MODEL OCENY JAKOŚCI IMPLEMENTACJI WZORCÓW PROJEKTOWYCH W OPROGRAMOWANIU

Rafał WOJSZCZYK

Streszczenie: Istnieje wiele metod i dobrych praktyk mających na celu zapewnienie jakości wytwarzanego oprogramowania, jedną z owych praktyk jest wykorzystanie wzorców projektowych. Celem artykułu jest przedstawienie autorskiej metody opartej na modelu oceny jakości implementacji wzorców projektowych. Metoda zakłada weryfikację różnych aspektów implementacji wzorców oraz wyrażenie otrzymanych wyników w postaci liczbowej. Analiza otrzymanych wyników umożliwia wykrywanie błędów, które są trudne do zidentyfikowania podczas inspekcji kodu lub testowania.

Słowa kluczowe: wzorce projektowe, jakość oprogramowania, ocena jakości.

1. Wstęp

Oprogramowanie komputerowe to bardzo specyficzna grupa produktów klasyfikowanych jako niematerialne. W odróżnieniu od produktów materialnych oprogramowanie: nie psuje się, nie zużywa oraz nie wymaga wymiany części. Wydawać by się mogło, że raz wytworzone oprogramowanie winno działać bez przerwy i raz na zawsze. Niestety rzeczywistość wielokrotnie udowodniła, że oprogramowanie musi się zmieniać i dostosowywać do potrzeb ludzi. Powszechnie znanym przykładem wymuszonych zmian w oprogramowaniu, na przełomie 1999 i 2000 roku, była tzw. pluskwa milenijna. Uważano wtedy, że większość komputerów przestanie prawidłowo pracować ze względu na dwucyfrowy system dat. Innym przykładem zmian, na które część programów komputerowych nie było przygotowanych to wprowadzenie nowych stawek VAT w Polsce w 2011 roku. Niektóre programy wymagały poniesienia dodatkowych kosztów przez producentów, tj. ingerencji w kod źródłowy, aby wprowadzić nowe stawki. Producenci mogli uniknąć kosztów poprzez wcześniejszą implementację odpowiednich mechanizmów rozszerzeń i modyfikacji, które są realizowane między innymi przez wzorce projektowe [1].

Wykorzystanie wzorców projektowych jest bardzo ważne w całym procesie wytwórczym oprogramowania, ponieważ są uznawane za pewnego rodzaju język komunikacji. Język ten sprawia, że kod programu, który zawiera wzorce projektowe będzie łatwiejszy w poznaniu i zrozumieniu niż ten, który nie zawiera wzorców. Jest to szczególnie ważne w sytuacji zaprzestania rozwoju oprogramowania lub zmian w składzie zespołu wytwórczego odpowiedzialnego za dane oprogramowanie.

Celem pracy jest przedstawienie autorskiej metody opartej na modelu oceny jakości implementacji wzorców projektowych. Metoda została zaimplementowana i zweryfikowana w dedykowanym pakiecie oprogramowania użytkowego.

Drugi rozdział pracy wyjaśnia wpływ wzorców projektowych na jakość oprogramowania oraz przedstawia wybrane badania związane z wzorcami projektowymi. Rozdział trzeci i czwarty to odpowiednio opis ram modelu oraz opartej na nim metody. W piątym rozdziale przedstawiono przykładowe wyniki. Ostatni, szósty rozdział, stanowi podsumowanie pracy.

2. Jakość oprogramowania oraz wzorce projektowe

Jakość oprogramowania to bardzo ogólne pojęcie i nie możliwe jest zdefiniowanie miary, która bezpośrednio wyznacza tę wartość. Analizując różne modele jakości oprogramowania [2], w tym charakterystykę konserwowalności (ang. maintainability) zawartą w trzeciej części normy ISO/IEC 9126 [3] można wywnioskować, że o wewnętrznej jakości programu (tj. jakości kodu programu) świadczą charakterystyki: łatwość rozbudowy, łatwość modyfikacji, łatwość zrozumienia kodu. Implementacja wzorców projektowych w oprogramowaniu sprzyja pozytywnym wartościom wymienionych charakterystyk (pozytywnym tzn. dzięki wykorzystaniu wzorców projektowych oprogramowanie jest łatwiejsze w rozbudowie itp.) [1]. Zatem można przyjąć, że jakość implementacji wzorców projektowych jest jednym z czynników świadczących o jakości oprogramowania.

