

Modele jakości oprogramowania historia i perspektywy

Radosław Hofman, EUR ING

Student studiów Doktoranckich na Wydziale Informatyki i Gospodarki Elektronicznej
Akademii Ekonomicznej w Poznaniu radekh@pbpolsoft.com.pl

Abstrakt. Niniejszy artykuł nawiązuje do opublikowanych modeli jakości oprogramowania jako produktu, oraz pokazuje ich związek z kontekstem w jakim modele te były wytwarzane. Artykuł prezentuje również model, który jest w trakcie opracowywania przez ISO/IEC JTC1/SC7, komitet w którym w charakterze obserwatora uczestniczy Polski Komitet Normalizacyjny. Na zakończenie autor przedstawia rozważanie na temat zapotrzebowania na standardy rynkowe oraz przykład inicjatywy ich ustanowienia.

1. Wprowadzenie

Produkty IT zwane potocznie „oprogramowaniem” zostały zdefiniowane i wyodrębnione spośród innych produktów w drugiej połowie XX w. Produkty te odróżniają się od większości obiektów wytwarzanych wcześniej przez człowieka – z jednej strony są niematerialne, a z drugiej działają jako narzędzia w rzeczywistym i materialnym świecie.

Z wytwarzaniem produktów od zawsze związane było pojęcie jakości produktu. Nie można dziś zbadać dokonań człowieka w zakresie porównywania jakości narzędzi i przedmiotów w epokach kamienia łupanego i kolejnych, jednakże pierwsze wzmianki o ocenie jakości płótna lnianego przypisywane są Egipcjanom, a datowane na XXI w.p.n.e. Z kolei w kodeksie Hammurabiego z XVIII-XVII w.p.n.e. znajdujemy wzmianki o odpowiedzialności skutnika i budowniczego w przypadku produktu o złej jakości [24]. W starożytnym Rzymie nazwano zasadę *Caveat emptor*¹, która w istocie jest również uregulowaniem problematyki jakości nabywanego produktu. Analizując spojrzenie na jakość panujące już w czasach starożytnych należy wspomnieć o pismach Arystotelesa, który to postrzegał jakość jako jedną z 10 tzw. Kategorii. Jakość wg. tego filozofa to zbiór cech odróżniających rzecz od innych tego samego typu [1].

W czasach średniowiecza pojawiła się po raz pierwszy udokumentowana rola zawodowa, którą dziś nazwano by „kontrolerem jakości” – na przełomie XII i XIII w. Jan, król Anglii zatrudnił niezależnego eksperta Williama Wrotham’a aby w jego imieniu sprawował kontrolę nad produkowanymi na zamówienia króla statkami.

W czasach nowożytnych problematyka jakości i kontroli jakości opisana jest niemalże wyczerpująco. Powstały różne podejścia do zapewnienia i kontroli jakości opisane zarówno w standardach ISO, normach krajowych jak i publikacjach z zakresu

¹ *lat.* niech kupujący strzeże się

SPC (Statistical Process Control), SQC (Statistical Quality Control), TQM (Total Quality Management), Six Sigma itd. Podejścia te opisują metody planowania procesów zapewniających możliwość produkcji wysokiej jakości produktów oraz metody kontroli jakości procesów i produktów. Metody te są bardzo powszechnie implementowane w przemyśle, a funkcjonujące automatyczne linie produkcyjne wyposażane są w stanowiska automatycznej kontroli jakości.

Tematyka jakości jest na tyle dynamiczną dziedziną, że spowodowała powstanie nowej dyscypliny badawczej kwalitologii [12] a w obszarze jakości oprogramowania inżynierii użyteczności (usability engineering) [31].

Zanim zostanie przedstawiona analiza pojęcia jakości w odniesieniu do oprogramowania komputerowego należy podjąć próbę zdefiniowania, czym jest „jakość” w odniesieniu do jakiegoś obiektu? Większość definicji w literaturze odnosi jakość do konkretnego modelu lub obszaru stosowalności obiektu, dla którego jest ona definiowana. *Jakość można zatem zdefiniować jako zdolność obiektu do spełnienia zdefiniowanych i niezdefiniowanych potrzeb w określonym kontekście użycia.* Definicja ta jest bardzo szeroka, stąd też pozostawia duże pole do analizy znaczeniowej. Takie definicje jakości pojawiają się zarówno w normach ISO [14,17,18] oraz literaturze [24].

W przypadku oprogramowania komputerowego, które, jak wspomniano powyżej, różni się znacząco od pozostałych produktów wytwarzanych przez człowieka, sytuacja z jednej strony jest analogiczna do innych produktów, natomiast z odmiennej perspektywy zupełnie różna. Po pierwsze oprogramowanie jest bytem niematerialnym. Jako taki byt posiada szereg cech statycznych i dynamicznych. W odróżnieniu od innych produktów, w przypadku oprogramowania cechy statyczne są niemalże pomijalne w aspekcie jakości oprogramowania. Z punktu widzenia użytkownika można powiedzieć, że oprogramowanie tak samo jak inne produkty posiada określoną zdolność do zaspokajania potrzeb użytkownika, co nadaje mu podobne właściwości do innych produktów.

