Student: Jakub Woźniak pd4135
Kurs: Budowa i integracja systemów informacyjnych (BYT) - studia podyplomowe SAB
Prowadzący: Włodzimierz Dąbrowski

Wybrane metryki projektu informatycznego dla firmy “Krasnale”

1. Wstęp

1.1 Cel dokumentu

Celem poniższej analizy jest zaproponowanie metryk, które mogą posłużyć do oceny projektu informatycznego dla firmy “Krasnale” (dalej: “Projekt”). Zaproponowane metryki mają na celu ocenić produkty występujące w Projekcie, procesy w nim realizowane oraz wykorzystywane zasoby. Dokument zawiera krótki opis metryk proponowanych dla każdej z trzech kategorii. Postaramy się przedstawić konstrukcję każdej z metryk, uzasadnić ich wybór każdej z metryk i - o ile to będzie możliwe - skwantyfikować oczekiwane wyniki pomiarów.

1.2 Zakres projektu

Według opisu przypadku pt. “Krasnale”, firma ta zajmuje się dystrybucją krasnali ogrodowych renomowanych światowych producentów. Głównym celem projektu informatycznego byłoby dostarczenie narzędzi do sprzedaży przez Internet oraz unowocześnienie istniejącej strony WWW. Sponsorem projektu został wybrany dyrektor działu sprzedaży, który jest największym działem firmy “Krasnale”. Zarząd oczekuje, że projekt będzie gotowy w ciągu 4 miesięcy.

2. Przegląd wybranych metryk

2.1 Metryki produktów

2.1.1 Charakterystyka

Produktem w projekcie informatycznym jest każdy przedmiot powstały w wyniku procesu. Mogą nim być kod źródłowy, specyfikacja projektowa, projekt architektury systemu, plan bazy danych, plan testów, dane wsadowe do testów, projekt wdrożenia, dokumentacja szkoleniowa itp. Produkty możemy podzielić na cztery kategorie: kod źródłowy, specyfikacje, projekty i dane testowe.

2.1.2 Propozycja metryk

Element	**Opis**
Nazwa metryki:	procent zakodowanych i przetestowanych modułów
Dlaczego stosujemy:	Chcemy upewnić się, że oceniamy kompletny i przetestowany system informatyczny, zgodny z wymaganiami. Zespół programistyczny firmy “Krasnale” nie wykonywał wcześniej takich projektów, więc chcemy się upewnić, że zakodowano i przetestowano wszystkie uzgodnione moduły.
Co mierzymy:	Kompletność i funkcjonalność oprogramowania
Jak mierzymy:	liczba zakodowanych i przetestowanych modułów podzielona przez liczbę planowanych modułów
Co oznacza wynik:	W jakim % projekt jest kompletny i funkcjonalny.
Jakiego wyniku oczekujemy:	Oczekujemy, że wszystkie zaplanowane moduły będą zakodowane i przetestowane.
Dopuszczalne odchylenie:	-

Element	**Opis**

Nazwa metryki:	procentowe pokrycie kodu testami
Dlaczego stosujemy:	Chcemy upewnić się, w jakim stopniu kod źródłowy został przetestowany. Biorąc pod uwagę niewielkie doświadczenie zespołu programistycznego w firmie “Krasnale” w podobnych projektach, napięty harmonogram i brak wiedzy o wybranej metodzie pracy (ryzyka projektowe), metryka ta może być dość ważna.
Co mierzymy:	Poziom pokrycia kodu źródłowego testami
Jak mierzymy:	Odsetek linii kodu pokrytego testami dzielony przez łączną liczbę linii kodu
Co oznacza wynik:	Generalnie, oczekujemy jak najwyższego poziomu pokrycia kodu testami. Wynik należy rozpatrywać łącznie z wiedzą o tym, jakiego rodzaju testy został wykonane (funkcjonalne, strukturalne, jednostkowe, automatyczne, manualne) oraz wynikami analizy błędów i wąskich gardeł w kodzie.
Jakiego wyniku oczekujemy:	75-100% z zastrzeżeniami powyżej
Dopuszczalne odchylenie:	25% z zastrzeżeniami powyżej

