PRACE NAUKOWE UNIWERSYTETU EKONOMICZNEGO WE WROCŁAWIU nr 82
Informatyka Ekonomiczna 14 2009

Cezary Hołub
Uniwersytet Ekonomiczny we Wrocławiu

METODYKA *AGILE* I NIEKTÓRE PRZYKŁADY WSPÓŁCZEŚNIE ROZWIJANYCH JEJ ODMIAN W WYTWARZANIU OPROGRAMOWANIA

Streszczenie: Obecnie istnieje wiele metod rozwoju oprogramowania, osoby zarządzające projektami jednak coraz częściej wybierają metodykę *agile*. W 2001 r. grupa ekspertów zorganizowała spotkanie, na którym zostało utworzone stowarzyszenie o nazwie Agile Alliance.

Głównym celem tego artykułu jest charakterystyka metodologii wytwarzania oprogramowania *agile* oraz opis kilku jej modyfikacji obecnie używanych w przemyśle IT. W artykule znajdziemy także opis jednej z najczęściej stosowanych „tradycyjnych" metod rozwoju oprogramowania, a mianowicie *waterfall*, wraz z krótką konfrontacją tych dwóch metod.

Słowa kluczowe: *agile*, *waterfall*, metodologia, programowanie, Scrum.

1. Wstęp

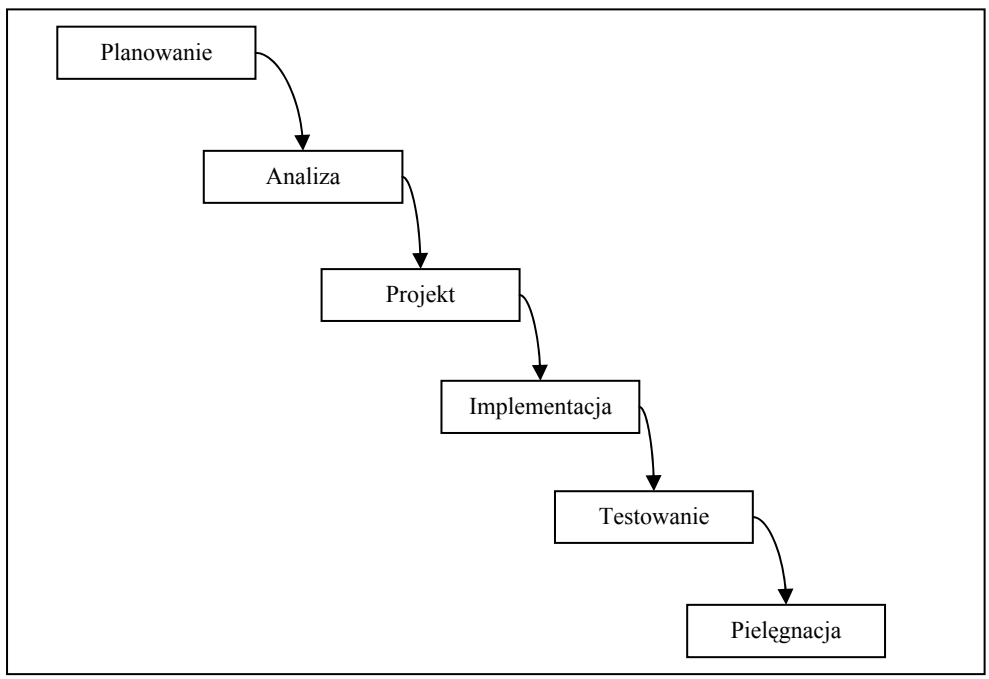
Obecnie istnieje wiele metodyk wytwarzania oprogramowania, jednak coraz częściej menedżerowie projektów wybierają metodykę zwinną (*agile*). W roku 2001 grupa ekspertów zaniepokojonych obserwowanymi zjawiskami (polegającymi m.in. na wpędzaniu się wielu korporacji w poważne kłopoty wskutek rozrastających się procesów) zorganizowała spotkanie, na którym powstało zrzeszenie nazwane Agile Alliance.

Celem artykułu jest charakterystyka metodyki zwinnej wytwarzania oprogramowania oraz krótki opis kilku jej odmian używanych obecnie w przemyśle. W artykule znajdziemy również opis jednej z najczęściej używanych metodyk „tradycyjnych", a mianowicie modelu kaskadowego (*waterfall*) wraz z krótką konfrontacją obu podejść.

