Gdańsk, dn. 2022/11/28

Recenzja rozprawy doktorskiej

mgr inż. Jarosława Hryszko

pt.

„zastosowanie predykcji defektów w komercyjnych projektach rozwoju oprogramowania – od rozważań teoretycznych do codziennej praktyki”.

Rozprawa została mi przekazana do recenzji w formie oprawionego maszynopisu (książki) liczącego 146 stron w języku polskim. Poniżej przedstawiam w kolejnych punktach moją szczegółową ocenę stopnia spełnienia przez nią trzech wymogów stawianych rozprawom doktorskim w art. 187, ust.1 i 2 ustawy „Prawo o szkolnictwie wyższym i nauce” z dnia 20 lipca 2018r. z późniejszymi zmianami (zwanej dalej w tej recenzji “Ustawą”) – dotyczących odpowiednio oryginalności opisanego rozwiązania, wykazanej wiedzy teoretycznej oraz zademonstrowanej umiejętności samodzielnego prowadzenia przez Doktoranta pracy naukowej. W podsumowaniu wskazuję także na zauważone przeze mnie niedociągnięcia i słabe strony Rozprawy oraz formułuję wniosek dotyczący jej dalszego procedowania.

Problem rozprawy i ocena jego rozwiązania

1. *Motywacja podjęcia badań i uzasadnienie celu Rozprawy:*

Chociaż inżyniera oprogramowania stanowiąca ważną gałąź informatyki technicznej nie dysponuje jeszcze zbiorem podstawowych praw i zależności odkrywanych przez nauki przyrodnicze jak w przypadku innych obszarów nauk technicznych, to tak jak one również zajmuje się tworzeniem produktów w postaci struktur i systemów użytecznych dla człowieka. I także jak one stara się dostarczać metody systematycznego wytwarzania swoich produktów – w sposób opłacalny dla wytwórców i na poziomie jakości akceptowalnym przez odbiorców. Stąd zainteresowanie Doktoranta zagadnieniem predykcji defektów, stanowiącym ważny komponent wszelkich metodyk zapewniania jakości produktu oraz projakościowym zarządzaniem procesem wytwórczym. Zagadnieniem tym rządzą dwie zasady empirycznie potwierdzone w niezliczonych projektach różnej skali. Pierwsza z nich sformułowana przed laty przez Barry Boehma mówi, że najgorsze (bo najbardziej kosztowne do usunięcia w późnych fazach projektu) są błędy wprowadzane do wytwarzanego produktu we wczesnych fazach projektu. W klasycznym modelu kaskadowym oznacza to fazę formułowania wymagań lub projektowania konstrukcji, zaś w nowszych modelach iteracyjnych, jak np. w modelu kolejnych wersji czy modelu spiralnym rozciąga się praktycznie na wszystkie fazy, w tym szczególnie na fazę kodowania. Druga zasada, zwana zasadą Pareto, ma swoje źródło w statystycznej technice podejmowania decyzji – gdy mamy do czynienia z niewielką liczbą czynników powodujących znaczący efekt ogólny. Dla zagadnienia predykcji defektów w kodzie oznacza to, że stosunkowo niewielkie fragmenty kodu mogą być odpowiedzialne za większość defektów dających się wykryć w drodze testowania. Stąd koncepcja automatycznej predykcji defektów w kodzie poprzez poszukiwanie najbardziej „defektogennych” fragmentów czy jednostek kodu.

Warto nadmienić, że są oczywiście znane i stosowane techniki zapewniania wysokiej jakości produktu bez predykcji defektów. Te jednak zakładają bardzo wymagające reżimy wytwórcze (jak np. CMMI) i dotyczą wielkonakładowych projektów, w których aspekt ekonomiczny ma mniejsze znaczenie niż ryzyko występowania skutków krytycznych (przemysł jądrowy, lotniczy czy kosmiczny). Inny przykład dotyczy metodyki Six Sigma, która analizuje prawdopodobieństwo wytworzenia dobrego produktu oparciu o szacowanie prawdopodobieństw prawidłowego wykonania poszczególnych kroków procesu wytwórczego. Celem tego ostatniego jest doskonalenie powtarzalnego procesu wytwórczego poprzez zapewnienie stabilności jego parametrów statystycznych, co w przypadku cyklicznych wydań oprogramowania może tłumić innowacyjność rozwijanego produktu. Podobnie, zalecane w inżynierii oprogramowania różne „dobre praktyki” zapewniania jakości procesu wytwarzania kodu, a także stosowanie w tym celu wzorców projektowych wspieranych przez wiele narzędzi wytwórczych, również mogą się do takiego tłumienia przyczyniać.