Jakość implementacji wzorców projektowych, podobnie jak ogólna jakość oprogramowania, nie posiada bezpośredniej miary. Przyjęło się w społeczeństwie programistów, że jednym z podstawowych czynników świadczących o implementacji wzorców jest weryfikacja poprawności struktury tworzącej wzorce względem ogólnie znanych szablonów z [1]. Szablony nie przewidują wszystkich możliwych wariantów, stąd pojawia się znaczące zróżnicowanie implementacji, co ostatecznie utrudnia weryfikację. Struktura tworząca wzorce jest wyłącznie jednym z aspektów w procesie oceny wzorców projektowych. Istnieje wiele innych cech, które wpływają na jakość implementacji, np. zniekształcenia i brudy (potocznie z ang. distortion oraz bad smells), nachodzenie na inne wzorce projektowe, typowe błędy. Ponadto występują jeszcze inne utrudnienia związane z analizowaniem implementacji wzorców projektowych, zostało to przedstawione między innymi w [4] oraz [5]. Różne utrudnienia mogą prowadzić do pozornej sytuacji, w której zachowana jest poprawność strukturalna implementacji wzorca, ale wykorzystanie nie jest zgodne z intencją lub nie spełnia założonego celu.

Wzorce projektowe opisane w [1] to szablony gotowych mechanizmów, które można wykorzystać do rozwiązania typowych problemów pojawiających się cyklicznie w projektowaniu i programowaniu obiektowym. Jednakże nie są to gotowe rozwiązania, ponieważ wykorzystanie każdego wzorca wymaga odpowiedniej implementacji zgodnie z kontekstem oprogramowania.

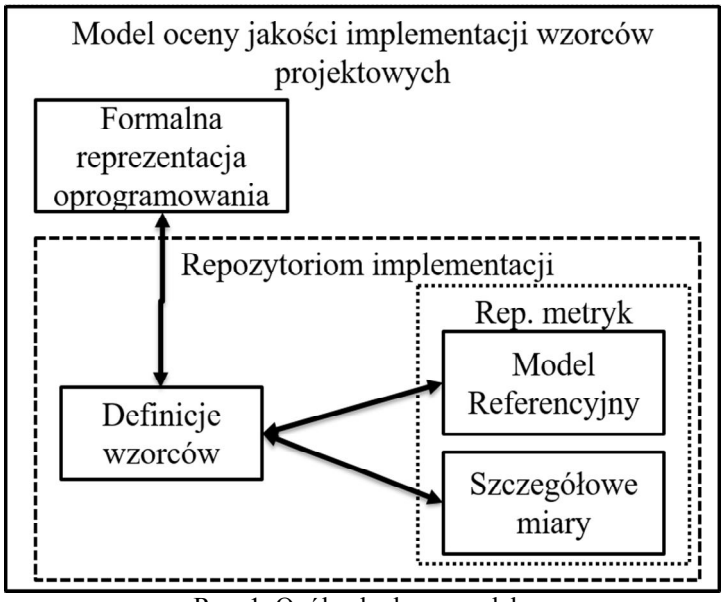
Zagadnienia poruszane w badaniach związanych z wzorcami projektowymi bardzo często dotyczą problemu wyszukiwania wystąpień wzorców projektowych w oprogramowaniu [6, 7, 8]. Miarą wymienionych badań jest przedstawienie ilości wystąpień wzorców, co jest niewystarczające aby uznać to za znamiona jakości. Podejmowane są również badania, które mają na celu głównie wykazanie poprawności strukturalnej implementacji wzorców [9], co również dostarcza zbyt małą ilość informacji. Inne próby liczbowego wyrażenia wzorców projektowych zostały przedstawione w [10] oraz [11], polegają na zastosowaniu znanych metryk oprogramowania obiektowego lub autorskich metryk do oprogramowania zawierającego wzorce projektowe. Wiele lat badań nad metrykami dostarczyło zalecane wartości metryk dla ogólnego przypadku oprogramowania, których uzyskanie wskazuje na wysoką jakość. Natomiast w [12] wykazano, że występowanie wzorców projektowych nie wpływa pozytywnie na wyniki uzyskiwane z metryk, zatem należałoby dokonać odpowiedniej nadinterpretacji metryk w zastosowaniu do wzorców, zostało to przedstawione w [13].