Pierwsze wzmianki o zapewnianiu jakości oprogramowania pojawiły się w publikacjach dotyczących Inżynierii Oprogramowania w literaturze lat 1960, gdzie IBM oraz Departament Obrony USA zdefiniowały po raz pierwszy takie pojęcie [34]. Od tamtej pory zdefiniowano wiele modeli jakości oprogramowania, a najważniejsze z nich zostaną omówione w rozdziale 2. W chwili obecnej trwają prace nad modelem SQuaRE [33], który wyrażany jest w normach ISO25000.

W wielu pozycjach w literaturze jakość oprogramowania jest wiązana z jakością procesu wytwarzania oprogramowania, choć są i takie pozycje które wyraźnie zaznaczają, iż jakość procesu nie ma wpływu na jakość produktu [10,23]. Trzecim istotnym punktem widzenia jest relacja dojrzałości organizacji i jakości produktów – związek taki jest prezentowany w samym modelu CMMI oraz w literaturze [11,7].

Abstrahując od sposobu wytwarzania oprogramowania można podjąć próbę zdefiniowania standardów jakości dla oprogramowania, prezentując różne punkty widzenia, co zostanie przedstawione w rozdziale 3. Dodatkową motywacją do zdefiniowania lub odnowienia modelu jest fakt zmiany sposobu korzystania z oprogramowania. Dziś zamiast oprogramowania coraz częściej mówi się o kompozycie usług świadczonych drogą elektroniczną, a co za tym idzie, kryteria jakości takie jak: łatwość instalacji, koszt utrzymania itp. nie mają jakiegokolwiek

zastosowania, z kolei pojawiają się nowe obszary definiujące użyteczność, czy jakość takiej usługi.

2. Modele jakości

2.1. Historyczne modele jakości

Jedynymi z pierwszych modeli w literaturze są modele McCall z 1977 roku [25] i Boehm z 1978 roku [4,28].

Model McCall, który z dzisiejszego punktu widzenia nazwalibyśmy perspektywą producenta oprogramowania, wprowadza zestaw oczekiwanych charakterystyk jakości, oraz pokazuje zestaw atrybutów/metryk, które na ową jakość mają wpływ.

Model ten, pomimo prób wdrożenia jest postrzegany jako ogólna koncepcja trudna do zastosowania w praktyce, ponieważ charakterystyki, które mają być mierzone są de facto bardzo subiektywne i nieprecyzyjne [30]

Model Boehm'a [4] zmienia punkt widzenia jakości oprogramowania przenosząc punkt ciężkości na perspektywę użyteczności oprogramowania. Według [28] jest to pierwsze wskazanie, że jakość oprogramowania może być postrzegana tylko wtedy, kiedy oprogramowanie jest użyteczne. Dla twórców modelu najważniejsze były 3 cechy oprogramowania: użyteczność (łatwość użycia, niezawodność i efektywność), utrzymanie (rozumienie, modyfikowanie i retestowanie), przenośność (rozumiana jako zmiana środowiska).

Kolejnym modelem, który zyskał szerokie uznanie na świecie był model zaprezentowany w 1991 roku w normie ISO9126. Model ten miał definiować charakterystyki jakości oraz być przewodnikiem dotyczącym sposobu ich oceny. Dostęcznie szybko stał się wyznacznikiem sposobu definiowania jakości oprogramowania [2].

Model jakości oprogramowania opisany we wspomnianej normie określał 6 głównych charakterystyk jakości: funkcjonalność (functionality), niezawodność (reliability), użyteczność (usability), wydajność (efficiency), łatwość utrzymania (maintanability), przenośność (portability).

Norma nie spełniła pokładanych w niej nadziei [28], głównie ze względu na ograniczenie się do perspektywy producenta, brak stanowiska w sprawie całościowej oceny jakości oraz brak dokładnych wytycznych jak należy mierzyć poszczególne parametry jakości.

Model zaproponowany przez Dromey'a w 1994 roku [8] i w 1998 [9] koncentruje się na jakości produktu w oderwaniu od sposobu jego wytworzenia, co było pewnym przełomem w postrzeganiu jakości oprogramowania w latach 90'tych XX wieku. Jakość produktu, według autora, ma być jakością artefaktów związanych z tym produktem. Przez produkt w tym modelu nie był rozumiany wyłącznie produkt końcowy, ale także dokumentacja, projekt itd.

Model jakości dzieli produkt na poziomy złożoności (od wyrażeń do złożonych systemów). Autor zwraca uwagę, że poszczególne cechy jakości technicznej określane dla jednostek prostych przenoszą się na jednostki wyższego poziomu kreując ich parametry jakościowe. Autor rozróżnia 3 grupy interesariuszy: Klient/Sponsor, Użytkownik, Administrator/Twórca i wskazuje główne cechy zainteresowania tych grup. Odpowiednio są to: Odpowiedniość (Funkcjonalność,

Pewność, Sprawność), Użytkowość (Łatwość opanowania, Łatwość obsługi), Adaptowalność (Łatwość pielęgnacji, Przenośność, Re-używalność kodu).