Metodyka *agile* to zbiorcza nazwa technik wytwarzania oprogramowania kładących nacisk na bezpośrednią komunikację w zespole wytwórczym i realizowanie projektu iteracyjnie. Metody przeznaczone są głównie dla małych zespołów wytwórczych, w których istnieje możliwość łatwej komunikacji, dzięki czemu nie ma potrzeby tworzenia dużej ilości dokumentacji. Pozwala to na łatwiejsze zrozumienie zagadnienia oraz zminimalizowanie ryzyka w projektach o krótkim czasie wykonania, ale za to wymaga dobrze zgranego i stałego zespołu.

2. Metoda kaskadowa jako przykład dotychczasowego podejścia

Najczęściej stosowanym podejściem przy tworzeniu oprogramowania przed metodyką *agile* była metodyka kaskadowa [Internet 1]. Model kaskadowy *(waterfall model)* to jeden z kilku rodzajów procesów tworzenia oprogramowania zdefiniowany w inżynierii oprogramowania. Polega on na wykonywaniu podstawowych czynności jako odrębnych faz projektowych, w porządku jeden po drugim (rys. 1). Każda czynność to kolejny schodek (kaskada). Jeśli któraś z faz zwróci niesatysfakcjonujący produkt, cofamy się, wykonując kolejne iteracje aż do momentu, kiedy otrzymamy satysfakcjonujący produkt na końcu schodków.



Rys. 1. Model kaskadowy wytwarzania oprogramowania

Źródło: opracowanie własne na podstawie [Internet 1].

Model kaskadowy ma następujące wady:

- Nie można przejść do następnej fazy przed zakończeniem poprzedniej.
- Model ten ma bardzo nieelastyczny podział na kolejne fazy.
- Iteracje są bardzo kosztowne – powtarzamy wiele czynności.

Tego typu modelu należy używać wyłącznie w przypadku, gdy wymagania są zrozumiałe i przejrzyste, ponieważ każda iteracja jest czasochłonna i wymaga dużych wydatków na ulepszanie [Internet 1].

3. Istota metodyki *agile*

Jak wspomnieliśmy we wstępie, celem grupy Agile Alliance była popularyzacja wartości i reguł, które skłonią zespoły programistyczne do szybkiego i elastycznego wytwarzania oprogramowania. Przez kilka miesięcy grupie Agile Alliance udało się opracować najważniejsze założenia. Efektem tej pracy był następujący dokument:

The Manifesto of the Agile Alliance:
(Manifest zwinnego wytwarzania oprogramowania)

Próbujemy odkrywać lepsze sposoby wytwarzania oprogramowania, realizując własne eksperymenty w tym obszarze i zachęcając do podobnych działań innych programistów. Wskutek tych eksperymentów udało nam się sformułować następujące wskazówki:

Programiści i ich harmonijna współpraca są ważniejsi od procesów i narzędzi.
Działające oprogramowanie jest ważniejsze od wyczerpującej dokumentacji.
Faktyczna współpraca z klientem jest ważniejsza od negocjacji zasad kontraktu.
Reagowanie na zmiany jest ważniejsze od konsekwentnego realizowania planu.

Oznacza to, że chociaż rozwiązania wymienione po prawej stronie mają pewną wartość, z punktu widzenia skuteczności projektów dużo bardziej wartościowe są działania opisane z lewej strony [Martin, Micah 2008].

Wartości respektowane w metodach zwinnych są następujące:

- Osoby i interakcje są usytuowane ponad procesami i narzędziami.
- Działające oprogramowanie przedkłada się ponad dokumentację projektową.
- Współpraca z klientem jest ważniejsza od negocjowania kontraktu.
- Bieżące reagowanie na zmiany jest istotniejsze niż postępowanie według planu.