Element	**Opis**
Nazwa metryki:	SLOC (source lines of code)
Dlaczego stosujemy:	Jest to jedna z najpowszechniejszych miar produktowych. Stosowana z innymi metrykami, pozwala ocenić rozmiar i złożoność projektu i pojedynczych modułów/obiektów. W przypadku projektu dla firmy “Krasnale”, pozwoli razem z innymi miarami oszacować koszt wytworzenia i późniejszego utrzymywania strony WWW, modułu e-commerce i systemu informacyjnego (forum, akcje marketingowe, celowany e-mailing, webankiety itp.).
Co mierzymy:	Mierzy rozmiar i złożoność Projektu.
Jak mierzymy:	Łączna suma linii napisanego kodu bez komentarzy.
Co oznacza wynik:	Sam wynik niewiele nam powie. Należy go raczej porównać z podobnymi projektami pisanymi w tym samym lub podobnym języku programowania, z użyciem podobnych narzędzi CASE i przez zespół z podobnym doświadczeniem. Wysoki SLOC jest często skorelowany z niską spójnością klasy (cohesion) i silnymi powiązaniami pomiędzy klasami (coupling). Oznacz to zwykle niską jakość kodu i podatność na awarie.
Jakiego wyniku oczekujemy:	200-1000 linii w jednej klasie, z zastrzeżeniami przedstawionymi powyżej
Dopuszczalne odchylenie:	30% z zastrzeżeniami przedstawionymi powyżej

Źródło: “Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults”, Yuming Zhou and Hareton Leung, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 32, NO. 10, OCTOBER 2006

Element	**Opis**
Nazwa metryki:	LCOM (lack of cohesion in methods), brak spójności pomiędzy metodami w klasie
Dlaczego stosujemy:	Chcemy ocenić jakość i czytelność kodu oraz przyszłą pracochłonność w utrzymaniu i rozwijaniu systemu. Pomimo przejścia pozytywnych testów, system z nadmiernym i nieprzemyślanym kodem będzie podatny na przyszłe awarie.
Co mierzymy:	brak spójności pomiędzy metodami w klasie, podatność kodu na awarie, nieczytelność kodu
Jak mierzymy:	Dla każdego atrybutu/pola klasy, mierzymy jaki odsetek metod klasowych z niego korzysta. Uśredniony wynik odejmujemy od 100 punktów procentowych.
Co oznacza wynik:	Generalnie, oczekujemy jak najniższego LCOM, czyli jak najwyższej spójności pomiędzy metodami w danej klasie. Klasa, w której jest niska spójność kodu, będzie podatna na awarie w przypadku przyszłych modyfikacji i może skutkować duplikowaniem kodu.
Jakiego wyniku oczekujemy:	70-100%
Dopuszczalne odchylenie:	30%

Źródło: j.w., https://www.enjoyalgorithms.com/blog/cohesion-and-coupling-in-oops

Element	Opis
Nazwa metryki:	CBO (coupling between object classes), zależności pomiędzy klasami
Dlaczego stosujemy:	Chcemy ocenić jakość i czytelność kodu oraz przyszłą pracochłonność w utrzymaniu i rozwijaniu systemu. Pomimo przejścia pozytywnych testów, system z nieprzemyślanym kodem będzie podatny na przyszłe awarie.
Co mierzymy:	poziom powiązań pomiędzy klasami, podatność kodu na awarie, nieczytelność kodu
Jak mierzymy:	Odsetek liczby klas powiązanych z innymi do łącznej liczby wszystkich klas. Jedna klasa jest zależna od drugiej, kiedy wykorzystuje jej pola i/lub jej metody. Do wyliczenia nie bierzemy klas dziedziczących od siebie.
Co oznacza wynik:	Wysoki odsetek oznacza wysoki coupling. Klasy są silnie od siebie współzależne, więc modyfikacja atrybutów/metod jednej może skutkować błędami w kodzie drugiej i/lub niespodziewanym działaniem drugiej klasy.
Jakiego wyniku oczekujemy:	0-10%
Dopuszczalne odchylenie:	5%

Źródło: j.w., https://devstyle.pl/2020/02/24/szesc-twarzy-couplingu/

2.2 Metryki procesów

2.2.1 Charakterystyka

Przez proces rozumiemy każde działanie w kontekście projektowania, wytwarzania i użytkowania oprogramowania. Tutaj zwykle mierzymy jakość procesu wytwarzania oprogramowania poprzez m.in. mierzenie czasu, nakładu pracy i produktywności w zespołach.

