STUDIA INFORMATICA 2012
Volume 33 Number 3A (107)

Paweł L. KACZMAREK
Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki

INTEGRACYJNA METODA WYTWARZANIA WIARYGODNYCH APLIKACJI ROZPROSZONYCH[1]

Streszczenie. W pracy przedstawiono metodę wytwarzania aplikacji w architekturze zorientowanej na usługi, w której uwzględniono interoperacyjność i wiarygodność integrowanych elementów. Zaproponowano rozszerzenie istniejących algorytmów wyboru usług o analizę interoperacyjności oraz wykorzystanie ekspertowego systemu wyboru technik tolerowania błędów bazującego na klasteryzacji. Metoda została zaimplementowana jako system internetowy.

Słowa kluczowe: architektura zorientowana na usługi, interoperacyjność, wiarygodność

INTEGRATION-ORIENTED DEVELOPMENT METHOD OF DEPENDABLE DISTRIBUTED APPLICATIONS

Summary. The paper presents an integration oriented method of application development in SOA that considers interoperability issues and dependability of services. It is proposed to extend existing algorithms of service selection with dedicated interoperability constraints and to use an expert-based system based on clustering for selection of fault tolerance techniques. The method was implemented in a web-based system.

Keywords: Service Oriented Architecture, interoperability, dependability

1. Wstęp

Architektura zorientowana na usługi (*Service Oriented Architecture – SOA*) zakłada, że złożone aplikacje są wytwarzane przez integrację już gotowych, prostych usług dostarczanych

przez niezależnych dostawców. Takie podejście pozwala na zredukowanie kosztów i czasu wytwarzania, jednak wymaga rozwiązania wielu nowych problemów.

- Usługi składowe posiadają różne cechy wiarygodności, czasami niewystarczająco wyspecyfikowane przez dostawców[4], co może negatywnie wpływać na wiarygodność końcowego rozwiązania.
- Usługi mogą być związane z heterogenicznymi środowiskami wykonania, co powoduje konieczność zapewnienia interoperacyjności. Współdziałanie między różnymi środowiskami może zostać ograniczone przez względy, takie jak: niezgodność ze standardami, implementacja różnych wersji standardów i błędy implementacji [2, 8].
- Istnieje wiele alternatywnych technik tolerowania błędów. Techniki powinny być stosowane zależnie od wymagań systemowych i wiarygodności komponentów w celu zwiększenia wiarygodności końcowego rozwiązania [6].

Biorąc pod uwagę powyższe trudności, zaproponowano metodę wytwarzania aplikacji bazującą na integracji gotowych usług przy uwzględnieniu ich wiarygodności oraz interoperacyjności. Metoda wykorzystuje technologię usług sieciowych (*Web services*) oraz procesów biznesowych (*workflow*), które są dojrzałymi mechanizmami stosowanymi w SOA. W doborze usług uwzględniono zarówno atrybuty usług, jak i techniki tolerowania błędów. Zastosowanie dodatkowych mechanizmów tolerowania błędów ma na celu zabezpieczenie wytwarzanej aplikacji przed skutkami nieprawidłowych działań.

2. Zasady i etapy działania metody

Zaproponowana metoda zakłada, że zostaną wykorzystane istniejące rozwiązania i mechanizmy w możliwie szerokim zakresie. Celem metody jest optymalizacja wiarygodności i innych parametrów *Quality of Service (QoS)* aplikacji, przy założeniu że logika działania aplikacji została zdefiniowana. Metoda przewiduje, że wytwarzana aplikacja jest złożoną usługą, logiczna struktura aplikacji jest opisana w formie procesu biznesowego.

W ramach prac nad metodą opracowano dwie bazy wiedzy, obejmujące systematyczny opis i ocenę istniejących mechanizmów.

- Baza wiedzy o interoperacyjności środowisk uruchomienia usług (*runtime environments*). Zawiera opis standardów Web services, serwerów aplikacji, opcji konfiguracyjnych oraz ocen zakresu integracji i złożoności prac deweloperskich.