Poniżej przedstawiony zostanie zbiór dwunastu szczegółowych zasad (praktyk) metodyki *agile*. Właśnie te reguły odróżniają praktyki programowania zwinnego od

tradycyjnych (przykładem jest omówiony wcześniej model kaskadowy) „ciężkich procesów" [Martin, Micah 2008]:

1. Satysfakcja klienta jest najwyższym priorytetem.
2. Zmiany wymagań powinny być uwzględnione w procesie wytwarzania.
3. Działające oprogramowanie należy dostarczać klientowi możliwie często i kontynuować ten proces na wszystkich etapach realizacji projektu.
4. Warunkiem zwinności projektu jest częsta i intensywna współpraca klientów, programistów i ośrodków biznesowych zaangażowanych w jego finansowanie.
5. Projekty powinny być tworzone przez zmotywowanych autorów.
6. Bezpośrednia konwersacja staje się najwydajniejszą formą przekazywania informacji.
7. Podstawowym miernikiem postępu prac nad projektem jest ilość i jakość działającego oprogramowania spełniającego wymagania klienta.
8. Zaleca się promowanie zrównoważonego rozwoju. Zespół programistów nie powinien od razu próbować osiągnąć maksymalnej wydajności, której utrzymanie w dłuższym okresie byłoby niemożliwe.
9. Pozytywny wpływ na zwinność projektu ma stała dbałość o doskonałość techniczną i właściwy projekt.
10. Kluczowym elementem pomyślnego zakończenia projektu jest prostota, czyli sztuka minimalizacji pracy, która jest do wykonania.
11. Najlepsza architektura wyrobu i projekty powstają w samoorganizujących się zespołach. Odpowiedzialność za poszczególne zadania nie powinna być przydzielana wybranym członkom zespołu z zewnątrz, tylko komunikowana całemu zespołowi.
12. Zespół w regularnych odstępach czasu debatuje nad poprawą efektywności i odpowiednio do wniosków zmienia swój styl działania. Członkowie takiego zespołu doskonale zdają sobie sprawę z tego, że środowisko, w którym pracują, stale się zmienia, i wiedzą, iż ich zwinność zależy w dużej mierze od zdolności dostosowywania się do tych modyfikacji.

4. Wybrane metody stosowane w metodyce *agile*

Metodyka *agile* nie stanowi monolitu, jest raczej wyznacznikiem dla bardziej konkretnych metodyk uszczegóławiających ją. Poniżej przedstawiono opis kilku najważniejszych przykładów współcześnie rozwijanych metodyk zwinnych.

- **XP (*eXtreme Programming*)**

Programowanie ekstremalne (*eXtreme Programming*) jest metodyką zakładającą codzienną współpracę z klientem. Pomysłodawcy XP uznają ciągłe zmiany wymagań projektu w trakcie jego trwania za coś naturalnego, nieuniknionego i pożądanego w prowadzeniu projektu. Wierzą oni, że możliwość adaptowania się do zmian w projekcie w każdej jego fazie jest bardziej realna i stanowi lepsze podejście niż zdefiniowanie wszystkich wymagań na początku projektu z oszacowaniem czasu po-

trzebnego do ich zrealizowania. Podstawą XP jest synergia wynikająca z uwzględnienia rozmaitych praktyk, które same w sobie mają wiele zalet, lecz mogą być trudne w pojedynczych zastosowaniach. Łączne użycie tych praktyk ma zapewniać wyeliminowanie niedogodności każdej z nich. Cechą charakterystyczną XP jest praca programistów w parach. Metodyka XP ma jednak kilka słabych punktów. Porównując ją z bardziej formalnym sposobem prowadzenia projektu opartego na dużej ilości dokumentacji, tutaj możemy mieć do czynienia z niestabilnością wymagań, brakiem dokumentacji projektowej czy też z brakiem udokumentowanych konfliktów i kompromisów z klientem.

- **Tworzenie oprogramowania oparte na modelu AMDD (*Agile Model Driven Development*)**

Jak nazwa wskazuje, jest to zwinna wersja modelu MDD, czyli tworzenie oprogramowania sterowane modelem. MDD jest podejściem do tworzenia oprogramowania, w którym szczegółowy model jest konstruowany, zanim zacznie się tworzenie kodu. Głównym przykładem MDD jest standard organizacji „Object Management Group" [Internet 2], nazwany MDA (*Model Driven Architecture*). Zwinna wersja MDD różni się od tej tradycyjnej tym, że zamiast tworzyć na początku dokładny model systemu (zaczynamy pisać kod), tworzymy model dobry w danej chwili, który umożliwi pokierowanie pracą w mniej więcej określonym kierunku, następnie w każdym kolejnym cyklu poprawiamy ten model.