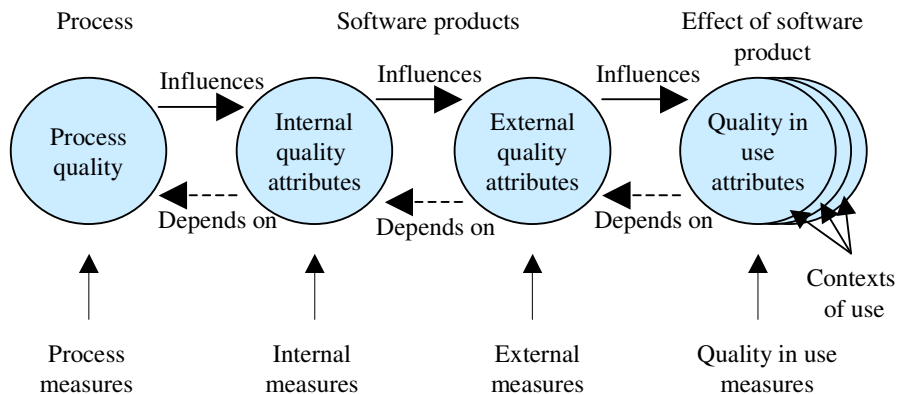
Podsumowanie spojrzeń na jakości oprogramowania zostały zaprezentowane przez Kitchenham & Pfleeger 1996 w publikacji [23] wyróżniając pięć perspektyw:

- jakość jako pojęcie metafizyczne, nieosiągalny ideał do którego zmierzamy
- jakość z punktu widzenia użytkownika jako odniesienie cech produktu do interesującego użytkownika kontekstu użycia
- jakość z punktu widzenia producenta jako zgodność z wymaganiami (analogiczny do spojrzenia ISO9001:1994
- jakość jako atrybut produktu wynikająca z konkretnych wyników pomiaru charakterystyk produktu
- ostatecznie jakość postrzegana jako wartość różna dla różnych grup interesariuszy

Autorzy podkreślali również, że jakość produktu jest zjawiskiem niezależnym od jakości samego procesu wytwórczego.

W 1999 przyjęto projekt prac nad modelem SQuaRE (Software Product Quality Requirements and Evaluation), który miał stać się drugą generacją modeli jakości oprogramowania. Pierwsze efekty tych prac zostały opublikowane w odnowionej normie ISO9126:2001. Norma ta radykalnie zmienia spojrzenie prezentowane w poprzedniej wersji rozróżniając trzy poziomy jakości oprogramowania i co za tym idzie definiując trzy zbiory charakterystyk: jakość wewnętrzna (internal quality), jakość zewnętrzna (external quality) oraz jakość użyteczna (in use quality) [33].

Model zdefiniowany w normie pokazuje również różne aspekty jakości dyskutowane we wcześniejszych modelach, czyli jakość produktu, jakość techniczną produktu, a jakość procesu, tak jak to widać na rysunku 1.



Rys. 1. Jakość w cyklu życia systemu [18]

Każdy z tych obszarów ma odmienne znaczenie i odmiennego interesariusza. Wewnętrzna i zewnętrzna jakość definiowana jest jako suma charakterystyk opisanych w poprzedniej wersji normy z 1991 roku, natomiast jakość użyteczna definiowana jest jako suma charakterystyk takich jak: przydatność (effectiveness),

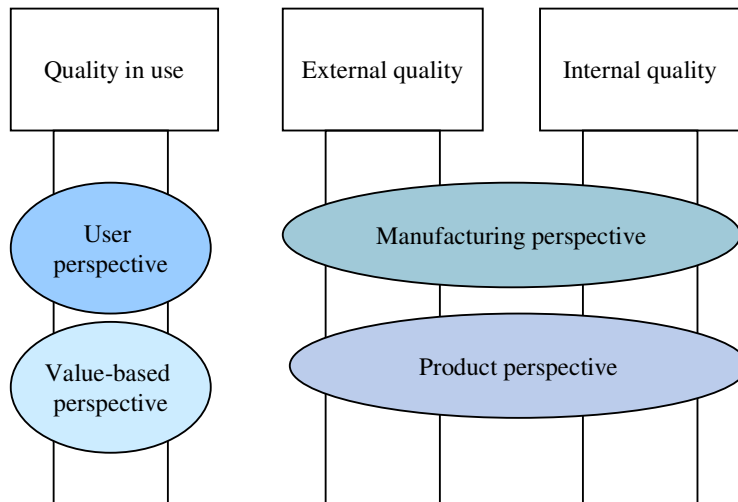
produktywność (productivity), bezpieczeństwo (safety) oraz zdolność do zaspokojenia potrzeb (satisfaction).

Warto zwrócić uwagę na to, że część z wspomnianych cech to cech obiektywnie mierzalne, a część to cechy postrzegane subiektywnie. W załącznikach do norm ISO9126-2 do -4 podane są przykłady skal (enumeratywna, uporządkowana, interwałowa, rankingowa, absolutna) oraz sposobów pomiaru cech abstrakcyjnych co pozwala traktować te cechy równorzędnie w stosunku do cech obiektywnych.

