SMESDaD – synergetyczna metodyka rozwijania i wdrażania oprogramowania korporacyjnego

Grzegorz Rogus, Paweł Skrzyński, Piotr Szwed, Michal Turek, Jan Werewka

AGH Akademia Górniczo-Hutnicza, Wydział EAIiE, Katedra Automatyki

Streszczenie: W opracowaniu przedstawiono zagadnienie rozwoju, wytwarzania i wdrożenia oprogramowania korporacyjnego. W takim procesie ważne jest, aby współpraca pomiędzy przedsiębiorstwem rozwijającym oprogramowanie a przedsiębiorstwem, w którym to oprogramowanie było wdrażane, była bardzo efektywna. Na podstawie modeli korporacyjnych obu przedsiębiorstw można odwzorowywać model współpracy pomiędzy przedsiębiorstwami. W pracy przedstawiono w skrócie metodykę, która ma za zadanie rozwój oprogramowania korporacyjnego na bazie powiązań synergetycznych pomiędzy przedsiębiorstwami. Metodyka w zamierzeniu ma integrować najlepsze rozwiązania z dziedziny modelowania korporacyjnego, architektur zorientowanych na usługi oraz modeli wdrażania oprogramowania.

Słowa kluczowe: architektura korporacyjna, modelowanie, SOA, rozwój oprogramowania, wdrażanie oprogramowania

1. Wstęp

Tworzenie, wdrożenie i utrzymanie opogramowania korporacyjnego jest dużym wyzwaniem zarówno dla organizacji rozwijającej i wdrażającej oprogramowanie, jak i dla organizacji, która ma tego oprogramowania używać. Prowadzone badania dotyczyły poszukiwania najlepszych rozwiązań dla obydwu organizacji. Zapewnienie efektywnego współdziałania takich organizacji wymaga budowy szerszych i głębszych relacji wykraczających poza proste zasady współpracy między zleceniodawcą i zleceniobiorcą.

Celem opracowania jest przedstawienie aktualnych wyników i rozwiązań oraz propozycji prac badawczych prowadzących do stworzenia synergetycznej metodyki rozwijania i wdrażania oprogramowania korporacyjnego SMESDaD (ang. *Synergetic Methodology for Enterprise Software Development and Deployment*). Metodyka dotyczy funkcjonowania i współpracy dwóch organizacji (korporacji). Jedna z tych organizacji jest przedsiębiorstwem informatycznym dostarczającym oprogramowanie (SDE – *Software Delivering Enterprise*) dla głównej działalności gospodarczej drugiego przedsiębiorstwa działającego na rynku (MOE – *Market Operating Enterprise*).

W artykule będzie stosowane pojęcie „korporacja", które jest popularnym w literaturze polskiej odpowiednikiem angielskiego słowa „enterprise". Pojęcie korporacji jest dosyć obszerne i obejmuje: organizacje sektora publicznego i prywatnego, obszar działalności gospodarczej, część dużego przedsiębiorstwa, połączenie kilku jednostek (np. *joint venture*), lub nawet wiele oddziałów przedsiębiorstw, którym wydzierżawiono operacje gospodarcze. W opracowaniu będą stosowane zamiennie słowa korporacja, przedsiębiorstwo lub organizacja. Pojęcie korporacji dotyczy także całego złożonego systemu zawierającego ludzi, informacje, technologie i związanego z działalnością gospodarczą (biznesem). Zwarty opis definicji, modeli, rozwiązań architektonicznych korporacji zawarty jest przykładowo w publikacji [1].

Organizacje gospodarcze w swoim rozwoju napotykały na dwa podstawowe problemy [10]:

– złożoność systemów – organizacje wydawały coraz więcej środków na tworzenie systemów informatycznych,

– słabe dostosowanie biznesowe – organizacje napotykały na coraz większe trudności w dostosowaniu tych coraz bardziej kosztownych systemów informatycznych do potrzeb biznesowych.

Ocenia się, że aktualnie został osiągnięty punkt krytyczny, w którym złożoność systemów informatycznych wzrasta wykładniczo, natomiast szanse na dostarczenie rzeczywistej wartości maleją.

Zakłada się, że dla organizacji dostarczającej oprogramowanie SDE głównym celem jest związanie się z przedsiębiorstwem rynkowym MOE celem zwiększenia jego korzyści biznesowych. MOE uzyskując korzyści część zysków przekaże do SDE. Należy zauważyć, że nie chodzi o uzyskanie zwykłej relacji „zleceniodawca-dostawca", lecz wypracowanie działań synergetycznych i symbiotycznych pozwalających na zwiększenie wartości biznesowej takiej relacji.

Z uwagi na złożoność możliwych i celowych relacji powinna zostać opracowana metodyka rozwoju oprogramowania bazująca na współpracy takich organizacji. Znane są liczne opracowania dotyczące architektury korporacyjnej dlatego ich rozwiązania zostaną wciągnięte do rozwijanej metodyki. Nowością badawczą proponowanego rozwiązania jest uwzględnienie architektury korporacyjnej obu przedsiębiorstw w celu uzyskania dodatkowych korzyści biznesowych. Wynikające z metodyki korzyści biznesowe będą bazować na optymalnym wyborze rozwiązań dotyczących: struktur obu przedsiębiorstw zapewniających sprawność działania, optymalizacji procesów wynikających z analiz biznesowych i oceny dotychczasowych efektów działań, doboru technologii łatwych do implementacji i do pełnego wykorzystania, umiejętność dotarcia do docelowych klientów korzystających z oprogramowania oraz utrzymania ich zadowolenia.

2. Założenia synergetycznej metodyki rozwoju i wdrażania oprogramowania

Celem metodyki jest objęcie obszaru działalności obu przedsiębiorstw w celu uzyskania efektów synergetycznych i symbiotycznych. Metodyka ta, w odróżnieniu od czystych metodyk rozwoju oprogramowania, uwzględnia obszary:

- cele strategiczne (MOE – wyznaczenie wysokopoziomowych wymagań oraz wyznaczenie zasad realizacji tych wymagań na podstawie dostarczonego oprogramowania – SDE),
- cele taktyczne (MOE – określenie programów i projektów oraz opracowanie programów i projektów implementacyjnych oraz wdrożeniowych – SDE),
- rozwój oprogramowania (MOE – określenie jakie własności powinno mieć oprogramowanie i jak zbudować oprogramowanie – SDE),
- wdrażanie oprogramowania (MOE – przygotowanie organizacji do użycia oprogramowania i do sposobu wdrożenia – SDE),
- utrzymanie oprogramowania (MOE – efektywne użytkowanie oprogramowanie oraz zapewnienie jakości jego użytkowania – SDE),
- produkcja oprogramowania (MOE – zapewnienie dostaw i zapewnienie produkcyjne – SDE);
- integracja oprogramowania (MOE – wykorzystanie obcego gotowego oprogramowania oraz efektywna i prosta integracja – SDE).

Na powyższym wykazie widać wspólne obszary obu organizacji, jednak, co zostało zaznaczone, każda z organizacji widzi te obszary z trochę innej perspektywy. Ponadto zakłada się, że dla wszystkich wymienionych obszarów współdziałanie organizacji powinno pozwolić uzyskać odpowiednią sprawność. Sprawność jest rozumiana jako własność polegająca na spełnieniu w odpowiednim stopniu następujących kluczowych parametrów:

- zwinność (ang. *agility*) – łatwa adaptowalność zmian,
- skalowalność (ang. *scalability*) – wykorzystanie zasobów proporcjonalne do obciążenia,
- ponowne użycie komponentów (ang. *reusable components*) – komponenty mogą pochodzić od różnych dostawców, przy czym istnieje możliwość wyboru komponentu dla zadanego typu usługi,
- wydajność (ang. *efficiency*) – efektywne wykorzystanie zasobów, w tym także spełnienia warunków czasu rzeczywistego (np. EDA),
- lekkość (ang. *lightness*) umożliwiająca redukcję działań i funkcjonalności tylko do potrzeb klienta,
- elastyczność (ang. *flexibility*) polegająca na doborze rozwiązań odpowiednich dla istniejących potrzeb.

3. Zagadnienie opisu architektury korporacyjnej

Celem metodyki jest opracowanie, implementacja, wdrożenie i utrzymanie architektury korporacyjnej dla obu przedsiębiorstw, z uwzględnieniem efektu synergii wynikającego z kooperacji. Model architektoniczny powinien być tak dobrany, by w ramach tego modelu istniała możliwość dokonania analizy rozwiązań w celu optymalizacji wyboru.

Architektura korporacyjna (AK) definiowana jest jako formalny opis:

- struktury i funkcji komponentów organizacji,
- relacji między nimi,
- wytycznych w zakresie zarządzania projektowaniem i zmianą tych komponentów w czasie.

Celem architektury korporacyjnej jest zbalansowanie jakości reakcji na zmiany w otoczeniu (cel: zwinność) w stosunku do spójności rozwiązań wynikających z tych zmian (cel: spójność). Równoważenie zmian wynikających z potrzeby reakcji na otoczenie (zwinność) i spójności obszarów i warstw w przedsiębiorstwie to zadanie, ale i wyzwanie dla architektury korporacyjnej. Jakość architektury w ujęciu architektury korporacyjnej powinna dotyczyć trzech warstw (biznesowej, aplikacji i infrastruktury). Wybiórcze spojrzenie na architekturę przedsiębiorstwa poprzez monitorowanie tylko jej wycinka nie spowoduje, że mierząc jakość architektury będziemy w stanie wyciągać wnioski, które następnie będzie można uznać za obiektywne.

W każdej z warstw wyróżnić należy trzy obszary – informacje, zachowania i struktury (rys. 1). W każdym z obszarów istotne są inne elementy architektury, i to te elementy powinny być poddane monitoringowi.