- **Tworzenie oprogramowania na podstawie zbioru cech FDD (*Feature Driven Development*)**

Możemy interpretować to podejście jako wytwarzanie oprogramowania sterowane funkcjonalnościami. FDD jest iteracyjnym i przyrostowym sposobem wytwarzania oprogramowania. FDD jest mieszanką „najlepszych praktyk" (*best practices*) wziętych z przemysłu informatycznego i zebranych w miarę spójną całość. Te praktyki są podporządkowane orientacji na funkcjonalności z punktu widzenia klienta. Głównym celem jest dostarczenie realnych, działających części oprogramowania zgodnie z wymaganiami czasowymi. Podobnie jak w metodyce Scrum tutaj również mamy kilka predefiniowanych ról, takich jak: kierownik projektu, główny architekt, eksperci dziedzinowi, kierownik programistów, główni programiści. W FDD wyróżniono pięć faz projektu, z których dwie ostatnie są powtarzane wielokrotnie podczas projektu: budowa ogólnego modelu, budowa listy funkcjonalności, planowanie według funkcjonalności, projekt według funkcjonalności, implementacja według funkcjonalności. To, co wyróżnia tę metodykę, to dość dokładne śledzenie postępu projektu. Dotyczy to szczególnie dwóch ostatnich faz.

- **Metodyki Crystal**

Metodyki te (których jest kilka) mogą być stosowane w małych grupach projektowych (do 10 osób). Przyjmuje się, że nie stosuje się tych metodyk do projektów krytycznych (tzn. mających wpływ na zdrowie lub bezpieczeństwo człowieka). Metodyki Crystal skupiają się na wydajności i elastycznej możliwości przystosowania się do istniejącej sytuacji jako na głównym składniku bezpieczeństwa projektu. Po-

nadto metodyki Crystal koncentrują się na ludziach (uczestnikach projektu), niekoniecznie zaś na procesach czy też artefaktach. Metoda ta wyewoluowała z obserwacji pracy zespołów, które odniosły sukces projektowy.

- ***Agile Unified Process* (AUP)**

Agile Unified Process jest uproszczoną wersją metodyki firmowanej przez firmę IBM, a mianowicie RUP (*Rational Unified Process*) [Internet 1]. Opisuje proste i łatwe w zrozumieniu podejście tworzenia aplikacji, używając technik zwinnych i koncepcji, które są związane z RUP. Metodyka AUP w celu zwiększenia wydajności korzysta z takich technik, jak: TDD (*Test Driven Development*) [Internet 1], modelowanie zwinne, zwinne zarządzanie wymaganiami. AUP składa się z siedmiu kategorii, takich jak:

- model – zrozumienie organizacji, dla której będzie produkt, oraz zdobycie wiedzy domenowej,
- implementacja – przetransformowanie modelu w działający kod i wykonanie prostych testów,
- testy – przeprowadzenie dokładnych testów. Sprawdzenie, czy są spełnione wymagania klienta,
- dostarczenie do klienta – plan dostarczenia produktu do klienta,
- zarządzanie konfiguracją – zapewnienie bazy danych z postępami prac i kolejnymi wersjami produktu z możliwością śledzenia zmian i ich porównywania,
- zarządzenie projektem – zarządzanie ludźmi i budżetem, szacowanie ryzyka,
- koordynacja prac; śledzenie postępów,
- środowisko pracy – zapewnienie sprzętu i oprogramowania do pracy.