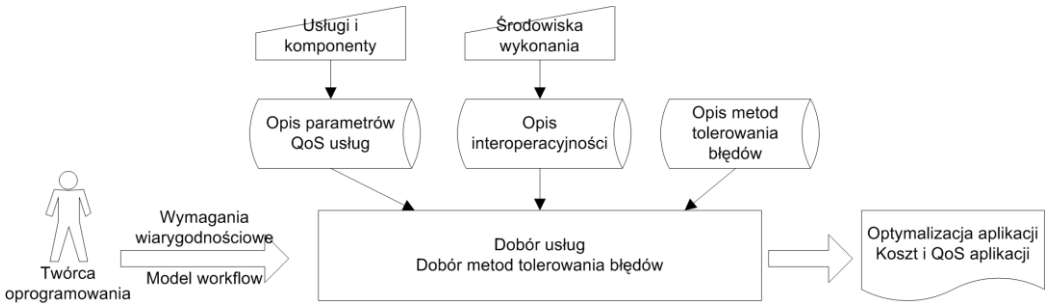
[1] Praca naukowa finansowana ze środków na naukę w latach 2009-2012 jako projekt badawczy nr N N519 172337 przez Narodowe Centrum Nauki/Ministerstwo Nauki i Szkolnictwa Wyższego.

- Baza wiedzy o metodach tolerowania błędów. Zawiera opis metod i miar wiarygodności [6] oraz opis wpływu metod na wytwarzanie aplikacji i jej działanie.

Bazy wiedzy stanowią uniwersalną infrastrukturę możliwą do wykorzystania niezależnie od realizowanego projektu. Po dostarczeniu wymaganych danych bazowych jest możliwe zastosowanie uniwersalnej metody projektowania wiarygodnych aplikacji. Główne kroki metody obejmują:

1. Określenie wymagań na projektowaną aplikację:
   a. zdefiniowanie logicznej struktury aplikacji w formie workflow,
   b. wyspecyfikowanie zestawu usług, ich atrybutów oraz usług alternatywnych, które mogą być wykorzystane do kompozycji aplikacji,
   c. określenie wymagań końcowej wiarygodności i poprawności działania aplikacji.
2. Wybór optymalnych usług, z uwzględnieniem ich atrybutów i interoperacyjności.
3. Wybór środowiska uruchomienia dla aplikacji workflow.
4. Wybór metod tolerowania błędów wraz z odpowiednimi wzorcami kodu.
5. Rekomendacja wybranych elementów deweloperowi aplikacji.

Na rys. 1 przedstawiono ogólny schemat komunikacji pomiędzy deweloperem, systemem kompozycji aplikacji i bazami wiedzy. Specyfikacja wymagań wiarygodnościowych oraz określenie alternatywnych usług są pozostawione w gestii projektanta. Natomiast wybór usług, środowisk wykonania oraz metod tolerowania błędów są dokonywane automatycznie na podstawie wymagań oraz zawartości baz wiedzy.



Rys. 1. Wymiana danych w integracyjnej metodzie wytwarzania oprogramowania
Fig. 1. Data exchange in the integration-oriented development method

Autorowi niniejszej pracy nie jest znane rozwiązanie, które umożliwia wybór usług i technik tolerowania błędów, uwzględniając czynniki interoperacyjności i wiarygodności końcowej aplikacji.

3. Bazy wiedzy interoperacyjności i technik tolerowania błędów wykorzystywane w metodzie

W ramach analizy interoperacyjności opracowano usystematyzowany katalog bazujący na koncepcjach związanych z usługami sieciowymi: standard, wersja standardu, opcje konfiguracyjne i wartości tych opcji [5]. Wśród skatalogowanych standardów znalazły się, poza podstawowymi standardami SOAP i WSDL, standardy z zakresu: adresacji, niezawodności komunikacji, bezpieczeństwa, transakcji oraz procesów biznesowych.

W proponowanym podejściu uwzględniono dwa aspekty interoperacyjności: czasu wytwarzania oraz czasu wykonania. Pierwszy rodzaj wiąże się z wymaganą pracochłonnością oraz wiedzą, które są potrzebne do wykonania integracji między usługami. Drugi rodzaj interoperacyjności opisuje zakres integracji między elementami systemu. W przypadku tego rodzaju znanych jest wiele metryk, które opisują integrację [3]. Zakres integracji określa, czy dwie usługi mogą prawidłowo wymieniać dane na poziomie działania Web services, co odpowiada poziomowi *Connected* w metryce *LISI* [3]. Oba atrybuty są wyrażone w ocenach z zakresu od 1 do 5. Dla każdej oceny określono warunki, które decydują o jej przyznaniu.