AK uwzględnia opis procesów biznesowych organizacji, jej danych i aplikacji, wykorzystywanych technologii oraz systemów IT. Podejście AK uwzględnia strategię organizacji, która wspiera stan obecny oraz definiuje sposób i środki osiągnięcia stanu docelowego

Ponieważ AK opisuje procesy biznesowe, dane i aplikacje, technologie oraz systemy IT, zazwyczaj dzielona jest na cztery podstawowe modele:

- Model biznesowy. Architektura Biznesowa to sposób funkcjonowania korporacji, a więc struktura organizacyjna, strategia biznesowa i kluczowe procesy biznesowe.
  - opisuje misję i cele organizacji,
  - dostarcza funkcjonalny widok organizacji.
- Model danych. Architektura Danych to typy i źródła oraz cykl życia danych niezbędnych do funkcjonowania korporacji. Zauważmy, że to właśnie dane są najważniejsze. Aplikacje mają za zadanie jedynie wspomagać zbieranie i przetwarzanie oraz przeszukiwanie tych danych. To jakie to są dane i w jaki sposób są przetwarzane w dużym stopniu determinuje sposób konstrukcji oprogramowania, które te dane ma za zadanie przetwarzać. Architektura Danych jest łącznikiem pomiędzy Architekturą Biznesu i Architekturą IT.
  - opisuje strukturę danych, ich właściwości i wzajemne powiązania.
- Model aplikacji. Architektura Aplikacji to znaczna część tego co można rozumieć pod pojęciem Architektury IT. Jest to sposób organizacji oprogramowania, tj. podział na poszczególne aplikacje i integracja tych aplikacji tak aby wspólnie zapewniały zaplanowane przetwarzanie danych i wspierały istniejące procesy biznesowe.
  - opisuje oprogramowanie używane w organizacji.

Rys. 1. Obszary modelowanie przedsiębiorstwa w architekturze korporacyjnej
Fig. 1. Enterprise architecture modeling

– Model techniczny. Architektura Infrastruktury IT to rozwiązania techniczne zapewniające funkcjonowanie oprogramowania zgodne z przyjętymi założeniami i istniejącymi możliwościami.

– opisuje architekturę systemowo-sprzętową.

Budowa modelu architektury biznesowej w dużym uproszczeniu polega na następujących etapach:
– określenie strategicznych celów organizacji,
– budowa modelu biznesu na wysokim poziomie abstrakcji,
– znalezienie ram strukturalnych organizacji,
– budowa modelu procesu biznesowego.

Powyższe punkty zasadniczo nie odbiegają od standardowej ścieżki budowy modeli procesów biznesowych, a czasem wymagań systemowych. Różnica polega na tym, że architektura korporacyjna swoim zasięgiem obejmuje szerszy kontekst działania organizacji niż tradycyjny model biznesowy:
– architektura korporacyjna pozwala na uporządkowanie portfela posiadanych systemów informatycznych i rozwiązań technologicznych,
– znajomość aktualnych zasobów IT i procesów biznesowych z nich korzystających (*as-is*) pozwala zaplanować rozwiązania docelowe (*as-will*),
– dzięki architekturze korporacyjnej łatwiej jest przedstawić działanie poszczególnych systemów informatycznych, co przekłada się na funkcjonowanie procesów biznesowych,
– architektura korporacyjna ułatwia rozbudowę i optymalizację funkcjonalności informatycznych, gdyż przedstawia aplikacje, systemy i platformy, które są powiązane złożoną siecią wzajemnych zależności.

Proponowana metodyka SMESDaD zakłada tworzenie architektury korporacyjnej oddzielnie dla SDE przedsiębiorstwa informatycznego dostarczającego oprogramowanie oraz dla MOE przedsiębiorstwa działającego na rynku. Dla obu modeli należy zdefiniować odwzorowania i powiązania, a następnie utworzyć model architektury współpracy, który powstanie poprzez uwzględnienie tylko części wspólnej modeli wyjściowych.

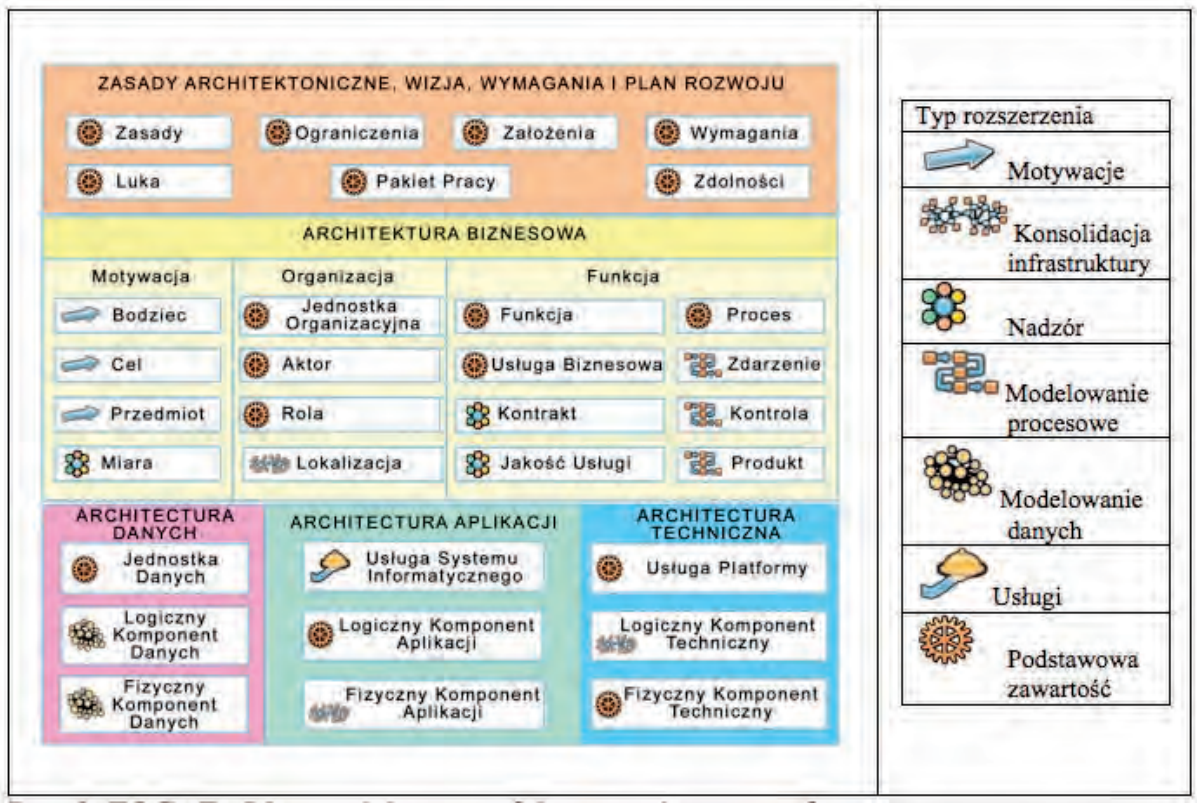
Przy analizie rozwiązań modeli architektury korporacyjnej skoncentrowano się na najpopularniejszych modelach architektonicznych:
– siatka Zachmana,
– TOGAF (The Open Group Architecture Framework),
– FEAF (Federal Enterprise Architecture Framework),
– The Gartner Methodology,
– DoDAF (Department of Defense Architecture Framework).

W wyniku analiz metod opisu architektur korporacyjnych przyjęto, że TOGAF wystarczająco dobrze nadaje się do zastosowania w metodyce SMESDaD.

TOGAF [5] ma na celu dostarczenie praktycznej, łatwo dostępnej standaryzowanej i przemysłowej metody projektowania architektury przedsiębiorstwa. TOGAF jest wystarczającym narzędziem do organizacji lub adaptacji metody rozwoju architektury przedsiębiorstwa, która może być stosowana także z innymi strukturami. TOGAF skupia się na kluczowych aplikacjach biznesowych, które używają otwartych bloków budujących. Struktura wciela w życie koncept kontynuacji architektury przedsiębiorstwa celem odzwierciedlenia poszczególnych poziomów abstrakcji w procesie architektury przedsiębiorstwa. TOGAF wprowadza kontekst do użycia w wielu strukturach, modelach i zasobach architektury w połączeniu z metodą rozbudowy architektury TOGAF.

TOGAF przedstawia ogólną architekturę korporacyjną w postaci metamodelu zawartości, przedstawionego na rys. 2. Metamodel pomaga definiować artefakty powstające w trakcie procesu rozwoju oprogramowania. TOGAF wprowadza pojęcie rozszerzenia, które może być użyte do opisu dodatkowych jednostek metamodelu, które powinny być rozważone przy rozwoju architektury.

Dzięki wykorzystaniu podejścia AK możliwe jest:
– wspieranie działań biznesowych organizacji poprzez uzgodnienie podstawowych technologii i struktury procesów dla strategii IT,
– uzyskanie równowagi między wydajnością systemów IT a innowacyjnością,
– bezpieczne wdrażanie innowacji w organizacji w celu osiągnięcia konkurencyjnej przewagi,



Rys. 2. TOGAF – metamodel zawartości z typami rozszerzeń
Fig. 2. TOGAF – metamodel with extension types

– uzyskanie maksymalnej synergii w rozwijanej (rozszerzanej) organizacji przy jednoczesnej realizacji wymagań strategii integracji IT.

4. Tworzenie architektury korporacyjnej w oparciu o paradygmat SOA

Tworzenie architektury korporacyjnej jest procesem iteracyjnym i dlatego należy się zdecydować na metodykę rozwoju architektury oraz wzorce projektowe, które należy zastosować. W metodyce SMESDaD postanowiono w tym celu użyć metodę rozwoju architektury ADM (Architecture Development Method – rys. 3) zawartą w TOGAF [3]. Metoda ta złożona jest z kilku kroków kojarzących się z cyklem rozwoju oprogramowania, tyle że tylko część etapów związana jest bezpośrednio z oprogramowaniem. Pierwsza faza metody ADM to w wielkim skrócie ustalenie kontekstu biznesowego i konsolidacja wokół inicjatywy wszystkich kluczowych interesariuszy. W fazie drugiej tworzona jest wizja korporacji, do której dążymy. Po tym etapie ma być wiadomym, co trzeba zmienić w firmie, nie tylko w IT. Dalsze etapy to właśnie rzutowanie zmian w biznesie na zmiany w IT, a więc opracowanie niezbędnego zakresu zmian Architektury Danych, Architektury Aplikacji i Architektury Infrastruktury IT oraz implementacja i wdrożenie tych zmian. Ten fragment pracy związany jest już *stricte* z rozwojem oprogramowania.