- ***Scrum***

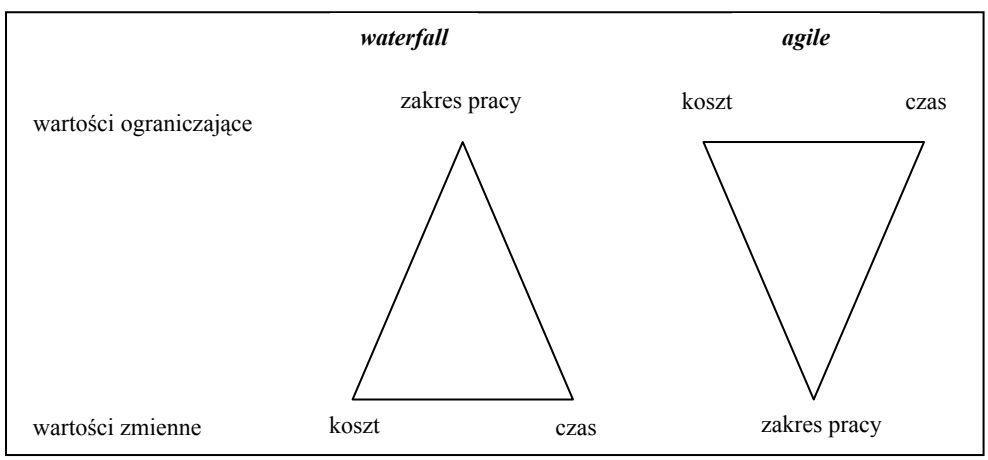
Nazwa *scrum*, czyli po polsku młyn, wywodzi się z terminu występującego w grze rugby. Używana jest często w przypadku przedsięwzięć o wysokim stopniu innowacyjności. Zespół projektowy pracuje w określonym przedziale czasowym zwanym przebiegiem (*sprint*). Efektem przebiegu za każdym razem powinno być dostarczenie użytkownikom kolejnego działającego produktu. Zasadą jest to, że zmiany wprowadzane w jednym przebiegu muszą być odczuwalne dla użytkowników. Powinny one wnosić wartość funkcjonalną. Przebieg może trwać od 2 do 6 tygodni. Zaleca się stosowanie przebiegów o stałych długościach. Zazwyczaj zespół liczy od 5 do 9 osób, ma charakter interdyscyplinarny, a poszczególne osoby reprezentują różne umiejętności. Osoby uczestniczące w zespole nie mogą uczestniczyć w innych zespołach. W zespole scrumowym występuje kilka predefiniowanych ról. Główne role to *scrummaster*, który jest kimś w rodzaju menedżera projektu, *productowner*, który jest przedstawicielem klienta i dostarcza wymagania do projektu (do każdego przebiegu dostarcza krótkie wymagania opisujące funkcjonalność przyszłego produktu), oraz *team*, czyli mieszany zespół programistów/testerów/architektów. Naczelną zasadą metodyki jest przeprowadzanie codziennych (ok. 15-minutowych) spotkań (*scrum meeting*), na których omawiane są zadania zrealizowane poprzednie-

go dnia i problemy występujące przy ich realizacji oraz zadania do wykonania w dniu spotkania. Przebieg kończy się spotkaniem podsumowującym (*sprint review*), na którym prezentowany jest wynik pracy zespołu przez przedstawianie produktu wykonanego podczas przebiegu. Powinni w nim uczestniczyć wszyscy zainteresowani projektem. Na spotkaniu każdy członek zespołu może zabrać głos i wyrazić opinię o produkcie. Po omówieniu produktu ustalany jest termin kolejnego spotkania dotyczącego następnego przebiegu.

Najczęściej stosowaną metodyką zwinną w praktyce jest właśnie *scrum*. Wynika stąd często nieporozumienie utożsamiania *agile* wyłącznie ze scrumem.

5. Porównanie metodyki *agile* i *waterfall*

W przypadku metodyki *waterfall* i metodyki *agile* różnica w podejściu do procesu wytwarzania oprogramowania jest znaczna. Model kaskadowy jest zorientowany na plan pracy (*plan driven*), co oznacza, że już na początku projektu dokładnie znamy zakres naszej pracy i to dopiero ma wpływ na czas i koszty. W przypadku metodyki *agile* jest odwrotnie; dysponując z góry określonym budżetem i ramami czasowymi, tak manewrujemy zakresem pracy, aby jak najlepiej spełnić oczekiwania klienta (*value/vision-driven*), co ilustruje rys. 2. Za jedną z kluczowych spraw uznano zaangażowanie klienta w projekt, tak aby stał się on integralną częścią zespołu. Dzięki temu mamy szansę stworzyć oprogramowanie w 100% spełniające oczekiwania klienta.



Rys. 2. Różnice między metodyką zwinną a kaskadową

Źródło: opracowanie własne.