W tym kontekście podstawowe założenie dla badań opisanych w Rozprawie, iż metodyka predykcji defektów kodu powinna być wbudowana w proces wytwórczy jest ze wszech miar słuszne. Z jednej strony jest bowiem konieczne zapewnienie możliwości szybkiego reagowania na zmiany zespołu ds. kontroli jakości (bez potrzeby nadmiernego wydłużania albo co gorsza powtarzania cyklu wytwórczego), a z drugiej daje możliwość wdrażania w kolejnych wydaniach oprogramowania przez zespół wytwórczy nowych funkcjonalności produktu. Realizacja tego skądinąd ambitnego założenia nie byłaby możliwa bez wsparcia firmy, udostępniającej Doktorantowi swoje środowisko produkcyjne. W mojej ocenie tak zarysowany cel Rozprawy jest w pełni zgodny z brzmieniem art. 187 ust.2 Ustawy i jego realizacja stanowi „oryginalne rozwiązanie w zakresie zastosowania wyników własnych badań naukowych w sferze gospodarczej".

2. *Istota problemu i jego aktualność*

Problem przewidywania i detekcji defektów w kodzie budził i nadal budzi duże zainteresowanie od czasów nawet jeszcze wcześniejszych niż rok 1968, umownie przyjmowany jako rok narodzin dyscypliny informatycznej zwanej inżynierią oprogramowania. Tak jak wtedy również i teraz jednym z jej podstawowych wyzwań jest minimalizacja czasu wytwarzania oprogramowania przy jednoczesnej maksymalizacji jakości produktu końcowego. Dodając do tego jeszcze biznesowy wymiar rozważany w Rozprawie należałoby to wyzwanie uzupełnić o sformułowanie „i w ramach optymalnie ponoszonych przez wytwórcę kosztów". Początkowo, gdy nie mogło jeszcze być mowy o automatyzacji tego procesu stosowano formalne inspekcje realizowane przez wyspecjalizowane zespoły ds. jakości (np. inspekcja Fagana z 1976 r.) czy mniej formalne przeglądy lub recenzje kodu przeprowadzane wspólnie z jego autorem lub bez przez członków zespołu wytwórczego. Inspekcje i przeglądy wymagają ponoszenia dodatkowych nakładów pracy związanych ze zbieraniem, gromadzeniem i analizą dużej ilości różnych danych. Pomimo tego, ostateczny bilans kosztów najczęściej wypada korzystnie dla wytwórcy. Dzisiaj, ze względu na panującą na rynku konkurencję równoczesna minimalizacja czasu wytwarzania i maksymalizacja jakości produktu informatycznego wymaga poszukiwania rozwiązań pozwalających ten proces zautomatyzować. Odpowiedzi na to wyzwanie szuka się w coraz powszechniej stosowanych w przemyśle metodach sztucznej inteligencji. Podejmowany przez Doktoranta problem wpisuje się w pełni w najbardziej związane z tym trendem aktualne zagadnienia „predictive maintenance" i „continuous improvement", tworzące obecnie swoisty „Święty Graal" nowoczesnej analityki przemysłowej. Jego istotę stanowią w szczególności: dobór metryk, których automatyczny pomiar dostarcza danych uczących do budowy modeli predykcji, metody ich obróbki wstępnej oraz określenie optymalnego wolumenu tychże z punktu widzenia jakości predykcji.