Efektywna implementacja architektury korporacyjnej powinna brać pod uwagę paradygmat SOA, w którym funkcjonalność systemu opisana jest pojęciem usługi w sposób niezależny od systemu operacyjnego czy języka programowania. Zaletą takiego podejścia jest odejście od praktyk „izolacjonistycznych", w których każdy departament w korporacji buduje swój system, często duplikując funkcjonalność, co podnosi koszty, zwiększa czas rozwoju i utrudnia utrzymanie. W paradygmacie SOA system składa się z luźno powiązanych ze sobą usług, które stanowią komponenty wielokrotnego wykorzystania i mogą występować w różnych kompozycjach. Cechami charakterystycznymi oprogramowania rozwijanego w oparciu o SOA są [13]:

Rys. 3. Metodyka rozwoju architektury (ADM) w TOGAF
Fig. 3. Enterprise architecture development in TOGAF

– możliwość wywoływania usług niezależnie od technologii i lokalizacji sieciowej,

– komunikacja jeden do jeden: w danej chwili czasowej konsument usługi (ang. *service consumer*) może wywołać tylko jedną usługę, przy czym komunikacja jest dwukierunkowa,

– przepływ inicjowany jest przez konsumenta usługi – wywołanie usługi,

– komunikacja synchroniczna.

Innym paradygmatem jest EDA (ang. *Event Driven Architecture*), w którym aplikacje i usługi wymieniają ze sobą w sposób asynchroniczny zdarzenia (ang. *events*). Z racji tego, że komunikacja pomiędzy komponentami przebiega w sposób asynchroniczny, EDA nazywane jest często „SOA zorientowanym na zdarzenia", przy czym paradygmat ten stanowi raczej uzupełnienie SOA, a nie jego konkurencję. Cechami charakterystycznymi architektury EDA są:

– komunikacja, w której źródła zdarzeń nie są świadome istnienia odbiorców zdarzeń,

– komunikacja wiele do wiele: jedno zdarzenie może być konsumowane przez wielu subskrybentów,

– przepływ inicjowany przez odbiorcę w zależności od typu zdarzenia,

– komunikacja asynchroniczna.

Co ciekawe, z punktu widzenia implementacyjnego obydwie architektury mogą wykorzystywać ten sam wzorzec projektowy – szynę korporacyjną ESB (ang. *Enterprise Service Bus*). Wzorzec ten integruje oba podejścia, co znacząco ułatwia integrację modułów biznesowych i różnych platform. ESB pełni rolę warstwy pośredniej, która umożliwia komunikację między różnymi procesami. Każda usługa, która jest wdrożona w ramach szyny może być inicjowana poprzez konsumenta lub zdarzenie. ESB dostarcza wszystkich możliwości określonych przez obydwa paradygmaty i z racji tego jest zalecanym sposobem implementacji architektury korporacyjnej w ramach niniejszego podejścia.

Jednym z pierwszych kroków podczas projektowania architektury zorientowanej na usługi jest identyfikacja usług. Żeby proces ten mógł zostać przeprowadzony, potrzebna jest znajomość dokładnego opisu procesów biznesowych, które działają w przedsiębiorstwie. Proces analizy jest procesem iteracyjnym (więcej o samym procesie w [15]), a jego celem jest zidentyfikowanie planów inwentarzy usług (ang. *service inventories*) – jest to pojęcie szersze od kompozycji usług – inwentarz taki stanowi spójny, zestandaryzowany, o jasno określonych granicach zbiór uzupełniających się usług reprezentujący przedsiębiorstwo lub jego znaczącą część. W przypadku gdy przedsiębiorstwo ma wiele inwentarzy usług, w celu ich odróżnienia wprowadza się jeszcze pojęcie domenowego inwentarza usług (ang. *domain service inventory*).

Charakterystyczną cechą tego procesu jest to, iż biorą w nim udział zarówno przedstawiciele biznesowi przedsiębiorstwa (którzy mają wiedzę dotyczącą jego funkcjonowania), jak i technicy odpowiedzialni później za projektowanie i implementację architektury zorientowanej na usługi. W związku z tym w metodyce SMESDaD jako punkt startowy przyjmuje się stworzenie modelu biznesowego działania przedsiębiorstwa przy pomocy notacji BPMN.

Notacja BPMN (ang. *Business Process Modelling Notation*) zyskuje obecnie popularność i jest już szeroko wykorzystywana do opisu procesów biznesowych. Można ją także traktować – co jednak nie jest do końca poprawne – jako dodatkową warstwę dostarczającą mechanizmów wizualizacji dla języków wykonywalnych, takich jak BPEL (ang. *Business Process Execution Language*). Jedną z zalet BPMN oraz BPEL jest właśnie zgodność z paradygmatem SOA.

Cechą wyróżniającą oprogramowanie korporacyjne jest to, iż każdy program (komponent) jest osadzony w ramach infrastruktury technologicznej (ang. *technology infrastructure*), na którą składa się oprogramowanie (systemy operacyjne, API systemowe, bazy danych i katalogi (np. LDAP), programy zarządzające transakcjami, kolejki komunikatów, oprogramowanie warstwy pośredniej – middleware i adaptery, programy zarządzające użytkownikami, technologie zapewniające bezpieczeństwo etc.) oraz sprzęt (serwery, routery i inne narzędzia sprzętowe, zapory ogniowe – firewall, systemy storage, okablowanie etc.). Oprogramowanie takie ma na celu wspomagać działanie przedsiębiorstwa na różnych polach.

Cechą odróżniającą oprogramowanie będące częścią infrastruktury technologicznej, czyli tzw. oprogramowanie korporacyjne, od oprogramowania standalone (np. edytora tekstu MS Word) jest to, iż każdy komponent w ramach tej infrastruktury jest dostępny dla różnych innych aplikacji/komponentów oraz systemów, przez co staje się dla korporacji zasobem (ang. *resource*). Paradygmat SOA reprezentuje model architektury, której celem jest zwiększenie możliwości reakcji na zmiany infrastruktury technologicznej przedsiębiorstwa (dodawanie/zmiana komponentów programowych) przy jednoczesnej redukcji kosztów wprowadzenia tych zmian. Architektura zorientowana na usługi ma umożliwić łatwą integrację różnego rodzaju działalności biznesowej oraz zapewnić interoperabilność heterogenicznych systemów. Cele te osiągnięte są poprzez wprowadzenie abstrakcyjnego pojęcia „usługa" (ang. *service*), używanego następnie w opisie logiki rozwiązania problemu – pojedyncza usługa enkapsuluje w sobie logikę biznesową, która stanowi część rozwiązania problemu.

Z punktu widzenia implementacyjnego SOA stanowi kombinację technologii, produktów, specyficznych API etc. Praktyczne wdrożenie takiej architektury jest specyficzne dla danej korporacji, przy czym częścią wspólną może być wykorzystanie platform (np. Java EE) ułatwiających tworzenie oprogramowania w oparciu o taką architekturę poprzez zapewnienie możliwości tworzenia, wykonania i ewolucji rozwiązań zorientowanych na usługi (ang. *service-oriented solutions*).

Na usługę można patrzyć jako na zbiór funkcjonalności w ramach określonego kontekstu, służący rozwiązaniu określonego problemu. Na poziome koncepcji nie ma żadnych założeń co do sposobu implementacji usługi. Nie należy przy tym mylić pojęcia usługi z pojęciem usługi sieciowej (ang. *web service*), która może być jednym ze sposobów implementacji usługi, aczkolwiek nie jedynym.

Termin „kompozycja usług" (ang. *service composition*) opisuje zbiór usług, które pozwalają na automatyzację jakiegoś zadania lub procesu biznesowego. Należy zaznaczyć, że termin ten odnosi się do agregatów, które zawierają przynajmniej dwie usługi oraz coś, co nazywane jest inicjatorem kompozycji (ang. *composition initiator*) – w innym przypadku mówimy o komunikacji typu punkt-punkt (ang. *point-to-point exchange*). Jednym z głównych założeń paradygmatu projektowania oprogramowania zorientowanego na usługi jest takie przygotowanie usług, by umożliwić im efektywne działanie w ramach różnych, złożonych kompozycji.

5. Przygotowanie do rozwoju architektury korporacyjnej w oparciu o paradygmat SOA

Przygotowanie nawet najlepszej architektury nic nie da, jeżeli nie będzie organizacji potrafiącej rozwijać, implementować i wdrażać oprogramowanie zgodne z tą architekturą. Dlatego należy również określić stopień przygotowania organizacji do takiego rozwoju oraz określić kroki w kierunku umożliwienia takiego rozwoju.

Tworzenie rozwiązań bazujących na SOA wymaga uzyskania odpowiedniej dojrzałości zarówno przez przedsiębiorstwo informatyczne dostarczające oprogramowanie (SDE), jak i przedsiębiorstwo działającego na rynku (MOE).

Dla firm informatycznych tworzone były modele dojrzałości, z których najbardziej znany jest model CMMI (ang. *Capability Maturity Model for Integration*) [7] zaproponowany przez Software Engineering Institute.

W przypadku stosowania SOA można zastosować model OSIMM (ang. *The Open Group Service Integration Maturity Model* [2]), który umożliwia ocenę dojrzałości organizacji w użyciu usług (rys. 4). Jest to model jakościowy w postaci macierzy dwuwymiarowej. Kolumny przedstawiają siedem poziomów dojrzałości, natomiast wiersze dotyczą siedmiu obszarów adaptacji usług. W modelu zawarte są zagadnienia dotyczące rozwoju oprogramowania (dotyczące bardziej SDE) oraz działalności biznesowej (bardziej dotyczących MOE).

