Magazine

Strategie testowania i jakości Agile: dyscyplina ponad retoryką

Cześć II z IV: Strategie wymagań Agile

Autor: Scott Ambler

O autorze: W lipcu 2006 r. Scott dołączył do zespołu IBM w Kanadzie jako Główny Metodolog Agile i Lean dla IBM Rational. Scott pracuje dla klientów IBM, w szczególności w obszarach coachingu i usprawniania procesów wytwórczych w IT. Scott pracuje w przemyśle IT od połowy lat 80-tych, a z technologią

obiektową od wczesnych lat 90-tych. Napisał kilka książek i *white papers* traktujących o obiektowym wytwarzaniu oprogramowania, procesie oprogramowania, Modelu Skalowania Agile (ASM), Wytwarzaniu Agile Kierowanym Modelem (AMDD), Technikach Baz Danych Agile, Agile UP (ang. *Unified Process*). Przemawiał na wielu konferencjach na całym świecie.

Kontakt: **Scott_Ambler@ca.ibm.com** twitter: **http://twitter.com/scottwambler**

Intermediate

Level

3

Magazine Number

Inżynieria oprogramowania

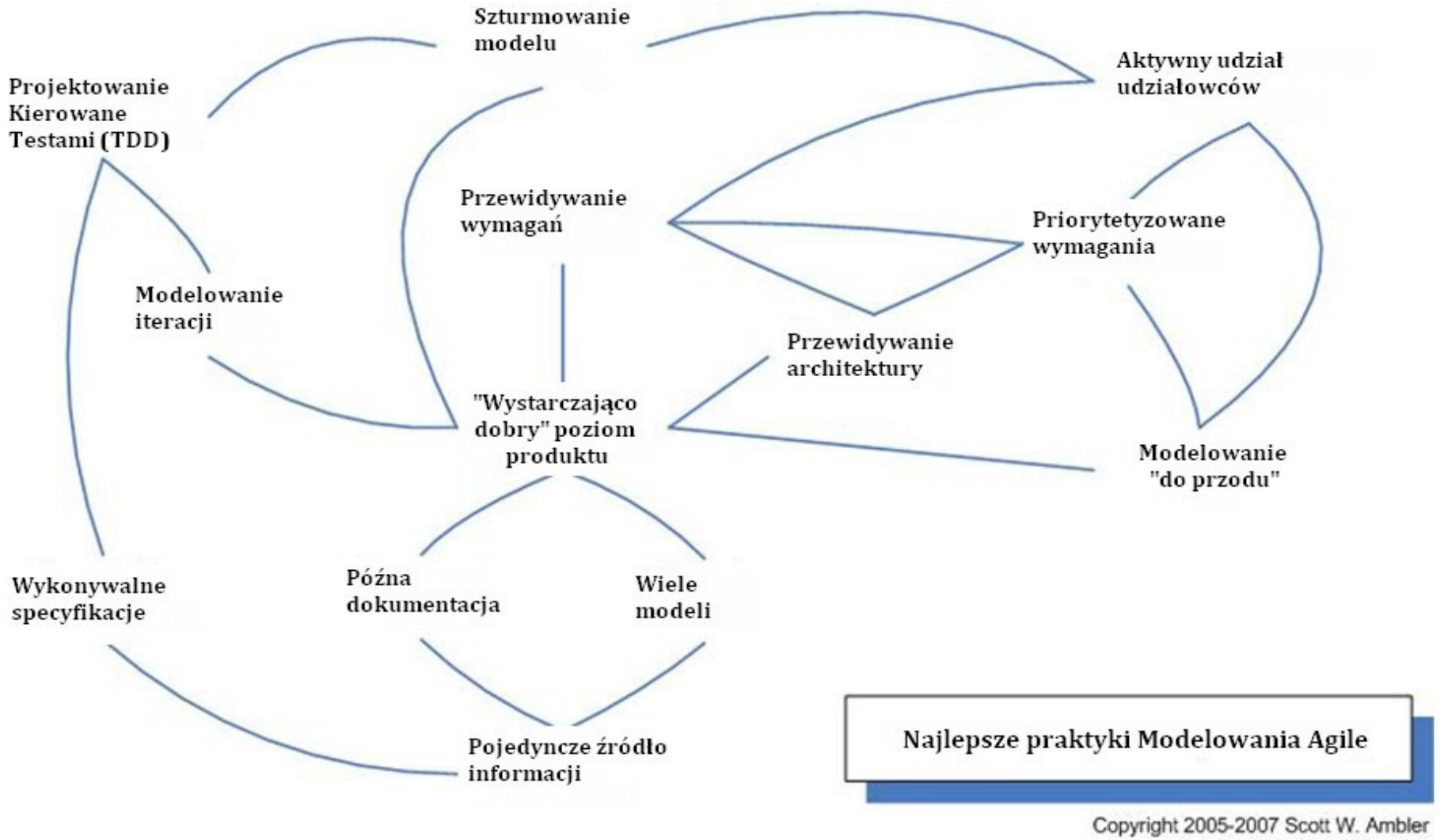
Section in the magazine

Wprowadzenie

