Stanisław Stanek*

ANALIZA WYBRANYCH KONCEPCJI W OBSZARZE PROJEKTOWANIA WYMAGAŃ

Wprowadzenie

W kontekście podejmowanej działalności projektowej pojawiają się zazwyczaj cztery podstawowe pytania: o cele, dla których system ma być realizowany *(*dlaczego?), o metareguły obowiązujące w procesie realizacji systemu (jak?), o interesariuszy, których system ma wspomagać (kto?), o usługi jakie system ma realizować (co?). Specyfikę projektów informatycznych odzwierciedla ciąg ukierunkowanych na informatykę zaleceń organizacji normalizacyjnych. W szczególności standard ISO/IEC 12207-2008, opracowany dla potrzeb łatwiejszego łącznego posługiwania się standardami 21207 oraz 15504, definiuje używane w artykule pojęcia klient[1], użytkownik[2], developer[3] oraz interesariusz[4]. Interesariusze będą wspomagać działania projektowe w nawiązaniu do postrzeganych potrzeb jakie projektowany system będzie zaspakajał. Ian Sommerville w jednej z częściej cytowanych pozycji [Somm03] podejmujących problematykę analizy wymagań pisze: „Problemy, które mają rozwiązywać inżynierowie oprogramowania, są często bardzo złożone. Zrozumienie ich natury może być trudne, zwłaszcza w wypadku nowych systemów. Trudno jest zatem dokładnie ustalić, co system powinien robić. Opisy usług i ograniczeń są wymaganiami stawianymi systemowi. Proces wynajdowania, analizowania, dokumentowania oraz sprawdzania tych usług i ograniczeń nosi nazwę „inżynierii wymagań".

* Prof. nadzw. dr hab. inż. Stanisław Stanek, Wyższa Szkoła Oficerska Wojsk Lądowych im. generała Tadeusza Kościuszki, 51-150 Wrocław, ul. Czajkowskiego 109, e-mail: stan_stanek@neostrada.pl

1 Customer – organizacja lub osoba, która otrzymuje produkt lub usługę.

2 User – jednostka lub grupa, korzystająca z systemu podczas jego użytkowania.

3 Developer – organizacja, która wykonuje zadania rozwoju (w tym analizy wymagań, projektowania, testowania) podczas procesu cyklu życia.

4 Stakeholder – osoba lub organizacja mająca prawo, akcje, roszczenia albo zainteresowana systemem lub jego charakterystykami nawiązującymi do potrzeb oraz oczekiwań. Można zauważyć (za [Some03, s. 122]), że w określaniu i analizowaniu wymagań mogą wziąć udział osoby z różnych stanowisk, np.: użytkownicy, którzy będą pracować z systemem, inżynierowie budujący lub pielęgnujący inne powiązane systemy, menadżerowie przedsiębiorstwa, eksperci z danych dziedzin, a być może nawet reprezentanci związków zawodowych.

Problematyka inżynierii wymagań wydzieliła się z obszaru badań nad inżynierią oprogramowania. Za prekursorów tej problematyki badawczej uważa się Bella oraz Thayera, którzy w 1976 roku w następstwie analizy badań empirycznych konkludowali, że:

- niekompletne, niewłaściwe, niezgodne lub niejasne wymagania są liczne i mają krytyczny wpływ na jakość oprogramowania,
- wymagania dla systemu nie powstają w sposób naturalny, przeciwnie, muszą być zaprojektowane a następnie konsekwentnie przeglądane i dostosowywane.

Rozwój podstaw metodycznych inżynierii wymagań oraz szerzej inżynierii oprogramowania, jaki nastąpił w latach 80., nie spowodował znaczącego postępu w obszarze prac projektowych, co ilustrują np. następujące często cytowane wyniki raportu Standish Group's:

- 31% projektów zostało przerwanych przed zakończeniem,
- 53% projektów kosztowało ponad 189% wartości estymowanych,
- tylko 16% projektów zakończyło się zgodnie z planem (termin, koszty),
- projekty ukończone dostarczały jedynie 42% oryginalnych cech oraz funkcji,

Ponadto trzy najczęstsze problemy projektowe to:

- brak informacji wejściowych pochodzących od użytkownika – 13%,
- niekompletność wymagań oraz specyfikacji – 12%,
- zmiany wymagań oraz specyfikacji – 12% [StGr94].

Wyniki te oznaczają, że na progu nowego tysiąclecia projektowanie wymagań w dalszym ciągu sprawiało znaczące trudności projektantom.

Dalsza interpretacja badań empirycznych z tego okresu utrwalała przekonanie o potrzebie bardziej adekwatnej pracy z użytkownikiem. Podejście zorientowane na użytkownika (UOD – User Oriented Design) uzyskało międzynarodową nobilitację najpierw w 1999 roku w postaci standardu ISO 13407, a ostatnio po aktualizacji, w nawiązaniu do numeracji zgodnej z innymi standardami użyteczności, w postaci obowiązującego obecnie standardu ISO 9241-210[5]. Standard jest postrzegany jako manifest skierowany do rozproszonych środowisk stosujących UOD nawołujący do zjednoczenia. Organizacje, które decydują się na jego wykorzystanie uzyskują terminologiczne i metodyczne wsparcie wypracowane przez międzynarodowe gremia ekspertów.

Krytyczną dyskusję dotychczasowych praktyk w obszarze specyfikowania wymagań wskazującą na przesłanki wprowadzenia podejścia zwinnego przedstawioną przez Dona Reinertsena we wprowadzeniu do ostatniej książki Deana Lefingwella można streścić następująco:

[5] W Polsce 21 lutego 2011 roku ukazał się w postaci normy PN-EN ISO 9241-210:2011 Ergonomia interakcji człowieka i systemu -- Część 210: Projektowanie ukierunkowane na człowieka w przypadku systemów interaktywnych.

- badania niezmiennie ukazują, że od 80% do 85% niepowodzeń w projektowaniu wiąże się z niepoprawnymi wymaganiami. Wiemy o tym od ponad dekady, jednak nie udało nam się dotychczas tego wyniku poprawić;
- dlaczego? Początkowo byliśmy organizowani funkcjonalnie, po prostu problem pozostawał poza granicami inżynierii – mogliśmy obwiniać marketing i zarządzanie produktem. Później, gdy przyjęto cross-funkcjonalne zespoły nakazano, aby słuchać głosu klienta, co miało rozwiązać problem;
- nie rozwiązało. Zignorowano fakt, że niejednokrotnie klienci nie wiedzą, czego chcą. Jeśli nawet wiedzą, to nie potrafią tego opisać. A jeśli potrafią, to raczej opisują swoje rozwiązanie problemu, a nie samą rzeczywistą potrzebę. Aby trzymać się prawdy, nie ma jednego „głosu klienta". Mamy do czynienia z kakofonią głosów wskazujących na różne rozwiązania. Jeśli skupimy się jedynie na użytkowniku, to możemy przeoczyć „wymagania niefunkcjonalne". A przecież, szczególnie w kontekście dynamiki zmian w otoczeniu, nie można interesariuszom odmówić prawa do rozwoju, do zmian, bo wówczas będziemy ich ograniczać, a nie wspomagać;
- tak więc musimy zaprzestać „chowania głowy w piasek", bardziej niż mozolnie i kosztownie tworzyć **idealne** wymagania, musimy inwestować w organizowanie procesów i infrastruktury, wspierających wykrywanie i korygowanie braku dopasowania pomiędzy naszymi rozwiązaniami a zmieniającymi się potrzebami klienta [Lefi11].

Celem artykułu jest analiza trzech wymienionych nurtów badawczych (inżynieria wymagań, podejście zorientowane na użytkownika, podejścia zwinne) w kontekście rozwiązań proponowanych dla potrzeb projektowania wymagań. Artykuł ma zidentyfikować, opisać oraz poddać pod dyskusję wybrane, proponowane w ramach wymienionych koncepcji, mechanizmy.

