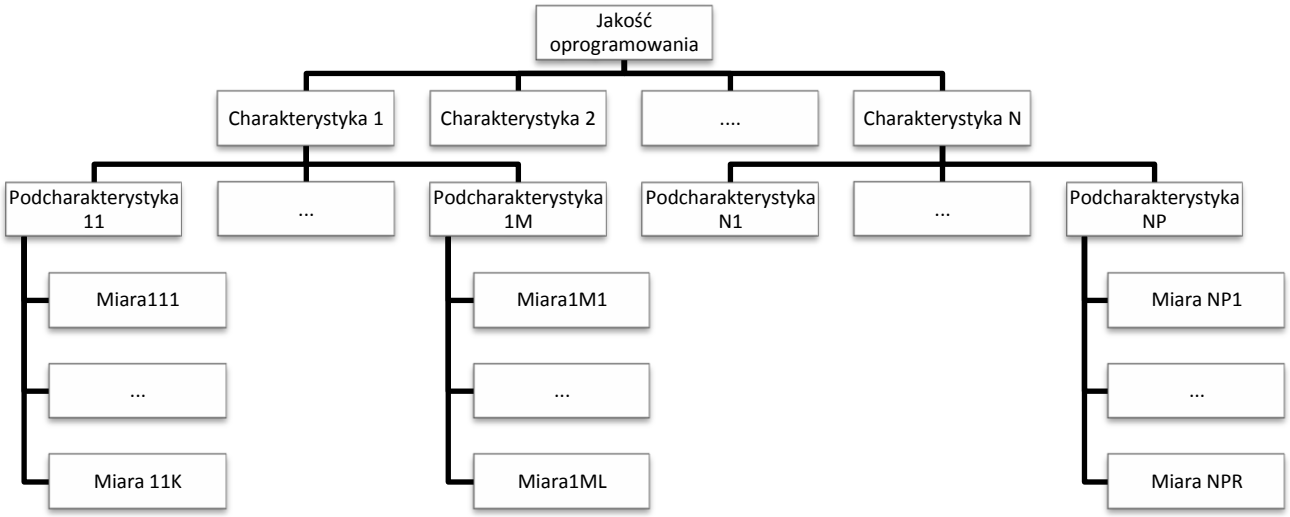
Jakość i modele jakości oprogramowania

Jakość oprogramowania norma ISO9126 oraz Kobyliński [31] opisują przy pomocy innych, bardziej konkretnych, lepiej zdefiniowanych i łatwiejszych do pomiaru cech zwanych *charakterystykami* jakości. Charakterystyki są jednak zbyt ogólne aby mogły być zmierzone bezpośrednio. Dlatego dokonuje się ich dekompozycji na elementy bardziej szczegółowe – *podcharakterystyki*, które pozwalają na łatwiejsze wprowadzenie miar. W końcu, w trzecim etapie podziału, przypisuje się każdej podcharakterystyce pewien zbiór bezpośrednio mierzalnych wskaźników zwanych często metrykami jakości. Ogólnie, opisany powyżej proces dekompozycji, można przedstawić w postaci drzewa atrybutów jakości oprogramowania przedstawionego na rysunku Rys. 2.



Rys. 1. Drzewo atrybutów jakości oprogramowania. Na podstawie: [31].

Organizacje takie jako ISO czy IEEE próbują standaryzować jakość oprogramowania, wprowadzając definicje pojęć i gotowe modele, które łączą charakterystyki i podchrakterystyki i pozwalają na ocenę jakości systemu. Norma ISO 9126 definiuje [28] jakość oprogramowania jako:

Zdolność produktu programowego do zaspokajania aktualnych i przewidywalnych potrzeb jego użytkowników.

W Inżynierii oprogramowania jest wiele typów modeli jakości, każdy z nich ma różny zestaw charakterystyk i podcharakterystyk.

Wybrane modele jakości oprogramowania

W niniejszym podrozdziale zamieszczono przegląd najbardziej znanych modeli jakości, które mogą pomóc przy ocenie oprogramowania pod względem pielęgnowalności.