Norma definiuje również pojęcia miar jakości oprogramowania, charakterystyk i pod-charakterystyk. Dzięki temu model jest zupełny w rozumieniu podejść opisanych w IEEE1061. W wielu wypadkach opisany w tej normie model jest opisywany jako zwieńczenie wcześniejszych prac McCall, Boehm [6].

Dalsze prace nad modelem SQuaRE zaowocowały propozycją uporządkowania norm w serię norm ISO25000 (kolejne normy dotyczą: wprowadzenie do SQuaRE i zarządzanie, model jakości, miary jakości, wymagania jakościowe, ewaluacja jakości).

Na najwyższym poziomie modelu jakości oprogramowania pozostawiono podział na jakość wewnętrzną, zewnętrzną i użyteczną. Jakość wewnętrzna i zewnętrzna oprogramowania zdefiniowana jest analogicznie do definicji w ISO9126:2001, z tym że zmieniono charakterystyki wysokiego poziomu – zamiast sześciu w nowym modelu jest osiem – doszły charakterystyki poufność (security) oraz zgodność techniczna (interoperability).



Rys. 2. Perspektywy jakości mapowane na model SQuaRE

Zmienił się znacząco model jakości użytecznej oprogramowania, a jej główne charakterystyki określane są jako: użyteczność (usability in use), zgodność z kontekstem (context in use), bezpieczeństwo (safety in use), poufność (security in use) oraz łatwość użycia (adaptability in use) [15].

Warto zauważyć, że model jakości SQuaRE opisuje zarówno oprogramowanie, jak i dane. Twórcy normy zauważają, że nie są to jedyne modele jakości dla bytu

nazywanego „systemem”, lecz norma odnosi się tylko do oprogramowania i danych. Norma nawiązuje również do wcześniej publikowanych modeli jakości oprogramowania mapując przedstawiane tam perspektywy na swoje poziomy jakości oprogramowania, adresując perspektywy opisane w [23] na swoje poziomy, co pokazano na rysunku 2.

Warto również wspomnieć również o tym, że model jakości SQuaRE składa się z trój-poziomowych charakterystyk, które powiązane ze sobą zapewniają spełnienie wymagań opisanych w IEEE1061 dotyczące śledzenia od wymagań jakości do wymagań technicznych i od pomiaru charakterystyk do ogólnej oceny jakości.

Literatura w Polsce

W literaturze dostępnej w Polsce bardzo często jakość produktu utożsamiana jest z jakością procesu, w którym produkt ten powstaje. Przykładowo w [3] autorka wręcz stwierdza się, że nie można mówić o jakości produktu, jeżeli nie mówi się o jakości procesów.

Ciekawym przykładem publikacji, która choć w zamierzeniu nie była publikacją naukową autorzy zaprezentowali bardzo nowoczesne spojrzenie. Mowa o zasadach organizacji projektowania, wdrażania i eksploatacji systemów informatycznych w Resorcie Obrony Narodowej [26] o aspektach jakościowych wymagań dla systemów praktycznie nie ma wzmianki. Autorzy zaznaczają jednakże, że podstawowym wymaganie dla systemu jest potwierdzenie jego bezpieczeństwa w aspektach poufności, integralności i dostępności danych. Podstawą odbioru i oceny systemu, którą to można rozumieć jako ocenę jakościową, jest całość rozwiązania rozumianego jako aspekty informacyjne, funkcjonalne, organizacyjne, technologiczne i techniczne. Inaczej mówiąc dla systemów realizowanych na potrzeby obronności kluczowym nie jest charakterystyka systemu jako takiego, ale charakterystyka systemu i jego otoczenia umieszczonego w konkretnym środowisku oraz przydatność i efektywność tak przygotowanego systemu. Pomimo, że w czasie kiedy dokument powstawał obowiązująca postać normy ISO9126:1991 koncentrowała się na aspektach technicznych oprogramowania, to ich postrzeganie jakości jest zbieżne z obecnym rozumieniem prezentowanym w ISO25000.

Istotną pozycją literaturową traktującą o problematyce jakości oprogramowania i procesów wytwarzania oprogramowania bez wątpienia jest [22]. W pozycji tej autor przedstawia obszerną kwerendę literaturową nt. problematyki jakości, prezentuje historię modeli jakości oprogramowania przywołując model McCall [25], Boehm [4], Boeinga [5], FURPS [29], CUPRIMDSO [20], ISO9126:2001 i inne. Autor analizuje wspomniane modele oraz proponuje ujednoczenie znaczenia charakterystyk jakości pojawiających się w tychże, oraz pokazuje mapę występowania poszczególnych charakterystyk w tych modelach.

Odnotować należy również, że pomimo przywołania normy ISO9126:2001 opisywanych we wcześniejszej publikacji tego autora [21], i wspomnienia o pojęciu quality in use, jako wręcz wyodrębniającej się dziedzinie usability engineering (za [31]), autor nie analizuje aspektów ukierunkowanych na zaspokojenie potrzeb użytkownika, oraz charakterystyk jakości użytecznej oprogramowania, postrzegając model zawarty w tej normie jako model jakości technicznej oprogramowania.