Zebrane wyniki oraz skatalogowane standardy Web services są dostępne pod adresem: http://www.as-interoperability.eti.pg.gda.pl/

Ocena interoperacyjności jest dokonywana dla konkretnych konfiguracji, w których są środowiska wykonania integrowane. Opis konfiguracji obejmuje opis wykorzystanych standardów, ich wersji, opcji konfiguracyjnych oraz wartości tych opcji. Ocena jest dokonywana przez dewelopera, który określa integrowane środowiska wykonania, wykorzystywane standardy, zakres integracji oraz zakres zmian w konfiguracji.

Techniki tolerowania błędów [6] (*Fault Tolerance Techniques – FTT*) różnią się między sobą zarówno w kontekście procesu wytwarzania (wprowadzając dodatkowy narzut pracy), jak i działania (zwiększając odporność aplikacji na błędy). W proponowanej metodzie opracowano i zastosowano system ekspertowy, który wspomaga programistów w wyborze FTT [7]. System wykorzystuje bazę wiedzy o znanych FTT oraz ich atrybutach. Atrybuty technik podzielono na trzy grupy: (i) atrybuty procesu wytwarzania (np.: ekonomiczność, prostota konfiguracji), parametry QoS działania aplikacji (np.: wydajność sekwencyjna, rozproszona), atrybuty wiarygodności (np. niezawodność, dostępność, bezpieczeństwo).

Baza wiedzy gromadzi informacje o technikach tolerowania błędów ze znanych katalogów i źródeł oraz ocenę tych technik przy użyciu powyższych atrybutów. Ocena technik została dokonana na podstawie ankiety przeprowadzonej wśród ekspertów z dziedziny wiarygodności. Szczegółowa lista technik została zarejestrowana w systemie: http://www.fttadvisor.eti.pg.gda.pl/

Wykorzystując katalog FTT, zaproponowano algorytm ekspertowy, który wspomaga programistów w wyborze takich technik, które będą posiadać wymagane atrybuty, ale jednocześnie będą możliwie różnorodne. Różnorodność technik ma na celu wyeliminowanie możliwie dużej liczby błędów przy jednoczesnej minimalizacji kosztów wytwarzania aplikacji. W celu określenia różnorodności (podobieństwa) technik zastosowano algorytm klasteryzacji K-średnich (*K-means clustering*).

Przyjęto następujący sposób mapowania danych o technikach na model odpowiedni dla algorytmu klasteryzacji: atrybuty (w liczbie *M*) tworzą przestrzeń S^M o wymiarze *M*, każda technika jest wektorem w przestrzeni $p \in S^M$, zaś atrybuty są wartościami w odpowiednich wymiarach. Liczba FTT do wyboru (*K*) odpowiada liczbie klastrów $\omega_1,...,\omega_K$. Celem algorytmu jest minimalizacja wartości odległości K-means wyliczonej zgodnie ze wzorem:

$$J = \sum_{k=1}^{K} \left(\sum_{j, p_j \in \omega_k} d(p_j, \mu_k) \right) \tag{1}$$

gdzie *d* oznacza odległość między punktem p i środkiem klastra μ Wybór odbywa się w następujących etapach:

1. Programista specyfikuje wymagane atrybuty, wagi oraz liczbę technik do wyboru (*K*).
2. Następuje podział technik na *K* grup podobieństw z wykorzystaniem klasteryzacji.
3. Z każdego klastra wybierana jest jedna technika, która posiada najwyższą wartość atrybutów z uwzględnieniem ich wag.
4. Techniki wraz ze wzorcami kodu są rekomendowane programiście.

4. Wykorzystywane algorytmy wyboru usług z ograniczeniami interoperacyjności

Odpowiedni wybór usług jest kluczowym zadaniem podczas kompozycji aplikacji w technologii SOA. Proponowana metoda wykorzystuje znane rozwiązania wyboru usług, rozszerzając je dodatkowo o elementy analizy interoperacyjności. Wytwarzanie usługi złożonej przez integrację usług prostych przyjmuje następujące założenia [9, 1].

- Usługa złożona (*Complex Service - CS*) składa się z N operacji połączonych instrukcjami sterującymi, każda operacja jest realizowana przez usługę z klasy usług $S_1,...,S_N$.
- Dla każdej klasy usług S_i istnieje jeden lub więcej alternatywnych usług s_{ij}, które różnią się parametrami QoS ($q_{ij} = [q_{ij}^1,...,q_{ij}^n]$).
- Zostaje zdefiniowana funkcja celu (*F*) zależna od wartości QoS serwisów i struktury *CS*.