Model McCall'a

Model McCall'a to przodek współczesnych modeli jakości, który został opracowany dla sił powietrznych USA. Na jego podstawie opracowano model ISO/IEC 9126. Model odzwierciedla zarówno widok użytkownika, jak i priorytety programisty. Ma on 3 główne perspektywy, które ilustrują podstawowe sytuacje, w których spogląda się na oprogramowanie pod innym kątem: rewizje produktu (zdolność podlegania zmianom), przejścia produktu (umiejętność przystosowywania się do nowych środowisk) i używanie

produktu. Do każdej perspektywy są przypisane charakterystyki (czynniki jakości), które z kolei dzielą się na podchrakterystyki (kryteria jakości). W Tab. 1 przedstawiono powiązania między nimi.

Perspektywa	Charakterystyki	Podcharakterystyki
Podleganie zmianom, rewizje produktu (ang. *product revisions*)	Pielęgnowalność	Prostota
		Zwięzłość
		Autodokumentacja, samo-opisywalność
		Modularność
	Elastyczność	Autodokumentacja, samo-opisywalność
		Rozszerzalność
		Ogólność
	Testowalność	Prostota
		Autodokumentacja, samo-opisywalność
		Modularność
		Podatność na analizę
Używanie produktu (ang. *product operations*)	Poprawność	Podatność na śledzenie
		Kompletność
		Spójność
	Efektywność	Wydajność wykonania
		Wydajność pamięciowa
	Niezawodność	Spójność
		Dokładność
		Tolerancja na błędy
	Integralność	Kontrola dostępu
		Audit dostępu
	Użyteczność	Operacyjność
		Szkolenia
		Komunikatywność
Przejścia produktu (ang. *product transition*)	Przenośność	Autodokumentacja, samo-opisywalność
		Niezależność systemowa
		Niezależność sprzętowa
	Wieloużywalność	Autodokumentacja, samo-opisywalność
		Ogólność
		Modularność
		Niezależność systemowa
		Niezależność sprzętowa
	Zdolność do współdziałania	Modularność
		Standaryzacja
		Standaryzacja danych

Tab. 1. Model jakości McCall'a. Na podstawie: [31] i [13].

Model Boehm'a

Model Boehm'a to drugi z bazowych modeli – przodków współczesnych modeli jakości. Powstał w tym samym okresie co model McCall'a i jest do niego dość podobny – również przedstawia hierarchię charakterystyk i podcharakterystyki. Najwyższy poziom struktury drzewa charakterystyk również stanowią perspektywy: przydatność, pielęgnowalność (łatwość konserwacji) i przenośność. Przypisane im charakterystyki i podcharakterystyki określają warunki konieczne i wystarczające do ich osiągnięcia. Przydatność (ang. *as-is utility*) to stopień w jakim oprogramowanie może być przydatne użytkownikowi. Mówi, że program powinien być wystarczająco wydajny, niezawodny i użyteczny. Pielęgnowalność (ang. *maintainability*) określana jest jako łatwość identyfikowania tego, co musi być zmienione oraz łatwość modyfikacji i testowania. A przenośność (ang. *portability*) to łatwość zmian oprogramowania w celu dostosowania go do nowego środowiska. Najniższy poziom drzewa stanowią podcharakterystyki odnoszące się

głównie do jakości kodu źródłowego, które mają stanowić podstawę do definiowania miar ilościowych. Model Boehm'a przedstawiono w tabeli Tab. 2.

Perspektywa	Charakterystyki	Podcharakterystyki
Przydatność – łatwość, wydajność i niezawodność używania oprogramowania.	Niezawodność – zachowuje się tak jak się oczekuje	Kompletność
		Dokładność
		Jednolitość
	Efektywność – realizuje swoje funkcje bez marnowania zasobów	Efektywność zużycia zasobów
		Dostępność
	Użyteczność	Komunikatywność
		Dostępność
Przenośność – łatwość i poprawność użytkowania oprogramowania przy konfiguracji innej niż domyślna		Niezależność sprzętowa
		Kompletność
Pielęgnowalność – stopień zrozumienia, modyfikowalności i testowalności.	Testowalność	Ocenialność
		Komunikatywność
		Strukturalność
		Autodokumentacja
	Zrozumiałość kodu	Zwięzłość
		Spójność
		Strukturalność
		Przejrzystość
	Modyfikowalność	Rozszerzalność
		Strukturalność

Tab. 2. Model jakości Boehm'a. Na podstawie: [31] i [13].