Niniejsza sekcja stanowi przegląd podejść Agile do pozyskiwania i zarządzania wymaganiami. Jest to o tyle ważne, ponieważ twoje podejście do wymagań idzie ramię w ramię z twoim podejściem do walidowania tych wymagań, tym samym aby zrozumieć, jak zdyscyplinowane zespoły Agile podchodzą do testowania i jakości, musisz najpierw zrozumieć, jak zespoły Agile podchodzą do wymagań. Rysunek 5 obrazuje mapę procesu najlepszych praktyk Modelowania Agile (ang. *Agile Modeling (AM)*), które adresują zwinne strategie modelowania oraz dokumentacji i w przypadku TDD (wytwarzania kierowanego testami) oraz wykonywalnych specyfikacji, niewątpliwie wpadają w testowanie. Sekcja podzielona jest na następujące tematy:

Niniejszy artykuł dzieli się na następujące tematy:

- Aktywny udział udziałowców
- Zarządzanie wymaganiami funkcjonalnymi
- Przewidywanie wstępnych wymagań
- Modelowanie iteracyjne
- Model szturmowy “Dostarczenia na czas” (ang. *Just in Time (JIT)*)
- Zarządzanie wymaganiami niefunkcjonalnymi
- Kto to robi?
- Implikacje dla testowania