- obszar biznesu dotyczy praktyk i polityk gospodarczych, rozważa koszt i elastyczność możliwości IT, zwinność biznesu i umów określających gwarantowany poziom usług,
- obszar organizacji i nadzoru dotyczy struktury i projektu organizacji, metryki wydajności organizacji w otoczeniu SOA,
- obszar metod dotyczy metod i procesów stosowanych przez organizację, dojrzałości cyklu rozwoju oprogramowania, praktyk inżynierskich w zakresie wytwarzania oprogramowania stosowanych przy rozwoju SOA,
- obszar aplikacji dotyczy stylu aplikacji, struktury i funkcjonalności obejmującej parametry ponownego użycia, elastyczności, niezawodności oraz rozszerzalności,
- obszar architektury dotyczy struktury stylu architektonicznego, decyzji dotyczącej architektury korporacyjnej, standardów i polityk, doświadczenia w implementacji serwisów, kryteriów zgodności z SOA itp.,
- obszar informacji dotyczy struktury informacji, metody dostępu do danych korporacji, modelu biznesowego informacji i zarządzaniu zawartością, abstrakcji danych, bezpieczeństwa danych itp.,
- Infrastruktura: dotyczy własności infrastruktury, jak zarządzanie usługami i operacjami IT, zarzadzanie i administracja IT, zgodność z SLA, oraz monitorowania i typów dostarczonych platform integracyjnych.

Model powinien zostać zastosowany do wyznaczenia strategii adaptacji SOA. Ocena poziomu dojrzałości określa informacje potrzebne do wyznaczenia atrybutów dojrzałości. Transformacja do SOA daje korzyści związane z efektywnością kosztową, zwinnością, adaptacyjnością.

6. Identyfikacja i formalizacja opisu elementów architektury korporacyjnej

W wyniku wyboru metodyki opisu architektury korporacyjnej uzyskaliśmy pewien szkielet zawierający nieformalny opis, który powinniśmy uzupełnić o opis formalny. Zdecydowano się na wybór języka Archimate [6] i BPMN do formalnego opisu elementów architektury korporacyjnej.

6.1. Archimate

W przedsiębiorstwach funkcjonują architektury dla różnych dziedzin, takich jak struktura organizacji, procesów biznesowych, aplikacji, danych i architektury technicznej. Każda dziedzina architektury ma własne koncepcje modelowania i wizualizacji wewnętrznej spójności. Te konkretne modele i wizualizacje uproszczają komunikację, refleksję i analizę dziedziny.

Jednak relacje między pojęciami w tych różnych dziedzinach w wielu przypadkach są niejasne. Co więcej, dziedziny częściowo się pokrywają, a co gorsza, używają różnych pojęć w celu wyrażenia tych samych idei, czasem nawet bez wiedzy osób zaangażowanych w proces. Języki takie jak UML, ARIS, IDEF czy BPMN nie są wystarczająco uniwersalne, aby sprostać zadaniu opisu architektury przedsiębiorstwa.

UML jest postrzegany jako język, który podczas kilkunastoletniej ewolucji stał się kolekcją różnego typu diagramów, proponowanych przez środowiska akademickie lub przemysłowe jako użyteczne do opisu wybranego aspektu rozwijanego systemu. Konsekwencją tego jest brak spójnego metamodelu, który zdefiniowałby to, w jaki sposób elementy poszczególnych warstw opisu powinny być ze sobą powiązane. W szczególności, za pomocą UML nie można zobrazować powiązania procesu biznesowego z usługą aplikacyjną (usługą IT), natomiast jest to możliwe przy użyciu języka Archimate [14].

Definicji języka Archimate towarzyszyło założenie, że aby zbudować ekspresywny model biznesowy, konieczne jest wykorzystanie relacji łączących całkowicie odmienne dziedziny: od motywacji biznesowych do procesów biznesowych, usług i infrastruktury. Archimate prowadzi tam, gdzie UML nie sięga: definiuje metamodel, na podstawie którego można tworzyć i obrazować relacje pomiędzy elementami różnych warstw. Wyspecyfikowane w metamodelu więzy i ograniczenia pozwalają na analizę, sprawdzanie spójności, identyfikację i śledzenie powiązań oraz zarządzanie. UML nie narzuca powiązania pomiędzy modelami, natomiast Archimate tak.

Poziomy dojrzałości — Obszary adaptacji usług	Poziom 1 Silos	Poziom 2 Zintegrowany	Poziom 3 Komponentowy	Poziom 4 Proste usługi	Poziom 5 Kompozy-towe usługi	Poziom 6 Wirtualne usługi	Poziom 7 Dynamicznie rekonfiguro-walne usługi
Biznes	Izolowany biznes - liniowy	Integracja procesów biznesowych	Procesy komponentowe	Oferowanie usług przez komponentow y biznes	Procesy poprzez kompozytowe usługi	Geograficznie niezależne centra usług	Dobieranie biznesu i możliwości w świadomości otoczenia
Organizacja i nadzór	Doraźna strategia i nadzór dla LOB i IT	Doraźna strategia IT i nadzór dla korporacji	Wspólne procesy nadzoru	Początkowa faza nadzoru SOA	Dostosowanie nadzoru SOA i IT	Dostosowanie infrastruktury nadzoru SOA i IT	Nadzór poprzez określoną politykę
Metody	Analiza i projektowanie strukturalne	Modelowanie zorientowane obiektowo	Rozwój bazujący na komponentach	Modelowanie zorientowane na usługi	Modelowanie zorientowane na usługi	Modelowanie zorientowane na usługi dla Infra (CSDP)	Modelowanie zorientowane na gramatyki biznesowe
Aplikacje	Moduły	Obiekty	Komponenty	Usługi	Integracja procesów poprzez usługi	Integracja procesów poprzez usługi	Dynamiczny montaż - świadome otoczenia wywołania
Architektura	Architektura monolityczna	Architektura warstwowa	Architektura komponentowa	Początkowa faza SOA	SOA	Grid-enabled SOA	Dynamicznie rekonfigu-rowalna architektura
Informacja	Specyficzna dla aplikacji	Specyficzna dla LOB lub korporacji	Modele kanoniczne	Informacja jako usługa	Słownik i repozytorium danych biznesowych	Wirtualne usługi danych	Semantyczne słowniki danych
Infrastruktura	Specyficzna platforma dla LOB	Standardy korporacyjne	Wspólna re-używalna infrastruktura	Projektowo zorientowane środowisko SOA	Wspólne środowisko SOA	Wirtualne środowisko SOA: S&R	Dynamiczne odczuwanie, decydowanie i odpowiadanie

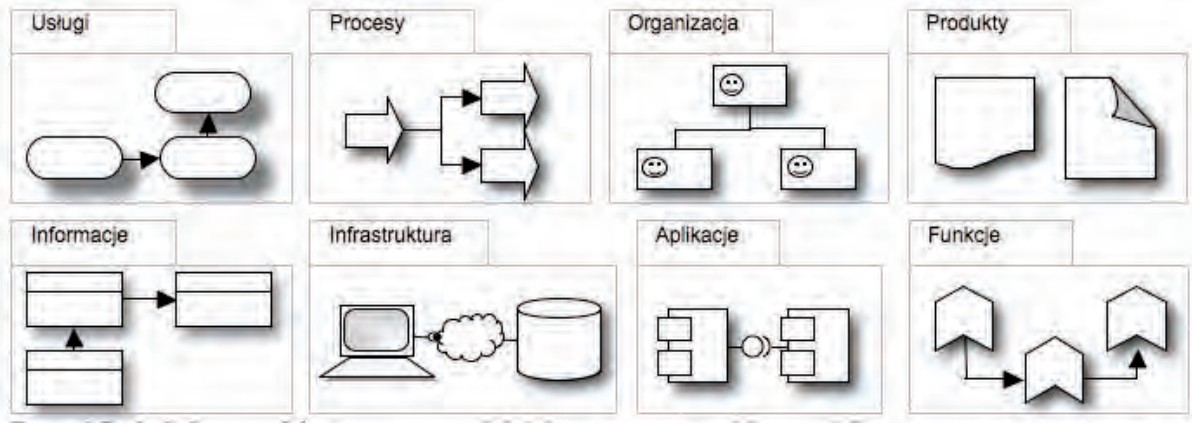
Rys. 4. OSIMM macierz dojrzałości SOA
Fig. 4. The Service Integration Maturity Model

służące ułatwieniu badania i opisywania relacji pomiędzy elementami, których szczegóły można opisywać właśnie przy użyciu w/w języków i notacji. Archimate dostarcza pełnego zestawu koncepcji opisu relacji w stosunkach między dziedzinami architektury, jednocześnie dając w wyniku prostą i jednolitą strukturę. Założeniem podstawowym języka jest to, że usługi odgrywają kluczową rolę w relacjach między domenami.

Architektura zbudowana z wykorzystaniem usług sieciowych (ang. Service Oriented Architecture SOA) może być wykorzystywana równie dobrze do budowy usług komercyjnych dla klientów organizacji, jak również dla usług wewnętrznych, wpierających procesy biznesowe, w których roli klienta nie odgrywa rzeczywisty klient organizacji. Koncepcja SOA jest naturalnym celem do budowania architektury z wykorzystaniem Archmate dlatego, że warstwowy widok modelu stanowi naturalny sposób patrzenia na modele zorientowane na usługi. Wyższe warstwy, korzystają z usług świadczonych przez warstwy niższe. Archimate wyróżnia trzy główne warstwy:

– warstwę biznesową, która oferuje produkty i usługi na rzecz klientów zewnętrznych lub wewnętrznych, które są realizowane w organizacji w ramach procesów biznesowych, przez aktorów i ich role,

– warstwę aplikacji, składa się ze zbioru aplikacji, które są realizowane przez oprogramowanie udostępniające usługi obsługujące warstwę biznesową,



Rys. 5. Dziedziny architektury przedsiębiorstwa w Archimate [6]
Fig. 5. Enterprise architecture domains in Archimate

realizowane w organizacji w ramach procesów biznesowych przez aktorów i ich role,

– warstwę aplikacji, która składa się ze zbioru aplikacji realizowanych przez oprogramowanie udostępniające usługi obsługujące warstwę biznesową,

– warstwę technologii, udostępniającą usługi w zakresie infrastruktury (np. przetwarzania, przechowywania i przesyłania) potrzebne do uruchomienia aplikacji realizowanych za pomocą komputerów, infrastruktury sieciowej i oprogramowania systemowego.