Model FURPS

Model FURPS został zaproponowany przez Roberta Grandy'ego i firmę Hewlett Packard w 1987 r. Łączy on charakterystyki jakości w grupy dotyczące dwóch kategorii wymagań: funkcjonalnych i niefunkcjonalnych. Wymagania funkcjonalne określane są poprzez definicje wejść i oczekiwanych wyjść. Wymagania niefunkcjonalne to użyteczność, niezawodność, wydajność i jakość wsparcia. Nazwa modelu jest akronimem podstawowych charakterystyk jakości w nim uwzględnionych: functionality (funkcjonalność), usability (użyteczność), realiability (niezawodność), performance (wydajność) i suportability (jakość wsparcia). Model FURPS został przedstawiony w tabeli Tab. 3.

Wymagania funkcjonalne	Funkcjonalność	Adekwatność
		Możliwości
		Uniwersalność
		Zabezpieczenie
Wymagania niefunkcjonalne	Użyteczność	Operacyjność
		Atrakcyjność
		Standaryzacja
		Dokumentacja
	Niezawodność	Dojrzałość
		Dokładność
		MTBF (ang. *Mean Time Between Failures)* – czas bezawaryjnej pracy
		Zdolność regeneracji
		Odporność
	Wydajność	Czas obrotu zadania
		Czas odpowiedzi
		Zużycie zasobów
		Efektywność użycia zasobów

	Jakość wsparcia	Łatwość konserwacji • Korygowalność • Rozszerzalność • Adaptacyjność
		Testowalność
		Podatność na analizę, analizowalność
		Współistnienie
		Łatwość konfigurowania
		Łatwość instalacji

Tab. 3. Model jakości FURPS. Na podstawie: [31] i [13].

Modele jakości ISO

ISO (The International Organization for Standardization) to międzynarodowa, pozarządowa organizacja standaryzacyjna, która w swoim gronie skupia przedstawicieli narodowych komitetów normalizacyjnych. ISO zajmuje się standaryzacją (ujednoliceniem) różnych dziedzin życia. Podstawowym przedmiotem działalności organizacji ISO jest standaryzacja różnych rozwiązań technicznych i organizacyjnych.

Model ISO/IEC 9126

Normy ISO serii 9000 dotyczą zarządzania jakością. Zawierają terminologię, wymagania i wytyczne dotyczące wprowadzania, doskonalenia i kontrolowania systemu zarządzania jakością. ISO 9126 definiuje model jakości dla oprogramowania jako produktu. Prace nad nim trwały od 1985 roku, a za status normy uzyskał w 1991 roku. W 2001 roku został on podzielony na dwa standardy: ISO/IEC 9126:2001 oraz ISO/IEC 14598. Pierwszy z nich opisuje model jakości, a drugi skupia się na sposobie oceny jakości produktów programowych.

Model ISO 9126 bazuje na modelach McCall'a i Boehm'a oraz FURPS i odnosi się zarówno do wymagań funkcjonalnych, jak i niefunkcjonalnych. Standard składa się on z czterech części – pierwsza definiuje model jakości, a pozostałe trzy opisują związane z nim metryki:

I. **Software Engineering–Product quality – "Quality model"** - część pierwsza opisująca model jakości. Bazuje on na modelach McCall'a i Boehm'a oraz FURPS i składa się z 6 charakterystyk: funkcjonalności, niezawodności, użyteczności, efektywności, łatwości utrzymania i przenośności. Do każdej charakterystyki są przypisane odpowiednie podcharakterystyki. Model został przedstawiony w tabeli Tab. 4.

II. **Software Engineering–Product quality – "External metrics"** – opisuje metryki jakości zewnętrznej, używane do pomiaru charakterystyk i podcharakterystyk zdefiniowanych w części pierwszej. Jakość zewnętrzna to charakterystyka końcowego produktu programowego postrzeganego z zewnątrz, wtedy gdy oprogramowanie zostało już wykonane np. z punktu widzenia kierownika projektu. Jakość zewnętrzna jest zależna od jakości wewnętrznej.