3. *Metody rozwiązania problemu*

W Rozdziale 1. Doktorant formułuje zasadniczy cel badawczy Rozprawy, którym było opracowanie metodyki zapewniania jakości procesu wytwarzania oprogramowania poprzez automatyczną predykcję defektów kodu z wykorzystaniem metod uczenia maszynowego oraz przedstawia cały szereg wynikających z niego celów szczegółowych; ich realizacja powinna umożliwić skuteczne wdrożenie wspomnianej metodyki w docelowym środowisku komercyjnym. Wywód poświęcony realizacji powyższych celów Doktorant prowadzi konsekwentnie przez kolejne rozdziały Rozprawy.

W Rozdziale 2. dokonuje systematycznego przeglądu i krytycznej analizy literatury pod kątem oceny aktualnego stanu wiedzy w obszarze predykcji defektów oprogramowania wykorzystującej metody uczenia maszynowego. W szczególności zwraca uwagę na problemy z tworzeniem modeli predykcji dla projektów, dla których historia naprawy defektów nie jest wystarczająco bogata by możliwe byłoby budowanie modeli predykcyjnych o zadowalającej dokładności oraz na fakt, że większość publikowanych w literaturze światowej wyników nie dotyczyła możliwości wdrożenia metod predykcji defektów oprogramowania w produkcyjnych środowiskach komercyjnych.

W konsekwencji w Rozdziale 3. typuje 5 komercyjnych produktów informatycznych współpracującej z nim firmy wytwarzającej oprogramowanie. Każdy z tych produktów analizuje pod kątem możliwości

wdrożenia metod predykcji defektów podczas rozwoju ich kolejnych wersji w oparciu o oryginalny zestaw szczegółowych wymagań dla ich procesu wytwórczego. Interesującym zagadnieniem na tym etapie wywodu jest przedstawiona w Rozdziale 4. nietrywialna analiza biznesowa opłacalności wdrożenia metody predykcji defektów w procesie wytwórczym. To zagadnienie jest istotne z punktu widzenia opłacalności wdrożenia tego typu metod w działalności firmy wytwarzającej oprogramowanie ze względu na dodatkowy nakład pracy i zasoby obliczeniowe niezbędne do zbierania danych pochodzących z różnych miejsc procesu wytwórczego oraz ich wykorzystania do wytrenowania predyktora. Z drugiej strony niska kultura pracy zespołu wytwórczego, charakteryzująca się brakiem odpowiednio wczesnego, systematycznego i odpowiednio precyzyjnego raportowania zmian w kodzie może skutkować znaczącym obniżeniem jakość predykcji i w konsekwencji doprowadzić do podwyższenia kosztów projektu zamiast spodziewanego ich obniżenia.

W oparciu o sformułowany jeszcze w Rozdziale 3. oryginalny zbiór minimalnych wymagań dla procesu wytwórczego w Rozdziale 4. Doktorant dokonuje pogłębionej analizy wspomnianych projektów i wybiera jeden z nich do przeprowadzenia szczegółowych badań porównawczych: analizuje jego dane historyczne i wylicza rzeczywiste koszty poniesione przez firmę na zapewnienie jakości produktu końcowego bez wdrożonego mechanizmu predykcji oraz szacuje na podstawie tych samych danych koszty zapewnienia jakości produktu przy symulowanym wdrożeniu tego mechanizmu. Wykazuje jednoznacznie, że pomimo dodatkowych nakładów jakie na podstawie jego obliczeń należałoby ponieść na zebranie danych, ich obróbkę wstępną i budowę modelu predyktora, a także na zakup odpowiednich narzędzi i dodatkowe szkolenia członków zespołu wytwórczego potencjalny zwrot netto firmy przy wdrożeniu predykcji w analizowanym projekcie byłby znaczący. Osiągnięty na tym etapie prac wynik miał kluczowe znaczenie dla dalszego powodzenia założonego w Rozdziale 1. planu badań, a mianowicie uzyskania zgody firmy wspierającej na faktyczne włączenie predykcji defektów do aktualnie realizowanego w firmie procesu wytwórczego komercyjnego produktu informatycznego, jak przedstawiono to w Rozdziale 5.

