Stanisław Jerzy Niepostyn[1], Andrzej Tyrowicz[2]

Zastosowanie metryki FBS *entropy-based* do szacowania kompletności i spójności architektury oprogramowania złożonych systemów IT na przykładzie systemów publicznych pl.ID (Źródło) oraz e-IRZ ARiMR

1. Wstęp

Wzrastające znaczenie i rola informatyzacji państwa w niemal każdej dziedzinie życia powodują potrzebę budowy coraz większych i bardziej złożonych systemów informatycznych. Jednocześnie szybko zmieniające się wymagania dla systemów IT prowadzą do coraz krótszych terminów ich realizacji. Wobec czasochłonnego i uciążliwego konstruowania pełnej architektury oprogramowania (ang. *upfront software architecture*), będącej podstawą budowy systemów informatycznych, coraz większą popularność zdobywa koncepcja zwinnej architektury oprogramowania (ang. *agile software architecture*).

Do oceny projektów złożonych systemów informatycznych, implementowanych w środowisku oprogramowania zorientowanego obiektowo (ang. *object-oriented programs*), można wykorzystać metrykę architektury oprogramowania FBS (ang. *Functional-Behaviour-Structure*). Metryka ta wylicza entropię[3] diagramów UML[4], które składają się na konstrukcję architektury oprogramowania w koncepcji Enhanced CMDA[5] (w skrócie nazwanej e-CMDA, gdzie CMDA oznacza Consistent Model Driven Architecture), umożliwiającej budowę zwinnej architektury oprogramowania z zastosowaniem reguł spójności (ang. *consistency rules*).

---

1 Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych.
2 Agencja Europejska, Andrzej Tyrowicz.
3 Tzn. miary odwrotności zawartości informacyjnej.
4 Unified Modeling Language, http://www.omg.org (25.08.2016).
5 S.J. Niepostyn, *Koncepcja architektury sterowanej regułami spójności modeli (CMDA) na przykładzie systemu e-IRZ dla ARiMR*, TIAPISZ 2015.

Jedną z głównych zalet metodyki e-CMDA jest możliwość porównywania i szacowania własności różnych architektur oprogramowania złożonych systemów IT ze względu na ich implementowalność przez obiektywne obliczenia ich spójności, kompletności i transformowalności. Niniejsza praca zawiera opis metryk złożoności architektury oprogramowania wraz z koncepcją autorskiej metryki FBS oraz przykłady jej użycia dla dwóch systemów publicznych: systemu *e-IRZ* dla Agencji Restrukturyzacji i Modernizacji Rolnictwa[6], systemu pl.ID, zwanego często systemem *Źródło*[7], oraz systemu Focus.

2. Metryki złożoności architektury oprogramowania

Metryki oprogramowania, proponowane od lat siedemdziesiątych ubiegłego wieku, miały służyć w przemyśle IT do oszacowania jakości, pracochłonności lub złożoności budowanego oprogramowania. Pomimo opracowania wielu metryk, takich jak COSMIC, IFPUG, czy ISO 25010, większość z nich nie cechuje się na tyle solidnymi podstawami, by stać się wiarygodnymi metrykami oprogramowania uznanymi przez środowiska badaczy[8].

Pod koniec XX w. zauważono[9], że przy budowie systemów IT metryki architektury oprogramowania mają o wiele większe znaczenie niż metryki kodu[10], gdyż umożliwiają zoptymalizowanie i oszacowanie systemu informatycznego jeszcze zanim ten kod źródłowy zostanie wytworzony.

Cechami takimi mogą się za to charakteryzować metody badania złożoności oprogramowania, związane z architekturą oprogramowania, podzielone przez Abran[8] na metody bazujące na danych jakościowych (ang. *qualitative models*) oraz ilościowych (ang. *quantitative models*). Modele jakościowe związane są z subiektywną oceną ekspercką poszczególnych atrybutów jakościowych oprogramowania, a modele ilościowe korzystają z matematycznych wyliczeń.