Usługi warstw oferowane są przez interfejsy, które stanowią opis sposobu udostępniania usług przez elementy aktywne każdej warstwy (komponent, rola biznesowa, urządzenie itd.). Elementy zachowania zarządzają danymi (informacjami) trzech obszarów zainteresowania architektury korporacyjnej.

6.2. BPMN

BPMN jest standardową notacją graficzną przeznaczoną do opisu procesów biznesowych dostarczoną przez Business Process Managament Initiative (BPMI), która została zaakceptowana przez OMG. BPMI współpracuje z organizacją OASIS [4] w zakresie opracowania standardów dla e-biznesu. Organizacja ta jest ogólnoświatowym konsorcjum non-profit zajmującym się opracowywaniem, łączeniem, i przyjmowanie standardów dla e-biznesu. Jej działania obejmują tworzenie standardów bezpieczeństwa, usług WWW, zgodności z XML, transakcji biznesowych, publikacji elektronicznych, map tematycznych oraz zasad bezproblemowej współpracy wewnątrz oraz pomiędzy rynkami zbytu.

Zasadniczym celem powstania BPMN było dostarczenie notacji, która byłaby łatwa i zrozumiała dla analityków biznesowych, którzy znają i tworzą procesy biznesowe, oraz specjalistów technicznych, którzy odpowiadają za ich implementację i automatyzację, a także kierowników i menedżerów, którzy zarządzają tymi procesami i je monitorują.

Kolejnym celem było zapewnienie możliwości wizualizacji językom opartym na XML, które umożliwiają wykonanie procesów – takich jak BPEL4WS (ang. *Business Process Execution Language for Web Services*). Zanim podjęto próbę standaryzacji notacji dla takich potrzeb, zauważono, że analitycy biznesowi mają tendencję do posługiwania się różnego rodzajami grafami przepływu (ang. *flowcharts*). Z drugiej strony notacja musiała brać pod uwagę rosnącą popularność języków opartych na usługach sieciowych i XML, które umożliwiały automatyczne wykonanie procesów. Języki te były zgodne z pardygmatem strukturalnym, natomiast BPMN musiał mieć równocześnie korzenie w diagramach grafowych, co było trudne do pogodzenia. Języki takie jak BPEL są zoptymalizowane pod kątem interoperacyjności systemów zarządzania procesami (BPM, ang. *Business Process Management Systems*), co czyni je trudnymi do zrozumienia przez osoby niebędące specjalistami. W związku z tym pojawił się rozdźwięk pomiędzy tym, co było używane przez analityków biznesowych do opisu procesów (nieformalne notacje grafowe) oraz językami używanymi przez techników do implementacji systemów. Notacja BPMN stanowi nie tylko próbę zasypania tej luki, ale uwzględnia też ludzki poziom interoperabilności, który nie był uwzględniany przez języki typu BPEL, nastawione na interoperabilność systemów informatycznych.

Wymienione wyżej cechy sprawiają, iż BPMN został zaadoptowany do metodyki SMESDaD jako pierwszy krok procesu opisywanego przez metodykę. Dodatkowo w celu podniesienia jakości stworzonego modelu BPM postuluje się przed przystąpieniem do opisu procesów biznesowych w notacji BPMN dokonanie próby stworzenia specyfikacji tekstowej tych procesów. Opis taki dostarcza informacji poglądowej o procesie i stanowi również cenną dokumentację. Wadą opisów tekstowych jest brak formalizmu, który uniemożliwia ich późniejszą automatyzację. Opis tekstowy powinien zawierać następujące informacje:

– cel i krótka charakterystyka procesu,

– wymagane zasoby,

– wytwarzane dokumenty,

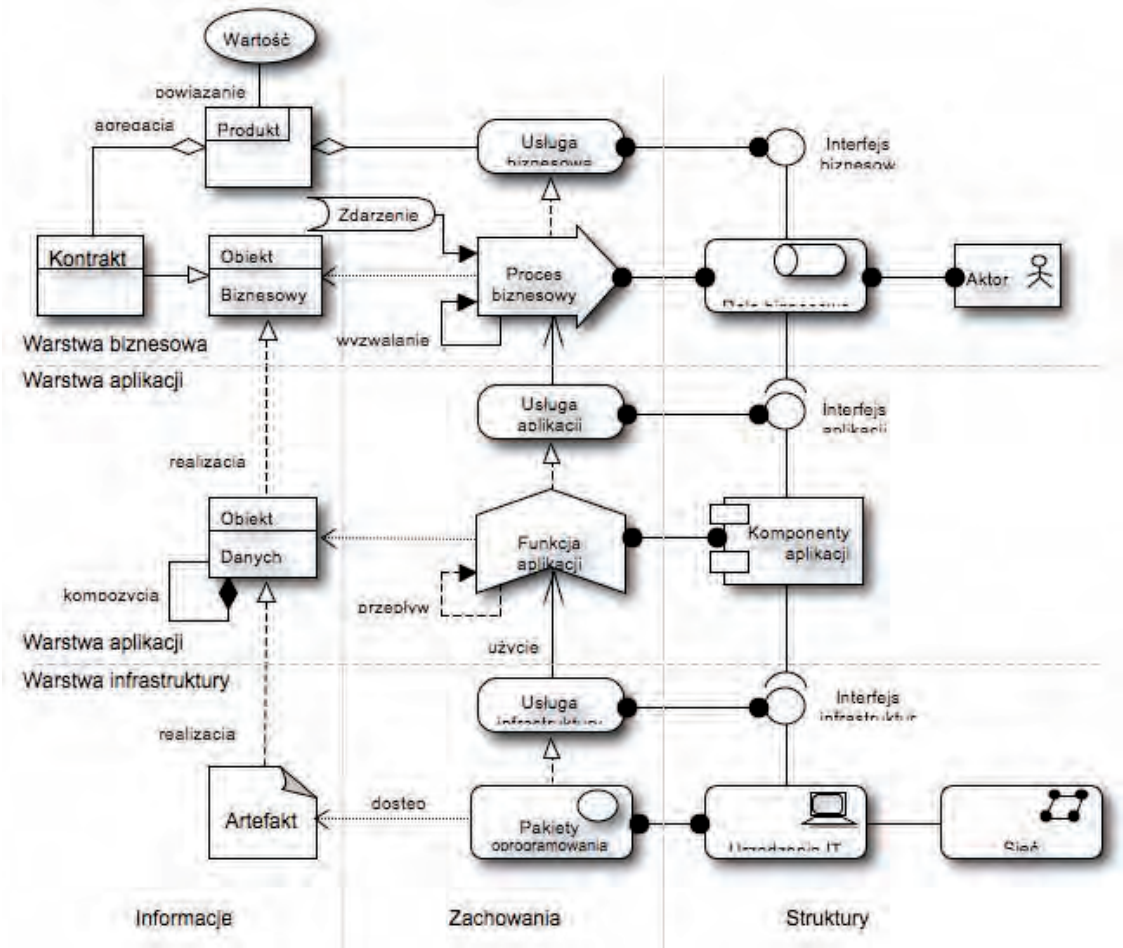
– warunki początkowe niezbędne do uruchomienia procesu,

– warunki końcowe,

– główny ciąg zdarzeń,

– systemy, osoby niezbędne do prawidłowego przebiegu procesu (w tym osoby oraz systemy, z którymi proces się komunikuje).

Zebranie tych informacji od osób dysponujących odpowiednią wiedzą biznesową nie jest trudne, a wydatnie



Rys. 6. Dziedziny architektury przedsiębiorstwa w Archimate
Fig. 6. Enterprise architecture domains in Archimate

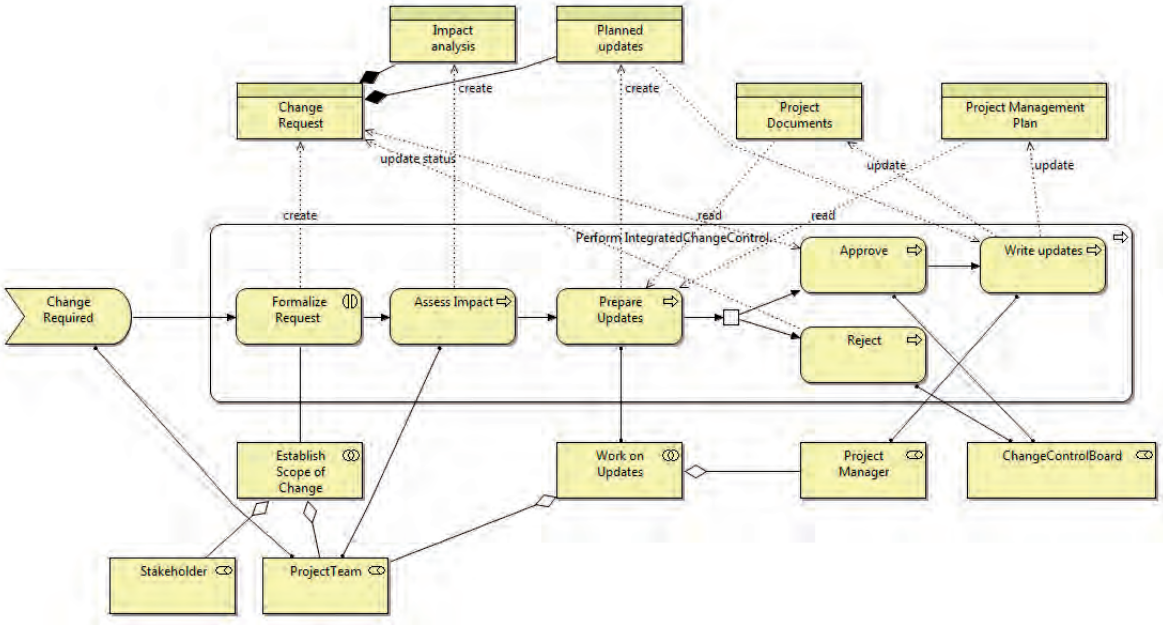
ułatwia stworzenie formalnego opisu procesu w notacji BPMN – jest to kolejny krok w łańcuchu wytwarzania oprogramowania wprowadzonym przez metodykę SMESDaD. Proces modelowania BPMN otrzymuje na wejściu specyfikację tekstową oraz inne specyfikacje, np. w postaci WSDL'i opisujących usługi sieciowe wystawiane przez systemy, z którymi proces się komunikuje. Proces tworzenia modelu w notacji BPMN proponuje się przeprowadzić w dwóch fazach:

– W pierwszej fazie tworzony jest wstępny diagram/diagramy będące odzwierciedleniem opisu tekstowego. Diagram na wyjściu tej fazy powinien zawierać tylko podstawowe symbole BPMN (tzw. *Core BPMN Elements*), dzięki czemu jest on łatwy w zrozumieniu przez analityków biznesowych, którzy powinni dokonać weryfikacji stworzonego modelu.

– W drugiej fazie diagramy BPMN są uszczegóławiane poprzez zastosowanie rozszerzonych elementów notacji BPMN (np. zdarzenia startowe pośrednie, końcowe, typy aktywności etc.) oraz uwzględnienie dodatkowych informacji (np. WSDL). Diagramy powstałe w tej fazie mają w pełni wykorzystywać możliwości ekspresji, jakie daje BPMN, w związku z tym są one w pełni zrozumiałe już tylko przez ekspertów biznesowych. Na tym etapie możliwa jest również formalna analiza i weryfikacja stworzonego modelu – np. poprzez zastosowanie logik temporalnych i wnioskowania dedukcyjnego. Wspomniane metody stanowią istotny obszar badawczy osób zaangażowanych w stworzenie metodyki SMESDaD.

Stworzone w drugiej fazie modelowania diagramy BPMN umożliwiają automatyczne wygenerowanie opisu procesów w języku wykonywalnym, jakim jest BPEL. Proces ten może, lecz nie musi przebiegać w sposób automatyczny ze względu na fakt, iż model BPMN jest semantycznie bogatszy od możliwości jakie stwarza BPEL oraz ze względu na fakt, iż nie jest to model w pełni formalny (w odróżnieniu od BPEL). Kolejnym obszarem badawczym twórców metodyki jest opracowanie transparentnych zasad tworzenia modelu w BPMN, który umożliwi automatyczną konwersję do języka wykonywalnego jakim jest BPEL. Prowadzone badania obejmują analizę teoretyczną oraz wytworzenie praktycznych narzędzi programistycznych ułatwiających proces konwersji.

Kolejnym krokiem procesu wytwarzania oprogramowania w ramach metodyki SMESDaD jest proces orkiestracji BPEL. Proces ten na wejściu otrzymuje szablon BPEL wygenerowany ze stworzonego modelu zapisanego w notacji BPMN i jego celem jest uzupełnienie kodu, które umożliwi jego automatyczne wykonanie na silniku procesowym. W trakcie generowania szablonu bardzo często tworzone są puste zadania (ang. *tasks*), których reprezentacja w BPMN nie była możliwa. Ponadto konieczna może



Rys. 7. Widok architektury korporacyjnej dla procesu zarządzanie zmianą (zgodnie z PMBOK)
Fig. 7. Enterprose architecture view for the change management process (PMBOK)

być implementacja przepływów komunikatów (ang. *message flows*) oraz stworzenie (lub transformacja) niezbędnych struktur danych. Ze względu na specyfikę BPEL zalecanym sposobem realizacji komunikacji jest technologia usług sieciowych (ang. *web services*). Proces SMESDaD opisany jest na rys. 8.

7. Repozytoria wzorców architektonicznych w metodyce SMESDaD

W rozwoju i przy wdrażaniu oprogramowania korporacyjnego przedsiębiorstwo informatyczne oraz przedsiębiorstwo, w którym to oprogramowanie będzie wdrażane, muszą współpracować na różnych obszarach. Dla przykładu rozważmy dwa obszary współpracy:

– Działalność projektowa. Dostarczanie wydań produktu w oparciu o projekty, mające na celu dostarczanie oprogramowania w terminie i zgodnego z zakresem;

– Działalność operacyjna. Produkcja oprogramowania mająca na celu dostarczanie oprogramowania zgodnego ze standardami jakości. Przedsiębiorstwo wdrażające oprogramowanie nie tylko będzie się interesować jakością produktu w momencie wydania, lecz w całym okresie jego użytkowania. W tym przypadku ważna jest współpraca między organizacjami w całym cyklu oprogramowania.

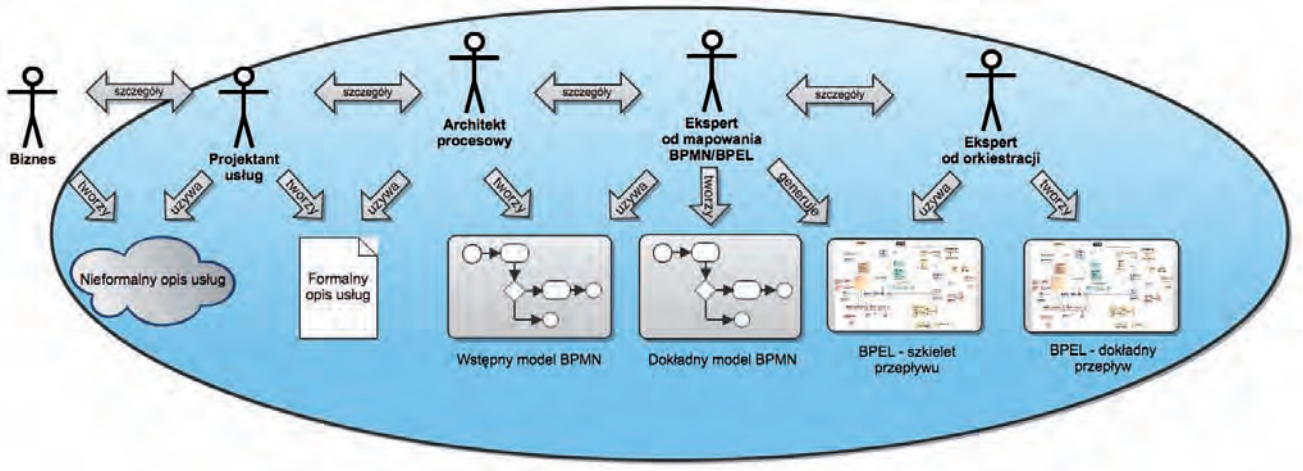
Zazwyczaj przedmiotem modelowania architektury przedsiębiorstwa jest działalność operacyjna. Pełny model obejmuje kolejne warstwy: biznesową, aplikacji i infrastruktury obejmujące rozmieszczone na kolejnych poziomach abstrakcji elementy składające się na procesy główne, pomocnicze i zarządcze.

W środowisku przedsiębiorstw branży IT, których podstawową formą jest działalność projektowa panuje pogląd, że poprawę efektywności wytwarzania oprogramowania i większą konkurencyjność można uzyskać przez wdrożenie sprawdzonych metodyk zarządzania projektami. Najczęściej firmy poszukują rozwiązań w obrębie metodyk klasycznych, do których należy PMBOK, PRINCE2, COBIT oraz metodyk zwinnych, których najbardziej popularnym przedstawicielem jest SCRUM.

Problem wyboru określonej metodyki i zdefiniowania organizacji projektu dostosowanej do lokalnych warunków jest bardzo złożony, ponieważ musi uwzględniać istniejące ograniczenia i uwarunkowania wynikające ze struktury organizacyjnej przedsiębiorstwa, dostępnych zasobów i wiedzy zgromadzonej w przedsiębiorstwie. Z drugiej strony oczekuje się, że zdefiniowana architektura procesów zarządzania projektem powinna być optymalna lub suboptymalna ze względu na rozmaite kryteria (np. wartość dodana, ryzyko projektowe).

Celem metodyki SMESDaD jest również zdefiniowanie architektury korporacyjnej, która będzie obejmowała również procesy projektowe. Ich cechą charakterystyczną jest często unikatowy charakter w ramach pojedynczego projektu (na przykład proces migracji danych może być wykonywany tylko raz). Jednocześnie cechą jest powtarzalność dla wielu projektów, zarówno prowadzonych wewnątrz organizacji, jak i zewnętrznych świadczonych na rzecz klientów korporacyjnych wdrażających oprogramowanie. W obu przypadkach, realizacja procesów projektowych wymaga zdefiniowania związków z obiektami biznesowymi wspierającymi zarządzanie projektami oraz interakcji i kolaboracji pomiędzy aktorami realizującymi projekt i będącymi jego nabywcą (interesariuszami).

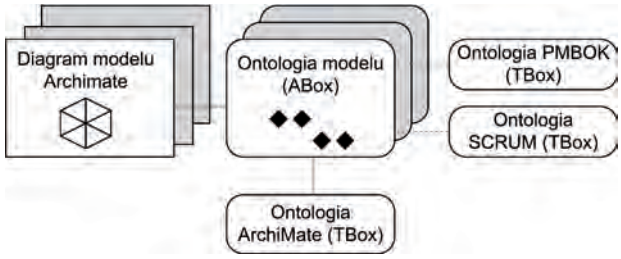
Definiując architekturę korporacyjną dla zarządzania projektami, wykorzystano istniejące i uznane metodyki, zarówno klasyczne (których przedstawicielem jest PMBOK), jak i zwinne, spośród których najpopularniejszy jest obecnie SCRUM. Konsekwencją przyjętych założeń jest z jednej strony pewna standaryzacja rozwiązań narzucona przez wymienione metodyki, ale również wynikająca z ich generycznego charakteru konieczność uściślenia i adaptacji do typu organizacji i typów prowadzonych przez nią projektów.