III. **Software Engineering–Product quality – "Internal metrics"** – identyfikuje metryki jakości wewnętrznej, używane do pomiaru charakterystyk i podcharakterystyk zdefiniowanych w części pierwszej. Jakość wewnętrzna to charakterystyka produktu programowego (najczęściej pośredniego) postrzeganego od strony producenta np. z punktu widzenia projektanta oprogramowania. Jest ona bazą do przewidywania i szacowania przyszłej jakości zewnętrznej produktu końcowego.

IV. **Software Engineering–Product quality– "Quality in use metrics"** – identyfikuje metryki jakości użytkowej, wykorzystywane do oceny oprogramowania z punku widzenia klienta/użytkownika produktu. Jest ona definiowana w książce „Cykl życia

oprogramowania" [6] jako *zdolność produktu programowego, która umożliwia określonym użytkownikom osiągnięcie konkretnych celów efektywnie, wydajnie, bezpiecznie i satysfakcjonująco w konkretnym kontekście użycia tego produktu*. Jakość użytkowa jest zależna od jakości zewnętrznej.

Jak łatwo zauważyć, pierwsze trzy części koncentrują się na opisie i miarach jakości produktu oprogramowania, podczas gdy część czwarta dotyczy jakości postrzeganej przez użytkownika. Na model jakości zaproponowany w części pierwszej normy, składają się zatem zgodnie z ISO/IEC 9126 [37] dwie główne części: model jakości wewnętrznej i zewnętrznej (ang. Internal and External Quality Model) oraz model jakości użytkowej (ang. Quality in use model). Na model jakości użytkowej składają się 4 charakterystyki: wydajność, produktywność, bezpieczeństwo i satysfakcja. Z kolei model jakości wewnętrznej i zewnętrznej oparty jest na sześciu charakterystykach: funkcjonalności, niezawodności, użyteczności, efektywności, łatwości utrzymania i przenośności. Każda z charakterystyk ma przypisany określony zestaw podcharakterystyk, a każda podcharakterystyka jest opisywana przez zewnętrzne i wewnętrzne atrybuty jakości, które mogą być mierzone za pomocą określonych metryk. Model ten został przedstawiony w tabeli Tab. 4.

Charakterystyka	Podcharakterystyka
Funkcjonalność (ang. Funcionality)	Adekwatność
	Dokładność
	Zdolność do współdziałania
	Zgodność funkcjonalna
	Zabezpieczenie
Użyteczność (ang. Usability)	Operacyjność
	Atrakcyjność
	Zrozumiałość
	Wyuczalność
	Zgodność użytecznościowa
Niezawodność (ang. Reliability)	Dojrzałość
	Zgodność niezawodnościowa
	Zdolność regeneracji
	Odporność
Wydajność (ang. Efficiency)	Szybkość działania
	Efektywność zużycia zasobów
	Zgodność efektywnościowa
Łatwość konserwacji, Pielęgnowalność (ang. Maintainability)	Modyfikowalność
	Stabilność
	Testowalność
	Podatność na analizę (analizowalność)
	Zgodność względem pielęgnowalności
Przenośność (ang. Portability)	Łatwość konfigurowania
	Łatwość instalacji
	Współistnienie
	Modyfikowalność
	Zgodność względem przenośności