1. Prace nad inżynierią wymagań

Podejście inżynierskie koncentruje się na wykorzystaniu sprawdzonych podstaw teoretycznych w procesie rozwiązywania problemów lub tworzenia produktów. Nawiązując do przytoczonej we wstępie opinii Iana Sommerville, inżynieria wymagań jest zorientowana na wyodrębnienie oraz analizę procesu wymagań w organizacji. Aktywności w procesie inżynierii wymagań analizowali m.in. Houdek oraz Pohl [HoPo00], Lamswerde [Lams09][6]:

- **tworzenie projektu**: aktywność związana z uruchomieniem projektu w celu opracowania nowego produktu lub modyfikowania istniejącego produktu;
- **odkrywanie wymagań** (określane też jako zbieranie pozyskiwanie, wydobywanie wymagań): aktywność, w ramach której developer zwykle w interakcji z interesariuszami działają w celu zdefiniowania dziedziny projektu usług, jakie system powinien zapewniać oraz przyszłych ograniczeń;

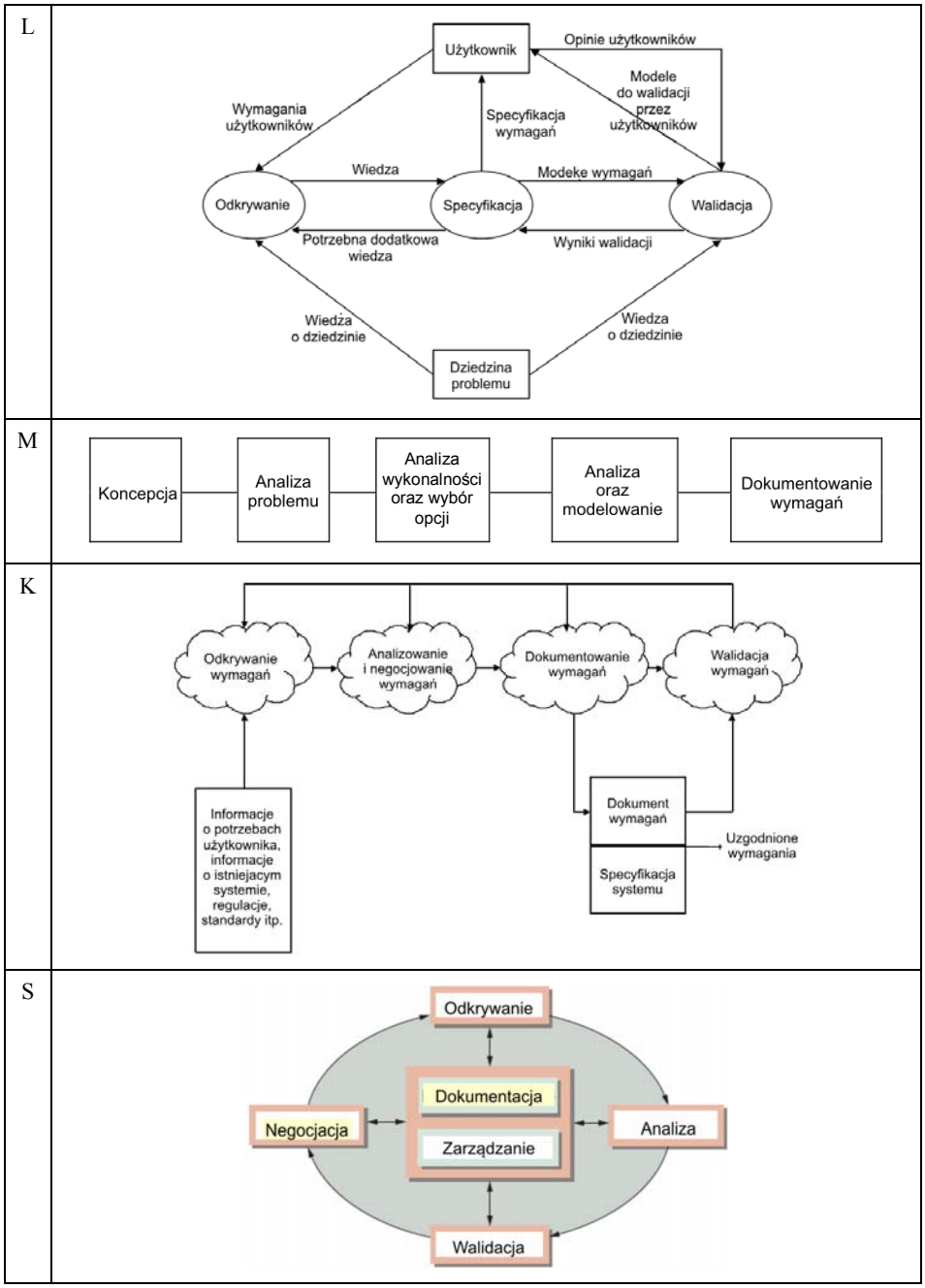
[6] Szerszy opis poszczególnych aktywności można także znaleźć np. w [Lams09].

- **analiza (doskonalenie) wymagań**: po odkryciu wymagania są interpretowane oraz strukturalizowane. Następnie są dokumentowane. Tutaj aktywność ta jest wyodrębniona, jednak często przeplata się z odkrywaniem wymagań, ponieważ analiza niezmiennie ma swoje miejsce w fazie odkrywania;
- **negocjacja wymagań**: aktywność związana z negocjacjami z interesariuszami w celu uzyskania porozumienia, co do definicji wymagań;
- **sprawdzanie wymagań**: weryfikacja formalna ma na celu zapewnienie formalnej poprawności wymagań. Nawiązuje do poprawności zapisu oraz wewnętrznej zgodności. Walidacja wymagań ma z kolei na celu stwierdzenie czy zebrane wymagania i stworzony dokument wymagań definiują system zgodny z oczekiwaniami interesariuszy. Dobrą praktyką jest organizowanie inspekcji wymagań, w których biorą udział interesariusze lub ich przedstawiciele;
- **zarządzanie wymaganiami**: dotyczy planowania oraz zarządzania zmianami wymagań. Zarządzanie zmianą sprawia, że ustandaryzowane informacje są gromadzone dla każdej zmiany oraz że całkowite koszty oraz korzyści proponowanych zmian są analizowane, dlatego też zarządzanie zmianą obejmuje ocenę ryzyka i analizę oddziaływań;
- **śledzenie wymagań**: pozwala na analizowanie pochodzenia wymagań, związków zachodzących pomiędzy wymaganiami oraz między wymaganiami a ich realizacją w projektowanym systemie. Ułatwia utrzymanie spójności w obliczu zmian.

Do wielokrotnie cytowanych później modeli procesu inżynierii wymagań można zaliczyć (rys. 1):

- L – model iteracyjny Loucopoulosa, Karakostasa [LoKa95],
- M – model liniowy Macaulaya [Maca96],
- K – model liniowy z iteracjami między działaniami Kotonaya oraz Sommervile'a [KoSo98],
- S – podsumowujący doświadczenia model cyklów aktywności Sommervile'a [Some05].

Przeprowadzane w dalszej kolejności, w dwóch firmach A i B, badania [SAJP02] zmierzające do wskazania, który z trzech modeli M, K, L adekwatnie odzwierciedla realizowane w praktyce prace nad wymaganiami, choć odsłoniły wiele prawidłowości, to jednak nie przyniosły rozstrzygnięcia. W dwóch projektach (A2, B2) inżynieria wymagań była realizowana jako wydzielona faza, a proces wymagań nawiązywał do modelu liniowego M. W trzech pozostałych projektach inżynieria wymagań była realizowana ciągle w czasie całego procesu projektowania.



Rys. 1. Modele inżynierii wymagań

Odnotowano wiele iteracji, co było szczególnie wyraźne, gdy w ramach przyrostów były realizowane prototypy (A3). Podsumowując, należy zauważyć, że proces inżynierii wymagań kształtuje się w nawiązaniu do szerszego kontekstu. Model przyrostowy inżynierii wymagań L w miarę dobrze oddaje iteracyjny charakter procesu projektowania, jednak nie pokazuje progresji w procesie projektowania. Mając na uwadze powyższe doświadczenia, za podstawę do dalszych rozważań przyjęto późniejszy model S (por. rys. 1).

Tabela 1

Badania nad modelami L, K, M

	A – duża firma produkcja			B – duża firma finanse	
	Proj. A1	Proj. A2	Proj. A3	Proj. B1	Proj. B2
Cel projektu	Wdrożenie ERP	Upgrade systemów	Wsparcie promocji	Rozwój witryny	Wymiana CRM
Rozmiar	Duży	Średni	Mały	Średni	Mały
Osób	26-100	11-25	1-10	1-10	1-10
Osobomiesięcy	540	24	3	100	8
Priorytet	Czas/Koszt	Czas/Funk.	Funk.	Czas	Czas/Funk.
Tworz. projektu	Nie	Explicite	Implicite	Implicite	Explicite
Odkrywanie wym.	Explicite	Explicite	Explicite	Explicite	Explicite
Analiza wym.	Explicite	Implicite	Explicite	Explicite	Implicite
Negocjacje wym.	Implicite	Implicite	Implicite	Explicite	Explicite
Zarządzanie wym.	Explicite	Implicite	No	No	Implicite
Śledzenie wym.	Implicite	Implicite	No	No	Implicite
Model	(K)	(M)	(L)	(K) (L)	(K) (M)

Źródło: Na podstawie: [SAJP02].