3. Ramy modelu oceny jakości

3.1. Ogólne założenia

Podstawowe założenia modelu zostały opisane w [4], natomiast rysunek 1 przedstawia ogólną budowę proponowanego modelu. Model zakłada opisanie badanego oprogramowania za pomocą formalnej reprezentacji, dokładniej jest to odpowiednio przygotowana struktura danych, wywodząca się z założeń paradygmatu programowania obiektowego [14]. Repozytorium implementacji wzorców odpowiedzialne jest za opisanie informacji o wzorcach projektowych, zawiera charakterystyki i miary jakości. Charakterystyki, dokładniej to aspekty oraz zawarte w nich cechy, opisują definicje wzorców. Metryki niezbędne do pomiaru wartości wspomnianych charakterystyk są opisane przez repozytorium metryk, które składa się z modelu referencyjnego oraz zbioru szczegółowych miar.

Model dostarcza niezbędne elementy do wyrażenia jakości w postaci liczbowej, co następnie umożliwia wyznaczenie wymaganego poziomu jakości względem badanego oprogramowania oraz porównywanie kolejnych wydań. Dodatkowo dane zawarte w metrykach pozwalają na uzyskanie informacji jakich zmian należy dokonać w badanym oprogramowaniu aby osiągnąć wyższą jakość.



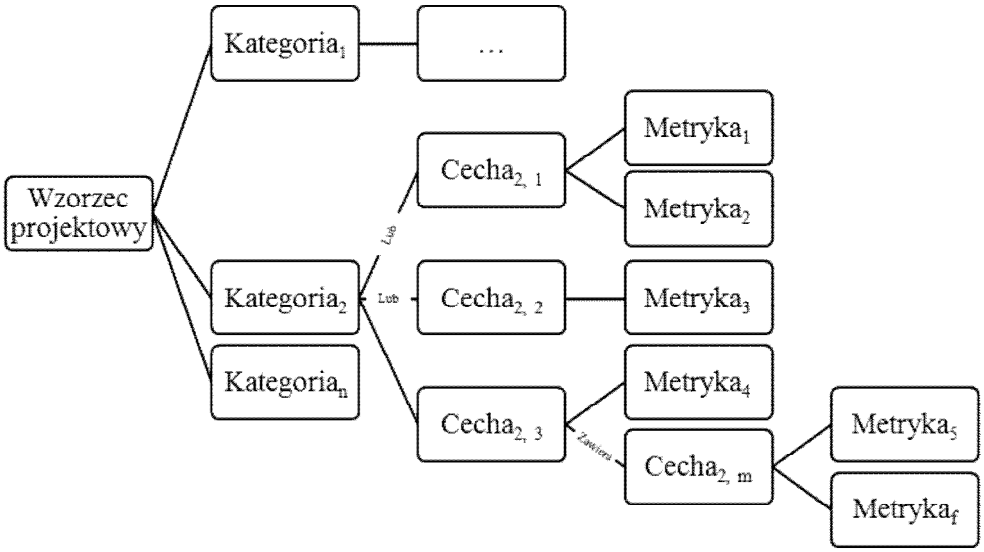
Rys. 1. Ogólna budowa modelu

3.2. Definicje wzorców projektowych

Definicje wzorców projektowych opisane są przez hierarchiczną strukturę danych, która reprezentuje zbiór cech wymaganych w implementacji rozpatrywanych wzorców. Rysunek 2 przedstawia symboliczną hierarchię definicji wzorców. Korzeniem w hierarchii jest konkretny wzorzec projektowy, pierwszym poziomem są ogólne kategorie (aspekty), w których pogrupowane zostały cechy. Poziom cech dotyczy konkretnych własności