Rys. 8. Fazy metodyki SMESDaD
Fig. 8. SMESDaD phases

W rezultacie można zaproponować wiele różnych modeli architektury korporacyjnej wspierającej działalność projektową, w tym różne procesy, zdarzenia, przetwarzane obiekty (dokumenty) oraz role wraz z obszarami ich interakcji i kolaboracji. Przykład widoku biznesowego takiej architektury przedstawiono na rys. 7. Diagram obrazuje wybrany aspekt procesu zarządzania zmianą zgodnego z PMBOK [18] – wybrane w modelu etapy zarządzania zmianą (ang. *Formalize Request, Asses Impact, Praepare Updates* itd.), przetwarzane obiekty biznesowe i ich zależności (ang. *Change Request, Impact Analysis, Planned Updates, Project Documents, Project Management Plan*) oraz role (ang. *Project Team, Stakeholder, Project Manager, Change Control Board*). Nieprzedstawiony ze względu na objętość tekstu pełny model architektury korporacyjnej dla tego procesu rozciąga się daleko głębiej, definiując możliwe role, obiekty danych, wspierające narzędzia IT, usługi świadczone przez Biuro Projektowe (ang. *Project Management Office*) itd.

Złożoność metodyk zarządzania projektami sprawia, że pełny model architektury korporacyjnej może być bardzo obszerny (PMBOK definiuje 42 procesy projektowe); możliwe także będą także ich wariacje będące wynikiem skalowania, refaktoryzacji oraz adaptacji do lokalnych uwarunkowań: charakteru organizacji uczestniczących w projekcie i typu projektu. Planowanym kierunkiem prac jest zbudowanie repozytorium modeli architektur korporacyjnych zapewniającego bogaty semantyczny opis dla składowanych modeli w postaci ontologii.



Rys. 9. Logiczna struktura elementów repozytorium
Fig. 9. Logical structure of the repository elements

Na rys. 9 przedstawiono logiczną strukturę elementów repozytorium. Przechowuje ono diagramy sporządzone w języku Archimate (na tym etapie posługujemy się narzędziem Archi [19]). Diagramom przechowywanym w postaci plików w repozytorium towarzyszy opis zawartości w postaci ontologii. Jej składowe to indywidua (ABox) odpowiadające elementom pojawiającym się na diagramie. Są one instancjami pojęć zdefiniowanych w ontologii języka (TBox). Dla wybranych modeli (związanych z procesami projektowymi) definiowane są odwzorowania do pojęć zdefiniowanych we wcześniejszych pracach ontologii metodyk zarządzania projektami PMBOK i SCRUM.

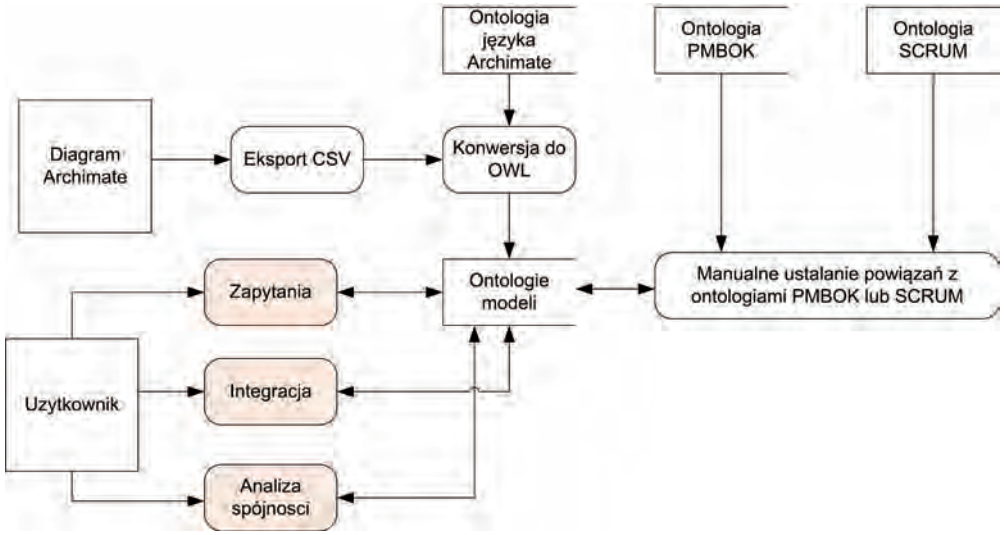
Na rys. 10 przedstawiono zakres funkcjonalności repozytorium. Ontologia modelu oraz powiązania z ontologią języka Archimate budowana jest automatycznie w momencie dodawania do repozytorium (w tym celu wykorzystywana jest funkcjonalność narzędzia Archi pozwalająca na eksport zawartości modelu w formacie CSV). Wynikowy plik tekstowy jest następnie konwertowany na formalną specyfikację w języku OWL. Na dalszym etapie mogą być manualnie ustalane powiązania z pojęciami występującymi w ontologiach PMBOK lub SCRUM (rolami, procesami, obiektami biznesowymi, zdarzeniami).

Planowane funkcje użytkowe repozytorium to:

- indeksowanie i wykonywanie złożonych semantycznych zapytań,
- szeroko rozumiana integracja modeli obejmująca manualne i automatyczne ustalanie odwzorowań,
- analiza spójności i ocena właściwości strukturalnych wytworzonych rozwiązań.

8. Zasada wdrożenia i utrzymania oprogramowania korporacyjnego

Bardzo istotnym zagadnieniem jest określenie zasad wdrożenia i utrzymania (serwisowania) oprogramowania korporacyjnego. W związku z zaletami i tendencjami technologicznymi celowe jest zastosowanie wdrożenia w postaci dostosowanej do potrzeb przedsiębiorstwa chmury obliczeniowej.



Rys. 10. Funkcjonalność repozytorium
Fig. 10. Repository functionality

W rozważaniach opieramy się na popularnej i uznanej definicji chmury obliczeniowej podanej przez National Institute of Standards and Technology (NIST) [9]: Chmura obliczeniowa jest modelem umożliwiającym wszechobecny, wygodny, sieciowy dostęp na żądanie dla dzielonej puli konfigurowalnych zasobów (takich jak sieci, serwery, pamięci, aplikacje i usługi), który może zostać szybko dostarczony (przez środowisko o ograniczonych zasobach) i wydany przy minimalnym wysiłku związanym z zarządzaniem lub przy minimalnym współdziałaniu dostawcy usługi. Model definiuje pięć podstawowych charakterystyk (samoobsługa na żądanie, dostęp poprzez sieć szerokopasmową, tworzenie puli zasobów, szybka elastyczność, mierzalne usługi), trzy modele chmury: oprogramowanie jako usługa (ang. *Software as a Service* – SaaS), platform jako usługa (ang. *Platform as Service* – PaaS) i infrastruktura jako usługa (ang. *Infrastructure as a Service* – IaaS), oraz cztery modele wdrożenia chmury (prywatna, wspólna, publiczna oraz hybrydowa).

Dla celów architektury korporacyjnej powinien być wybrany model wdrożeniowy chmury. Jednym z prostszych sposobów kategoryzacji chmur obliczeniowych jest Cloud Cube Model, zaproponowany przez Jericho Forum [11]. Model ten definiuje cztery wymiary:

- Wewnętrzny/zewnętrzny (chmura prywatna/publiczna). Określamy tu, czy chmura ma być wykorzystywana wewnątrz, czy na zewnątrz organizacji.
- Własny/otwarty. Własna chmura oznacza, że organizacja dostarczająca usługi dostarcza środki do utrzymania chmury. Chmury z otwartym dostępem stosują otwarte technologie dla wielu dostawców.
- Dostępność/niedostępność. Niedostępność oznacza zastosowanie tradycyjnych zapór sieciowych, natomiast dostępność oznacza dostarczanie danych opakowanych w metadane oraz zabezpieczonych przed niepowołanym dostępem.
- Nie zlecane/zlecane (*sourced/outsourced*). Ten wymiar określa, czy usługa jest dostarczana przez własny zespół, czy też jest zlecana do wykonania na zewnątrz.

Bazując na tych definicjach, zakładamy, że architektura korporacyjna będzie wdrażana jako wewnętrzna, niezlecana w pierwszym etapie oraz niedostępna w pierwszym etapie, lecz dostępna w kolejnym. Dla tego typu zastosowań zakładamy użycie prywatnej chmury oraz że organizacja posiada infrastrukturę i jest odpowiedzialna za zarządzanie infrastrukturą. W następnej kolejności infrastruktura może zostać przekazana na zewnątrz. Chmura będzie umieszczona w zaufanym środowisku, traktowanym jako część organizacji. W następnej kolejności chmura może zostać umieszczona w środowisku obcym, niestanowiącym części organizacji.

W przypadku wdrożenia opartego na chmurze powstaje pytanie dotyczące sukcesu wdrożenia. Sukces wdrożenia zależy od modelu biznesowego oraz zrównoważonego (ang. *sustainability*) rozwoju. Zrównoważony rozwój oznacza zdolność adaptacji do zewnętrznych i wewnętrznych zmian. Do opisu słabych i mocnych stron wdrożenia w chmurze służy model heksagonalny [1], określający komponenty typu ludzie (konsumenci i inwestorzy), biznes (popularność i wartość), czy możliwości wykonania zadania (ang. *Get The Job Done* – GTJD). W większości przypadków firmy informatyczne dysponują ograniczonymi zasobami i kompetencjami do wykonywania całego oprogramowania korporacyjnego. Jednym z ważnych zagadnień dla organizacji rozwijającej oraz tej, w której oprogramowanie będzie wdrażane, jest określenie, jakie oprogramowanie należy zrobić, a jakie zakupić w formie gotowej. Decyzja ta powinna bazować na analizie SWOT z uwzględnieniem wskaźników ROI i TCO.