---

6 W ramach zamówienia publicznego na doradztwo IT.

7 W ramach zamówienia publicznego na ekspertyzę IT.

8 A. Abran, *Software Metrics and Software Metrology*, Wiley-IEEE Computer Society, 2010.

9 J. Zhao, *On Assessing the Complexity of Software Architectures*, ISAW '98 Proceedings of the third international workshop on Software architecture, 1998, pp. 163–166.

10 Z punktu widzenia użyteczności szeregu metryk kodu źródłowego opracowanych w ostatnich dekadach, takich jak liczba linii kodu, złożoność cyklomatyczna, metryki Halsteada czy metryki obiektowe, okazuje się, że nie jest możliwe wnioskowanie na ich podstawie na temat spójności, kompletności czy pracochłonności budowy systemu.

W autorskiej metryce FBS uzyskane wartości, na podstawie wag elementów diagramów UML, umożliwiają określenie ich spójności bądź jej braku, a także oceny kompletności modeli bądź braku tej kompletności. Spójność albo kompletność określa się zazwyczaj dla modeli (grup) diagramów. W niniejszym tekście przyjęto nomenklaturę standardu UML oraz ISO/IEC 42010:2011 [IEEE 42010]. Jednocześnie założono, że architektura oprogramowania składa się z czterech perspektyw zawierających po dwie warstwy, które można utożsamiać z modelami. Zatem jeśli modele (warstwy) są spójne i kompletne, to perspektywy są kompletne i spójne – podobnie cała architektura oprogramowania. W dalszej części zdefiniowano perspektywy: kontekstową (opis z ekonomicznego punktu widzenia); biznesową (opis z punktu widzenia procesów biznesowych), systemową (opis zewnętrzny systemu informatycznego) oraz komponentową (opis wewnętrzny systemu informatycznego). Każda z nich składa się z dwóch warstw: wejściowej (wizualizacji przypadku użycia) oraz wyjściowej (dekompozycji zwizualizowanego scenariusza na diagramy opisujące funkcjonalność, behawioryzm oraz strukturę).

3. Metryka złożoności oprogramowania FBS *entropy-based*

Metryka FBS wykorzystuje metodę wieloatrybutowej macierzy decyzyjnej[11], w której wyliczana jest znormalizowana entropia dla określonego zbioru danych, gdzie za zbiór ten przyjmuje się zestaw wartości rodzajów elementów diagramu UML dla trzech wymiarów: funkcjonalności, behawioryzmu i struktury[12]. Algorytm obliczania metryki FBS jest ulepszeniem pierwszej części algorytmu zaproponowanego przez Yi[11], przez wyznaczanie entropii dla trzech wymiarów diagramu UML. Grupy elementów tego diagramu umieszczono w wierszach macierzy decyzyjnej, a wartości tych grup otrzymano przez zsumowanie wartości elementów danej grupy dla określonego wymiaru.

Każdy element UML można przedstawić za pomocą uporządkowanej 3-ki:

$$X=\left\{X_{functionality};X_{behavior};X_{structure}\right\} \tag{1}$$

[11] T. Yi, *On the Application of Information Entropy-Based Multi-Attribute Decision in UML Class Diagram Metrics*, "International Journal of u- and e-Service" 2015, vol. 8, pp. 105–116.

[12] J.S. Gero, *Design Prototypes: a Knowledge Representation Schema for Design*, „AI Magazine" 1990, vol. 11, no. 4, pp. 26–36.

Po wielokrotnej weryfikacji (obliczenia na zbiorze 500 diagramów UML) wartości metryki dla diagramów maszyny stanów UML zdefiniowano poniżej następujące wartości wag dla 4 rodzajów elementów, które najczęściej występują na tym diagramie: UML State = {0;3;0}; UML Transition = {0;3;0}; UML Region = {0;2;1}; UML PseudoState = {0;2;0}.

Inne elementy UML wymienione w grupie elementów *UML StateMachine* dziedziczą własności z wymienionych wyżej 4 rodzajów elementów. W podobny sposób ustalono wagi pozostałych elementów diagramów UML.