Obserwowane w praktyce duże zróżnicowanie procesów inżynierii wymagań po części tłumaczy koncepcja dojrzałości procesowej. Dojrzałość procesowa wyraża się zakresem, w jakim procesy są formalnie: zdefiniowane, zarządzane, elastyczne, mierzone i efektywne [Gaje03]. W latach 90. wielu developerów oprogramowania, w celu doskonalenia procesów wewnetrznych, podjęło prace (por. np. [Paulk95]) nad wdrożeniem opracowanego przez Software Engineering Institute SEI[7], modelu Capability Maturity Model for Software (SW-CMM). W 2000 roku wydał CMMI-SE/SW zintegrowany model obejmujący zarówno problematykę oprogramowania, jak również problematykę inży-

[7] Przy Cornegie Melion University, por. http://www.sei.cmu.edu/.

nierii systemów. Zarówno SW-CMM, jak również CMMI-SE/SW dostarczają specyficzne rekomendacje dla rozwijania oraz zarządzania wymaganiami. Krótki opis tych modeli z punktu widzenia problematyki wymagań można znaleźć w załączniku do książki Karla Wiegersa [Wieg03]. Modele te nie pokrywają całości procesu wymagań (por. np. [Rog98; BeHR03]), co zaowocowało dużą ilością modeli dojrzałości ukierunkowanych na doskonalenie procesu wymagań u developerów. Oto najlepiej zweryfikowane modele, zalecane do przeanalizowania: REGPG [SoSa97], REPM [GoTe02], RMM [Heum03], R-CMM [BeHR03], [TrAl08], Uni-REPM [Nguy10], IAG [WWW1]. Ze szczególnym zainteresowaniem spotkał się pierwszy z wymienionych modeli (por. np. [KaAK02; SoRa05; XuSS06; SDYS09]). Jak zauważają autorzy, „(…) odmiennie niż CMM lub ISO 9001-3, REGPG w swoim zamiarze nie ma służyć jako standard lub do celów akredytacji, lecz jako praktyczny przewodnik łatwy do zrozumienia, a także łatwy w stosowaniu". W procesie oceny developera rozważono 66 dobrych praktyk sklasyfikowanych według ośmiu obszarów oraz trzech poziomów (Zał. 1). Przydzielono oceny powszechności stosowania każdej z praktyk.

- 0 – praktyka nigdy niestosowana (lub prawie nigdy),
- 1 – praktyka stosowana sporadycznie (jeśli ktoś zadecyduje o jej zastosowaniu),
- 2 – praktyka często stosowana (nie jest jednak standardem),
- 3 – praktyka zawsze stosowana (jest standardem).

Obliczono wskaźniki sumaryczne S1 – suma ocen za kryteria z poziomu podstawowego oraz S23 – suma ocen za kryteria z pozostałych poziomów. Poziom dojrzałości odczytano z poniższej tabeli.

Tabela 2

Poziomy dojrzałości modelu REGPG

Lp.	Poziom dojrzałości	Warunek	Komentarz
1	Początkowy	S1 < 56	Proces wymagań planowany ad hoc. Wynik zależy od realizatorów
2	Powtarzalny	S1 > 55, S23 < 41	Developer otwarty na narzędzia wspomagające prace oraz na ulepszanie praktyk
3	Zdefiniowany	S1 > 85, S23 > 40	Developer stosuje dobre praktyki inżynierii wymagań na co dzień

Źródło: Na podstawie: [SoSa97].

Słabo ocenione praktyki stają się wskazówką dla potrzeb dalszego doskonalenia. Poziomy 1 oraz 2 korespondują z odpowiednimi poziomami modeli SEI, w przypadku poziomu trzeciego, w nawiązaniu do znaczącej nieokreśloności w obszarze wymagań nie da się dokładnie sprecyzować poziomu korespondującego modelu SEI (3, 4 lub 5).

W analizie przypadków skonfrontowano jak przedstawione modele i zalecenia są realizowane w praktyce. Arao, Goto i Nagata [ArGN05] podejmują istotny problem przejścia od modelu biznesowego do modelu systemowego, zestawiając „jaki być powinien proces wymagań" z „jaki proces wymagań jest" por. tab 3.

Tabela 3

Lp.	Jak być powinno?	Jak jest?
1	Klienci i developer zgadzają się na nowy proces biznesowy dla realizacji celów projektu	Spojrzenie klienta oraz developera na nowy proces nie jest jednoznaczne
2	Developer adekwatnie wydobywa oraz definiuje wymagania klientów dotyczące systemu/oprogramowania	Developer określa wymagania klienta do pewnego stopnia, ale nie na poziomie szczegółowym
3	Developer definiuje specyfikację, która spełnia wymogi klienta	Developer odkrywa wymagania klienta w nawiązaniu do postępu prac nad specyfikacją wymagań
4	Developer zarządza zmianami wymagań szybko i dokładnie, zgodnie ze zmianami wymagań klienta	Developer nie jest zdolny zarządzać zmianami wymagań adekwatnie oraz terminowo, z powodu zbyt dużej liczby zmian

Popularne wyjaśnienie dla rozbieżności między tym, co być powinno, a tym, co jest, głosi, że użytkownik nie jest zdolny stanowić o swoich preferencjach przy przejściu do modelowania systemowego bez kontaktu z artefaktami świata rzeczywistego. Autorzy rozważanych badań nie do końca podzielają powyższe wyjaśnienie. Analizując przykładowy proces „śledzenie wyników biznesowych w tygodniu poprzedzającym złożenie zamówienia", dostrzegają złożoność techniczną (m.in. potrzeba współdziałania punktu sprzedaży utrzymującego sprzedaż z ubiegłego tygodnia z systemem finansowym naliczającym wskaźniki oraz mechanizmami adekwatnej prezentacji). Każdemu procesowi odpowiada wiele funkcji systemowych, następuje zmiana terminologii oraz dodanie szczegółów technicznych, które mogą w istotny sposób oddziaływać na proces biznesowy. Narzuca się tutaj powiedzenie, że niestety „diabeł tkwi w szczegółach" takiej operacji. W rzeczywistości ponieważ niemożliwe jest wyartykułowanie wymagań w tym samym czasie zarówno dla klientów, jak i developerów, to proces musi być do pewnego stopnia ciągły i interaktywny, aby wymagania okazały się zrozumiałe i dokładne. Autorzy budują narzędzia autorskie wspomagające

specyfikowanie oraz śledzenie (tracing) wymagań pozyskiwanych w procesie przejścia od modelu biznesowego do SRS na podstawie Microsoft Excel oraz Access. Dalsze prace projektowo wdrożeniowe prowadzą do następujących konkluzji:

- nie można rozpoczynać procesu przechodzenia do SRS zanim spojrzenie biznesowe nie będzie ustabilizowane,
- należy starać się negocjować model wymagań ze wszystkimi interesariuszami,
- proponowane rozwiązanie sprawdzało się w różnorodnych realizowanych projektach,
- brak szkoleń członków projektu oraz liderów powoduje niepowodzenie procesu implementacji.

2. Podejście zorientowane na użytkownika

Podejście zorientowane na użytkownika (UOD – User Oriented Design) uzyskało międzynarodową nobilitację najpierw w 1999 roku w postaci standardu ISO 13407, a następnie po aktualizacji, w nawiązaniu do numeracji zgodnej z innymi standardami użyteczności, w postaci obowiązującego obecnie standardu ISO 9241-210[8]. Standard jest postrzegany jako manifest skierowany do rozproszonych środowisk stosujących UOD nawołujący do zjednoczenia. Organizacje, które decydują się na jego wykorzystanie uzyskują terminologiczne i metodyczne wsparcie wypracowane przez międzynarodowe gremia ekspertów.

Rozważania rozpoczęto od przypomnienia pryncypiów podejścia zorientowanego na użytkownika (Por. Zał. 3).

W standardzie ISO 9241-210 podstawową, wyróżnioną aktywnością jest identyfikowanie i specyfikowanie wymagań. Zaleca się utworzenie jednoznacznie określonych wymagań użytkownika w stosunku do zamierzonego kontekstu wykorzystania oraz celów biznesowych systemu.

Tradycyjne podejścia inżynierii wymagań koncentrowały się na identyfikacji funkcjonalnych wymagań oraz na zapewnieniu, aby rozwijany produkt je spełniał. Inne niefunkcjonalne wymagania (wydajności, niezawodności, użyteczności, łatwości konserwacji, rentowności) miały mniejsze znaczenie. Dopiero z perspektywy użytkownika, niefunkcjonalne wymagania stają się krytyczne dla powodzenia we wdrożeniu nowego systemu. Z perspektywy użytkownika zasadne staje się pytanie czy dotychczasowe procesy, również te, które wiążą się z działaniem przyszłego systemu, ale nie będą informatyzowane,

[8] W Polsce 21 lutego 2011 ukazał się w postaci normy PN-EN ISO 9241-210:2011. Ergonomia interakcji człowieka i systemu -- Część 210: Projektowanie ukierunkowane na człowieka w przypadku systemów interaktywnych.

nie powinny być zreorganizowane. W szerszym oraz lepiej umocowanym gronie interesariuszy rozwija się problematyka modelowania biznesowego, która nabiera większego znaczenia oraz uzyskuje potrzebne możliwości realnego oddziaływania. Zaczyna uwierać niezrozumiała dla użytkownika tradycyjna specyfikacja. Rozwija się alternatywa tradycyjnej specyfikacji, w tym prototypy i bardziej iteracyjne oraz przyrostowe podejścia, znajdujące swoje miejsce w rozszerzonej specyfikacji wymagań, która przekonująco opisuje w jaki sposób przyszły system będzie funkcjonował. Przyszły użytkownik wiele uwagi poświęca ważnej i zrozumiałej dla niego specyfikacji wejść i wyjść. Napotykając na różnorodne problemy w komunikacji bezpośredniej, można dostrzec potrzebę modelowania umysłowości odbiorcy naszej pracy, utworzenia tzw. modelu mentalnego. Model mentalny daje głębsze zrozumienie ludzi, ich motywacji, procesów myślowych, a także otoczenia emocjonalnego i filozoficznego działań. Może być reprezentowany poprzez diagram afiniczny (podobieństwa) zachowań użytkowników [You08].