Rysunek 5 Najlepsze praktyki Modelowania Agile

1.1. Aktywny udział udziałowców

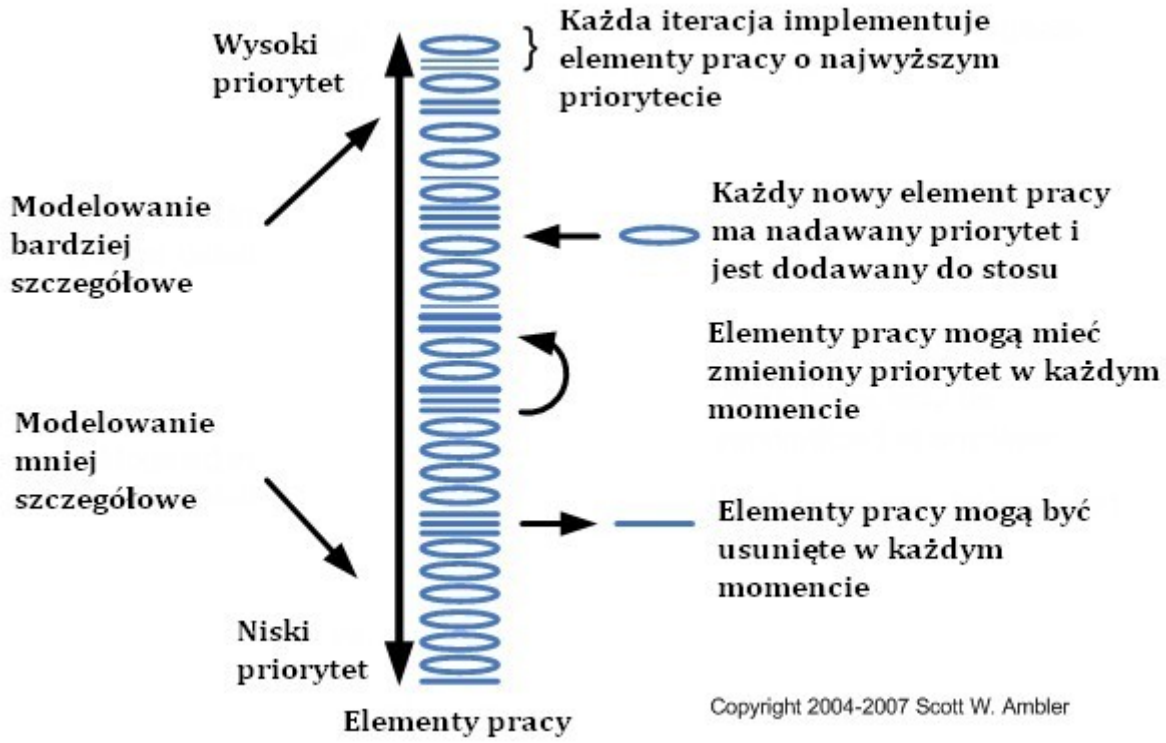
Praktyka Modelowania Agile o nazwie „Aktywny udział udziałowców" mówi, że udziałowcy powinni punktualnie dostarczać informacje, punktualnie podejmować decyzje i być aktywnie zaangażowani w proces wytwarzania poprzez wykorzystanie narzędzi i technik. Kiedy udziałowcy pracują blisko z developmentem, zwiększa to szanse powodzenia projektu poprzez zwiększanie:

- Szansy, że developerzy zrozumieją rzeczywiste potrzeby udziałowców
- Zdolności udziałowców do kierowania (sterowania) projektem poprzez rozwijanie ich wymagań w oparciu o obserwowanie działającego oprogramowania, które wytwarza zespół
- Jakości tego, co jest wytwarzane poprzez aktywne zaangażowanie w testy akceptacyjne podczas cyklu życia

Tradycyjne podejście polegające na zapewnieniu udziału udziałowców w realizowanej wcześnie w projekcie fazie pozyskiwania wymagań, a następnie ich odejściu aż do końcowej fazy projektu i testów akceptacyjnych na końcu cyklu życia, okazuje się w praktyce bardzo ryzykowne. Ludzie nie są za dobrzy w definiowaniu swoich wymagań od początku – w wyniku seryjnego podejścia do wytwarzania, inwestuje się znaczące nakłady w budowanie i testowanie oprogramowania, które nie jest użyte nawet raz, do momentu gdy system jest już na produkcji. Aby uniknąć takich problemów, praktycy Agile wolą podejście ewolucyjne, w którym udziałowcy są aktywnie zaangażowani – podejście, które okazało się bardziej efektywne w dostarczaniu oprogramowania, niż ludzie naprawdę chcieli.