Autor proponuje również własny model jakości oprogramowania GMSPQ (Generalized Model of Software Product Quality), wg. którego najważniejszymi charakterystykami jakości oprogramowania są:

- funkcjonalność (adekwatność, możliwości, uniwersalność, dokładność, zdolność do współdzielenia, współlistnienie, zabezpieczenia)
- użyteczność (zrozumiałość, wyuczalność, operacyjność, zasób danych we/wy, współczynnik we/wy, atrakcyjność, łatwość instalacji, łatwość konfigurowania, serwis, szkolenia, dokumentacja)
- niezawodność (dojrzałość, odporność, zdolność regeneracji, bezpieczeństwo)
- wydajność (szybkość działania, efektywność użycia zasobów, wymagane zasoby)
- łatwość konserwacji (korygowalność, elastyczność)

Jak widać, model pogłębia modele historyczne, nie naruszając ich spojrzenia. Autor sam przypisuje tym modelom koncentrację na jakości technicznej, stąd też należy uznać, że model GMSPQ jest również modelem jakości technicznej produktu.

3. Podsumowanie i wizja rozwoju modeli jakości

Perspektywy aktorów

Jak widać z lektury modeli jakości oprogramowania, modele były ściśle związane z określoną perspektywą patrzenia na jakość. Perspektywy opisane w [23] określają różne wymiary jakości jako wymiar użytkownika, producenta, produktu jako takiego, oraz wartości zmierzanej. Z kolei w normach ISO9126:2001 i ISO25010 rozróżnia się trzy poziomy jakości: wewnętrzną, zewnętrzną i użyteczną. Na rysunku 2 przedstawiono mapowanie wymienianych wcześniej punktów widzenia pokazujące relacje poszczególnych wymiarów jakości.

Należy rozważyć, czy modele jakości opisane w poprzednim rozdziale wyczerpują różne perspektywy jakie można przypisać aktorom, czyli osobom mającym związek z oprogramowaniem. Biorąc pod uwagę model zarządzania usługami IT (ITIL) [27], traktujący oprogramowanie szerzej niż tylko jako produkt, a mianowicie postrzegając oprogramowanie wraz z jego dynamicznym udostępnieniem jakości jako jeden byt zwany usługą, wprowadza pojęcia następujących klas aktorów:

1. Dostawca IT – przedstawiciel firmy zajmującej się utrzymaniem usług IT,
2. Klient – właściciel biznesowy podejmujący decyzję o potrzebach oraz ograniczeniach ekonomicznych,
3. Użytkownik – osoba korzystająca z usługi IT

Ten podział nie jest odległy od opisanych wcześniej modeli, jednakże po wprowadzeniu pojęcia „usługi IT” należy zauważyć, że pod pojęciem Dostawcy IT, kryje się szereg ról, których postrzeganie jakości może być zupełnie inne:

4. Właściciel - klasa przedstawicieli producenta których celem jest maksymalizacja ekonomicznych aspektów związanych z wytwarzaniem/rozwojem/utrzymaniem oprogramowania,
5. Deweloper - przedstawiciele producenta będący osobiście zaangażowani w proces wytwarzania oprogramowania,
6. Operator/Serwisant - przedstawiciel wykonawcy, dostawcy usługi IT lub wewnętrznego IT Klienta będący osobiście zaangażowany w proces utrzymania działającego oprogramowania.

Twierdzenie, że każda z powyższych klas aktorów postrzega jakość oprogramowania w inny sposób jest oczywistością. W poniższej tabeli nakreślono

główne (w sposób nie wyczerpujący) aspekty jakości oprogramowania istotne dla danej klasy aktorów:

Aktor	Główne cechy jakości
Klient (Acquirer)	Całkowity Koszt Utrzymania, użyteczność
Użytkownik (User)	Użyteczność, ergonomia
Właściciel IT (Supplier)	Koszt całkowity wytworzenia i utrzymania, przychody związane z oprogramowaniem
Deweloper (Developer)	Prostota technologiczna, przejrzystość części wewnętrznej
Operator/Serwisant (Operator/Maintainer)	Niezawodność oprogramowania, łatwość instalacji i odtworzenia

Tabela 1. Perspektywy jakości oprogramowania względem aktorów

Rozważając czy powyższa tabela jest wyczerpująca i oddaje wszystkie możliwe spojrzenia na jakość oprogramowania należy oczywiście odpowiedzieć, że nie. Niemniej jednak obserwując rozwój rynku IT oraz rosnący poziom organizacji wewnętrznej dostawców, należy spodziewać się, że zmierzanie rynku w stronę rosnącej liczby usług świadczonych w oparciu o infrastrukturę IT, modele jakości będą rozróżniały co najmniej powyższą listę perspektyw.