W wielu pracach są analizowane zalety i wady modelowania biznesowego oraz systemowego z wykorzystaniem UML oraz innych rozwiązań. Wiele interesujących, z perspektywy problematyki specyfikowania wymagań, analiz przedstawił Martin Glinz ze współpracownikami. W artykule z 2000 roku [Glin00] w nawiązaniu do analizy przykładowego systemu (zdalnej opieki medycznej) przedstawił, szeroko wzmiankowane w innych pracach, dziewięć słabości analizy wymagań opartej na UML[9]. W kolejnym artykule [FrGK06] w nawiązaniu do wcześniejszych badań podjęto obserwację oraz analizę pracy wyróżnionego zespołu projektowego. W trakcie pierwszego miesiąca pracy zespół poszukiwał, techniką prób i błędów, adekwatnej do kontekstu, metody pracy z wymaganiami. Poczynając od najprostszych metod, rozwijał swój arsenał przydatnych w danej sytuacji rozwiązań, dochodząc do metody Ericssona-Penkera, która została przyjęta jako metoda wiodąca. W trakcie kolejnych dwóch miesięcy udało się zespołowi z wykorzystaniem wypracowanych, w szerokim zakresie autorskich, podstaw metodycznych opracować adekwatną specyfikację wymagań. W trakcie poszukiwań zespół zaadaptował Enterprise Architecta, które to narzędzie zostało z powodzeniem wykorzystane w trakcie dalszych prac.

[9] 1. Brak elementów aktywnych w diagramach przypadków użycia. 2. Brak możliwości reprezentowania kontekstu – zależności między aktorami. 3. Brak możliwości adekwatnego modelowania struktury przypadków użycia i hierarchii. 4. Brak możliwości adekwatnego specyfikowania interakcji między przypadkami użycia. 5. Trudność modelowania zależnych od stanu zachowań systemu. 6. Nieporęczne modele przepływu informacji. 7. Brak możliwości modelowania zachowań charakterystycznych dla komponentów wysokiego poziomu, takich jak podsystem. 8. Trudności przy modelowaniu dekompozycji systemu rozproszonego. 9. Brak aspektowo zorientowanego spojrzenia na złożony system.

3. Podejście zwinne

Podejście zwinne wyłoniło się w kontekście dyskusji nad cyklem życia oprogramowania na początku naszego wieku. Dominujące w poprzednim wieku modele: kaskadowy, spiralny oraz prototypowania, wypracowane przez takie autorytety, jak Winston Royce, Barry Bohm, Daniel McCracen, Michael Jacobson oraz Garry Russel Gladden dały podstawę dla metodyk określanych jako twarde, z racji ich ukierunkowania na rozległą dokumentację, rozbudowane procesy, długie oraz sekwencyjne fazy. Jak wynika jednakże również z przedstawionych wcześniej rozważań, w praktyce szczególnie w niewielkich firmach rozwiązania te w całości nie przyjęły się oraz trwały poszukiwania alternatywnych (lekkich) rozwiązań (por. tab. 4).

Tabela 4

Przykładowe lekkie rozwiązania metodyczne
jakie wyłoniły się w latach 90.

Rok	Metodyka	Twórca	Liczba praktyk	Liczba ról	Liczba artefaktów
1991	Crystal Methods	Alistair Cockburn	14	10	25
1993	Scrum	Jeff Sutherland	5	3	5
1993	Dynamic Systems Dev.	Grupa firm brytyjsk	15	12	23
1998	Extreme Programming	Kent Beck	28	7	7

Twórcy podejść alternatywnych dostrzegając potrzebę wymiany doświadczeń, w 2000 roku skorzystali z zaproszenia Kenta Becka i spotkali się po raz pierwszy na konferencji w Oregon. Bob Martin we wrześniu 2000 roku zaprosił liderów wyłaniającego się lekkiego podejścia na następne spotkanie w 2001 roku. Alister Cockburn rekomendował zastąpienie, używanego dla potrzeb identyfikacji proponowanych rozwiązań metodycznych, określenia „lekkie" określeniem „zwinne". Następne spotkanie odbyło się w dniach 11-13 lutego 2001 roku w Utah, gdzie zgromadziło się 17 entuzjastów nowego kierunku. Rezultatem spotkania było powołanie zrzeszenia nazywanego Agile Alliance mającego na celu promowanie podejść zwinnych i wspieranie osób zainteresowanych [WWW4]. Ogłoszono Manifest Zwinnego Wytwarzania Oprogramowania sformułowany w postaci czterech ogólnych reguł oraz dwunastu szczegółowych zasad (por. np. [MaMa08, s. 39-46]). Już pierwsze spojrzenie na pryncypia podejścia zwinnego wskazuje na zasadniczą zmianę optyki patrzenia na problematykę analizy wymagań. Rozwój polega na rozszerzeniu dotychczasowego logicznego, systemowego spojrzenia o: kontekst retrospektywnej, krytycznej

analizy dotychczasowych praktyk, przejęte z nauk zarządzania zagadnienia organizacji pracy wysoko wydajnych zespołów oraz tworzenie samoorganizujących się, szczupłych rozwiązań, a także na fascynacji kulturą kaizen.

Bardziej szczegółową analizę rozpoczęto od zagadnienia wyłaniania wizji systemu. Podstawowym założeniem podejścia zwinnego jest ograniczenie wstępnych prac koncepcyjnych (tzw. paraliżu przez analizę), krótki cykl dostarczenia produktu, a następnie szybkie adaptacje produktu zgodnie z aktualną odpowiedzią płynącą z rynku. Niemniej jednak osoba reprezentująca interesariuszy, w SCRUM – właściciel produktu, w trakcie sesji poświęconej planowaniu wydania poddaje pod dyskusję wizję produktu. Aby dobrze spełnić swoje zadanie, właściciel projektu powinien zastanowić się nad obszarami projektu (tzw. drajwery projektu), w których oczekuje na generowanie wartości, patrząc z punktu widzenia biznesu. Uczestnicy mogą zastanawiać się nad zwrotem z inwestycji. W nawiązaniu do problemu projektowania produktu dobrze jest wcześniej opracować tzw. minimalny zestaw jego cech marketingowych (MMFS). Zwinna wizja komunikuje intencje strategiczne oraz odpowiada na następujące pytania:

- dlaczego budujemy ten produkt, system lub aplikację?
- jaki problem rozwiązuje?
- jakie cechy oraz korzyści dostarcza?
- komu są dostarczane te cechy oraz korzyści?
- jaka wydajność, niezawodność i skalowalność wiąże się z produktem?
- jak wygląda porównanie produktu z istniejącymi już na rynku oraz z przygotowywanymi do wydania?
- jak będziemy uzyskiwać zysk na sprzedaży? Jakie będą źródła przychodów oraz jaki będzie model biznesowy?
- czy mamy zdolność wykonania produktu? Czy będziemy w stanie produkt rozwijać? Jaka będzie roadmapa rozwoju produktu?

Analizując dobre praktyki, szczególnie w obszarze RUP, gdzie dokument wizji jest jednym z kluczowych, szczegółowo opracowywanym artefaktem, Dean Leffingwell zaleca [Leff11], aby również w zwinnych projektach, zwłaszcza w dużych, taki dokument opracowywać. Postuluje jednak, aby w nawiązaniu do zasad podejścia zwinnego opracowywać tylko jeden dokument wizji liczący od 5 do 10 (maks. 20) stron, tak aby aktualizacja tego dokumentu nie była uciążliwa. W załączniku do ostatniej książki Deana można znaleźć przykładowy wzorzec dokumentu wizji dla potrzeb projektów zwinnych (4 strony).

