje wiele elementów z Worldwide Project Management Method (jedna z metodyk zarządzania projektami). [1]

4.3.2 Fazy metodyki SOMA

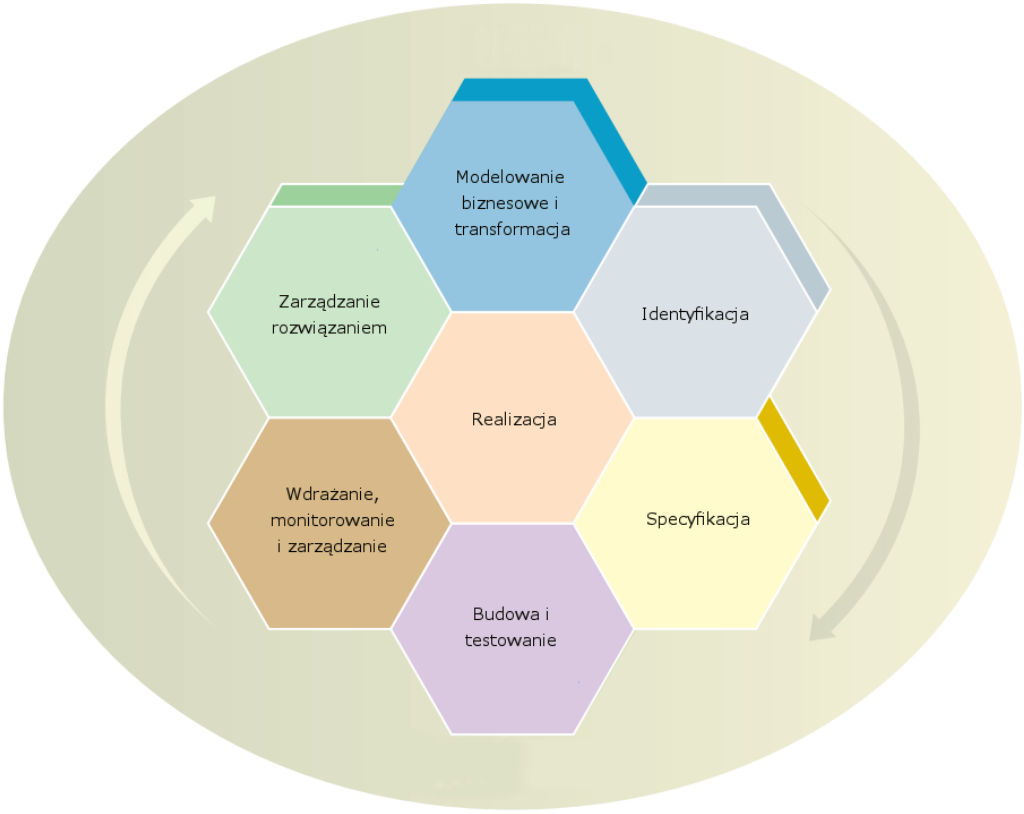
SOMA bazuje na siedmiu podstawowych fazach:

- 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.

Można również mówić o fraktalnej charakterystyce metodyki. 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 usługi zmieniają się w trakcie cyklu życia 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.



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

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 celów biznesowych. [11]

W fazie modelowania biznesowego i transformacji funkcjonowanie organizacji jest modelowane, symulowane i optymalizowane. 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 metodyki, 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. Występuję kilka możliwych

scenariuszy: wytwarzania oprogramowania (ang. *custom development*), integracji pakietów aplikacji (ang. *package application integration*). W tym etapie tworzona jest również konfiguracja metodyki dostosowana do potrzeb danego projektu. Ponadto zdefiniowaniu podlegają: zadania, role, produkty pracy oraz poradniki.[1]

Faza identyfikacji związana jest z identyfikacją trzech elementarnych dla SOA konstrukcji: usług, komponentów oraz usług.[11] Do zadań tego etapu należy: dekompozycja obszarów biznesowych (Domain Decomposition), modelowanie usług w kontekście celów biznesowych (Goal-Service Modeling) oraz analiza istniejących zasobów IT (Existing Assets Analysis).[24] 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. Metodologia dostarcza gotowe szablony, wzorce oraz 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 niefukncjonalnych - 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.