- Zostają zdefiniowane ograniczenia na wartości niektórych atrybutów usługi złożonej $Q_c = [Q_c^1,..,Q_c^m]$.

Algorytm wyboru usług ma na celu optymalizację (osiągnięcie maksymalnej wartości) funkcji celu *F* oraz jednocześnie zachowanie ograniczeń Q_c dla złożonej usługi.

Proponowana metoda jest adekwatna dla dwóch głównych modeli rozwiązania tego problemu: modelu kombinatorycznego oraz modelu grafowego. Model kombinatoryczny traktuje wybór usług jako wielokryterialny problem plecakowy, w którym klasy usług to grupy elementów, usługi to elementy, parametry QoS to wymagane zasoby i dostarczane zyski odpowiednio, zaś ograniczenia Q_c są traktowane jako dostępne zasoby w plecaku. Model matematyczny odpowiada całkowitoliczbowemu programowaniu liniowemu (*Integer Linear Programming – ILP*):

$$\begin{aligned} &Max \sum_{i=1}^{N} \sum_{j \in S_i} F_{ij} x_{ij} \\ &subject\ to \sum_{i=1}^{N} \sum_{j \in S_i} q_{ij}^q * x_{ij} \le Q_c^a \ (a = 1,...,m) \\ &\sum_{j \in S_i} x_{ij} = 1, \quad x_{ij} \in \{0,1\}, \quad i = 1,...,N, \end{aligned} \tag{2}$$

gdzie $x_{ij} = 1$, jeżeli usługa atomowa s_{ij} została wybrany do kompozycji usługi złożonej.

Model grafowy zakłada, że każda usługa atomowa jest węzłem w grafie. Jeżeli klasa S_i przesyła dane do klasy S_j, to istnieje krawędź od każdej usługi klasy S_i do każdej usługi klasy S_j. Z każdym węzłem są związane wartości atrybutów q_{ij} odpowiadające usłudze. Problem optymalnego wyboru usług jest równoznaczny znalezieniu ścieżki w grafie takiej, że wartość funkcji *F* dla węzłów na ścieżce będzie maksymalna, zaś ograniczenia Q_c zostaną zachowane.

4.1. Opis interoperacyjności podczas wyboru usług

Proponowana metoda modeluje elementy odpowiedzialne za interoperacyjność oraz uwzględnia ją w procesie obliczania optymalnej selekcji usług.

Niech $q^{comm}(s,s')$ oznacza interoperacyjność jako atrybut QoS reprezentujący koszt komunikacji między usługami, $q^{comm}(s,s')$ jest wartością binarną:

$$\begin{aligned} &q^{comm}(s,s') = 0, \quad jeżeli\ s,s'\ mogą\ wymieniać\ dane, \\ &q^{comm}(s,s') = 1, \quad jeżeli\ s,s'\ nie\ mogą\ wymieniać\ danych. \end{aligned} \tag{3}$$

Interoperacyjność jest analizowana w dwóch modelach komunikacji:

- Model zdalnego wywołania metod (*Remote Method Invocation – RMI*) – istnieje centralne środowisko wykonania, które hostuje usługę złożoną i wywołuje kolejno usługi atomowe (np. standard BPEL oraz Microsoft WWF).
- Model przesyłania komunikatów (*Message Passing – MP*) – usługi bezpośrednio komunikują się ze sobą (np. standard BPMN).

4.2. Rozszerzenia algorytmów dla modelu RMI

W przypadku modelu RMI konieczne jest uwzględnienie dodatkowo środowiska wykonania, które uruchamia usługę złożoną. Niech $R^{CS} = \{r_1^{CS},..,r_T^{CS}\}$ oznacza zbiór alternatywnych środowisk wykonania dla usługi złożonej.

Wprowadzane jest dodatkowe oznaczenie dla interoperacyjności między środowiskiem wykonania usługi złożonej oraz usług atomowych.

$$\begin{aligned} &q^{comm,cs}(s,cs,r^{CS}) = 0, \quad \textit{jeżeli } s \textit{ może wymieniać dane z } cs \textit{ na } r^{CS}, \\ &q^{comm,cs}(s,cs,r^{CS}) = 1, \quad \textit{jeżeli } s \textit{ nie może wymieniać danych z } cs \textit{ na } r^{CS}. \end{aligned} \tag{4}$$

Rozszerzenia algorytmów wyboru usług w tym modelu komunikacji polegają na filtrowaniu usług w zależności od interoperacyjności i wybranego środowiska wykonania r^{CS}. Możliwe jest zastosowanie dowolnego z istniejących algorytmów selekcji (ozn. SEL_ALG). Algorytm działa według następujących kroków.