W odpowiedzi na pytanie jak wypełnić wzorzec wizji dla potrzeb projektów zwinnych Roman Pichler [Pich10] zaleca takie techniki, jak: prototypy (papierowe, sketches, spikes, mock-up), personas and scenerios (identyfikowanie się z klientem oraz zbadanie jak produkt oddziałuje na jego życie), vision box lub trade journal review (zastanowienie się nad trzema wypunktowaniami „sprzedającymi" produkt do centralnego umieszczenia na opakowaniu lub przez

retrospekcje przeanalizowanie, co chciałoby się przeczytać o produkcie po jego wydaniu), Kano model (projektowanie funkcjonalności produktu poprzez wyróżnienie funkcji podstawowych, wydajnościowych oraz wywołujących zachwyt).

Znaczącym wkładem podejścia zwinnego do problematyki specyfikowania wymagań jest wyartykułowana (Kent Beck) pierwotnie w ramach podejścia ekstremalnego (XP) koncepcja krótkich wypowiedzi użytkowników (historii użytkownika) – spisu rzeczy jakie system powinien zdaniem użytkownika robić. Historia użytkownika rozpoczyna tzw. grę planistyczną, w ramach której najpierw w wyniku intensywnych kontaktów między developerami oraz użytkownikami zbyt duże historie są dzielone na mniejsze oraz zbyt małe są scalane w większe. Developerzy przypisują każdej historii użytkownika koszt jej wykonania, a użytkownicy wybierają historie, które będą realizowane w następnej iteracji, nawiązując do szacowanej wydajności zespołu. W wyniku prac Mike'a Cohna historie użytkownika zostały zaadaptowane również w SCRUM jako narzędzie do budowy rejestru produktowego oraz przy definiowaniu zawartości poszczególnych iteracji – tzw. Sprintów. W nawiązaniu do problematyki specyfikowania wymagań historia użytkownika może być postrzegana jako wysokopoziomowy opis wymagań do zaimplementowania. W zrozumieniu tego związku pomaga rozwijana w ramach podejścia zwinnego koncepcja poruszającego dokumentu[10] – spełniającego rolę pamięci zewnętrznej ilustrującej w obrazowej, poruszającej wyobraźnie formie bieżącą komunikację w zespole projektowym. Poruszające dokumenty silnie nawiązują do kontekstu, są dokumentacją do bieżącego wykorzystania w odróżnieniu od dokumentów reprezentacyjnych (por. np. [Elss10]). W podejściu zwinnym historia użytkownika jest przykładem poruszającego dokumentu o bardziej sformalizowanej postaci (np. według Mike'a Cohna „Jako <rodzaj użytkownika>, chcę <cel>, tak aby <uzasadnienie>" [Co04, s. 127]), który przykładowo mógłby przyjąć następującą postać: „Jako użytkownik standard chcę, aby przy dużym jednorazowym zakupie był udzielany rabat, tak aby duży jednorazowy zakup był bardziej konkurencyjny".

W podejściach zwinnych centralne znaczenie, z perspektywy wymagań, ma proces opracowywania testów w nawiązaniu do praktyk: zautomatyzowane testy programisty, wytwarzanie kończone testami lub wytwarzanie poprzedzone testami (WPT). Jakkolwiek dalsze rozważania są w dużej części prawdziwe również dla dwóch pierwszych praktyk, to najbardziej wyraźne powiązania można uzyskać w przypadku wdrożenia praktyki WPT. Na wstępie należy zauważyć, że sztuka pisania testów rozwiązuje w ramach podejść zwinnych problem wyższego piętra specyfikacji wymagań. Odnosi się to zarówno do testów jednostkowych (białej skrzynki), gdy programista powodowany wymaganiami wyraża swoje intencje w formie testu, jak również akceptacyjnych (czarnej skrzynki),

[10] Ang. *evocative document*.

w których interesariusz sprawdza czy jego wymagania zostały właściwie zrozumiane, a programista implementuje wymagania w postaci kodu. W jednej z konkluzji książki Roberta Martina i Mican'a Martina można przeczytać, że „(…) zarówno testy jednostkowe, jak i akceptacyjne pełnią funkcję swoistej dokumentacji. Ponieważ dokumentacja w tej formie może być kompilowana i wykonywana, jej wiarygodność i precyzja nie powinny budzić najmniejszych wątpliwości. Co więcej, tego rodzaju testy są pisane w jednoznacznych językach zrozumiałych dla właściwych grup odbiorców" [MaMa08, s. 82]. Nie można zakończyć analizy problematyki dokumentowania wymagań w podejściach zwinnych bez spojrzenia z perspektywy refaktoryzacji kodu, stanowiącego w swojej istocie również kolejne piętro specyfikacji wymagań. Dla programisty kod nierefaktoryzowany jest kodem ciężkim, zwykle niezrozumiałym, a jego leczenie w formie ekstremalnej nawiązuje do leczenia objawowego według mechanizmu czarnej skrzynki. Przeciwnie lekki, refaktoryzowany kod stanowi o zdolności do przetrwania oprogramowania w burzliwym, zmieniającym się otoczeniu.

Z uwagi na wymogi dotyczące objętości artykułu, przeanalizowano jedynie wyzwanie jakie stoi przed twórcami gier komputerowych. Dynamika rozwoju kosztów jest w tym obszarze znacznie większa niż dynamika rozwoju rynku. Branża staje na rozdrożu: Czy czeka ją kryzys podobny do tego z 1983 roku, czy też uda się wypromować nowe rynki, a może wystarczy odchudzająca terapia z wykorzystaniem podejścia zwinnego? W tym sektorze rynku liczy się radość z grania, o której można orzekać trzymając w ręku manipulator. Każdy dzień zyskany dla bardziej obiecującego projektu poprzez precyzyjniejszą identyfikację oraz wycofywanie projektów brnących ku klęsce może stanowić o „być albo nie być" firmy. Znamienne są doświadczenia firmy CCP Games zatrudniającej ponad 400 pracowników w trzech lokalizacjach: Reykjavik, Atlanta oraz Shanghai. Jesienią 2008 roku, CCP podjęła ambitne wyzwanie zaprojektowania dodatku do EVE Online o nazwie Apokryfy w ciągu sześciu miesięcy. W przypadku Apocrypha udana realizacja oznaczała współdziałanie ponad 120 developerów w 13 zespołach Scrum zatrudniających 9 właścicieli produktu oraz pomieszczonych na trzech kontynentach (por. [Keit10]). Zwinne rozwiązania sprawdzają się więc również w dużych projektach.

Podsumowanie

Uczyniono znaczące wysiłki badawcze w zakresie poszukiwania modelu inżynierii wymagań. Weryfikacje empiryczne wskazują, że pierwsze wielokrotnie cytowane modele mają charakter raczej normatywny niż opisowy. Model Sommerville'a wydaje się być rozsądnym kompromisem. Ciekawą własnością tego modelu jest jego podobieństwo do zalecanego dla procesów innowacyjnych modelu Stewarta – Deminga PDCA.

Kolejny rozważony problem dotyczy rozwoju dojrzałości provaiderów w obszarze wymagań. Modele CMMI-SW dostarczają stabilną, choć nie zawsze wystarczającą podstawę. Wskazane środowiska badawcze oraz provaiderzy rozwijają te rozwiązania dodając własne narzędzia, które doskonalą zwarty schemat myślowy zaproponowany w jednym z pierwszych tego typu rozwiązań, zaproponowany przez Sommerville'a oraz Sawyera. Z uwagi na brak opracowania tego modelu w literaturze krajowej został on w artykule szerzej scharakteryzowany.

Umiejętne wykorzystanie narzędzi, również autorskich, wspomagających proces zarządzania wymaganiami może znacząco wpłynąć na skuteczność developera. Wykorzystanie tego typu narzędzi niestety nie jest powszechne. Aktualny krótki przegląd narzędzi można znaleźć np. w [ShSM11] .

Wydanie normy PN-EN ISO 9241-210:2011 inspiruje do przedstawienia oraz dalszego przedyskutowania pryncypiów podejścia zorientowanego na użytkownika. Podejście to rekomenduje potrzebę, metody oraz dobre praktyki głębszego uwzględnienia kontekstu użytkownika w procesie prac nad wymaganiami.

Introspektywne badania Glinza identyfikują występujący w praktyce proces poszukiwania adekwatnych do kontekstu konkretnych rozwiązań w nawiązaniu do ogólnych modeli, norm oraz praktyk . Ten kluczowy, słabo opisany, proces determinuje sukces lub porażkę projektu.

Kontekst wymagań w podejściu zwinnym jest istotny, coraz lepiej rozumiany oraz opisany w aktualnie ukazującej się literaturze. Wyłaniają się cztery poziomy szczegółowości rozważań:

- poruszająca, kolektywnie rozwijana wizja,
- napędzające kolejne iteracje, adekwatnie skalowane oraz wybierane do realizacji historie użytkownika,
- sprawujące piecze nad zmianami testy,
- refaktoryzowany kod.

Wypracowane w ramach różnych podejść rozwiązania mogą być integrowane adekwatnie do potrzeb. Przykładem może być rozwijanie dojrzałości wymagań w zespołach wykorzystujących podejście zwinne.