Poniżej przedstawiono wzór na wartość prawdopodobieństwa wystąpienia rodzaju elementu x na badanym diagramie UML, posługując się wzorem na wieloatrybutową macierz decyzyjną.

$$p_{ij} = \frac{x_{ij}}{\sum_{i=1}^{m} x_{ij}} \tag{2}$$

gdzie *j* jest wymiarem elementu UML, jak w równaniu (1), natomiast *i* jest rodzajem elementu UML, a *m* jest liczbą rodzajów elementów UML występujących na badanym diagramie UML.

Wartość entropii (wartość metryki FBS) dla określonego wymiaru architektury oprogramowania opisywanego przez badany diagram UML wylicza się zgodnie ze wzorem Shannona opisującym entropię:

$$FBS_j = \frac{-1}{\ln m_1} \sum_{i=1}^{m} \ln p_{ij} \tag{3}$$

gdzie *m* jest liczbą rodzajów elementów zidentyfikowanych na diagramie UML, natomiast m_1 liczbą wszystkich elementów.

4. Definicja Metryki złożoności oprogramowania FBS

Formalny opis zagadnienia niespójności modeli zaproponowali w 2001 r. G. Spanoudakis i A. Zisman[13]. Opis tego zagadnienia podano w artykule

[13] Z. Spanoudakis, A. Zisman, *Inconsistency Management in Software Engineering: Survey and Open Research Issue*s, w: *Handbook of Software Engineering and Knowledge Engineering*, red. S.K. Chang, World Scientific Publishing Co., Singapore 2001, pp. 329–380.

Niepostyna[5]. Przedstawiono tam również koncepcję Gero[12], w której opis dowolnego systemu (artefaktu) powinien posiadać informacje o jego funkcjonalności, strukturze i behawioryzmie, by taki projekt mógł być wytworzony. Uwzględniając to, przedstawiono poniższe definicje spójności, kompletności i transformowalności architektury oprogramowania.

D1. Definicja kompletności opisu architektury oprogramowania

Architektura oprogramowania jest kompletna (ang. *complete*), gdy składa się z kompletnych warstw diagramów opisujących system informatyczny. Warstwa jest kompletna wtedy, gdy opisuje trzy wymiary architektury oprogramowania (funkcjonalność, behawioryzm, strukturę), tzn. gdy wszystkie wymiary metryki FBS diagramów, opisujące daną warstwę, są mniejsze od 1.0. Poniżej przedstawiono sformalizowaną definicję kompletności warstw, które można interpretować jako modele.

Dana jest para modeli S_i i S_j oraz dwie metryki FBS $FBS_i = \{f_i, b_i, s_i)$ i $FBS_j = \{f_j, b_j, s_j)$ tych modeli S_i i S_j. Mówimy, że S_i i S_j *są kompletne* w odniesieniu do FBS_i i FBS_j, gdy $(f_i < 1.0$ *lub* $f_j < 1.0)$ i $(b_i < 1.0$ lub $b_j < 1.0)$ i $(s_i < 1.0$ lub $s_j < 1.0)$.

D2. Definicja spójności opisu architektury oprogramowania

Architektura oprogramowania jest spójna (ang. *consistent*), gdy składa się z oddzielnych i spójnych warstw diagramów opisujących system informatycznego. Warstwa jest spójna wtedy, gdy składające się na nią diagramy nie opisują razem tego samego wymiaru architektury oprogramowania (funkcjonalność, behawioryzm, strukturę), tzn. gdy tylko jeden diagram opisujący daną warstwę posiada wartość mniejszą niż 1.0 dla co najmniej jednego wymiaru metryki FBS. Poniżej przedstawiono sformalizowaną definicję spójności warstw, które można interpretować jako modele.

Dana jest para modeli S_i i S_j oraz dwie metryki FBS $FBS_i = \{f_i, b_i, s_i)$ i $FBS_j = \{f_j, b_j, s_j)$ tych modeli S_i i S_j opisanych w następujący sposób:

1. S_i i S_j *nie pokrywają się* w odniesieniu do FBS_i i FBS_j jeśli:
   $(f_i = 1.0$ *i* $f_j \leq 1.0$ *lub* $f_i \leq 1.0$ *i* $f_j = 1.0)$ i $(b_i = 1.0$ *i* $b_j \leq 1.0$ lub $b_i \leq 1.0$ i $b_j = 1.0)$ i $(s_i = 1.0$ *i* $s_j \leq 1.0$ lub $s_i \leq 1.0$ *i* $s_j = 1.0)$.