Faza realizacji związana jest ze sprawdzaniem stabilności i realizowalności zaprojektowanego rozwiązania poprzez budowę prototypów. Zadanie to

powinno mieć miejsce we wczesnym etapie projektu, aby zminimalizować niepowiedzenia i wyeliminować niepotrzebne ryzyka. Faza ta stanowi pewnego rodzaju formę technicznego studium wykonywalności. Każdy poziom rozwiązania wiąże się również z wybieranym podejściem lub sposobem realizacji prac.

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 wpływać na realizowalność zadań.

W fazach implementacji i wdrożenia oraz monitorowania i zarządzania są konstruowane, generowane i łączone usługi, komponenty funkcjonalne i techniczne oraz przepływy. Konstruowane są również poszczególne elementy opakowujące oraz odpowiednie mechanizmy dla istniejących komponentów. 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 ekspozycję encji.

Oprócz wymienionych modeli metodyka 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 metodyki 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*.

Do wad SOMA można zaliczyć fakt, że mimo szczegółowego opisu projektu technicznego usług biznesowych nadal elementy dotyczące ich wdrożenia są przedstawiane tylko szczątkowo.

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 tymi dwiema fazami jest niejednoznaczna i często w projektach powstają z tego z tego powodu różne nieścisłości (jak na przykład duplikaty usług)[29].

4.4 Metoda Papazoglou

4.4.1 Service-Oriented Design and Development Methodology by Papazoglou

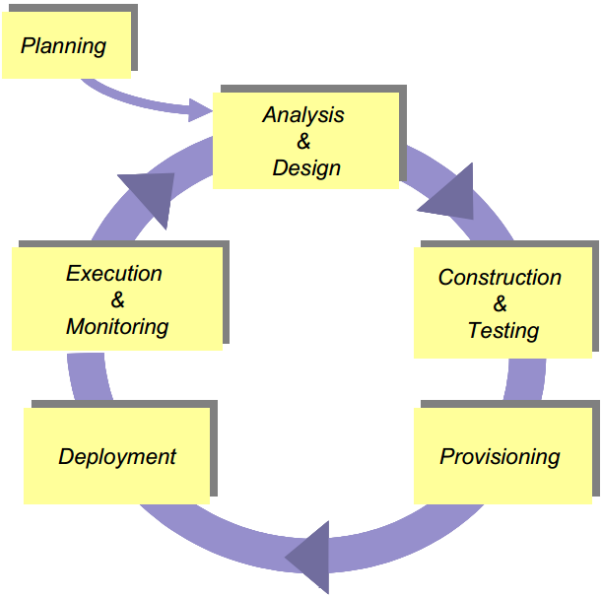
Service-Oriented Design and Development by Papazoglou również stanowi metodę projektowania architektury systemów informatycznych zorientowanych na usługi. Metoda opiera się ponadto na kilku podstawowych założeniach jak poniżej.

- 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 narzędzie 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 - kolejno: projektowania, implementacji, testowania oraz transformacji procesów biznesowych w kompozytowe aplikacje. W ten sposób wymagania biznesowe mogą być śledzone przez cały cykl życia - od celów biznesowych przez projektowanie oprogramowania oraz zasobów kodu, aż do kompozytowych aplikacji.

4.4.2 Fazy metodyki

Metodyka utworzona przez Papazoglou składa się z elementów (rys.) bazujących na innych metodykach takich jak: *Rational Unified Process*, *Component-based development* oraz modelowania procesów biznesowych zaproponowanego przez Paula Harmona w 2003 roku[18].



Rysunek 4.2: Fazy metodyki zaproponowej przez Papazoglou.[18]

Podstawowym celem metodyki jest jak najlepsza integracja usług oraz osiągnięcie jak najwyższego poziomu ich interoperacyjności. Poszczególne fazy, które wchodzą w składy metodyki są wymienione poniżej.