1.2. Zarządzanie wymaganiami funkcjonalnymi

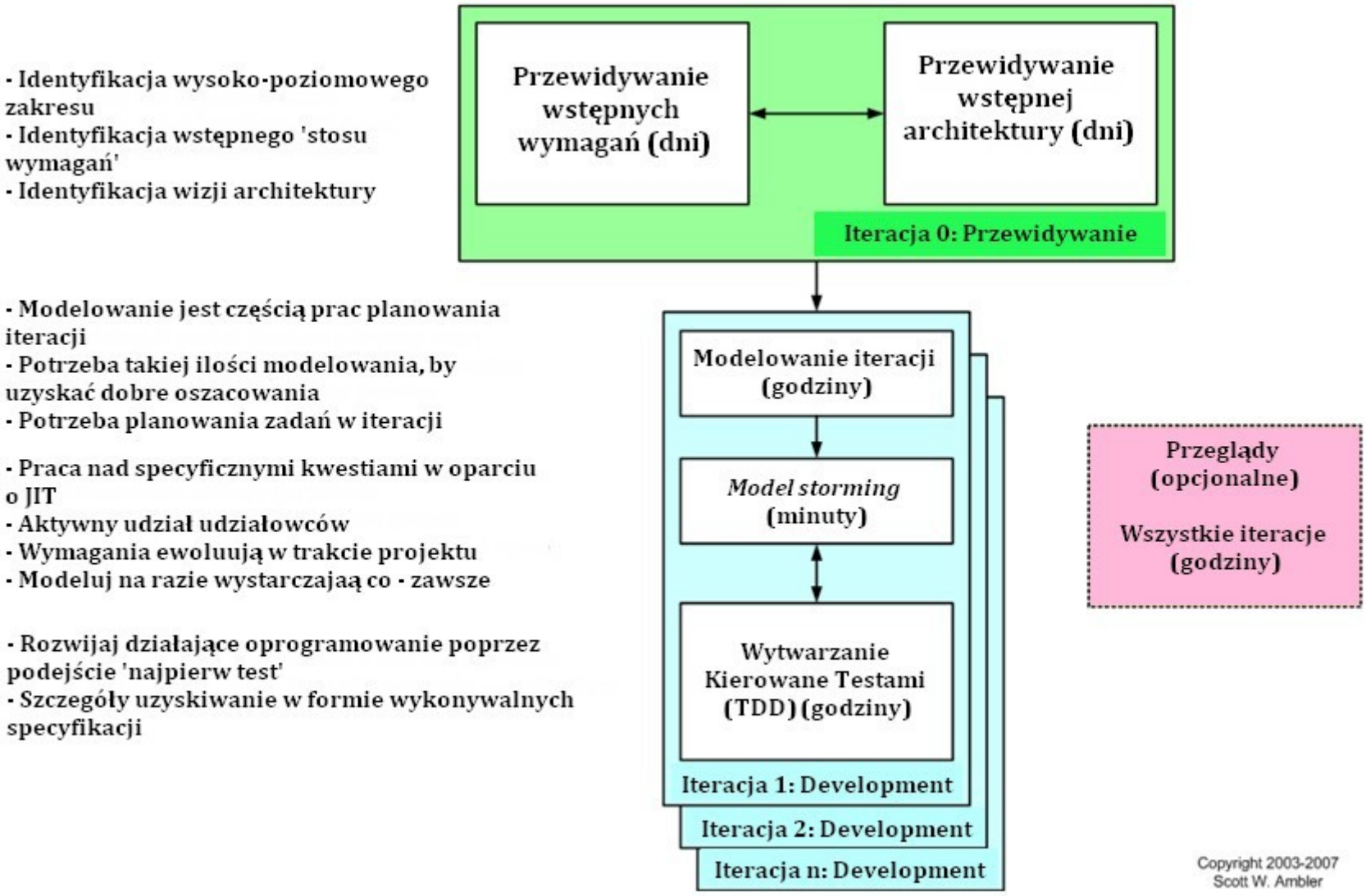
Podstawową praktyką Agile jest Spriorytetyzowany Stos Wymagań (ang. *Prioritized Requirements Stack*), w terminologii Scrum znany jako Product Backlog. Podstawowe zasady, pokazane na Rysunku 6, mówią, że powinieneś implementować wymagania w porządku określonym priorytetami i pozwolić swoim udziałowcom zmieniać wymagania w trakcie projektu, w miarę, jak się uczą. Rysunek prezentuje również kilka zaawansowanych koncepcji Agile. Po pierwsze, to naprawdę stos elementów pracy, a nie tylko wymagania funkcjonalne (na tym stosie pojawiają się również raporty defektów, jak widać na Rysunku 2 – więcej o tym później; musisz też zaplanować prace takie jak przegląd artefaktów od innych zespołów, urlopy). Po drugie, by zmniejszyć ryzyko związane ze złożonymi składnikami pracy, nie wszystkie składniki są tworzone równo z innymi; musisz rozważyć modelowanie z wyprzedzeniem, niezależnie od tego czy złożony element pracy znajduje się w najbliższej, czy w następującej po niej iteracji.