Modele S_i i S_j, które nie pokrywają się, nie mogą być niespójne, zatem modele S_i i S_j, które nie pokrywają się, są na pewno spójne.

D3. Definicja transformowalności opisu architektury oprogramowania

Architektura oprogramowania jest transformowalna, gdy składa się z oddzielnych i transformowalnych warstw diagramów opisujących system informatyczny. Warstwy są transformowalne wtedy, gdy występuje w nich co najmniej jedna para odpowiadających sobie elementów, z których każdy należy do różnych warstw, tzn. wartość co najmniej jednej składowej metryki FBS dla powiązań pomiędzy warstwami posiada wartość mniejszą niż 1.0. Poniżej przedstawiono sformalizowaną definicję transformowalności warstw, które można interpretować jako modele. Dana jest para modeli S_i i S_j oraz metryka $FBS = \{f, b, s\}$ powiązanych elementów tych modeli S_i i S_j. Mówimy, że:

1. S_i i S_j *są transformowalne funkcjonalnie* w odniesieniu do *FBS*, gdy $f<1.0$;
2. S_i i S_j *są transformowalne behawioralnie* w odniesieniu do *FBS*, gdy $b<1.0$;
3. S_i i S_j *są transformowalne strukturalnie* w odniesieniu do *FBS*, gdy $s<1.0$;
4. S_i i S_j *są transformowalne* w odniesieniu do *FBS*, gdy $f<1.0$ lub $b<1.0$ lub $s<1.0$.

5. Przykłady wykorzystania metryki FBS

W niniejszym rozdziale opisano zastosowanie metryki FBS w przemysłowych projektach budowy systemów informatycznych w Polsce. W tabeli 1 zestawiono metryki FBS architektury oprogramowania tych systemów w podziale na poszczególne warstwy kolejnych perspektyw oraz rodzaje (akronimy) diagramów projektowych. Metryki te obliczono w części rejestracji zgłoszenia siedziby stad dla systemu e-IRZ, w części rejestracji komunikatu dla systemu Focus EO oraz w części wydania aktu urodzenia dla systemu pl.ID. Na końcu tabeli oszacowano przybliżoną pracochłonność skonstruowania architektury oprogramowania oraz czas implementacji systemu IT.

Poniżej scharakteryzowano krótko systemy zamieszczonych w tabeli.

- e-IRZ – system e-IRZ (Identyfikacja i Rejestracja Zwierząt) zaprojektowano w 2015 r. dla Agencji Restrukturyzacji i Modernizacji Rolnictwa[14] do obsługi ponad 1 miliona użytkowników, około 1 300 tys. stad oraz 117 mln zgłoszeń

[14] Pełny opis systemu można znaleźć na stronie Agencji Restrukturyzacji i Modernizacji Rolnictwa: http://www.arimr.gov.pl/uploads/media/Model_Architektury_Systemu_e-IRZ.docx (25.08.2016).

zwierzęcych. Rocznie system miał rejestrować 14 tys. nowych producentów, 25 tys. nowych siedzib stad, ponad 6 mln zgłoszeń zwierzęcych.

- Focus EO – system umożliwia elektroniczną wymianę not egzekucyjnych i informacji o klientach banków pomiędzy bankami, a w szczególności instytucjami egzekucyjnymi w ramach systemu Ognivo zarządzanego przez Krajową Izbę Rozliczeniową[15]. System obsługuje dziennie do kilkudziesięciu komunikatów związanych z zajęciami komorniczymi kont bankowych dla około 30 użytkowników jednego Banku.
- pl.ID – system pl.ID zaprojektowano w Centralnym Ośrodku Informatyki w latach 2012–2015 dla ówczesnego Ministerstwa Spraw Wewnętrznych w ramach integracji Systemu Rejestrów Państwowych. Jednym z głównych celów była integracja systemu PESEL z rejestrami prowadzonymi przez Urzędy Stanu Cywilnego[16]. System w założeniach ma obsługiwać (w części BUSC[17]), łącznie około 80 mln tzw. przypisków, natomiast rocznie przewidziano obsługę około 1,2 mln aktów urodzenia, małżeństwa i zgonu. Liczba użytkowników założona została na poziomie około 7,5 tys. (prawie 2500 gmin w całej Polsce ze średnio 3 stanowiskami na jeden urząd).