1. Algorytm analizuje w pętli kolejne środowiska wykonania r^{CS}.
2. Dla każdego środowiska filtruje usługi proste, które mogę współpracować z r^{CS} oraz wybiera optymalne rozwiązanie wykorzystując SEL_ALG.
3. Jako wynik sugeruje r^{CS} o najwyższej funkcji celu (*F*).

4.3. Rozszerzenia algorytmów dla modelu Message Passing

W przypadku modelu MP interoperacyjność jest modelowana różnie zależnie od przyjętego modelu algorytmu wyboru usług (kombinatoryczny, grafowy). Modelu kombinatoryczny zostaje rozszerzony o równanie, które określa interoperacyjność usług

$$\sum_{j \in S_i} \sum_{l \in S_k} q_{s_{ij}, s_{kl}}^{comm} * (x_{ij} * x_{kl}) = 0 \tag{5}$$

dla każdej pary s_{ij} s_{kl}, takiej że s_{ij} przesyła dane do s_{kl}. Równanie wymaga, aby koszt interoperacyjności pary usług był 0, jeżeli usługi zostały wybrane.

Warunek ten jednak powoduje, że model programowania całkowitoliczbowego liniowego jest zamieniony na model mieszany nieliniowy, co znacznie komplikuje obliczenia. Powyższa forma równania zostaje zamieniona na formę rozdzielnego programowania (*separable pro-*

gramming), która jest względnie łatwo rozwiązywalna przez znane metody. Zastosowano podstawienie $x_{ij} * x_{kl} = y^2_{1,ijkl} - y^2_{2,ijkl}$, gdzie $y^2_{1,ijkl} = 0{,}5*(x_{ij} + x_{kl})$, zaś $y^2_{2,ijkl} = 0{,}5*(x_{ij} - x_{kl})$.

Powyższa forma może zostać wykorzystana przez algorytmy selekcji usług dla małych procesów biznesowych lub dla lokalnej optymalizacji.

Model grafowy / MP umożliwia stosunkowo łatwą reprezentację ograniczeń interoperacyjności, gdyż interoperacyjność $q^{comm}(s,s')$ jest bezpośrednio traktowana jako dodatkowy atrybut usługi. Zasady przetwarzania opierają się na dwóch krokach. Zakładając, że usługa s_{ij} przesyła dane do usługi s_{kl}, atrybut $q^{comm}(s,s')$ jest dodawany do każdej krawędzi idącej od wierzchołka s_{ij} do s_{kl}. Wektor Q_C jest rozszerzony o dodatkowy element Q_C^{comm}, który reprezentuje ograniczenia interoperacyjności:

$$Q_C^{'} = [Q_C^1,...,Q_C^m,Q_C^{comm}] \tag{6}$$

gdzie $Q_C^{comm} = 0$, Q_C^{comm} jest sumą $q^{comm}(s,s')$ na wybranej ścieżce w grafie CS.

Proponowane rozszerzenie nie zmienia struktury grafu ani zasad jego przetwarzania, gdyż interoperacyjność została potraktowana jako atrybut QoS. Model grafowy jest bardziej odpowiedni do analizy dla komunikacji na zasadach przesyłania wiadomości.

5. System wspomagający integracyjne wytwarzanie wiarygodnych aplikacji SOA

W ramach prac zaimplementowano internetowy system, który realizuje operacje proponowanej metody. System dostępny jest pod adresem http://kask.eti.pg.gda.pl/WorkflowFTI

Funkcjonalność systemu pokrywa się z bazami wiedzy i zadaniami realizowanymi w proponowanej metodzie i obejmuje moduły: opis interoperacyjności (ASInteroperability), opis i wybór technik tolerowania błędów (FTTAdvisor), wybór usług (WorkflowFTI).

Zaimplementowane zostały wybrane algorytmy wyboru usług, które miały na celu zweryfikowanie, czy proponowane rozszerzenia mogą zostać skutecznie wykorzystane. Prace implementacyjne dotyczyły modeli przetwarzania BPEL oraz BPMN. W systemie zaimplementowano algorytmy, takie jak: algorytm lokalnej optymalizacji dla BPEL oraz algorytmy *explicit enumeration*, uproszczony algorytm MSCP oraz algorytm zachłanny dla BPMN.