Rysunek 6 Proces zarządzania zmianami wymagań Agile

1.3. Przewidywanie wstępnych wymagań

Rysunek 7 obrazuje cykl życia AMDD (ang. *Agile Model Driven Development*). Jak można zobaczyć na Rysunku 7, podczas Iteracji 0 praktycy Agile będą wykonywać ze swoimi udziałowcami wstępne modelowanie wymagań, aby zidentyfikować początkowe, wysoko-poziomowe, wymagania dla systemu. Celem przewidywania wstępnych wymagań jest wykonanie modelowania wystarczającego do określenia zakresu systemu i stworzenia wstępnego stosu wymagań, który formuje podstawę dla twojej spriorytetyzowanej listy elementów pracy (to nie pojawia się magicznie samo pewnego dnia). Celem nie jest stworzenie szczegółowej specyfikacji wymagań, ponieważ w rzeczywistości taka strategia zwiększa ryzyko projektowe.



Rysunek 7 Cykl życia Agile Model Driven Development (AMDD)

W zależności od kwestii logistycznych (może być trudno zebrać wszystkich właściwych ludzi dokładnie w tym samym czasie) i zdolności twojej organizacji do podejmowania decyzji w rozsądnym czasie, Iteracja 0 może trwać przez okres od kilku dni do kilku miesięcy. Jednakże twoja praca w modelowaniu wstępnych wymagań powinna zabrać tylko kilka dni z tego okresu. Zauważ również, że w Iteracji 0 jest nieco więcej, niż wstępne modelowanie – cykl życia AMDD na Rysunku 7 obrazuje tylko czynności modelowania. Ważną czynnością podczas Iteracji 0 jest zdobywanie wstępnego wsparcia i budżetu dla projektu, czegoś, co wymaga zrozumienia wstępnego zakresu. Możesz już mieć zapewnione inicjalne wsparcie w wyniku czynności planowania przed-projektowego (część zarządzania portfolio), ale realistycznie – w pewnym momencie ktoś zapyta, co mamy zamiar osiągnąć, ile to będzie kosztować i jak długo ma to trwać. Jeśli masz zamiar uzyskać pozwolenie na pracę nad projektem, musisz być w stanie udzielić rozsądnych, a mimo to potencjalnie zmieniających się, odpowiedzi na te pytania. W wielu organizacjach możesz musieć zrobić krok naprzód i ocenić swój projekt poprzez studium wykonalności (ang. *feasibility study*).

1.4. Modelowanie iteracji

Jak widać na Rysunku 6, zespół Agile będzie implementował wymagania w porządku określonym priorytetami, poprzez zdejmowanie elementów pracy w danej iteracji ze stosu. Aby zrobić to skutecznie, musisz umieć odpowiednio oszacować pracę wymaganą dla każdego wymagania, następnie bazując na prędkości twojej poprzedniej iteracji (miara tego, ile pracy zrealizowałeś) zdejmujesz taką ilość pracy ze stosu. Dla przykładu – jeśli w ostatniej iteracji zrealizowałeś pracę o wartości 15 punktów, wtedy zakładamy, że w tej iteracji będziesz w stanie wykonać rzeczy o takiej właśnie wartości. To implikuje, że na początku każdej iteracji konstrukcyjnej zespół Agile musi oszacować i zaplanować pracę, którą będą wykonywać w danej iteracji. Aby odpowiednio oszacować każde wymaganie, musisz zrozumieć pracę wymaganą do jego implementacji – i tu wkracza modelowanie. Dyskutujesz, w jaki sposób zamierzasz zaimplementować każde wymaganie, modelując tam, gdzie właściwe jest eksplorowanie lub komunikujesz pomysły. To modelowanie jest w efekcie analizą i projektem wymagań, które będą implementowane w danej iteracji. Moje doświadczenie mówi, że dwutygodniowa iteracja będzie miała w przybliżeniu pół dnia planowania iteracji, włączając modelowanie, podczas gdy dla iteracji 4-tygodniowej to zadanie zwykle zabiera dzień. Celem jest odpowiednie zaplanowanie pracy dla iteracji, identyfikacja elementów pracy o najwyższym priorytecie, które będą adresowane i w jaki sposób. Innymi słowy, myślenie o rzeczach krótkoterminowo. Celem nie jest stworzenie wszechstronnego wykresu Gantta, czy szczegółowej specyfikacji pracy, która ma być wykonana. Koniec końców - to z tej techniki wynika wymagana czynność modelowania – często zaniedbywany aspekt ”planning pokera” według Mike’a Cohn’a.

