**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

# Rozdział 1

# Wstęp

W ostatnich latach zarysowuje się coraz większa potrzeba wdrażania systemów informatycznych w celu usprawniania funkcjonowania firm. 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 firm może wywoływać różnorodne efekty: techniczne (związane z zastosowaniem techniki komputerowej: zwiększenie szybkości przetwarzania informacji, wzrost dokładności przetwarzania, wzrost szczegółowości informacji, poprawa bezpieczeństwa poufnych informacji) ekonomiczne (związane pośrednio ze wzrostem efektywności i szybkości podejmowania decyzji), organizacyjne (przede wszystkim związane są z usprawnieniami struktury organizacyjnej i procesów zachodzących w przedsiębiorstwie: podniesienie sprawności obiegu dokumentów, eliminacja zbędnej pracy administracyjnej, poprawa koordynacji zadań, eliminacja błędów), socjopsychologiczne (związane przede wszystkim z rozszerzeniem zakresu komunikacji pomiędzy pracownikami, usprawnieniem systemu ocen pracowniczych, polepszeniem kultury organizacyjnej). [1]

Utworzenie systemu informatycznego dla dużych firm i instytucji nie stanowi zadania trywialnego. 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 sprawnie działającego, spełniającego wymagania klienckie systemu informatycznego. Bardzo istotnym elementem jest wówczas stosowanie odpowiedniej metodyki, która pozwoli na usystematyzowane podejście do projektowania tego rodzaju systemów.

## 1.1 Cel pracy

Podstawowym celem pracy jest opracowanie nowej metodyki projektowania systemów o architekturze korporacyjnej, która będzie opierała się na wykorzystaniu języka SoaML. Dotychczas utworzono już podobne metodyki, jednakże żadna nie była w stanie w pełni sprostać wszystkim wymaganiom stawianym przez problematykę projektowania systemów o architekturze korporacyjnej.

Projekt metodyki będzie się opierał na wybraniu najmocniejszych stron oraz pominięciu wad istniejących metodyk. Autor pracy podejmie również próbę wdrożenia innowacyjnych elementów do zaprojektowanej metodyki, 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 Założenia dla pracy

Oprócz założenia głównego dla pracy – utworzenie własnej metodyki projektowania architektury korporacyjnej z zastosowaniem SoaML- 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

## 2.2 Czym jest architektura korporacyjna?

## 2.3 Definicje głównych pojęć związanych z architekturą korporacyjną

## 2.4 Korzyści z wdrożenia architektury korporacyjnej

## 2.5 Podejście do budowy architektury korporacyjnej

## 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. [6]

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). [2]

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. [9, 4] SOA stanowi pewnego rodzaju paradygmat, który prowadzi do określonej architektury. [9]

Mianem usługi w SOA określa się zbiór funkcjonalności pewnej aplikacji, który jest udostępniany jako interfejs. [3] 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.[JSOAcookkb]

### 2.6.2 Historia SOA

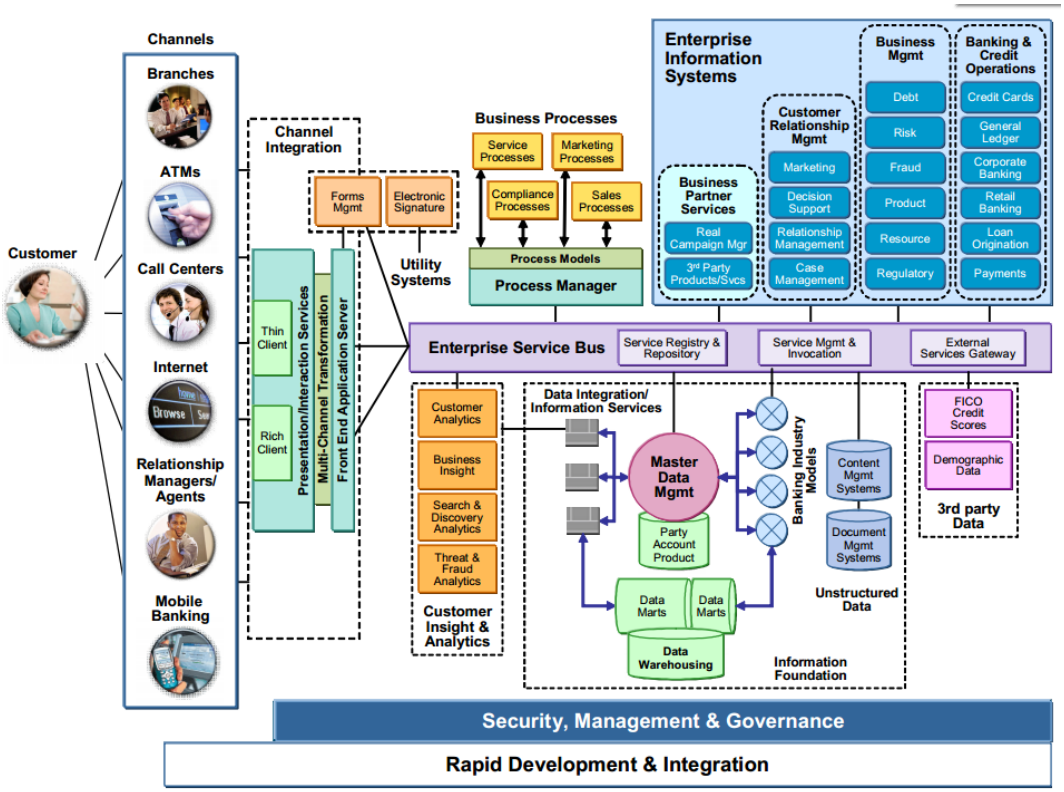
Określenia „SOA" lub „Architektura zorientowana na usługi" zostały po raz pierwszy wykorzystane w pracy naukowej analityka z firmy Garnter Yefim V. Natis w dniu 12 kwietnia 1996 roku.

### 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 ob-

szarach funkcjonowania firmy. Ściśle wiąże się z panującymi w niej procesami biznesowymi i wymaga odpowiedniego dostosowania.[SOAPAIesdasb]

Systemy opierające się na SOA składają się z zestawu aplikacji biznesowych pogrupowanych w niezależne komponenty, które komunikują się ze sobą wymieniając usługi. [SOAwJBBC, SOAidntech].




Rysunek 2.1: Przykładowa architektura SOA. [?]

Rys. [PRZArchSOA] Przykładowa budowa systemu SOA. 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 „każdy z każdym". [SOAidntech] 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. [SOAidntech, SOAfddum]

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 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. [9] 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. [9]
- 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.

## 2.7 Adaptacja SOA w architekturze korporacyjnej

# Rozdział 3

# Metodyki projektowania rozwiązań w architekturze usługowej

## 3.1 Wstęp

## 3.2 RUP4SOA

### 3.2.1 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. [8] 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*).[5]

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.[8]

### 3.2.2 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. [6] 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. [6]

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.

### 3.2.3 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. [7]

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. [6]

## 3.3 SOMA

# Rozdział 4

# Przegląd języków do projektowania systemów informatycznych o architekturze SOA

## 4.1 SoaML

### 4.1.1 Czym jest SoaML?

### 4.1.2 Główne elementy notacji

### 4.1.3 Modelowanie z wykorzystaniem SoaML

## 4.2 ArchiMate

## 4.3 Inne języki

# Rozdział 5

# Opracowanie metody projektowania systemów informatycznych o architekturze SOA

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

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

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

[4] 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.

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

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

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

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

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