Perspektywy rodzajowe

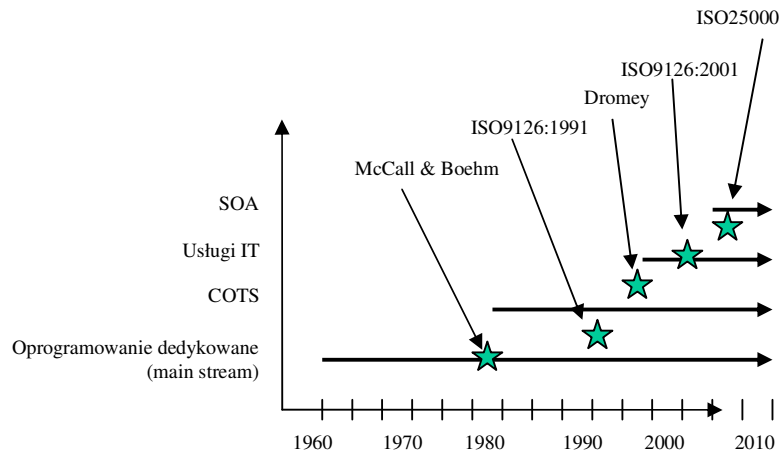
W opisanych w poprzednim rozdziale modelach jakości oprogramowania wprost, lub nie-wprost rozróżnia się typ tego oprogramowania. Modele tworzone w latach 70-tych ubiegłego wieku siłą rzeczy koncentrowały się na najczęściej spotykanych rodzajach oprogramowania, jakim były programy dedykowane. W latach 80-tych i 90-tych przy wzroście liczebności typu programów powtarzalnych (COTS Commercial off-the-shelf) modele jakości zaczęły z większą atencją traktować cechy oprogramowania istotne dla tego rodzaju produktów. Najnowsze modele, przygotowywane po zrewolucjonizowaniu spojrzenia na IT, jakich dokonały publikacje nt. ITIL, uwzględniają dedykowane usługi świadczone w oparciu o infrastrukturę IT. Wzrost liczby takich usług znajduje swoje odbicie w modelu SQuaRE.

Innym rodzajem rozróżnienia jest techniczny kontekst użycia danego fragmentu oprogramowania. Przykładowo [9] rozróżnia 10 poziomów złożoności fragmentów oprogramowania rozpoczynając od wyrażań, a kończąc na systemach będących zbiorem programów. Widać na tym przykładzie interesujący wymiar jakości – każdy produkt finalny może stać się komponentem bardziej złożonego systemu, a niektóre charakterystyki jakości tego komponentu staną się składowymi jakości systemu złożonego. Stwierdzenie to nabiera dodatkowo istotności w przypadku komponowania funkcjonalności dla użytkownika w oparciu o usługi świadczone w sieci, gdzie każda usługa jest produktem finalnym dla serwującej ją firmy, a elementem składowym nieznannej liczby różnorodnych procesów.

Jakie będą przyszłe spojrzenia na modele jakości ze względu na rodzaje oprogramowania? Otóż wschodzącą gwiazdą na rynku IT stają się usługi publiczne świadczone drogą elektroniczną. Usługi te różnią się od usług dedykowanych tym samym czym oprogramowanie COTS od oprogramowania dedykowanego. Dostawca i użytkownik stają się anonimowi, nie negocjują SLA (Service Level Agreement), nie

rozliczają się w oparciu o przedyskutowane raporty. Niemniej jednak należy definiować i mierzyć jakość oprogramowania bazującego na takich usługach i to uwzględniając wszystkie perspektywy aktorów opisanych w rozdziale 3. Usługi te zostaną zbudowane w oparciu o oprogramowanie mające swoją jakość wewnętrzną, zewnętrzną i użyteczną. Całościowa jakość z punktu widzenia użytkownika będzie jednak czymś więcej. Autorzy modelu SQuaRE również zauważyli tę tendencję i ten przyszły model jakości nazwali „Quality in use” odróżniając jakość użyteczną oprogramowania jako „Software quality in use”.

Reasumując modele jakości oprogramowania mogą odnosić się do różnych rodzajów oprogramowania: oprogramowania powtarzalnego (COTS), oprogramowania dedykowanego oraz usług IT. Takie spojrzenie po części wyjaśnia również przyczyny różnic w omawianych wcześniej modelach jakości oprogramowania.



Rys. 3. Sposoby korzystania z oprogramowania wraz z modelami jakości w czasie

Jeżeli model SOA będzie się popularyzował z pewnością wpłynie to na rozszerzenie listy podstawowych charakterystyk SQuaRE o parametry opisywane chociażby w serii norm ISO20000, jak np. dostępność, pojemność, ciągłość itd.

4. Standardy rynkowe

4.1.1. Czy są potrzebne

Pytanie zawarte w tytule tej sekcji jest raczej pytaniem retorycznym. Brak standardów jakości na rynku oprogramowania to szereg problemów i wątpliwości nie tylko dla Klientów i Użytkowników oprogramowania, ale także dla producentów. W przypadku producentów wątpliwości dotyczą tego co należy zrobić, aby osiągnąć produkt wysokiej jakości, lub inaczej mówiąc aby jakość produktu stała się elementem przewagi konkurencyjnej.

Powszechne wdrażanie normy ISO9001 oraz entuzjastyczne przyjęcie pierwszej wersji ISO9126:1991, jak również adaptowanie praktyk CMMI pokazują, że

producenci nie obawiają się inwestycji i nakładów na szeroko rozumiane zapewnianie jakości.