2.2.2 Propozycja metryk

Element	Opis
Nazwa metryki:	SLOC (source lines of code), liczba linii uruchomionego i przetestowanego kodu źródłowego (bez komentarzy)
Dlaczego stosujemy:	Chcemy monitorować wielkość produkowanego kodu w okresie czasu
Co mierzymy:	Rozmiar produkowanego kodu
Jak mierzymy:	Podobnie jak w przypadku metryki produktowej, z zastrzeżeniem, że mierzymy także trend czasowy np. w ciągu ostatniego miesiąca
Co oznacza wynik:	Wynik jest dość subiektywny i zależny m.in. od rozwiązywanych zadań programistycznych w badanym okresie oraz składu i doświadczenia zespołu. Nagły spadek może oznaczać np. wąskie gardło w procesie lub niedostateczny skład zespołu.
Jakiego wyniku oczekujemy:	Trudne do określenia.
Dopuszczalne odchylenie:	Trudne do określenia.

Element	Opis
Nazwa metryki:	Odsetek rozwiązanych problemów programistycznych
Dlaczego stosujemy:	Potrzebujemy metryki do mierzenia produktywności zespołu i postępów w pracy nad Projektem. W przypadku firmy “Krasnale” Projekt jest ryzykowny, ponieważ przewiduje wytworzenie oprogramowania we własnym zakresie przez wewnętrzny zespół nie posiadający doświadczenia w podobnych projektach.
Co mierzymy:	Poziom zaangażowania zespołu programistycznego, jakość pracy, produktywność, postęp w pracy nad projektem
Jak mierzymy:	Liczba rozwiązanych problemów programistycznych dzielone przez łącznie liczbę rozpoznanych problemów programistycznych. Metrykę monitorujemy w czasie.

Co oznacza wynik:	Niski poziom utrzymujący się przez dłuższy czas i/lub brak zmian np. w przeciągu kilku dni może oznaczać wąskie gardła w implementacji, niedostateczny skład zespołu, napotkanie nietypowych problemów itd. Metryka może nam pomóc we wczesnym wykryciu problemów w procesie.
Jakiego wyniku oczekujemy:	Generalnie spodziewamy się szybko rosnącego poziomu w czasie. Wynik może być jednak subiektywny, ponieważ liczba problemów programistycznych może się zmieniać w czasie.
Dopuszczalne odchylenie:	Trudne do określenia.

Element	**Opis**
Nazwa metryki:	Częstotliwość uruchamianego kodu (“deployment frequency”)
Dlaczego stosujemy:	Chcemy monitorować produktywność zespołu programistycznego.
Co mierzymy:	Produktywność
Jak mierzymy:	Ile razy np. w ciągu tygodnia kod produkowany przez programistów jest uruchamiany. Możemy mierzyć per programista lub per zespół.
Co oznacza wynik:	Poziom produktywności i sprawności zespołu.
Jakiego wyniku oczekujemy:	Generalnie poziom uruchomień od kilku razy dziennie do kilku razy w tygodniu jest przyjmowany jako oznaka wysokiej produktywności/sprawności zespołu.
Dopuszczalne odchylenie:	Trudne do określenia.

Źródło: https://www.pluralsight.com/blog/software-development/software-engineering-metrics

Element	**Opis**
Nazwa metryki:	Odsetek poprawek w kodzie (“% of rework”)
Dlaczego stosujemy:	Chcemy monitorować produktywność zespołu programistycznego. Dodatkowo, Projekt niesie ze sobą ryzyko zmian w wymaganiach i/lub niejasnych wymagań.
Co mierzymy:	Produktywność, jakoś produkowanego kodu
Jak mierzymy:	Odsetek poprawek w kodzie danej osoby w okresie czasu np. w ciągu ostatnich 3 tygodni
Co oznacza wynik:	Produktywność, jakoś produkowanego kodu. Nagły wzrost metryki może oznaczać np. pojawienie się nietypowego problemu programistycznego lub niejasne wymagania na oprogramowanie/zmianę w wymaganiach.
Jakiego wyniku oczekujemy:	20-30% dla doświadczonego programisty
Dopuszczalne odchylenie:	Trudno określić.

Źródło: j.w.