Dzięki temu opracowana przez Doktoranta metodyka DPDQA została poddana walidacji w warunkach rynkowych – jako istotna innowacja procesowa w firmie informatycznej. Celem tych prac było zbadanie jak dalece automatyczna predykcja defektów może podnieść zarówno jakość procesu wytwórczego (poprzez promowanie i doskonalenie najlepszych praktyk programowania przez zespół wytwórczy) jak i jakość produktu końcowego (poprzez spadek liczby defektów w kodzie finalnego produktu, w szczególności defektów o znaczeniu krytycznym). Opisane w Rozdziale 5. badania miały charakter porównawczy. Doktorant badał przez cały pierwszy rok przebieg pełnego cyklu wytwórczego jednego wydania (wersji) pewnego produktu kluczowego dla działalności firmy wspierającej z zastosowaniem metodyki wykorzystującej klasyczne podejście oparte na testowaniu, a przez cały drugi rok kolejny pełny cykl wytwórczy następnej wersji tego produktu, tym razem z zastosowaniem swojej metodyki DPDQA predykcji defektów. Szczególnego podkreślenia wymaga fakt, iż badania Doktoranta były bezpośrednio „wbudowane" w komercyjny proces wytwórczy, przez co ich znacznie rozciągnięty w czasie przebieg był silnie uwarunkowany przez cały szereg czynników pozostających poza jego kontrolą, w tym w szczególności przez narzucony harmonogram prac nad bieżącym wydaniem. Z drugiej strony takie usytuowanie bezpośrednio we wnętrzu realizowanego procesu produkcyjnego dało mu bieżący dostęp do wszystkich jego wykonawców, narzędzi i źródeł danych. Opisane w Rozdziale 5. porównanie obu badanych wydań wspomnianego produktu w oparciu o te same kryteria oceny jakości potwierdziło hipotezę postawioną wcześniej przez Doktoranta w Rozdziale 4. dotyczącą możliwości zapewnienia ekonomicznie opłacalnej redukcji defektów w rynkowym produkcie informatycznym przy zastosowaniu metodyki DPDQA.

Podczas gdy w zasadzie treść Rozdziałów 4. i 5. stanowi ostateczne zwieńczenie wywodu prowadzonego konsekwentnie od początku Rozprawy, treść Rozdziału 6. stanowi ciekawy przyczynek do dalszych badań w zakresie metod predykcji defektów z wykorzystaniem metod uczenia maszynowego. Oprócz bowiem zbioru klasycznych metryk produktowych wykorzystanych do budowy modelu predyktora Doktorant zbadał możliwość rozszerzenia swojej metodyki o zapachy kodu – stosunkowo nową koncepcję tworzenia metryk, próbujących obiektywizować (poprzez tworzenie mierzalnych reprezentacji) różne

praktyki programowania uznane w danym środowisku wytwórczym za „dobre”. Wykorzystał do tego celu narzędzia statycznej analizy kodu C# dostępne w środowisku Microsoft Visual Studio, a dane pozyskał z analizowanego w Rozdziale 5. produktu komercyjnego. Przeprowadzane przez Doktoranta eksperymenty wykazały, że wartości metryk dotyczące zapachów kodu (przynajmniej tych, których analizę umożliwia Visual Studio) nie wnoszą zauważalnej poprawy jakości predykcji względem tej wykorzystującej tylko metryki klasyczne. Zapewne metryki klasyczne lepiej agregują istotne właściwości analizowanego kodu – te związane z wysiłkiem umysłowym ponoszonym przez programistów podczas kodowania – niż dotychczas zdefiniowane metryki zapachów kodu potrafią odwzorować stopień przestrzegania przez nich obowiązujących praktyk programowania intuicyjnie uznawanych za „dobre”. Wynik przedstawiony w tym Rozdziale nie ma jednak w mojej opinii charakteru rozstrzygającego – zapachy kodu mają raczej charakter uznaniowy; być może zaproponowane zostaną w przyszłości jakieś metryki, które znajdą zastosowanie w predykcji defektów z wykorzystaniem uczenia maszynowego. Słusznie jednak Doktorant nie rozwija dalej tego wątku, bowiem wymagałoby to podjęcia zupełnie nowych badań.