W powyższym procesie należy zidentyfikować komponenty oprogramowania, które należy kupić i zintegrować z resztą oprogramowania. Przykładowo, jedną z istotnych części SOA jest korporacyjna magistrala usług ESB (ang. *Enterprise Service Bus*). Ogólnie rzecz biorąc, ESB dostarcza funkcjonalności w pięciu obszarach: architekturze, komunikacji, mediacji, instrumentacji (orkiestracji), zmianie i kontroli. Wybór ESB powinien być dokładnie rozważony. Dla przykładu Forrester Research zdefiniował kryteria ewaluacyjne [12], na podstawie których ocenił najbardziej

popularne komercyjne i otwarte rozwiązania: Fuse Source, IBM, Mule Soft, Oracle, Progress Software, Red Hat, Software AG, Tibco Software oraz WSO2.

Przejście do modelu chmury obliczeniowej spowoduje przesunięcie aktywności przedsiębiorstwa w kierunku działań operacyjnych. Jest to, ogólnie rzecz biorąc, pozytywna zmiana, lecz należy uwzględniać utrzymanie systemów korporacyjnych. ITIL (ang. *Information Technology Infrastructure Library*) może być dobrym odnośnikiem w tym kierunku, gdyż jest popularnym standardem zarządzania usługami informatycznymi (*IT Service Management*). W standardzie ITIL zdefiniowano spójną mapę procesów, relacji, ról, kluczowych pojęć i mierników, przyjętą w IT.

9. Wnioski

Zagadnienie rozwoju oprogramowania korporacyjnego z uwzględnieniem architektury organizacji wytwarzającej oprogramowanie i organizacji, w której to oprogramowanie jest wdrażane, oraz wzięciem pod uwagę wszystkich powiązań (a w szczególności współpracy) pomiędzy organizacjami, jest zagadnieniem obszernym. Rozpatrywanie rozwoju oprogramowania w takim kontekście jest bardzo istotne ze względu na wzrost konkurencji powodującej konieczność skracania czasu rozwoju i dostarczania oprogramowania.

Dotychczasowe prace autorów koncentrowały się bardziej na zagadnieniach modelowaniu biznesowego i ontologicznego metodyk zarządzania projektami i ich integracji w warunkach funkcjonowania firmy informatycznej [8]. Również zajmowano się zagadnieniem tworzenia oprogramowania korporacyjnego w oparciu o SOA dla odziedziczonych systemów [17].

Autorzy pracy – uczestnicząc w różnych projektach przemysłowych – wiedzą, że w wielu przypadkach zakończenie sukcesem rozwoju tak dużych systemów jest możliwe przy ścisłej współpracy organizacji, a ta współpraca musi mieć charakter synergetyczny.

Bibliografia

1. Lankhorst M. et al. (2009): *Enterprise Architecture at Work*, Springer, 352.
2. The Open Group (2009): *The Open Group Service Integration Maturity Model (OSIMM), Technical Standard*, 73, http://www.opengroup.org/bookstore/catalog/c092.htm.
3. The Open Group (2009): *SOA Governance Framework, Technical Standard* (C093), www.opengroup.org/bookstore/catalog/c093.htm.
4. OASIS Reference Model for SOA (SOA RM), Version 1.0, OASIS Standard, 12 October 2006; http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf.
5. The Open Group Architecture Framework (TOGAF); http://www.opengroup.org/architecture/togaf9.
6. Archimate 1.0 Specificattion, The Open Group, http://www.opengroup.org, 2009, 122.
7. Capability Maturity Model Integration (CMMI) Version 1.1, CMMI for System Engineering, Software Eng., Integrated Product and Process Development and Supplier Sourcing. Carnegie Mellon SEI, CMMI Product Team, 2002, 724.
8. Werewka J., Szwed P., Rogus G. (2010): *Integration of classical and agile project management methodologies based on ontological model, Production engineering in making*, ed. Piotr Łebkowski, Krakow, AGH University of Science and Technology Press, 7–28.
9. Mell P., Grance T. (2011): *The NIST Definition of Cloud Computing (Draft), Recommendations of the National Institute of Standards and Technology, The National Institute of Standards and Technology (NIST)*, U.S. Department of Commerce, Special Publication 800-145, p. 6., 7–28.
10. Homann U., Tobey J. (2006): *From Capabilities to Services: Moving from a Business Architecture to an IT Implementation*, http://msdn.microsoft.com/en-us/library/aa479075.aspx.
11. Cloud Cube Model: Selecting Cloud Formations for Secure Collaboration, www.jerichoforum.org, Version 1.0, 2009, 7.
12. Vollmer K., Gilpin M., Rose S. (2011): *The Forrester Wave^{TM}: Enterprise Service Bus*, Q2 2011, www.forrester.com, 15.
13. Reference Architecture Foundation for Service Oriented Architecture, Version 1.0, Committee Draft 02, OASIS, http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-cd-02.pdf, 2009, 119.
14. Malik N. (2009): *Will there be a battle between Archimate and the ULM?* http://blogs.msdn.com/b/nickmalik/archive/2009/04/17/will-there-be-a-battle-between-Archimate-and-the-uml.aspx.
15. http://SOAMethodology.com.
16. The Open Group (2011): The Open GroupUsing TOGAF to define and Govern Service-Oriented Architectures, Open Group Guide, Maj 2011, http://www.opengroup.org, 59.
17. Werewka J., Rogus G. (2011): *A solution for adaptation of legacy enterprise software for private cloud computing model, Information Systems Architecture and Technology*, Library of Informatics of University Level Schools, Service Oriented Networked Systems, 32nd International Conference on Information Systems Architecture and Technology, 61–75.
18. A Guide to the Project Management Body of Knowledge Fourth Edition (PMBOK Guide), Approved American National Standard ANSI/PMI 99-001-2008, PMI 2008.
19. Archi, Archimate Modelling Tool, http://archi.cetis.ac.uk/download.html, 2011.

SMESDaD – Synergetic Methodology for Enterprise Software Development and Deployment

Abstract: The paper presents the problem of the development, production and deployment of enterprise software. The success of this process relies on effective cooperation between a company developing software and enterprise in which software is to be deployed. A model of cooperation between enterprises can be developed on the basis of corporate models of both companies. The paper outlines the methodology for corporate software development based on the synergetic relationships between the companies. The

methodology is intended to integrate the best solutions in the field of corporate modeling, service-oriented architectures and software deployment models.

Keywords: corporate architecture, modeling, SOA, software development, software deployment

dr inż. Grzegorz Rogus

Doktor nauk technicznych w dyscyplinie informatyka. Zainteresowania naukowe koncentrują się głównie wokół szeroko rozumianej inżynierii oprogramowania. Interesuje się tematyka zwiazaną z projektowaniem architektury korporacyjnej i wytwarzaniem oprogramowania w oparciu o paradygmat SOA.
e-mail: rogus@agh.edu.pl

dr inż. Piotr Szwed

Doktor nauk technicznych w dyscyplinie informatyka. Zainteresowania naukowe koncentrują się wokół problemów inżynierii oprogramowania, modelowania biznesowego, wykorzystania ontologii, sieci Petriego oraz weryfikacji oprogramowania z wykorzystaniem metod formalnych.
e-mail: pszwed@ia.agh.edu.pl

dr inż. Michal Turek

Absolwent kierunku Informatyka na Wydziale EAIiE AGH (2002, na tym samym wydziale obronił doktorat w dyscyplinie informatyka) oraz kierunku Zarzadzanie na Wydziale Zarządzania AGH (2004). Zainteresowania naukowe obejmują grafikę komputerową oraz szeroko rozumianą inżynierię oprogramowania.
e-mail: mitu@agh.edu.pl

dr inż. Paweł Skrzyński

Absolwent kierunku Informatyka na Wydziale EAIiE AGH (2002, na tym samym wydziale obronił doktorat w dyscyplinie informatyka) oraz kierunku Zarządzanie na Wydziale Zarządzania AGH (2004). Zainteresowania naukowe obejmują zastosowanie technologii informatycznych w ekonomii oraz nauce o zarządzaniu (projektowanie architektury korporacyjnej, paradygmat SOA, EDA, modelowanie biznesowe). Miłośnik technologii Java. Posiada bogate doświadczenie w zakresie kierowania projektami informatycznymi. Prywatnie interesuje się sportem: tenis, narciarstwo oraz maraton (rekord życiowy: 3 h 44 min.)
e-mail: skrzynia@agh.edu.pl

dr hab. inż. Jan Werewka, prof. AGH

Jan Werewka jest absolwentem Liceum Ogolnoksztalcacego w Brzesku. Studia ukończył na Uniwersytecie Technicznym w Dreznie na Kierunku Elektronika Przetwarzania Danych. Pracę doktorską obronił na Wydziale Elektrotechniki Akademii Gorniczo-Hutniczej z dziedziny automatyki. W 1989 r. obronił pracę habilitacyjną z zakresu informatyki „Symulacja rozproszona systemów komunikacyjnych" na Uniwersytecie Humboldta w Berlinie na Wydziale Matematyki – Nauk Przyrodniczych. Praca została opublikowana w monografiach naukowych Akademii Nauk w Berlinie. Prowadzone prace badawcze i dydaktyczne dotyczą dwóch obszarów: modelowania i integracji zarzadzania projektami informatycznymi w oparciu o metodyki PMBOK i Scrum, oraz rozwijania skalowalnych systemow informatycznych w oparciu o architekturę korporacyjną. Jest członkiem organizacji międzynarodowych PMI i IEEE. Posiada bogate doświadczenie związane z prowadzeniem i uczestnictwem w projektach na uczelni oraz zarządzaniem przedsiębiorstwami informatycznymi.
e-mail: werewka@agh.edu.pl