Element	**Opis**
Nazwa metryki:	Odsetek usterek (znalezionych podczas testów)
Dlaczego stosujemy:	Chcemy upewnić się, że oddajemy system wolny od błędów i niezawodny w przyszłości. Jest to szczególnie istotne biorąc pod uwagę: 1) niski poziom doświadczenia zespołu programistycznego w firmie “Krasnale” oraz 2) wysokie ryzyko reputacyjne Projektu – oddanie użytkownikom systemu o niskiej jakości może oznaczać zmarnowanie czasu i pieniędzy, a także ryzyko utraty klientów/rynku.
Co mierzymy:	Jakość i niezawodność wytworzonego oprogramowania
Jak mierzymy:	Odsetek usterek dzielony przez liczbę roboczogodzin
Co oznacza wynik:	Wynik porównujemy z podobnymi projektami i monitorujemy jego poziom w czasie postępu fazy testów. Względnie wysoki poziom usterek w porównaniu z podobnymi projektami może świadczyć o niskiej jakości produktu. Wyniki mogą nam pomóc w oszacowaniu MTTF (Mean Time To Failure), stosowanego do prognozowania niezawodności systemu.
Jakiego wyniku oczekujemy:	Nie dopuszczamy żadnych niepoprawionych usterek.

Dopuszczalne odchylenie:	j.w.

Źródło: “1061-1992 - IEEE Standard for a Software Quality Metrics Methodology”, https://ieeexplore.ieee.org/document/237006, p69

2.3 Metryki zasobów

2.3.1 Charakterystyka

Przez zasoby rozumiemy każdy element niezbędny do realizacji procesu. Mogą nimi być zatem osoby, oprogramowanie, sprzęt komputerowy, biuro itp. Poza przedstawionymi poniżej metrykami zasoby mogą być także mierzone w takim zakresie jak doświadczenie zespołu, jakość komunikacji w zespole, ergonomia pracy czy jakość stosowanych narzędzi.

2.3.2 Propozycja metryk

Element	**Opis**
Nazwa metryki:	% wykorzystania zespołu
Dlaczego stosujemy:	Koszty zespołu programistów są najważniejszym kosztem w projekcie firmy “Krasnale”. Monitorując obłożenie zespołu zadaniami, będziemy w stanie kontrolować poziom wykorzystania tego zasobu.
Co mierzymy:	Poziom obłożenia zespołu programistycznego
Jak mierzymy:	Roboczogodziny faktycznie przepracowane do łącznej liczby roboczogodzin w miesiącu
Co oznacza wynik:	Wynik w okolicach 100% oznacza pełne obłożenie zespołu. Wyniki powyżej 100% mogą oznaczać pracę w nadgodzinach i mogą wynikać ze spiętrzenia zadań w Projekcie. Utrzymujący się przez dłuższy czas niski poziom obłożenia może świadczyć o zbyt dużym zespole lub o zbyt małej liczbie zadań programistycznych.
Jakiego wyniku oczekujemy:	90-105%
Dopuszczalne odchylenie:	10-20%

Element	**Opis**
Nazwa metryki:	Wariancja kosztów osobowych
Dlaczego stosujemy:	Koszty zespołu programistów są najważniejszym kosztem w projekcie firmy “Krasnale”.
Co mierzymy:	Odchylenie faktycznych kosztów osobowych od planowanych (budżetowanych).
Jak mierzymy:	Faktyczne koszty osobowe w miesiącu minus koszty budżetowane
Co oznacza wynik:	Duże odchylenia in plus mogą wynikać ze słabego planowania, zatrudnienia osób do zespołu za stawkę wyższą niż planowaną, wyższego niż planowano poziomu fakturowania przez programistów-podwykonawców.
Jakiego wyniku oczekujemy:	Wynik w okolicach wyniku budżetowanego
Dopuszczalne odchylenie:	5-10%

Element	**Opis**
Nazwa metryki:	Poziom rotacji pracowników
Dlaczego stosujemy:	Branża IT charakteryzuje się wysoką konkurencyjnością i wysokimi wymaganiami wobec pracowników. Wg niektórych badań, rotacja w IT jest najwyższa spośród wszystkich branż i wynosi 13-18% (źródło poniżej). Nagły wzrost poziomu rotacji pracowników w zespole programistów w firmie “Krasnale” mógłby świadczyć o rosnącym ryzyku całego projektu.
Co mierzymy:	Stabilność zespołu, ryzyko sparaliżowania pracy przez odejście kluczowej osoby (wysokie w przypadku “Krasnali”)

Jak mierzymy:	Odsetek osób, które opuściły zespół w ciągu ostatnich 12 miesięcy
Co oznacza wynik:	Im niższy, tym większa stabilność zespołu programistycznego.
Jakiego wyniku oczekujemy:	<5%
Dopuszczalne odchylenie:	2-3%

Źródło: https://www.linkedin.com/pulse/look-tech-retention-crisis-causes-key-trends-plausible-yara-abboud/

***