4. <u>Oryginalność i przydatność dorobku rozprawy</u>

Do osiągnięć Doktoranta opisanych w Rozprawie zaliczam dwa zasadnicze rezultaty badawcze, przedstawione odpowiednio w Rozdziałach 4. i 5.:

1. Opracowanie całościowego „procesowego” podejścia do problemu predykcji defektów w kodzie produktu informatycznego, które oprócz zagadnienia jakości samej predykcji uwzględnia także koszty jej wdrożenia przez zespół wytwórczy realizujący określone przedsięwzięcie. Doktorant w przekonujący sposób wykazał opłacalność takiego podejścia w oparciu o rzeczywiste dane zebrane z komercyjnych projektów informatycznych. Podejście to pozwala skutecznie zracjonalizować stosowanie całego szeregu dobrych praktyk wytwarzania oprogramowania w projektach informatycznych oraz formułować wymagania bazowe dla narzędzi CASE i CAST nowej generacji, wychodzących poza dotychczas stosowane metodyki QA zapewniania jakości produktu.
2. Walidację opracowanego podejścia w warunkach produkcyjnych przy bieżącej współpracy z zespołem wytwórczym przez wszystkie etapy projektu realizowane w dwóch kolejnych cyklach rocznych. Wartość wyniku uzyskanego przez te dwa lata pracy, tj. wykazanie że wdrożenie opracowanej metodyki DPDQA prowadzi ostatecznie do produktu informatycznego o wyższej jakości i jest opłacalne, potęguje fakt iż działania Doktoranta o charakterze badawczym musiały być silnie skorelowane z cyklem produkcyjnym w firmie i były obarczone dużym ryzykiem – zarówno dla firmy (ryzyko wzrostu kosztów w razie konieczności powrotu do klasycznych metod zapewniania jakości produktu w przypadku fiaska proponowanej przez Doktoranta metody predykcji) jaki i Doktoranta (uzależnienie powodzenia prowadzonych badań od zewnętrznie narzuconych reżimów czasowych oraz ryzyk niezwiązanych z założonym planem badawczym realizowanej Rozprawy).

Na szczególne podkreślenie zasługuje przemysłowy kontekst prowadzonych badań – na każdym ich etapie wsparty rzeczywistymi danymi, nad którymi Doktorant nie miał kontroli. Mogły one równie dobrze (jak pokazał to Doktorant w Rozdziale 6. na przykładzie metryk zapachów kodu) wcale nie dać rozstrzygających wyników. Ale ryzyko się opłaciło – Rozprawa dostarcza społeczności komercyjnych wytwórców oprogramowania przekonujących argumentów za tym, by stosować predykcję defektów z wykorzystaniem modeli uczenia maszynowego oraz pokazuje jak to należy robić. W mojej ocenie to jest oryginalne i wartościowe osiągnięcie Doktoranta.

Zademonstrowana w rozprawie wiedza autora i znajomość literatury

Lektura Rozdziału 2. Rozprawy potwierdza dobrą znajomość literatury i wiedzę Doktoranta w zakresie metodyk QA zapewniania i oceny jakości produktu informatycznego, co pozwoliło mu swobodnie formułować miarodajne oceny i identyfikować wyzwania badawcze dotyczące metod predykcji defektów kodu. W oparciu o systematyczny przegląd aktualnego stanu wiedzy opisany w poszczególnych punktach Rozdziału 2. Doktorant identyfikuje lukę jaka istnienie w metodykach QA w zakresie wykorzystania mechanizmów