wzorców projektowych a semantyka tego poziomu pozwala na definiowanie relacji "lub", "i" oraz "zawiera" pomiędzy cechami. Każda cecha może być skojarzona z elementami w repozytorium metryk, które szczegółowo opisuje daną cechę. Dodatkowo cechy scharakteryzowane są przez dane liczbowe informujące o krotności wystąpień oraz ocenie, co wyjaśnia tabela 1. Współczynnik wagi pozwala na określenie ważności danej cechy. Współczynniki mogą być ustalane na podstawie wiedzy eksperta, literatury oraz w zależności od kontekstu, stąd możliwe jest profilowanie definicji wzorców (dobór odpowiednich współczynników wagi w zależności od rodzaju badanego oprogramowania).

Oddzielenie definicji wzorców od metryk pozwala na zwiększenie szczegółowości opisywanych danych poprzez dobór najbardziej odpowiednich metryk, dodatkowo umożliwia to rozwój na wypadek, gdyby dotychczasowe metryki okazały się niewystarczające.



Rys. 2. Symboliczna reprezentacja definicji wzorców

Tab. 1. Dane charakteryzujące cechę

Nazwa własności	**Opis**
Nazwa	Słowne określenie cechy, zrozumiałe dla człowieka.
Zależność od cechy	Wskazanie na inną cechę i określenie rodzaju zależności od niej: i, lub, zawiera.
Ocena	Złożona struktura danych scharakteryzowana przez współczynnik wagi (ważności danej cechy) oraz próg aktywacji.
Krotność	Złożona struktura danych, która zawiera informacje o ilości wystąpienia danej cechy w badanym oprogramowaniu. Określa oczekiwaną wartość w postaci przedziałów: od do, równe lub większe, mniej niż, dokładnie.
Skojarzone metryki	Złożona struktura danych, która wskazuje na szczegółowe miary oraz skojarzone z nimi możliwe wartości.

3.3. Repozytorium metryk

W modelu przewidziane zostały dwa rodzaje metryk: opis szablonów wzorców projektowych w modelu referencyjnym oraz szczegółowe miary. Pierwszy rodzaj jest dedykowany do opisu generycznej struktury tworzącej wzorce, ponieważ jest to jeden z podstawowych aspektów, który występuje w każdym wzorcu. Model referencyjny składa się z rozbudowanej struktury danych, która bazuje na założeniach paradygmatu programowania obiektowego [15]. Umożliwia zdefiniowanie różnych wariantów poszczególnych elementów wzorców projektowych z określeniem stopnia dopasowania do założonego ideału. Drugi rodzaj metryk, tj. szczegółowe miary, reprezentuje funkcje matematyczne, które operują na mierzalnych atrybutach programowania obiektowego [13]. Z każdą funkcją skojarzony jest zbiór zalecanych wartości, co ponownie przekłada się na poziom dopasowania.

Metryki, pod względem zwracanego wyniku, są wobec siebie polimorficzne. Wynik każdej metryki powinien być zwracany w postaci ustandaryzowanej, jako maksyment oraz w określonym przedziale wartości.

4. Realizacja metody

Model wyznacza pewne ramy, w oparciu o które została zaproponowana autorska metoda pozwalająca na weryfikację modelu. Metoda charakteryzuję się dwoma etapami postępowania, które zostały zaimplementowane jako jedno oprogramowanie użytkowe wykonane w technologii Microsoft .NET. Aktualne zastosowanie narzędzia ogranicza się do wybranych wzorców z [3].

4.1. Pozyskanie oprogramowania

Pierwszym etapem jest pozyskanie oprogramowania, które będzie poddane ocenie jakości implementacji wzorców projektowych. Pozyskanie polega na przekształceniu kodu źródłowego programu do formalnej reprezentacji oprogramowania, które wyłącznie w tej postaci jest wykorzystywane przez dalsze etapy. Wykorzystana technologia usprawniła proces pozyskania badanego oprogramowania, który sprowadza się do akwizycji struktury obiektowej z kodu zarządzanego CIL przy wykorzystaniu jednej z metod inżynierii odwrotnej, np. Mono.Cecil [14]. Wykorzystanie kodu zarządzanego pozwoliło na uniknięcie typowych błędów syntaktycznych występujących w kodzie źródłowym oraz wyeliminowało z badanego oprogramowania zbędne fragmenty kodów, np. kod testów jednostkowych, dyrektywy preprocesora i inne. Dodatkowa korzyść wynikająca z wykorzystanej technologii to możliwość zastosowania metody do różnych języków programowania kompilowanych do kodu zarządzanego. Formalna reprezentacja danych zawierająca pozyskane oprogramowanie została zrealizowana jako baza danych w środowisku Microsoft SQL Server.