Pomimo tego, że na rynku pojawiają się bardziej szczegółowe wytyczne odnośnie wymogów dla oprogramowania niż w ISO9126:2001 lub ISO25000 (na przykład w ISO9241) to brakuje mechanizmów i standardów, które pozwoliłyby materializować wysiłki producentów i byłyby rozpoznawalne dla użytkowników.

4.1.2. Podjęta inicjatywa

Należy rozważyć jakie cechy powinny mieć standardy jakości dla oprogramowania aby ich pojawienie się wniosło realną wartość dla producentów i użytkowników. Autor tej publikacji wymienia kilka cech, które zostały zidentyfikowane w ramach prac nad stworzeniem standardów jakości oprogramowania [32]:

4.1.3. Perspektywa klienta i użytkownika

Przyjęto, że standardy rynkowe mają dostarczać informacji o produkcji użytkownikom i klientom branży IT. Wynika z tego, że zarówno w obszarze definiowania standardu główny głos powinni zabierać użytkownicy i klienci jak również, że to właśnie perspektywa użytkownika powinna być dominującym zestawem cech jakościowych.

Warto zauważyć, że choć drugi z wyżej wymienionych postulatów zauważono już w modelu Boehm [4], o tyle dotychczas budowane modele powstawały głównie na podstawie analiz naukowych lub prac organizacji wywodzących się z branży IT.

4.1.4. Oparcie merytoryczne o osiągnięcia dziedzinowe

Postulat ten jest raczej oczywisty i w pewnym sensie dobrze uzupełnia opisany powyżej. Celem jaki mają do osiągnięcia standardy jest stworzenie porozumienia pomiędzy społecznością odbiorców i producentów oprogramowania. Jednym z podstawowych wymagań dla takiego porozumienia jest jednoznaczna komunikacja, stąd też oparcie się o dotychczasowe dokonania w obszarze modelowania jakości oprogramowania i korzystanie z bieżących i przyszłych prac w tej dziedzinie wydaje się być nieodzowne.

4.1.5. Obiektywizm i własność standardu

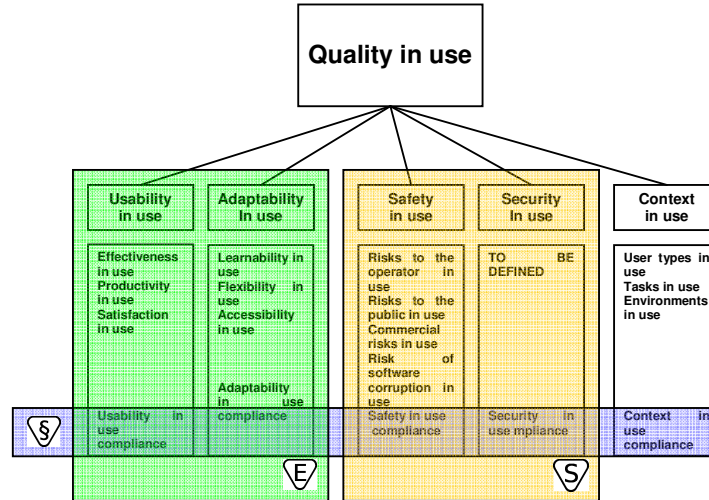
Standard jakości oprogramowania nie może preferować producentów oprogramowania, konkretnych technologii czy platform. Wynika z tego, że właścicielem standardu powinna być organizacja non-profit skupiająca użytkowników i przedstawicieli IT.

4.1.6. Porównywalność

Kolejnym wymogiem dla standardu jest możliwość odniesienia go do różnych produktów i wyciągnięcie wniosków o ich jakości z możliwością wprowadzenia relacji porządku. Patrząc na model SQuaRE można zauważyć, że część charakterystyk dotyczy kontekstu użycia aplikacji, który ze swojej natury nie nadaje się na obiektywny element standardu (mógłby powstać szereg standardów dla konkretnych dziedzin zastosowania). W modelu wybrano zatem te charakterystyki, które zapewniają możliwość stworzenia obiektywnych i porównywalnych miar, a ich odniesienie do SQuaRE prezentuje rysunek 4.

4.1.7. Dostęp do dokumentacji i możliwości ewaluacji

Standard może upowszechnić się jedynie wtedy, kiedy jego dokumentacja będzie powszechnie dostępna (najlepiej nieodpłatnie) oraz będzie istniało wiele podmiotów na rynku, które będą doradzały producentom w zakresie stosowania standardu lub dokonywały ewaluacji oprogramowania (nadania odpowiedniego znaku jakości).



Rys. 4. Trzy rodzaje certyfikatów mapowane na model SQuaRE w ISO25010-CD

5. Podsumowanie

Niezależnie od liczby zdefiniowanych do tej pory modeli jakości oprogramowania na rynku brakuje standardów, które pozwoliłyby producentom dyskontować swoje inwestycje w jakość produktów IT. Standardy odnoszące się do obecnych modeli zastosowania oprogramowania oraz przyszłych wyzwań, takich jak SOA mogą stać się istotnym elementem uwagi producentów, oraz dać im możliwość stosowania rozwiązań, których oczekuje społeczność użytkowników.