predykcji defektów w komercyjnych środowiskach wytwarzania oprogramowania. Przegląd ten rozpoczyna się od krótkiego przedstawienia historii badań nad predykcją defektów oprogramowania na przestrzeni ostatniego półwiecza, z licznymi odwołaniami do literatury i różnymi przełomowymi etapami rozwojowymi – od miar złożoności McCabe'a i Halstead'a sprowadzających prognozowanie obecności defektów w kodzie do swoistej „reguły kciuka" aż po dwa kluczowe wyzwania stojące przed badaczami obecnie. Pierwsze z nich dotyczy ekonomicznej opłacalności metod predykcji obszarów kodu podatnych na defekty dokonywanej przy okazji każdej modyfikacji kodu, a drugie związane jest z brakiem wystarczającej liczby danych historycznych do przeprowadzenia predykcji o odpowiedniej jakości. Oba te wyzwania są także punktem odniesienia dla badań Doktoranta opisanych w Rozprawie. O dobrym przygotowaniu Doktoranta do podjęcia tychże świadczy pogłębiony przegląd wyników opisanych w literaturze przedmiotu dla całego szeregu zagadnień szczegółowych. Zagadnienia te warunkują skuteczne podjęcie wspomnianych wyzwań i obejmują m.i miary do oceny jakości predykcji, metryki produktowe i procesowe produktu informatycznego, modele predykcji i ocenę ich dokładności, problemy obróbki wstępnej danych uczących czy predykcję międzyprojektową wykorzystującą uczenie transferowe.

Umiejętność samodzielnego prowadzenia pracy naukowej

Redakcja Rozprawy i sposób prowadzenia wywodu potwierdza umiejętność Doktoranta w planowaniu, realizacji i ocenie wyników złożonego przedsięwzięcia o charakterze B+R, którego celem jest innowacja procesowa. Oprócz wiedzy teoretycznej zademonstrowanej przede wszystkim w Rozdziale 2. i technicznej zilustrowanej licznymi przykładami w Rozdziałach 3., 4. i 5. w zakresie metodyk i narzędzi inżynierii oprogramowania Doktorant wykazał się dużą dozą kompetencji biznesowych, nadających osiągniętym przez siebie wynikom badawczym sens praktyczny i stanowiących wymierną wartość dla przedsiębiorstw wysokich technologii (HiTech). Na szczególne uznanie zasługuje fakt realizacji opisanych badań w docelowym środowisku produkcyjnym konkretnego produktu komercyjnego. Dzięki temu wynikiem Rozprawy nie jest li tylko samodzielnie przeprowadzony „Proof of Concept" (co już byłoby wystarczającym osiągnięciem z punktu widzenia celów rozprawy doktorskiej o charakterze B+R), ale rzeczywiste skuteczne i opłacalne wdrożenie rozwiązania postawionego przez siebie problemu przy użyciu aktualnie dostępnych technologii oraz do wskazania dalszych możliwości doskonalenia opracowanego przez siebie rozwiązania.

Niedociągnięcia i słabe strony Rozprawy

Rozprawa została napisana poprawnym językiem, stosowane formy gramatyczne, słownictwo, interpunkcja, itp. na ogół nie budzą zastrzeżeń. Nieliczne moje zastrzeżenia w tym zakresie wymieniam poniżej:

- Niekiedy Doktorant posługuje się żargonem, np. odmieniając angielski rzeczownik „commit" przez polskie przypadki (narzędnik „commicie", str. 29), co niestety jest dość powszechną praktyką w oficjalnych polskojęzycznych tekstach dotyczących informatyki.
- Próba tłumaczenia zwrotu angielskiego (jak można się domyśleć) „enforce development best practices" jako „napędzanie [..] praktyk rozwojowych" na str. 84 nie wypada najlepiej.
- Nagminne użycie archaicznej formy „koszta" (stosowanej chyba już tylko w księgowości) zamiast słowa „koszty" (tylko w jednym miejscu na str. 13 znalazłem słowo „koszty"), może być dość irytująca dla czytelników tekstów z zakresu zarządzania projektami informatycznymi.
- Nieprofesjonalnie wygląda w tekście z zakresu nauk technicznych użycie w wyrażeniach arytmetycznych operatora iloczynu kartezjańskiego '$\times$' zamiast zwykłego mnożenia '$\cdot$' (wzory 4.2 i 4.3).
- Rysunek 4.2 (str. 62) używa zarówno języka polskiego jak i angielskiego. Wszystkie napisy powinny być jednolicie po polsku, jak w pozostałych rysunkach Rozprawy.
- W spisie literatury została dwukrotnie podana ta sama cytata ([156] i [157]).
- Cytowanie w Rozprawie naukowej kultowego tekstu (skądinąd świetnego) satyryka Jana Tadeusza Stanisławskiego nie jest poważne. To tak jakby w przypadku roprawy napisanej w języku angielskim króliczek Bugs albo kaczor Duffy z Hanna Barbera Cartoons mówił do czytelnika "That's all folks".