- Faza planowania (ang. *Planning phase*) - określa prawdopodobieństwo wykonalności, 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 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 tych wymagań 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*), 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: jedna do produkcji usług (ewentualnie z wcześniej utworzonych elementów), natomiast druga związana jest z asemblacją usług reużywalnych (wielokrotnego użytku). Projektowanie usług, podobnie jak analiza usług posiada swoje własne specjalne charakterystki i techniki. Faza projektowania rozpoczyna się od rozważenie kilku istotnych problemów: zarządzania usługami i ziarnistostością 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 specyfing phase*) - składa się z trzech równie istotnych elementów: specyfikacji strukturalnej (ang. *Structural specification*), specyfikacji zachowań (ang. *Behavioural 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. *Specifying business pro-*

cesses) - dzieli się na 3 mniejsze etapy: opisywanie struktury procesów biznesowych (ang. *Designing the business process structure*), opisywanie ról biznesowych (ang. *Describing 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 opisów interfejsów oraz definicję opisów 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 zdyscyplinowaną metodologią ustalenia dopuszczalnych poziomów usług, które dotyczą celów biznesowych, procesów i kosztów. W fazie tej następuje wykonanie pomiarów, monitorowanie, raportowanie i poprawianie jakości usług i aplikacji dostarczanych przez rozwiązania zorientowane usługowo.[18]

4.4.3 Zalety i wady metody Papazoglou

Do zalet metody Papazoglou można zaliczyć przede wszystkim pokrywanie 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. [19]

4.5 Metoda Thomasa Erl'a

4.5.1 Opis metody

Metoda Thomasa Erl'a stanowi pierwszą metodologię 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. *define business 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 *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 identyfikowani oraz uwzględniani na modelach „usług kandydujących"(ang. *service candidates*). [25]

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 jak *Web service'y*.

4.5.2 Zalety i wady

Metodologia projektowania SOA zaproponowana przez Thomasa Erl'a może być używana jedynie w połączeniu z innymi metodologiami. 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. Ponadto role podmiotów biorących udział w działaniu systemu nie są 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, 19]

Rozdział 5

Opracowanie metody projektowania systemów informatycznych o architekturze SOA

5.1 Wstęp

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

Stopień złożoności biznesowej niektórych organizacji jest naprawdę wysoki. Bez stosowania odpowiedniej metodyki nie jest możliwe utworzenie sprawnie funkcjonującego systemu informatycznego. Występuje ich już wiele jednakże każda wykazuje pewne niedociągnięcia.

Większość metodyk łączy opis sposobu przejścia od stanu początkowego do końcowego. Każda z nich stanowi hierarchicznie uporządkowany zbiór elementów takich jak: fazy, aktywności, zadania oraz kroki. Opierają się również na wykorzystywaniu różnorodnych technik oraz proponowanych języków modelowania i notacji[17].

W rozdziale tym zostanie podjęta próba zdefiniowania nowej metodyki, a następnie porównanie jej z obecnie dostępnymi.

5.2 Metodyka MatSOA

MatSOA stanowi autorską metodykę projektowania architektury korporacyjnej. Obecne metodyki posiadają zarówno wiele wad jak i zalet. Różnią się podziałami na etapy, wykorzystywanymi technologiami oraz zakresem sto-

sowalności. MatSOA stanowi połączenie najlepszych cech z już istniejących wzorców. Wprowadza również pewne własne elementy usprawniające projektowanie architektury korporacyjnej.

Cechą charakterzstzcynA MatSOA jest fakt, że do specyfikacji usług wykorzystuje język SoaML stanowiący rozszerzenie dla UML.

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

Rozdział 6

Weryfikacja koncepcji na przykładzie systemu bankowego

6.1 Projektowanie systemu bankowego w oparciu o utworzoną koncepcję

6.1.1 Opis ogólny

6.1.2 Analiza systemu

6.1.3 Implementacja

6.1.4 Proces wdrożenia

6.1.5 Testy

6.2 Cel systemu w odniesieniu do zaprojektowanej koncepcji

Rozdział 7

Podsumowanie i wnioski

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

[7] 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.

[8] *Specyfing services using the Service Oriented Architecture modeling language (SoaML): A baseline for specification of Cloud-based Services*, Elvesæter Brian, Berre Arne-Jørgen, Sadovykh Andrey, 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'Reillyl, 2007

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

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

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

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

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

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

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

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

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

[24] Suda Michał, Marcin Roszczyk, *Projektowanie Modeli Usług dla rozwiązańtypu SOA*, IBM, 2006.

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

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

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

[28] The Open Group, `http://www.opengroup.org/subjectareas/enterprise/archimate`.

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