Literatura

[ArGN05] Arao T., Goto E., Nagata T.: *„Business Process" Oriented Requirements Engineering Process*. RE '04 Proceedings of the 13th IEEE International Conference on Requirements Engineering. IEEE Computer Society, Washington.

[BeHr03] Beecham S., Hall T., Rainer A.: *Software Process Improvement Problems in Twelve Software Companies: An Empirical Analysis*. „Empirical Software Engineering" 2003, Vol. 8, No. 1.

[Co04] Cohn M.: *User Stories Applied for Agile Software Development*. Addison--Wesley, 2004.
[Elss10] Elssamadisy A.: *Agile wzorce wdrażania praktyk zwinnych*. Helion, Gliwice 2010.
[FrGK06] Fricker S., Glinz M., Kolb P.: *A Case Study on Overcoming the the Requirements Tar Pit*. „Journal of Universal Knowledge Management" 2006, Vol. 1, No. 2.
[Gaje03] Gajewski P.: *Koncepcja struktury organizacji procesowej*. TNOiK, Toruń 2003.
[Glin00] Glinz M.: *Problems and Deficiencies of UML as a Requirements Specification Language*. Procedings of the 10th International Workshop on Software Specification and Design (IWSSD-10), San Diego, November 2000.
[GoTe02] Gorschek T., Tejle K.: *A Metod for Assessing Requirements Engineering Process Maturity in Software Projects*. Blenking Institute of Technology, Master Thesis Computer Science no. MSC-2002:2, 2002.
[Heum03] Heumann J.: *The Five Levels of Requirements Management Maturity*, http://www.therationaledpe.com/conIenI/feb 03/f_managementMaturity_jh. jsp [dostęp: 28.11.2012].
[HoPo00] Houdek F., Pohl K.: *Analyzing Requirements Engineering Process a Case Study*. Proceedings of the 11th International Workshop on Database and Expert Systems Applications, Greenwich, UK 2000.
[KaAK02] Kauppinen M., Aaltio T.. Kujala S.: *Lessons Learned from Applying the Requirements Engineering Good Practice Guide for Process Improvement*. In: Proceedings of the 7lh European Conference on Software Quality, Helsinki.
[Keit10] Keith C.: *Agile Game Development with Scrum*. Addison-Wesley, Boston 2010.
[KoSo98] Kotonya G., Sommerville I.: *Requirements Engineering – Processes and Techniques*. John Wiley & Sons. UK 1998.
[Lams09] Lamsweerde A.: *Requirements Enginering*. John Wiley & Sons, 2009.
[Leff11] Leffingwell D.: *Agile Software Requirements*. Pearson Education, 2011.
[Maca96] Macaulay L.A.: *Requirements Engineering*. Springer-Verlag, Berlin, Heidelberg, New York 1996.
[MaMa08] Martin R., Martin M.: *Agile programowanie zwinne*. Helion, Gliwice 2008.
[Nguy10] Nguyen T.: *The Creation of Uni-REPM*. Blenking Institute of Technology, Master Thesis Computer Science no. MSE-2010:27, 2010.
[Paulk95] Paulk M., Weber C., Curtis B., Chrissis M.: *The Capability Maturity Model: Guidelines for Improving the Software, Process*. Addison-Wesley, Reading--MA, Boston 1995.
[Pich10] Pichler R.: *Agile Product Management with SCRUM*. Addison-Wesley, 2010.
[PISO11] PN-EN ISO 9241-210:2011 Ergonomia interakcji człowieka i systemu -- Część 210: Projektowanie ukierunkowane na człowieka w przypadku systemów interaktywnych.

[Rog98] Rogoway P.: *How to Reap the Business Benefit from SPI: Adding SPICE while Preserving the CMM (Motorola): SPI NEWSPAPER*, http://www.iscn.at/select_newspaper/assessments/motorola.html [dostęp: 28.11.2012].

[SAJP02] Sacha M., Aurum A., Jeffery J., Paech B.: Proceedings of the Seventh Australian Workshop on Requirements Engineering Process Models in Practice, http://www.deakin.edu.au/events/awre02/Melbourne, 2002, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-69 [dostęp: 28.11.2012].

[SAJP02] Sacha M., Aurum A. Jeffery R., Paech B.: *Requirements Engineering Process Models in Practice*. *The seventh Australian Workshop on Requirements Engineering: proceedings*, Melbourne, Victoria, School of Information Systems, Deakin University, 2002.

[SDYS09] Shrivastava A., Darhan M., Yagyasen D., Singh V.: *An Efficient Evaluation of Requirements Engineering Maturity Measurement Framework For Medium and Small Scale Software Companies*. Proceedings of the 3rd National Conference; INDIA Com-2009, Computing For Nation Development, February 26-27, 2009.

[ShSM11] Shahid M., Ibrahim S., Mahrin M.: *An Evaluation of Requirements Management and Traceability Tools*. World Academy of Science, Engineering and Technology 78, 2011, http://www.waset.org/journals/waset/v54/v54-117.pdf [dostęp: 28.11.2012].

[Somm03] Sommerville I.: *Inżynieria oprogramowania*. WNT, Warszawa 2003.

[Somm05] Sommerville I.: *Integrated Requirements Engineering: A Tutorial*. IEEE Software, 2005.

[SoRa05] Sommerville I., Ransom J.: *An Empirical Study of Industrial Requirements Engineering Process Assessment and Improvement*. ACM Transactions on Software Engineering and Methodology. 14(1).

[SoSa97] Sommerville L Sawyer P.: *Requirements Engineering: A Good Practice Guide*. Wiley, Chichester, l997.

[StGr04] Standish Group: *Charting the Seas of Information Technology – Chaos*. The Sandish Group International, West Yarmouth 1994.

[TrAl08] Tripathi S., at all: *An Efficient Evaluation of Requirements Engineering Process Maturity Assessment and Improvement*. Proceedings of the 2nd National Conference; INDIA Com-2008, Computing For Nation Development, February 08-09, 2008.

[Wieg03] Wiegers K.: *Software Requirements*. Microsoft Press, 2003.

[WWW1] http://www.iag.biz/business-analysis-resources/landing/self-assessment-tool.html [dostęp: 28.11.2012].

[WWW4] http://www.agilealliance.org [dostęp: 28.11.2012].

[XuSS06] Xu H., Sawyer P., Sommerville I: *Requirement Process Establishment and Improvement from the Viewpoint of Cybernetics*. „The Journal of Systems and Software" 2006, 79.

[Youn08] Young I.: *Mental Models. Aligning Design Strategy with Human Behavior.* Rosenfeld Media, 2008.

Załącznik 1

Zestawienie wybranych technik inżynierii wymagań

Technika	Mocne strony	Słabe strony
Wywiad Dla zewnętrznego obserwatora przypomina rozmowę, jednak role są zróżnicowane. Osoba przeprowadzająca wywiad – ankieter dąży do pozyskania informacji oraz może posługiwać się wcześniej przygotowanymi pytaniami tzw. wywiad ustrukturalizowany lub jedynie dyspozycjami do wywiadu, czyli luźno sformułowanymi problemami – tzw. wywiad swobodny, nieustrukturalizowany (sytuacja bardziej zbliżona do naturalnej rozmowy). Osoba odpowiadająca – respondent stara się przekazać wiedzę w odpowiedzi na pytania. Pytania mogą być zamknięte, gdy zawierają określone możliwości odpowiedzi tzw. kafeterie (koniunktywne lub dysjunktywne) lub otwarte gdy respondentowi daje się swobodę odpowiedzi, a ankieter przytacza odpowiedź dosłownie.	• możliwość pozyskania bogatych oraz szczegółowych danych; wywiad penetruje zakamarki zagadnienia • pozwala na rozwinięcie / uzupełnienie wiedzy ankietera • u respondenta rozwija się zaangażowanie oraz zrozumienie projektu • naturalna komunikacja z uwzględnieniem elementów niewerbalnych ułatwia podejmowanie / zrozumienie trudnych problemów, otwartość oraz szczerość wypowiedzi.	• wymaga znaczącego zaangażowania czasu oraz wysiłku, głównie ankietera, ale także respondenta • wymaga zaangażowania osób o wysokich umiejętnościach w obszarze komunikacji interpersonalnej • trudna do zastosowania w przypadku dużej ilości zróżnicowanych interesariuszy – potrzeba wspomagania poprzez ankiety • efekty uboczne, takie jak społecznych oczekiwań, halo, Rosenthala, ...
Wywiad grupowy zogniskowany[1] Druga co do powszechności wykorzystania, po przypadkach użycia, technika zaliczana do jakościowych metod gromadzenia danych, polegająca na przeprowadzeniu i analizie kontrolowanej dyskusji „kolektywnej konwersacji" na dany temat pomiędzy uczestnikami małych (6-12 osób) starannie selekcjonowanych grup. Zwykle nie dążymy do uzyskania konsensusu, raczej chodzi o prezentacje/negocjacje znaczeń, idei, opinii, oczekiwań, obserwacji, postaw oraz zachowań.	• wspierają kreowanie wizji, formułowanie pytań oraz weryfikowanie koncepcji, gdy brak jest idei, hipotez, wiedzy, danych • możliwość obserwowania oraz wpływania na zachowania w grupie • możliwość bardziej efektywnego wykorzystania czasu niż w przypadku wywiadu, tematy bardziej wszechstronnie naświetlone	• wysokie kwalifikacje osób prowadzących oraz interpretujących wyniki • rezultaty różnią się zazwyczaj od wcześniejszych oczekiwań, mogą być specyficzne • wkład uczestników jest często zróżnicowany • szczegółowe, specyficzne, sensytywne zagadnienia mogą okazać się nieodpowiednie