1.5. Model szturmowy “Dostarczenia na czas” (ang. *Just in Time (JIT)*)

Szczegóły wymagań modelowane są w oparciu o podejście *just in time,* w sesjach szturmowania modelu (ang. *model storming*) podczas iteracji developerskich. *Model storming* to modelowanie dokładnie na czas: określasz kwestię, którą chcesz rozwiązać, szybko zgarniasz kilku kolegów z zespołu,

którzy mogą ci pomóc, grupa bada zagadnienie, a następnie każdy wraca do swoich zajęć. Jednym z powodów, dla których modelujesz „szturmowo” jest przeanalizowanie szczegółów wymagania. Dla przykładu – możesz implementować *user story*, które wskazuje, że system, który budujesz musi umożliwiać edycję informacji o studencie. Wyzwaniem jest to, że *user story* nie zawiera żadnych szczegółów na temat tego, jak powinien wyglądać ekran – w świecie Agile lubimy mówić, że *user story* są „przypomnieniem, by odbyć rozmowę z udziałowcami” – co innymi słowy mówi, by wykonać nieco szczegółowego modelowania wymagań. Tak więc, by zebrać szczegóły, wzywasz właściciela produktu (*Product Owner*) i wspólnie tworzycie zarys wyglądu ekranu, rysując kilka przykładów zanim nie dojdziecie do wspólnego porozumienia co do tego, co ma być zbudowane. Innymi słowy – modelujesz szturmowo szczegóły.

1.6. Wymagania niefunkcjonalne

Wymagania niefunkcjonalne, znane też jako „wymagania techniczne” lub wymagania „jakości usługi” (ang. *quality of service (QoS)*) skupiają się na aspektach zwykle krzyżujących się z wymaganiami funkcjonalnymi. Popularne wymagania niefunkcjonalne obejmują dokładność, dostępność, równoległość, konsumpcyjność/użyteczność, zagadnienia środowiskowe, internacjonalizację, kwestie operacyjne, wydajność, zagadnienia prawne, wiarygodność, bezpieczeństwo, możliwości serwisowe i dotyczące wsparcia. Ograniczenia, które dla ułatwienia ujmę w wymaganiach niefunkcjonalnych, określają ograniczenia twojego rozwiązania, takie jak: wymaganie by gromadzić wszystkie dane firmowe w DB2 dla twojej architektury korporacyjnej czy umożliwienie używania tylko oprogramowania Open Source (OSS), co przekłada się na określony poziom licencji OSS.

Ograniczenia mogą często wpływać na twoje wybory techniczne, poprzez ograniczanie specyficznych aspektów architektury, definiowanie sugerowanych możliwości ponownego użycia, a nawet punktów dostosowania architektury. Mimo, że wielu developerów udaje się to okiełznać, rzeczywistość jest taka, że ograniczenia często powodują, iż dla twojego zespołu rzeczy stają się o wiele prostsze, ponieważ niektóre techniczne decyzje zostały już podjęte za ciebie. Lubię myśleć o tym w taki sposób – praktycy Agile będą mieli odwagę podejmować jutrzejsze decyzje jutro, zdyscyplinowani praktycy Agile z pokorą respektują również wczorajsze decyzje.

Mimo, że zespoły Agile dość dobrze odkryły, jak efektywnie adresować wymagania funkcjonalne, większość ciągle walczy z wymaganiami niefunkcjonalnymi. Niektóre zespoły tworzą techniczne *story*, by zarządzać wymaganiami niefunkcjonalnymi w tak prosty sposób, jak zarządzają wymaganiami funkcjonalnymi za pomocą *user story.* Jest to doskonałe dla celów dokumentacyjnych, ale szybko się rozpada z punktu widzenia zarządzania i implementacji. Opisane wcześniej strategie zarządzania wymaganiami zakładają, że wymagania są „samo-zawierające się” i mogą być adresowane w skończonym okresie czasu; założenie, które nie zawsze jest prawdziwe dla wymagań niefunkcjonalnych.