Moduły systemu udostępniają zarówno interfejs webowy dla użytkowników, jak i interfejs Web services dla innych modułów. System został zaimplementowany w środowisku .NET 3.5 i uruchomiony na serwerze Microsoft Internet Information Services. Dane w systemie są przechowywane w bazie danych MS SQL.

6. Podsumowanie

Przedstawiona w pracy metoda integracyjnego wytwarzania wiarygodnych aplikacji koncentruje się na uzyskaniu aplikacji o optymalnych parametrach przy istniejących ograniczeniach zasobów i interoperacyjności. W metodzie zastosowano znane i sprawdzone rozwiązania z zakresu tolerowania błędów i wyboru usług, dzięki czemu dotychczasowe modele teoretyczne mogą być zastosowane w technologii SOA. Zaimplementowany system zademonstrował działanie metody oraz jej skuteczność dla testowych przypadków doboru usług i ograniczeń interoperacyjności.

Dalsze prace badawcze przewidują zastosowanie metody podczas wytwarzania złożonych aplikacji w technologii SOA. Zastosowanie tej metody będzie miało na celu poprawienie jej skuteczności oraz wprowadzenie dodatkowych rozszerzeń.

BIBLIOGRAFIA

1. Czarnul P.: Modeling, run-time optimization and execution of distributed workflow applications in the jee-based beesycluster environment. The Journal of Supercomputing, Springer Netherlands, 2010.
2. Egyedi T.: Standard-compliant, but incompatible?! Computer Standards & Interfaces, 29 (2007), s. 605÷613.
3. Ford T., Colombi J., Graham S., Jacques D.: A survey on interoperability measurement [in:] 12th International Command and Control Research and Technology Symposium (ICCRTS) Adapting C2 to the 21st Century, 2007.
4. Gorbenko A., Kharchenko V., Popov P. et. al.: Development of Dependable Web Services out of Undependable Web Components, Univ. of Newcastle upon Tyne, 2004.
5. Kaczmarek P. L., Nowakowski M.: A developer's view of application servers interoperability. 9th International Conference on Parallel Processing and Applied Mathematics (in print), Toruń, Polska, 2011.
6. Resilience for Survivability in IST, European Network of Excellence ReSIST, http://www.resist-noe.org, 2006.
7. Kaczmarek P. L., Roman M. Ł.: A Methodology for Selection of Fault Tolerance Techniques in Application Development, accepted for: 11th Intern. Conf. on Artificial Intelligence and Soft Computing ICAISC, Zakopane, Polska, 2012.
8. Web Services Interoperability Consortium: Interoperability: Ensuring the Success of Web Services, 2004.

9. Yu, T., Zhang, Y., and Lin, K.-J.: Efficient algorithms for web services selection with end-to-end qos constraints. ACM Transactions on the Web, 2007.

Wpłynęło do Redakcji 12 marca 2012 r.

Abstract

In Service Oriented Architecture (SOA), there may exist many alternative services supplying required functionality but differing in non-functional attributes such as dependability, performance or price. Application development requires effective algorithms for service selection as well as resolution of interoperability and dependability issues. The paper presents a method of SOA application development that considers service interoperability and dependability attributes of the final application (Fig. 1.). A systematic classification of concepts related to Web services standards is proposed, which enables developers to rate interoperability of runtime environments in concrete configurations. Using the rating, existing service selection algorithms are extended with dedicated constraints that model interoperability in two computational models: the combinatorial model and the graph model, Eq.: (2), (3), (4). In order to improve dependability of service integration, the methodology proposes an advisory system that recommends fault tolerance techniques (FTT) depending on required attributes of the development process, such as cost and time, and the runtime operation, such as performance and reliability. The advisory system uses the K-means clustering algorithm – Eq. (1) – to determine similarities between techniques and select those techniques that are possibly different but simultaneously conform developer requirements. As a part of the research, a system that supports development of dependable SOA applications was implemented. System implementation covers: interoperability knowledge base, the FTT advisory system, and representative service selection algorithms.

Adres

Paweł L. KACZMAREK: Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki, ul. Narutowicza 11/12, 80-233 Gdańsk, Polska, pawel.kaczmarek@eti.pg.gda.pl