[1] tzw. fokus od ang. Focus Group Interview FGI

cd. załącznik 1

Etnografia Rozległy obszar wiedzy o kontekstualizacji, badania prowadzone są na podstawie zapisywania wyników bezpośrednich obserwacji. Technika uznawana jako niekompleksowa, zazwyczaj łączona z innymi technikami np. wywiadem, prototypowaniem lub analizą przypadków użycia. Opisano wiele rozszerzających strategii, takich jak np. obserwacji (jawnej, niejawnej, uczestniczącej, nieuczestniczącej, shadowing) i fotografii socjologicznej, fotografii dnia roboczego, obserwacje migawkowe, grupy fokusowe z elementami badania etnograficznego – wywiady grupowe, na które respondenci przynoszą przedmioty, które stają się bodźcem do dyskusji.	• pozwala na weryfikację faktów oraz założeń • umożliwia precyzyjne pomiary • pomaga odkrywać niejawne oddziaływania oraz wymagania (wiedza ukryta) • ułatwia analitykom pracę nad wymaganiami (wizualizowanie, zwiększenie zainteresowania, zrozumienia, procesy pamięci) • zwykle ponosimy relatywnie niewielkie koszty	• ludzie mogą świadomie lub nieświadomie realizować inaczej swoje działania, gdy są obserwowani • zaobserwowane przypadki mogą nie być typowe • obserwując złożone obszerne procesy można wyciągnąć mylne, niekompletne wnioski • w pewnych przypadkach niemożliwa lub niepraktyczna
Warsztat wymagań Doświadczenia praktyki wskazują na potrzebę zorganizowania warsztatu wymagań, podczas którego kluczowi interesariusze spotykają się „pod przywództwem" moderatora, aby intensywnie pracując, z wykorzystaniem technik wspomagania kreatywności, w ciągu 1-2 dni wypracować konsensus będący podstawą definicji wymagań. W wyniku warsztatów powstają ściśle zawczasu określone rezultaty.	• pomaga w budowaniu zespołu, którego członkowie zobowiązali się dążyć do sukcesu projektu • wszyscy interesariusze mają możliwość wypowiedzenia się, kształtuje się polityka realizacji projektu, rozwija się zrozumienie wspólnie przyjętej definicji systemu oraz modeli wymagań, zwiększa się innowacyjność projektu	• czynniki krytyczne związane z przygotowaniem warsztatów oraz oddziaływaniem osoby prowadzącej • dwa wątki osób reprezentujących stronę biznesową oraz osób tworzących oprogramowanie • niezbyt przydatne w przypadku małych projektów o niewielkim ryzyku
Przypadki użycia Koncepcję przypadków użycia stworzył Ivar Jacobson w latach 80. Podstawą konstrukcji jest scenariusz interakcji, a ponadto identyfikator, nazwa, specyfikacja aktorów, wyjątków i rozszerzeń – w wersji podstawowej. W latach 90. przypadki użycia zostały włączone do języka UML oraz uzyskały atrakcyjną prostą reprezentację graficzną. Cieszą się w dalszym ciągu wielkim zainteresowaniem (są używane w ponad 50% projektów) oraz rozwijają się w nawiązaniu do rozwoju koncepcji user stories oraz orientacji aspektowej, o czym Ivar Jacobson pisze na swojej stronie internetowej.	• możliwy jest prosty opis wymagań, nawet dla dużych systemów, zróżnicowani interesariusze mogą uczestniczyć oraz wnosić własny wkład • dobrze wkomponowują się w proces rozwoju systemu, stopniowo są uszczegóławiane oraz doskonalone, szczególnie dobrze integrują się z testowaniem oraz szacowaniem i śledzeniem prac	• z przypadkami użycia wiąże się wiele mitycznych opinii, które utrudniają ich stosowanie, takich jak: że są techniką dekompozycji, że są odpowiednią i wystarczającą techniką do opisu wszystkich wymagań, że służą tylko do analizy wymagań lub że ta analiza musi być nadmiernie pracochłonna

cd. załącznik 1

Prototyp Podkreślmy różnorodność, szeroki zakres badań i doświadczeń nad wykorzystaniem wstępnych wersji, pierwowzorów przyszłego rozwiązania. Kontinuum prototypów rozważa się w czterech wymiarach: zakres, funkcjonalność, interakcyjność, dizajn. Prototypy o niewielkim zaawansowaniu, określane terminem low-fidelity, obrazują koncepcję, alternatywne rozwiązania, układ ekranu. Ważne znaczenie mają w tej grupie tzw. papierowe prototypy tworzone przy pomocy szkiców odręcznych, naklejek, oprogramowania wspomagającego. Z kolei prototypy high-fidelity są zazwyczaj w pełni interaktywne, stanowią funkcjonalności produktu podstawowego, często są rozwijane z wykorzystaniem wyspecjalizowanych narzędzi (Smalltalk, Visual Basic, wzorce, SharePoint). Podział na prototypy horyzontalne oraz wertykalne jest z kolei związany z układem dwuwymiarowym (zakres, funkcjonalność). Szersza perspektywa podejmuje problem interakcji wzajemnych oraz szczegóły dizajnu.	• dostarcza konkret, wokół którego ogniskuje się dialog interesariuszy • pozwala na koncentrowanie się na wybranych aspektach wymagań • zwiększa zaangażowanie użytkownika, przeciwdziała efektom ubocznym utrudniającym wywiad, w szczególności papierowe prototypy pozwalają na aktywne doskonalenie założeń przez użytkownika • szczególnie odpowiedni w przypadku tzw. „trudnego użytkownika", który nie jest w stanie formułować precyzyjnych, abstrakcyjnych założeń • prototypy low-fidelity mogą być rozwijane przy niewielkich kosztach	• zwiększa wymagania wobec projektantów, w szczególności działania muszą być efektywne (trudne bo informatycy z reguły są perfekcjonalistami), narzuca reżim krótkich wydań, znajomości różnorodnych narzędzi, umiejętności komunikacyjne • często spotykamy efekt „to mi już wystarcza", związany z inspirowaną zwykle przez użytkownika chęcią poprzestania na aktualnym rozwiązaniu, bez zrozumienia, że prototyp nie spełnia wymagań np. skalowlności i rozwojowości • przy stosowaniu prototypów high-fidelity dochodzą nowe wymiary ryzyka
Modelowanie biznesowe Mimo swojej powszechności pogląd, że możliwe jest wykorzystanie technologii do wyeliminowania niedoborów organizacji nie kwestionując struktury organizacji jako całości, okazuje się zazwyczaj fałszywy. Jest to technika symetryczna do prototypowania, w tym sensie, że wspiera zwiększenie zrozumienia i aktywności przez analityków współpracujących z użytkownikami.	• zwiększa się zrozumienie problematyki, problemów oraz potrzeb interesariuszy • uproszczone, wyeliminowane lub uzasadnione zostają procedury co do których mogłyby powstać wątpliwości • wyłaniają się liderzy chętni i zdolni do współpracy w kolejnych etapach	• słabo rozwinięte są podstawy metodyczne w zakresie integrowania modelu organizacyjnego z aktywnościami inżynierii wymagań • rozległa reinżynieria poprzedzająca prace projektowe jest złożonym przedsięwzięciem opatrzonym znacznym ryzykiem