Dla systemu e-IRZ architektura jest zgodna z metodyką e-CMDA, a realizacja przeznaczona na platformę BPMS. Architektura Focus EO również jest zgodna z metodyką e-CMDA, przy czym wdrożenie nastąpiło na platformie.NET. Architektura oprogramowania systemu pl.ID była opracowana przez COI MSW w formie tzw. metafor zalecanych w przyjętej przez COI metodyce SCRUM. W rezultacie część metafor była opracowana w formie diagramów UML, a część w postaci tekstowej (m.in. opis biznesowych przypadków użycia). System pl.ID zaimplementowano na platformie JEE (Java Enterprise Edition).

[15] Wykonała i wdrożyła firma Risco Software sp. z o.o. w 2014 r. dla jednego z największych banków na świecie.

[16] „Raport z wykonania umowy nr 674/DEP/4.8/2014 z Project-Media Stanisław Niepostyn", https://mc.gov.pl/files/sb_16061609290.pdf (25.08.2016).

[17] System pl.ID posiada ponadto takie moduły jak rejestracja dowodów osobistych (RDO), system odznaczeń państwowych (SOP), centralny rejestr sprzeciwów (CRS), a także obsługa rejestru PESEL.

Tabela 1. Wartości metryk FBS przemysłowych projektów budowy systemów IT (źródło własne)

Perspektywa (View)	Warstwa (Layer)	Symbol diagramu	e-IRZ {F;B;S}	Focus EO {F;B;S}	pl.ID system {F;B;S}
Kontekstowa	Kontekstowa	CTXD {X}	{0.22;0.38;0.37}	{0.23;0.40;0.36}	{0.21;0.21;0.31}
	Dekompozycji procesów	BUCD {B}	{0.32;1.00;1.00}	{0.33;1.00;1.00}	
		PROCD {R}	{1.00;0.24;0.27}	{1.00;0.23;0.28}	
Biznesowa	Procesów biznesowych	RBUCD {A}	{0.29;0.32;0.29}	{0.29;0.44;0.49}	{0.38;0.28;0.24}
	Logiczna	SUCD {U}	{0.29;1.00;1.00}	{0.51;1.00;1.00}	
		BCLD {C}	{1.00;1.00;0.18}	{1.00;1.00;0.95}	{1.00;1.00;0.87}
		BSMD {S}			
Systemowa	Użytkownika	RSUCD {Z}	{0.39;0.28;0.30}		opis tekstowy
	Wewnętrzna	IUCD {Y}	{0.57;1.00;1.00}	{0.30;1.00;1.00}	
		SCLD {J}		{1.00;1.00;0.11}	
		SSMD {T}			
Implementacyjna	Sekwencji	RIUCD {Q}	{0.21;0.27;0.19}	{0.21;0.32;0.19}	
	Komponentowa	CMPD {M}	{0.81;0.81;0.97}	{0.57;0.57;0.79}	{1.00;0.83;0.91}
Czas budowy architektury*liczba osób			1,5 miesiąca*2	2 miesiące*1	5 miesięcy*6
Czas budowy oprogramowania			-	4 miesiące	17 miesięcy
RBUCD – diagram realizacji biznesowego przypadku użycia; BUCD – diagram biznesowych przypadków; BCLD – diagram klas biznesowych; DEPD – diagram wdrożenia; SCLD – diagram klas systemowych; CMPD – diagram komponentów; RIUCD – diagram realizacji wewnątrz systemowych przypadków użycia; PROCD – diagram dekompozycji procesów; BSMD – diagram biznesowej maszyny stanów; SSMD – diagram systemowej maszyny stanów; SUCD – diagram systemowych przypadków użycia; CTXD – diagram (kontekstowy) głównego procesu systemu informacyjnego; IUCD – diagram wewnątrz systemowych przypadków użycia; RSUCD – diagram realizacji systemowego przypadku użycia					

Źródło: opracowanie własne.