Istnieją cztery podstawowe strategie, z których wszystkie powinny być zastosowane, do adresowania wymagań niefunkcjonalnych w projekcie Agile:

Wstępne przewidywanie. To podczas przewidywania wstępnych wymagań, zidentyfikujesz wysokopoziomowe wymagania funkcjonalne i niefunkcjonalne. Wszystkie formy wymagań będą kierować pracami przewidywania architektury, które wykonywane są iteracyjnie i równolegle do przewidywania wymagań. Celem prac przewidywania wymagań jest określenie wymagań wysokopoziomowych, a celem prac przewidywania architektury jest zapewnienie, że twoja wizja architektur efektywnie adresuje te wymagania. W tym momencie nie musisz pisać szczegółowych specyfikacji, ale powinieneś upewnić się, że podążasz w dobrym kierunku.

Szturmowanie modelu JIT. Szturmowanie modelu „dokładnie na czas" podczas konstrukcyjnego cyklu życia celem eksplorowania szczegółów.

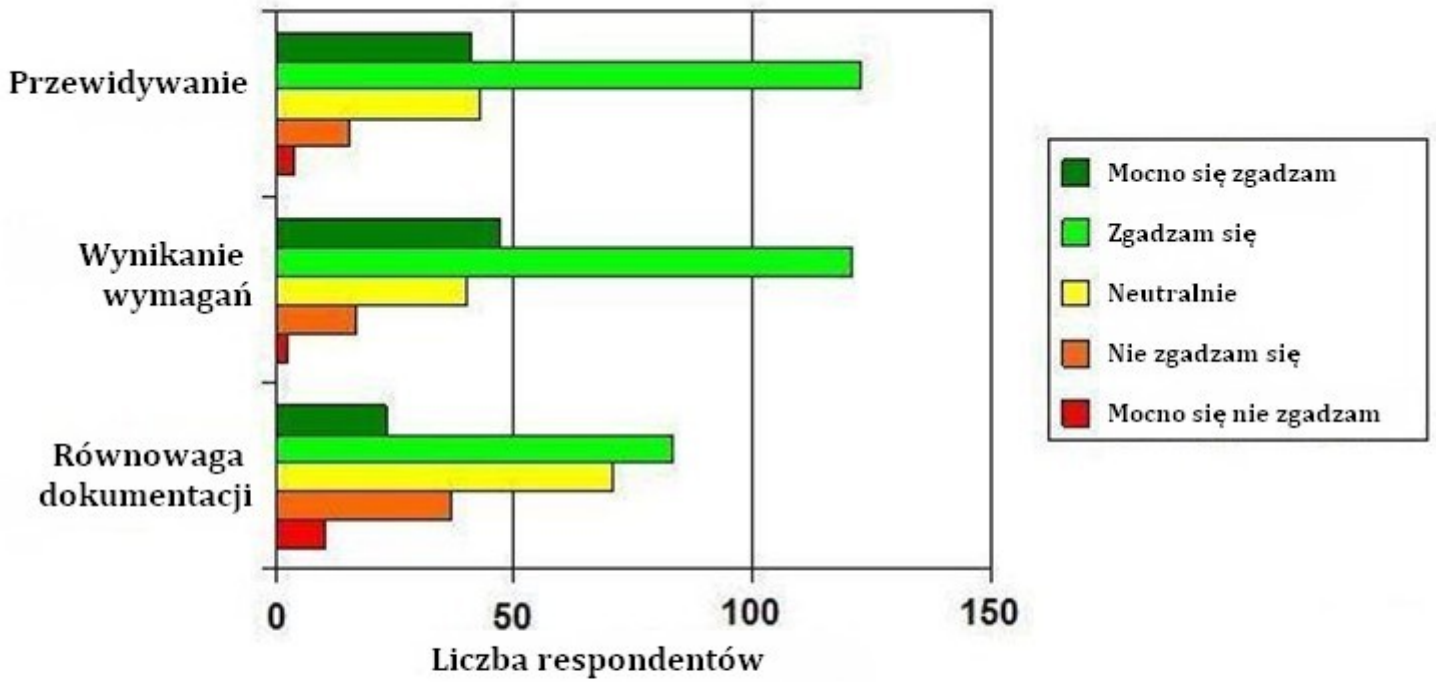
Niezależne testowanie równoległe. Jest wykonywane podczas całego cyklu życia, by upewnić się, że system odpowiednio adresuje wymagania niefunkcjonalne. Więcej o tym – w dalszej części.