4.2. Analiza oprogramowania

Etap analizy wykorzystuje dane zawarte w reprezentacji formalnej oraz dane opisywane przez repozytorium implementacji wzorców projektowych. Analiza cech odbywa się zgodnie z hierarchią opisaną przez definicje wzorców, tj. aby uzyskać charakterystykę danej cechy wzorca projektowego, należy uprzednio wyznaczyć wartości metryk, które

opisują tą cechę. Dokładniej opisuje to następująca procedura: dla każdej kategorii (inaczej aspektu) należy wykonać weryfikację cech, które posiada, zaś dla każdej weryfikowanej cechy należy wyznaczyć wartości metryk charakteryzujących daną cechę. W każdym kroku weryfikacji uzyskiwany jest wynik cząstkowy. Jeśli wynik nie spełnia progu aktywacji to element lub cecha nie występuje, w szczególnym przypadku jeśli próg aktywacji wynosi 0 to wystąpienie elementu lub cechy jest obowiązkowe. Każdy etap weryfikacji obarczony jest sprawdzeniem występujących ograniczeń, np. związanych z krotnością (ilością) występujących elementów. Opracowana struktura opisu definicji wzorców pozwoliła na przystępną realizację głównego algorytmu analizy: w pętli iterowane są kolejne cechy i dla każdej wykonywana jest analiza. W przypadku zależności „zawiera" wykonywana jest funkcja rekurencyjna w głąb hierarchii. Algorytm analizy traktuje metryki polimorficznie, dzięki czemu rodzaj metryki nie ma znaczenia.

Repozytorium metryk jest jednokrotnie uzupełniane przed pierwszym wykorzystaniem metody. Przykładowe dane opisujące wzorzec projektowy Singleton przedstawia tabela 2. Metoda nie przewiduje wyszukiwania wystąpień wzorców, stąd wymagane jest wcześniejsze wskazanie rdzenia wzorca (np. klasa wzorca Singleton, interfejs wzorca Strategia) w badanym oprogramowaniu.

Tab. 2. Przykładowy opis wzorca Singleton, rodzaj metryki odpowiada odpowiednio: MR - model referencyjny, SM - szczegółowe miary