Załącznik 2

Dobre praktyki inżynierii wymagań z podziałem na obszary oraz poziomy

Obszary dobrych praktyk	Podstawowy (36)	Pośredni (21)	Zaawansowany (9)
1	2	3	4
I. Dokument wymagań	• zdefiniuj strukturę standardowego dokumentu specyfikacji wymagań • wyjaśnij jak korzystać z dokumentu • dołącz streszczenie wymagań • utwórz uzasadnienie biznesowe • zdefiniuj specjalistyczne terminy • zadbaj o czytelność dokumentu • pomóż czytelnikowi znaleźć informacje • uczyń dokument łatwym do zmiany	–	–
II. Kreowanie wymagań	• oceń wykonalność systemu • bądź wrażliwy na kwestie organizacyjne i polityczne • zidentyfikuj i konsultuj się z udziałowcami • utrzymuj źródła wymagań • zdefiniuj środowisko operacyjne • używaj listy kontrolne przy odkrywaniu wymagań	• wyszukaj ograniczenia dziedzinowe • utrzymuj uzasadnienia wymagań • wyszukując wymagania patrz z różnych punktów widzenia • prototypuj słabo zrozumiałe wymagania • wykorzystuj scenariusze przy odkrywaniu wymagań • definiuj procesy operacyjne	• wielokrotnie używaj te same wymagania

cd. załącznik 2

1	2	3	4
III. Analiza i negocjowanie wymagań	• zdefiniuj granice systemu • używaj listy kontrolne do analizy wymagań • zaopatrz się w oprogramowanie do wspomagania negocjacji • zaplanuj postępowanie w przypadku konfliktów i ich rozwiązywanie • nadaj priorytety wymaganiom	• klasyfikuj wymagania wykorzystując wielowymiarowe podejście • użyj macierzy interakcji, aby wykryć konflikty i nakładające się wymagania	• przypisz ryzyko do wymagań
IV. Opis wymagań	• zdefiniuj standardowe szablony do opisu wymagań • używaj prostego i ścisłego języka • adekwatnie używaj diagramów • uzupełnij opis w języku naturalnym innymi sposobami opisu	• specyfikuj wymagania ilościowo	–
V. Modelowanie systemu	• utwórz komplementarne modele systemu • modeluj otoczenie systemu • modeluj architekturę systemu	• używaj metod strukturalnych do modelowania systemu • używaj słownika danych • dokumentuj powiązania pomiędzy interesariuszami a modelami systemu	–
VI. Walidacja wymagań	• sprawdź zgodność dokumentu wymagań z twoimi standardami • zorganizuj formalne inspekcje wymagań • wykorzystuj interdyscyplinarne zespoły do przeglądu wymagań • zdefiniuj listy kontrolne dla walidacji wymagań	• użyj prototypowania, aby animować wymagania • napisz szkic podręcznika użytkownika • zaproponuj przypadki testowe dla wymagań	• stosowanie notacji formalnych • parafrazuj modele systemu

cd. załącznik 2

1	2	3	4
VII. Zarządzanie wymaganiami	• jednoznacznie identyfikuj wymagania • zdefiniuj politykę zarządzania wymaganiami • zdefiniuj politykę śledzenia • utrzymuj przewodnik śledzenia	• użyj bazy danych do zarządzania wymaganiami • zdefiniuj zarządzanie zmianą • zidentyfikuj globalne wymagania systemowe	• identyfikuj wymagania ulotne • przechowuj odrzucone wymagania
VIII. Inżynieria wymagań dla systemów krytycznych	• utwórz listę kontrolną dla wymagań bezpieczeństwa • zaangażuj zewnętrznych recenzentów w procesie walidacji	• identyfikuj i analizuj zagrożenia • z analizy zagrożeń wyprowadź wymogi bezpieczeństwa • sprawdź powtórnie wymagania operacyjne i funkcjonalne wobec wymagań bezpieczeństwa	• specyfikuj system przy użyciu metod formalnych • zbieraj opisy incydentów • wyciągaj wnioski z opisów incydentów • dbaj o rozwój kultury bezpieczeństwa w organizacji

Źródło: Na podst. [SoSa97].

Załącznik 3

Przypomnienia pryncypiów podejścia zorientowanego na użytkownika

I. Projekt opiera się na dogłębnym zrozumieniu użytkowników, zadań oraz otoczenia. Powinni zostać zidentyfikowani interesariusze związani z projektem, uwzględniając tych, na których system będzie oddziaływał (bezpośrednio lub pośrednio). Charakterystyki użytkowników, zadań oraz otoczenia są nazywane **kontekstem wykorzystania**[1]. Kontekst wykorzystania jest głównym źródłem informacji dla ustalenia wymagań, dostarcza zasadniczy wkład w procesie projektowania. W komentarzach do tego pryncypium często wskazuje się na trójkę {użytkownik, system, otoczenie}, którą trzeba rozpoznać w trakcie prac projektowych.

II. Użytkownicy biorą udział w całym procesie projektowania i rozwoju. Zaangażowanie użytkowników w projektowanie pozwala na pozyskanie wartościowej, źródłowej wiedzy o kontekście wykorzystania, o zadaniach, a także jak użytkownicy mogą pracować w przyszłości z produktem, systemem lub usługą. Powinno ono być aktywne niezależnie czy użytkownik uczestniczy w projektowaniu, czy jest źródłem istotnych danych lub czy ocenia rozwiązania. Poprzez zaangażowanych w całym procesie projektowania i rozwoju użytkowników organizacja zamawiająca system ma możliwość oddziaływania na rozwiązanie. Takie zaangażowanie oraz współudział może mieć wpływ na poziom akceptacji oraz późniejszego wykorzystania.

III. Projekt jest prowadzony i udoskonalany z wykorzystaniem ocen zorientowanych na użytkownika. Oceny projektów oraz ich poprawianie na podstawie otrzymanego od użytkowników sprzężenia zwrotnego pozwalają minimalizować ryzyko, polegające na tym, że wykonany system nie będzie zaspakajał potrzeb użytkownika lub potrzeb organizacji (włączając wymagania ukryte lub trudne do jawnego wyspecyfikowania). Wstępne rozwiązania są weryfikowane w ramach zbliżonych do sytuacji rzeczywistych scenariuszy, wyniki są podstawą progresywnie poprawianych rozwiązań. Zorientowane na użytkownika ocenianie powinno być również wykorzystane w ramach testowania, końcowej oceny projektu, a także na etapie eksploatacji, dając podstawę dla przyszłych projektów.

---

[1] Ang. *context of use.*

IV. **Proces jest iteracyjny**. Bez iteracji nie można zazwyczaj osiągnąć najbardziej odpowiedniego projektu dla systemu interaktywnego. Wiele potrzeb oraz oczekiwań u użytkowników oraz u innych interesariuszy, pojawia się jedynie w czasie rozwoju, gdy projektanci doskonalą swoje zrozumienie użytkowników i ich zadań, gdy w obliczu potencjalnych rozwiązań u użytkowników doskonali się zdolność wyrażania potrzeb. W celu minimalizacji ryzyka, że rozwijany system nie spełni oczekiwań użytkownika, gdy nowe informacje są otrzymywane, opisy, specyfikacje i prototypy są zmieniane oraz poprawiane. Tak więc iteracje są wykorzystywane do stopniowej eliminacji niepewności oraz ryzyka. W humorystycznych komentarzach do tego pryncypium wskazuje się, że aby dowiedzieć się czego użytkownik potrzebuje można mu na początek dać to czego zapewne nie potrzebuje (nasz pierwszy projekt) oraz obserwować jego reakcję.

V. **Projekt odnosi się do całego doświadczenia użytkownika**. Kontekst użyteczności rozpatrujemy szerzej niż jedynie łatwość użycia produktu. Patrząc z punktu widzenia osobistych celów użytkownika, może on obejmować zarówno percepcyjne oraz emocjonalne aspekty zazwyczaj połączone z doświadczeniami użytkownika, jak również zagadnienia, takie jak satysfakcja z pracy oraz eliminowanie monotonii. Mocne strony użytkowników, ich ograniczenia, preferencje i oczekiwania powinny być brane pod uwagę podczas określania, jakie działania są przeprowadzane przez użytkowników i które funkcje są realizowane przez technologię. Tego pryncypium w ISO 13407 nie było. Dołączono je zapewne, dla podkreślenia, konieczności jeszcze bardziej szczegółowego analizowania kontekstu użytkownika.

VI. **Zespół projektowy wykorzystuje umiejętności oraz spojrzenie interdyscyplinarne**. Zorientowany na użytkownika zespół projektowy nie musi być liczny, ale powinien być wystarczająco różnorodny, aby w wymaganym czasie wypracowywać kompromisowe decyzje dotyczące projektu oraz jego wdrożenia. W czasie współpracy w zespołach, których członkowie mają rozległy zestaw umiejętności wyzwala się dodatkowa kreatywność oraz pomysły, które są wykorzystywane w projekcie. Dodatkowa korzyść z interdyscyplinarności oraz ze spojrzenia z różnych perspektyw[2] polega na tym, że członkowie zespołu stają się bardziej świadomi ograniczeń oraz realiów innych dyscyplin; dla przykładu eksperci techniczni stają się bardziej wyczuleni na problemy użytkowników, zaś użytkownicy mogą zyskać większą świadomość ograniczeń technicznych.

[2] Ang. *Multi-perspective approach*.

ANALYSIS OF SOME CONCEPTS IN THE FIELD OF DESIGN REQUIREMENTS

Summary

On the eve of the new millennium, the design requirements continue to make significant difficulties to designers. This article aims to analyze three research trends: software engineering, user-oriented approach and agile approach. The paper is to identify, describe, and to discuss mechanisms selected from among those trends.