Edukacja. Edukacja developerów w taki sposób, by rozumieli podstawy całego zakresu zagadnień architektonicznych opisanych w wymaganiach.

1.7. Kto to robi?

Rysunek 8 podsumowuje pewne wyniki z Badania Praktyk i Zasad Agile wykonanego w roku 2008 przez Ambysoft. Jak widać, dość popularne dla zespołów Agile jest wykonywanie z góry pewnego przewidywania wymagań oraz to, że szczegóły wymagań będą się wyłaniać z upływem czasu (poprzez modelowanie iteracyjne i szturmowanie). Na Rysunku 9 przedstawiono widok z punktu widzenia narzędzi, który podsumowuje pewne wyniki z Badania Wytwarzania Kierowanego Testami, wykonanego w roku 2008 przez Ambysoft. Mimo, że wokół akceptacji TDD istnieje sporo retoryki, faktem jest, że nie tylko nie zastąpiło ono technik modelowania wymagań Agile, ale nawet nie jest tak popularne. Implikacją jest to, że wymagania w zespołach Agile są odkrywane przez kilka technik – i słusznie, ponieważ pojedyncza strategia rzadko wystarcza dla sytuacji zachodzących w prawdziwym świecie.

Praktyki Wymagań w projektach Agile



Czy zgadzasz się ze stwierdzeniami:

1. **Wykonujemy pewne wstępne modelowanie wymagań (przewidywanie) na początku projektu dla celów określenia zakresu i planowania.**
2. **Szczegóły wymagań pojawiają się w projektach Agile z upływem czasu.**
3. **Nasze zespoły Agile rozumieją właściwą równowagę pomiędzy dokumentacją a innymi artefektami do dostarczenia**

Copyright 2008 Scott W. Ambler

Source: Ambysoft 2008 Agile Practices and Principles Survey

Rysunek 8 Praktyki związane z wymaganiami w projektach Agile

Rysunek 9 Praktyki pozyskiwania wymagań w projektach Agile

1.8. Implikacje dla testowania

Istnieje kilka ważnych implikacji, jakie dla testowania Agile mają strategie wymagań Agile:

Testowanie Agile musi być iteracyjne. Zwinne czynności związane z wymaganiami, czynności projektowe i konstrukcyjne są z natury iteracyjne. Tak samo musi być w przypadku czynności testowych.

Testerzy Agile nie mogą polegać na posiadaniu kompletnej specyfikacji. Jak widziałeś na Rysunkach 2 i 7, wymagania są określane, eksplorowane i implementowane w ciągu całego cyklu życia. Nie ma pojedynczej fazy wymagań, która produkuje wszechstronną specyfikację wymagań, dlatego też twoje strategie testowania nie mogą opierać się na posiadaniu dostępnej kompletnej specyfikacji.

Testerzy Agile muszą być elastyczni. Testerzy muszą być przygotowani do pracy najlepiej, jak potrafią, z informacją dostarczoną im na czas, z pełnym zrozumieniem, że informacje, na których opierają swoją pracę dziś, mogą zmienić się jutro.

Dobrą wiadomością jest to, że istnieją techniki testowania Agile, które odzwierciedlają te implikacje. Wyzwaniem jest to, że musisz chcieć je zastosować.

c.d.n.

W następnym numerze: Strategie testowania i jakości Agile: dyscyplina ponad retoryką. Cześć III z IV: Strategie testowania Agile.