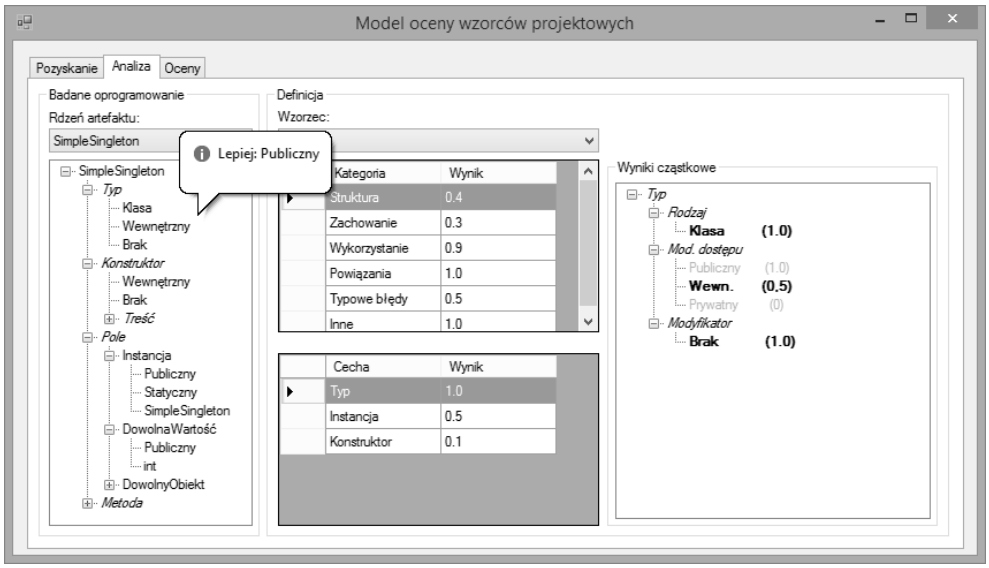
Kategoria	**Cecha**	**Wartości opisujące cechę**	**Rodzaj metryki**
Struktura tworząca wzorzec	Typ	Nieabstrakcyjna klasa, brak dziedziczenia	MR
	Instancja	Prywatne, statyczne pole zwracane przez publiczną statyczną właściwość identycznego typu jak klasa zawierająca, sugerowana nazwa to Instance	MR
	Konstruktor	Prywatny	MR
Zachowanie	Inicjalizacja	Sprawdzanie istnienia obiektu oraz tworzenie przy pierwszym wykorzystaniu	MR i SM
	Wielowątkowość	Synchronizacja dostępu do instancji	SM
Wykorzystanie	Poprzez asocjację	Przynajmniej jeden udziałowiec i nie więcej niż 1/2 sumy klas i interfejsów w badanym oprogramowaniu	SM
	Poprzez dziedziczenie	Brak dziedziczenia z klasy wzorca	MR
Inne	Błędy	Zabronione są konstruktory inne niż prywatne, powinien występować wyłącznie jeden element pełniący rolę instancji	MR
	Zawartość	Odpowiednia ilość niestatycznych elementów składowych (pola, metody) zawartych w klasie wzorca	SM
	Powiązania z innymi wzorcami	Fabryka abstrakcyjna	MR i SM

4.3. Zastosowanie

Główny obszar zastosowania metody to etap implementacji w procesach wytwórczych oprogramowania. Metoda może być wykorzystana równolegle z innymi narzędziami dbającymi o zapewnienie jakości produktu programowego, a w tym dobrych praktyk oraz czystości kodu. Jest to szczególnie przydatne w firmach lub zespołach deweloperskich, dla których wysokim priorytetem jest zapewnienie łatwej konserwacji i rozbudowy oprogramowania.

Prototypową realizację narzędzia przedstawia rysunek 3. Po wybraniu żądanej kategorii, a następnie cechy (zob. rys. 3), program wyświetla dodatkowe informacje wynikające z analizy danej cechy. W widocznych po prawej stronie wynikach metryk zaznaczone są wartości, które wystąpiły dla badanego oprogramowania (w nawiasie podany jest poziom dopasowania zgodnie z modelem referencyjnym). W przypadku szczegółowych miar prezentowany jest uzyskany wynik oraz informacja o możliwych wartościach. Natomiast po lewej stronie widoczny jest ekwiwalent badanego oprogramowania oraz wskazywane są podpowiedzi lepszych rozwiązań i komunikaty o wykrytych błędach z sugestią poprawy.

Bardzo praktycznym zastosowaniem byłoby zintegrowanie narzędzia z środowiskami programistycznymi, aby na bieżąco śledzić i oceniać zmiany związane z implementowanymi wzorcami projektowymi. Ostatecznie warto zastanowić się nad realizacją metody pozwalającej na działanie w przeciwnym kierunku, tj. do przynajmniej półautomatycznej implementacji wzorców w oprogramowaniu na podstawie dostarczonych definicji.