Z poważniejszych merytorycznych zagadnień jakie nasunęły mi się podczas lektury Rozprawy zgłaszam następujące:

1. Jako naturalne rozwinięcie motywacji podjęcia badań opisanych w Rozprawie należałoby się spodziewać na zakończenie Rozdziału 1. jednoznacznie sformułowanej tezy. Zamiast tego Doktorant podaje tylko listę dziewięciu celów szczegółowych planowanych badań. Ich realizacja została opisana w poszczególnych rozdziałach Rozprawy, jednak brakuje stwierdzenia co ich realizacja konkretnie wykazała. Niewątpliwie trzon Rozprawy stanowią Rozdziały 4. i 5. – nie jest jednak dla mnie jasne czy w wyniku badań przedstawionych w Rozprawie można twierdzić, że predykcja defektów w kodzie jest ekonomicznie opłacalna przy spełnieniu podanych kryteriów procesu wytwórczego, a opracowana przez Doktoranta metodyka DPDQA jedynie ilustruje jak te kryteria stosować, czy też wspomniana metodyka jest warunkiem koniecznym dla osiągnięcia tej opłacalności.
2. Jakie są kryteria (charakterystyki, atrybuty) „wystarczającej dojrzałości" projektu (np. str. 56) by wdrożenie predykcji defektów było możliwe? Czy te kryteria mają jakiś ogólny charakter, czy są specyficzne dla metodyki DPDQA? Brakuje odniesień do standardów dokumentowania projektów, w szczególności standardów dokumentowania testów (*IEEE Software and Systems Engineering Standards*, http://standards.ieee.org/findstds/standard/software_and_systems_engineering.html, czy *Space engineering – Software, ECSS-E-ST-40C, European Cooperation for Space Standardization, ESA-ESTEC*, http://ecss.nl/standards/ecss-standards-on-line/active-standards).
3. Ze względu na brak w Rozprawie porządnej definicji i klasyfikacji analizowanych defektów trudno zinterpretować ich zastanawiająco wysoką liczbę wyliczoną wzorem 4.12 na str. 67 czy podaną u dołu str. 74.
4. W rozważaniach dotyczących projektu komercyjnego realizowanego w firmie brakuje odniesienia się do aspektów związanych ze zjawiskiem fluktuacji kadry, które jest permanentnie wbudowane w procesy wytwórcze w firmach informatycznych (por. S. N.C., S. Majumder and T. Menzies, "Early Life Cycle Software Defect Prediction. Why? How?," 2021 IEEE/ACM 43rd Int. Conf. on Software Engineering (ICSE), 2021, pp. 448-459, doi: 10.1109/ICSE43902.2021.00050).
5. W podsumowaniu w Rozdziale 7. warto by było przedyskutować ogólne kwestie skuteczności i opłacalności metodyk predykcji błędów w różnych modelach cyklu wytwarzania komercyjnego, w tym w szczególności w modelu spiralnym oraz SPL (Software Product Line). Także warto przyjrzeć się perspektywie zastosowania metod „explainable AI" w predykcji defektów w kodzie.

Nadmieniam, że wszystkie moje wymienione wyżej uwagi mają charakter polemiczny i traktuję je jako punkt wyjścia dla dyskusji podczas publicznej obrony.

Wniosek końcowy

Wszystkie poruszone w poprzednim punkcie Rozprawy zagadnienia nie wpływają na moją pozytywną ocenę recenzowanej Rozprawy. Stwierdzam, że spełnia ona wymagania stawiane w Ustawie rozprawom doktorskim i wnoszę o jej dopuszczenie do publicznej obrony.

prof. dr hab. inż. Bogdan Wiszniewski