Z przedstawionych danych wynika, że nie wszystkie rekomendowane diagramy zostały opracowane[18]. Warto zwrócić uwagę, iż:

- wszystkie systemy posiadają diagramy kontekstowe, które są kompletne, to znaczy opisują wszystkie wymiary architektury oprogramowania;
- wszystkie systemy posiadają diagramy komponentów, które są kompletne, to znaczy opisują wszystkie wymiary architektury oprogramowania (wyjątkiem jest diagram komponentów systemu pl.ID, dla którego brak opisu wymiaru funkcjonalności – pierwsza składowa równa 1.00);
- tylko architektura oprogramowania systemu pl.ID nie posiada kompletnej warstwy komponentowej perspektywy implementacyjnej – jest to spowodowane stosowaniem się do metodyki SCRUM, która nie przewiduje opracowania kompletnej architektury oprogramowania;
- każda warstwa architektury oprogramowania jest spójna dla wszystkich systemów, oprócz systemu pl.ID (brak diagramów);
- kompletność architektury systemów Focus EO oraz e-IRZ w perspektywie biznesowej i logicznej jest względna[19], gdyż z jednej strony platformy BPMS dla systemu e-IRZ wymagają przed wszystkim opisu procesów biznesowych (diagramy RBUCD), z drugiej strony system Focus posiadał już działające formularze (nie opisywano diagramów RSUCD);
- architektura systemu pl.ID nie jest kompletna ani w perspektywie biznesowej, ani w systemowej.

Warto zwrócić uwagę, że w systemach Focus EO oraz e-IRZ można pokazać uporządkowany szereg diagramów od kontekstowego do implementowalnego. Brak szeregu uporządkowanych diagramów w architekturze systemu pl.ID może budzić podejrzenia, że jego opis w perspektywie implementacyjnej nie ma odpowiedniego uzasadnienia. Sytuacje takie zazwyczaj występują w przypadku, gdy system budowany jest w oderwaniu od jego założeń czy wymagań.

Pracochłonność zamodelowania architektury oprogramowania w przypadku zastosowania metodyki e-CMDA nie przekracza 3 osobo-miesięcy bez względu na wielkość systemu[20], natomiast dla projektów IT bez stosowania niniejszego

---

[18] Wynika to z braku potrzeby modelowania takich diagramów (np. dla platform BPMS najważniejszymi są diagramy procesów biznesowych), korzystaniu z tzw. metodyk zwinnych czy innych niedojrzałych metodyk budowy oprogramowania bądź też z braku wiedzy lub doświadczenia w konstruowaniu architektury oprogramowania dużych i złożonych systemów.

[19] Zdefiniowana jako sytuacja, w której obecność wybranych diagramów jest wystarczająca do zbudowania poprawnego systemu.

[20] System e-IRZ jest o wiele większy i bardziej skomplikowany aniżeli system Focus EO, ale porównywalny z systemem pl.ID.

podejścia pracochłonność może być nawet kilkunastokrotnie większa, jak to się okazało przy ocenie projektu pl.ID. Wykorzystanie metryki FBS w metodyce e-CMDA umożliwia skonstruowanie zrozumiałej, przejrzystej, spójnej, kompletnej i transformowalnej architektury w porównaniu z tradycyjną architekturą oprogramowania (wodospadową czy też stosowaną w metodykach typu RUP) bądź tzw. metaforami czy innymi artefaktami architektury w tzw. zwinnych metodach budowy oprogramowania. Okazało się, że czas budowy oprogramowania w przypadku systemu Focus EO przez zespół dwóch programistów to 4 miesiące, natomiast dla systemu pl.ID wyniósł on 17 miesięcy w zespole złożonym z około 15 programistów.

6. Podsumowanie i kierunki dalszych badań

W świetle powyższych obserwacji można wysnuć następujące wnioski:

- obliczanie i analizowanie entropii diagramów UML za pomocą metryki FBS jest sposobem udoskonalenia metody budowy architektury oprogramowania, prowadzącym do obiektywizacji wyników oceny tych architektur.
- Systemy informatyczne budowane na podstawie spójnej i kompletnej architektury zgodnej z e-CMDA z zastosowaniem metryki FBS cechują się krótszym czasem budowy, mniejszą liczbą błędów, a także mniejszą wielkością zasobów niezbędnych do ich budowy i wdrożenia.