Rys. 3. Prototypowe narzędzie wspierające metodę

5. Interpretacja wyników

Opisana wcześniej metoda pozwala na dostarczenie wyników metryk dla badanego oprogramowania. Wyniki o postaci liczbowej wyrażają ogólną jakość implementacji danej cechy wzorca projektowego, jak też po uwzględnieniu wagi każdej cechy możliwe jest

wyznaczenie wyniku dla całego aspektu. Wyniki przyjmują zakres wartości od 0 do 1, zatem w sytuacji, gdy wynik przynajmniej jednej z metryk jest poniżej 1 należy przeanalizować dane zawarte w metrykach (np. alternatywne rozwiązania opisane w modelu referencyjnym) i podjąć działania mające na celu poprawienie implementacji danego wzorca projektowego. Wyniki poniżej 1 oznaczają możliwość wystąpienia potencjalnych błędów, a dla każdej metryki można przewidzieć grupy takich problemów. Jednakże nie w każdym przypadku może być wymagany możliwie wysoki wynik, np. w aplikacji, w której nie występuje wielowątkowość niepotrzebna jest synchronizacja dostępu do instancji wzorca Singleton.

Przykład uzyskanych wyników (zaokrąglonych do części dziesiętnych) dla trzech różnych wystąpień wzorca Singleton został przedstawiony w tabeli 3.

Tab. 3. Przykładowe wyniki

Kategoria	**Przypadek 1**	**Przypadek 2**	**Przypadek 3**
Struktura	1	0,4	1
Zachowanie	0,7	0,3	0,7
Wykorzystanie	0,6	0,9	0,1
Powiązania z innymi wzorcami	1	1	0
Typowe błędy	1	0,5	1
Inne	1	1	0,3

Implementację w przypadku nr 1 można uznać za dobrej jakości. Została zachowana wysoka poprawność strukturalna oraz powiązania z innymi wzorcami. Maksymalna ocena w kategorii typowe błędy świadczy o tym, że takowe błędy nie wystąpiły. Gorsza ocena w kategorii zachowania wynika z braku synchronizacji odwołań do instancji singletonu w środowiskach wielowątkowych. Natomiast niższa ocena w kategorii wykorzystania wynika z zbyt dużej ilości odwołań do instancji co może wskazywać na zbyt dużą odpowiedzialność klasy z wzorcem Singleton.

Przypadek nr 2 wykazał niską jakość implementacji w kategorii poprawności strukturalnej, typowych błędów oraz zachowania. Potencjalne błędy w oprogramowaniu zawierającym taką implementację objawią się nieprawidłowym działaniem programu, np. wystąpią błędy związane z niezgodnością danych lub tzw. błędy runtimowe. Błędy te powinny zostać wykryte jeszcze na etapie wytwarzania oprogramowania (faza testowania) a koszt naprawy powinien zmieścić się w przewidzianym budżecie (na poprawy błędów) w ogólnym procesie wytwórczym. Jednakże należy mieć na uwadze, że wzorzec projektowy Singleton jest uznawany za jeden z najprostszych wzorców, zatem koszty naprawy mogą wzrosnąć w przypadku innych wzorców.

Przypadek nr 3 wykazał niską jakość implementacji w kategoriach wykorzystania, powiązania z innymi wzorcami oraz pozostałych cechach związanych z nadmierną ilością elementów składowych. Analiza tego przypadku wykazała, że wzorzec został zaimplementowany niezgodnie z intencją wykorzystania lub został niepoprawnie zrozumiany przez implementującego dewelopera. Problemy wynikające z owej implementacji są trudne do wykrycia, zarówno podczas inspekcji kodu jak też w fazie testowania. Najbardziej zauważalne błędy pojawią się w trakcie konserwacji i rozwoju oprogramowania. Bardzo trudno jest przewidzieć koszt naprawy.

6. Podsumowanie

W pracy wyjaśniono krótko wpływ jakości implementacji wzorców projektowych na wewnętrzną jakość oprogramowania oraz przedstawiono wybrane badania związane z weryfikacją implementacji wzorców projektowych.

Omówiono model oraz opartą na nim autorską metodę, która umożliwia wyrażenie jakości implementacji wzorców projektowych w postaci liczbowej. Model składa się z hierarchicznej struktury opisującej definicje wzorców (charakterystyki) oraz zbioru metryk niezbędnych do wyznaczenia wyników. Proponowana metoda dostarcza wyniki metryk, które są pogrupowane odpowiednio w różnych kategoriach. Wyniki poniżej zalecanej wartości oznaczają ryzyko wystąpienia potencjalnych błędów. Analiza wyników oraz metryk pozwala przewidzieć możliwe utrudnienia związane z rozbudową oprogramowania i niepoprawnym działaniem.