Przygotowywane standardy powinny bazować na dotychczasowych dokonaniach naukowych w obszarze modelowania jakości oprogramowania oraz rozszerzać te modele o konkretne decyzje o włączaniu, wyłączaniu i wadze poszczególnych charakterystyk. Jedną z pierwszych tego typu inicjatyw bazującą na modelu SQuaRE i nie opublikowanych jeszcze normach ISO25010 i innych z serii ISO25000 powstała w Polsce [32]. Inicjatywa ta nie jest skazana na sukces, ale z pewnością może wpłynąć zarówno na wewnętrzny rynek IT w Polsce jak również wizerunek polskiego IT na zewnątrz.

6. Bibliografia

1. Arystoteles, Kategorie, w Dzieła Wszystkie, Wydawnictwo Naukowe PWN, 1990

12 Radosław Hofman, EUR ING

2. Bazzana G., Anderson O., Jokela T., ISO9126 and ISO9000: friends or foes?, Software Engineering Standards Symposium
3. Begier B., Inżynieria Oprogramowania - Problematyka Jakości, WPP 1999
4. Boehm B., Brown J., Lipow M., MacClead G., Characteristics of software quality, NY, American Elsevier, 1978
5. Bowen T., Wigle G., Tsai J., Specification of Software Quality Attributes, RADC-TR-83-37, Vol 1-3, Boeing Aerospace Company, 1985
6. Côté M-A, Suryan W., Georgiadou E., "Software Quality Model Requirements for Software Quality Engineering", Software Quality Management & INSPIRE Conference (BSD) 2006
7. Diaz M., Sligo J., How Software Process Improvement Helped Motorola, IEEE Software 17(5), 1997
8. Dromey R., A Model For Software Product Quality, Australian Software Quality Research Institute 1994 http://www.sqi.gu.edu.au/docs/sqi/technical/Model_For_S_W_Prod_Qual.pdf
9. Dromey R., Software Product Quality: Theory, Model and Practises, Australian Software Quality Research Institute 1998 <http://www.sqi.gu.edu.au/docs/sqi/misc/SPO-Theory.pdf>
10. Dromey R., Cornering the Chimera, IEEE Software 13(1), 1996
11. Eickelman N., An Insider's View of CMM Level 5, IEEE Software 20(4), 2003
12. Hamrol A., Mantra W., Zarządzanie jakością, PWN 2002
13. IEEE1998 – Standard for a Software Quality Metrics Methodology, IEEE 1061, 1998
14. ISO25000:2005 – JTC1/SC7, Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE), International Standardization Organization, 2005
15. ISO25010-CD – JTC1/SC7, Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE), Dokument wewnętrzny ISO/IEC JTC1/SC7, Obecnie w stadium CD – Commission Draft, 2007
16. ISO9001:2000 – TC176/SC2, Quality management systems – Requirements, International Standardization Organization, 2000
17. ISO9126:1991 – JTC1/SC7, Information Technology – Software Product Quality, International Standardization Organization, 1991
18. ISO9126:2001 - JTC1/SC7, Software engineering - Product quality, International Standardization Organization, 2001
19. ISO9241-11:1998 – TC159/SC4, Ergonomic requirements for office work with visual display terminals (VDTs) , International Standardization Organization, 1999
20. Kan S., Metrics and Models in Software Quality Engineering (second edition), Addison Wesley, 2003
21. Kobyliński A., ISO/IEC 9126 - analiza modelu jakości produktów programowych, w: Systemy Wspomagania Organizacji 2003, T. Porębska-Miącz, H. Sroka [red.], Prace Naukowe AE w Katowicach, 10, 2003.
22. Kobyliński A., Modele jakości produktów i procesów programowych, Oficyna Wydawnicza Szkoły Głównej Handlowej, 2005
23. Kitchenham S., Pfleeger, Software Quality: The Elusive Target, IEEE Software 13(1) 1996
24. Kiliński A., Jakość, WNT 1979
25. McCall J., Richards P., Walters G., Factors In software quality, Griffiths Air Force Base, NY, Rome Air Development Center Air Force Systems Command, 1977
26. Tymczasowa Instrukcja Organizacji Projektowania, Wdrażania i Eksploatacji Systemów Informatycznych w Resorcie Obrony Narodowej, Sygn. Łączn. 1000/98, Ministerstwo Obrony Narodowej 1999
27. ITIL The Key To Manageing IT Services, Office of Government Commerce, 2000
28. Pfleeger S., Software Engineering: theory and practise, Secondo editio, Upper Sadle River, NJ, Prentice Hall, 2001
29. Pressman R., A Manager's Guide to Software Engineering, McGraw-Hill, 1993
30. Pressman R., Software Engineering: A practitioner's approach, fifth edition, Boston McGraw-hill, 2001
31. Rautenberg M., Sikorski M., Usability Engineering in Software Development, Tutorial Notes, Politechnika Gdańska, 1998
32. Strona Polskiego Stowarzyszenia na rzecz Atestacji i Standaryzacji Oprogramowania, <http://www.saso.org.pl>, 2007
33. Suryan W., Abran A., ISO/IEC SQuaRE. The second generation of standards for software product quality, IASTED2003
34. Voas J., Assuring Software Quality Assurance. IEEE Software, 20(3)