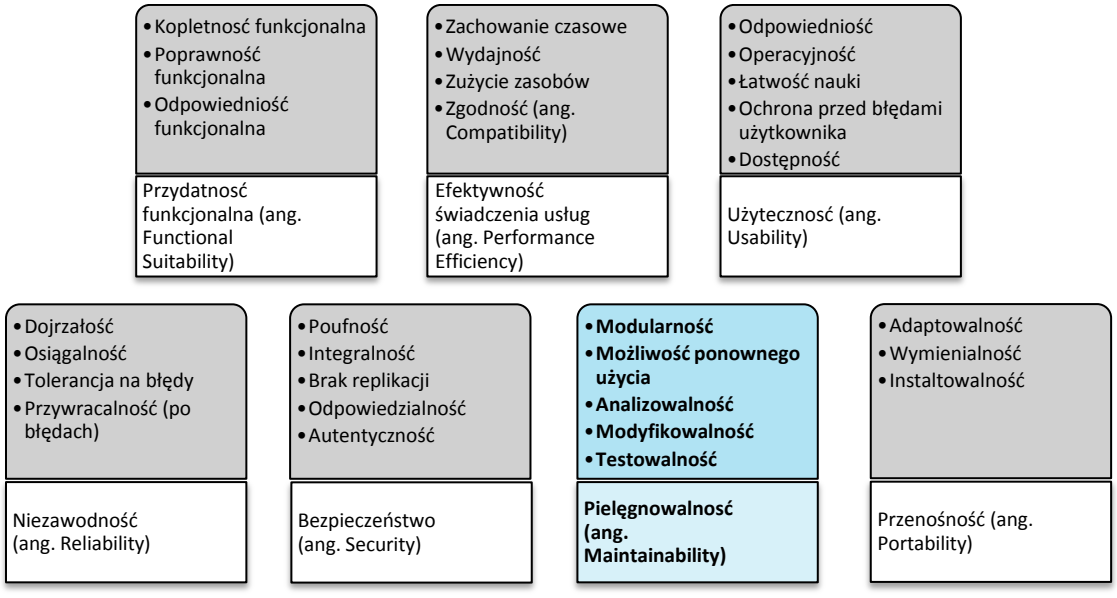
Tab. 4. Model jakości wewnętrznej i zewnętrznej ISO 9126. Na podstawie [31] i [13].

Model jakości ISO/IEC 25010 (SQuaRE)

Jak podano w opisie standardu ISO 25010 [26], zastępuje on standard ISO/IEC 9126-1:2001. ISO/IEC 25010 należy do zbioru norm SQuaRE (Software Product Quality Requirements and Evaluation), który ma na celu logiczną organizację, wzbogacenie i unifikację serii norm obejmujących specyfikowanie wymagań dotyczących jakości oprogramowania oraz ocenę jakości oprogramowania. Seria norm SQuaRE składa się z następujących działów:

- **ISO/IEC 2500n – Dział Zarządzania Jakością** (ang. Quality Management Division) – definiuje modele, terminy i definicje, o których mowa w innych standardach SQuaRE.
- **ISO/IEC 2501n – Dział Modeli Jakości** (ang. Quality Model Division) – prezentuje opis modeli jakości systemów komputerowych i produktu programowego oraz opis jakości użytkowej i danych.
- **ISO/IEC 2502n – Dział Pomiaru Jakości** (ang. Quality Measurement Division) – zawiera porady odnośnie specyfikacji wymagań jakościowych, użycia modeli i miar jakości.
- **ISO/IEC 2503n – Dział Wymagań Jakości** (ang. Quality Requirements Division) – pomaga specyfikować wymagania jakościowe, bazujące na modelach i miarach jakości.
- **ISO/IEC 2504n – Dział Oceny Jakości** (ang. Quality Evaluation Division) – pomaga oceniać jakość produktu programowego.
- **ISO/IEC 25050-25099 – Dział rozszerzeń** (ang. SQuaRE Extension Division) – zawiera wymagania na jakość oprogramowania komercyjnego tzw. oprogramowania "z półki" (ang. Commercial Off-The-Shelf software) oraz odnośnie formatów raportów użyteczności.

Standard ISO/IEC 25010 definiuje dwa rodzaje modelów jakości: model jakości użytkowej (ang. quality in use model) oraz jakość produktu (ang. produkt quality model). Model jakości użytkowej składa się z pięciu charakterystyk, które dalej podzielone są na podcharakterystyki. Są to: skuteczność (ang. effectiveness**),** wydajność (ang. efficiency), satysfakcja (ang. satisfaction), brak ryzyka (ang. freedom from risk) i pokrycie kontekstu użycia (ang. context coverage). Model jakości produktu składa się z ośmiu charakterystyk: przydatność funkcjonalna, efektywność świadczenia usług, użyteczność, niezawodność, bezpieczeństwo, pielęgnowalność, przenośność. Podzielone są one na podcharakterystyki.



Rys. 2. Model jakości proguktu ISO/IEC 25010. Źródło: [26].