Kierunek dalszych prac to rozbudowa metody, w tym: opracowanie szczegółowych miar, automatyzacja przewidywania błędów oraz sugerowania zmian na podstawie uzyskanych wyników, rozbudowa i integracja narzędzia. Następnie przewidywane jest opracowanie definicji i metryk dla większej ilości wzorców projektowych.

Literatura

1. Gamma E. i inni: WZORCE PROJEKTOWE Elementy oprogramowania wielokrotnego użytku. Helion, Gliwice, 2010.
2. Kan S. H.: Metryki i modele w inżynierii jakości oprogramowania, PWN SA, Warszawa, 2006.
3. ISO/IEC TR 9126-3, Software Engineering – Part 3: Internal metrics, ISO/IEC 2003.
4. Wojszczyk R.: Koncepcja hybrydowej metody do oceny jakości zaimplementowanych wzorców projektowych. Zeszyty Naukowe Wydziału Elektroniki i Informatyki nr 7, strony od 17 do 26, Wydawnictwo Uczelniane Politechniki Koszalińskiej, ISSN 1897-7421, Koszalin 2015.
5. Rasool G.: Customizable Feature based Design Pattern Recognition Integrating Multiple Techniques. Dysertacja Doktorska, Technische Universitat Ilmenau, Ilmenau 2010.
6. Tsantalis N. i inni: Design Pattern Detection Using Similarity Scoring. IEEE Transactions on Software Engineering, Volume: 32, Issue: 11, s. 896-908, 2006.
7. Singh Rao R., Gupta M.: Design Pattern Detection by Greedy Algorithm Using Inexact Graph Matching. International Journal Of Engineering And Computer Science, Volume 2 Issue 10, s. 3658-3664, 2013.
8. Binun A.: High Accuracy Design Pattern Detection. Dysertacja doktorska, Rheinischen Friedrich Wilhelms Universitat Bonn, 2012.
9. Blewitt A.: HEDGEHOG: Automatic Verification of Design Patterns in Java. Dysertacja doktorska, University of Edinburgh, 2006.
10. Khaer Md. A. i inni: An Empirical Analysis of Software Systems for Measurement of Design Quality Level Based on Design Patterns. Computer and information technology, IEEE, 2007.
11. Masuda G., Sakamoto N., Ushijima K.: Evaluation and Analysis of Applying Design Patterns. IWPSE - International Workshop on Principles of Software Evolution, 1999.
12. Hernandez J. i inni: Selection of Metrics for Predicting the Appropriate Application of design patterns. 2nd Asian Conference on Pattern Languages of Programs, 2011.

13. Wojszczyk R.: Zestawienie metryk oprogramowania obiektowego opartych na statycznej analizie kodu źródłowego. Zarządzanie projektami i modelowanie procesów, strony od 95 do 107, Zeszyty Rady Naukowej Polskiego Towarzystwa Informatycznego, ISBN 978-83-7518-599-7, Warszawa 2013.
14. Wojszczyk R.: Pozyskiwanie struktury obiektowej z kodu zarządzanego przy wykorzystaniu metod inżynierii odwrotnej. Inżynieria oprogramowania: badania i praktyka, strony od 199 do 213, Zeszyty Rady Naukowej Polskiego Towarzystwa Informatycznego, ISBN 978-83-63919-15-3, Warszawa 2014.
15. Wojszczyk R.: Weryfikacja poprawności implementacji struktury wzorców projektowych w oparciu o model referencyjny, w: Od procesów do oprogramowania: badania i praktyka, strony od 73 do 83, Zeszyty Rady Naukowej Polskiego Towarzystwa Informatycznego, ISBN 978-83-60810-73-6, Warszawa 2015.

Mgr inż. Rafał WOJSZCZYK
Zakład Podstaw Informatyki i Zarządzania
Wydział Elektroniki i Informatyki
Politechnika Koszalińska,
75-453 Koszalin, ul. Śniadeckich 2
tel. (94) 343 34 85, tel. (94) 3478-706, fax (94) 343 34 79
e-mail: rafal.wojszczyk@tu.koszalin.pl