Koszty, jakość oraz czas realizacji[21] dużych i złożonych systemów informatycznych realizowanych ze środków publicznych często odbiegają od początkowych założeń. Jest to zazwyczaj spowodowane brakiem bądź wadliwą konstrukcją architektury oprogramowania. Metryka złożoności architektury oprogramowania FBS byłaby dobrym narzędziem do niezwykle szybkiej i obiektywnej oceny architektury oprogramowania. Już przed rozpoczęciem budowy oprogramowania zamawiający system informatyczny miałby efektywne narzędzie do określenia stopnia przygotowania wykonawcy do budowy systemu informatycznego. Co więcej, po uruchomieniu zamówionego systemu IT zamawiający mógłby również ocenić stan i jakość dokumentacji (architektury oprogramowania) oddanego systemu IT przy zastosowaniu metryki FBS. Również w trakcie budowy systemu informatycznego zamawiający mógłby w prosty i szybki sposób

[21] Systemy te realizowane są najczęściej przez podmioty zewnętrzne wyspecjalizowane w budowie i integracji systemów IT.

sprawdzić prawidłowość modyfikowanej architektury oprogramowania. Zatem metryka FBS wydaje się narzędziem niezwykle szybko diagnozującym wymienione niedostatki informatyzacji państwa.

Bibliografia

Abran A., *Software Metrics and Software Metrology*, Wiley-IEEE Computer Society, 2010.

Gero J.S., *Design Prototypes: a Knowledge Representation Schema for Design*, „AI Magazine" 1990, vol. 11, no. 4.

Niepostyn S.J., *Koncepcja architektury sterowanej regułami spójności modeli (CMDA) na przykładzie systemu e-IRZ dla ARiMR*, TIAPISZ 2015.

Spanoudakis G., Zisman A., *Inconsistency Management in Software Engineering: Survey and Open Research Issues*, w: *Handbook of Software Engineering and Knowledge Engineering*, red. S.K. Chang, World Scientific Publishing Co., Singapore 2001.

Yi T., *On the Application of Information Entropy-Based Multi-Attribute Decision in UML Class Diagram Metrics*, "International Journal of u- and e-Service" 2015, vol. 8.

Zhao J., *On Assessing the Complexity of Software Architectures*, ISAW '98 Proceedings of the third international workshop on Software architecture, 1998.

Źródła sieciowe

Model architektury systemu e-IRZ, http://www.arimr.gov.pl/uploads/media/Model_Architektury_Systemu_e-IRZ.docx (25.08.2016).

Raport z wykonania umowy nr 674/DEP/4.8/2014 z Project-Media Stanisław Niepostyn, https://mc.gov.pl/files/sb_16061609290.pdf (25.08.2016).

Unified Modeling Language, http://www.omg.org (25.08.2016).

* * *

Applying FBS Entropy-Based Metric to Assess the Completeness and Transformability of the Complex IT Systems Software Architecture for the Business Cases of Public Systems pl.ID (Źródło) and e-IRZ ARiMR

Abstract

An improved method of complex computer systems design could be implemented in the environment of the object-oriented software, in particular in the BPM environment (Business Process Management) and using the Enhanced CMDA concept (shortly named e-CMDA, where CMDA denotes the Consistent Model Driven Architecture). Thus, it enables automating the design of the underlying software architecture while applying consistency rules of UML diagram series which are based on the calculation of entropy of UML diagrams using the FBS software architecture metric.

IT systems produced along the e-CMDA are built up quicker, more productively, and less erroneous, which makes such architectures particularly useful for agile development methods. The methodology of e-CMDA facilitates direct comparison from the implementability assessments of different software architectures by means of objective calculation of consistency, completeness and transformability of diagrams.

Keywords: software architecture, consistency, completeness and transformability of UML diagrams, UML diagrams entropy, software architecture dimensions, FBS, UML, agile software architecture