Kolejną cechą różniącą obie metodyki jest podział pracy. W metodyce *waterfall* zadania są wypełniane kaskadowo. Zbieramy wymagania klienta, dokonujemy ich analizy i na tej podstawie możemy zaprojektować cały program. Za tym idzie kodo-

wanie, testy i finalne oddanie projektu klientowi. Negatywną stroną modelu jest to, że etap kodowania grozi przyspieszonym tempem pracy w jego końcowej części (nerwowa atmosfera i presja czasu). Dotrzymanie terminu często odbywa się kosztem testów, skutkiem czego klient nie zawsze dostaje w pełni sprawdzony program. *Agile*, jako metodyka promująca technikę TDD (*Test Driven Development*), nie zostawia testów na sam koniec pracy. Swego rodzaju innowacją jest to, że testy są przygotowywane najpierw, zanim programiści przystąpią do właściwego kodowania. Cały projekt dzieli się na poszczególne iteracje i dla każdego z tych etapów przewidziane są osobne testy jednostkowe.

6. Podsumowanie

Dla kogo jest metodyka *agile*? Czy tylko bogate korporacje mogą sobie pozwolić na zastosowanie takiej metodyki, czy może to projekt powinien mieć określoną wielkość, aby mógł spełniać wszystkie założenia manifestu? Odpowiedź może być sporna w przypadku tzw. projektów wysokiego ryzyka, jak np. kontrola lotów. Ciężko byłoby sobie wówczas wyobrazić iterację projektu, w której samolot „tylko" startuje (tutaj lepiej sprawdzi się metodyka kaskadowa). Z drugiej strony niezaprzeczalnie silną stroną podejścia *agile* są dobre testy, kluczowe dla systemów krytycznych. Można zauważyć natomiast duże korzyści dla mniejszych firm, które zdecydują się na metodykę *agile*. Dzięki metodzie kroków firma szybko widzi efekty i – co więcej – może je od razu wdrażać. Jako przykład może posłużyć strona internetowa, np. pizzerii, która zanim będzie dysponować pełną funkcjonalnością, na dzień dobry może oferować spis asortymentu i dane teleadresowe.

Obecnie istnieje wiele metodyk wytwarzania oprogramowania (w artykule brano pod uwagę model kaskadowy i zwinny), jednak najlepszym sposobem na prowadzenie projektu wydaje się stosowanie podejścia indywidualnie dobranego do projektu. Metodyka zwinna jest dość nowa, ale nie należy bezwzględnie stosować tego podejścia, metodyka kaskadowa ciągle znajduje uznanie w wielu projektach. Sama metodyka *agile* nie jest monolitem, jest zbiorem konkretnych metodyk (i praktyk), które stosuje się w codziennej pracy. Samo zdecydowanie się na metodykę *agile* nie mówi wiele, należy stosować jedną z implementacji tej metodyki.

W trakcie podejmowania decyzji dotyczącej wprowadzenia tej metodyki ważna jest odpowiedź na tylko jedno pytanie. Czy zwinne wytwarzanie oprogramowania pomoże w osiąganiu większych sukcesów w naszym projekcie? Kiedy zespół odpowie na to pytanie, będzie wiedział, czy powinien stosować właśnie zwinne programowanie [Shore, Warden 2008].

Literatura

Martin R.C., Micah M., *Agile. Programowanie zwinne: zasady, wzorce i praktyki zwinnego wytwarzania oprogramowania w C#,* Helion, Gliwice 2008.
Object Manegement Group (OMG), http://www.omg.org.
Shore J., Warden S., *Agile Development. Filozofia programowania zwinnego*, Helion, Gliwice 2008.

Źródła internetowe

[1] http://pl.wikipedia.org.
[2] http://www.omg.com.

AGILE METHODOLOGY AND SOME OF THEIR MODERN VARIETIES IN SOFTWARE DEVELOPMENT INDUSTRY

Summary: A lot of methodologies of software development are presented, however, project management often chooses methodology called Agile. In 2001 a group of experts worried about observed phenomena (consisting among others in landing a lot of corporations introuble as a result of growing processes) organized a meeting on which they created an association called „Agile Alliance".

The main goal of this article is the of characteristic Agile methodology of software development with a short description of its few modifications currently used in IT industry. In this article we can also find a description of one of the most often used "traditional" methodology – „Waterfall", with a short confrontation of these two methodologies.

Agile methodology is a name of some kind of collection of software development techniques that put emphasis on direct communication in development team and realized project with using iterations. These methods are used mainly for small teams (or companies) where we can communicate with each other easily, without creating a lot of project documentation. This allows to understand easily the main issue of the project and minimize the risk in a project which requires short time. However, we need a team which is well-integrated and permanent.