2.Metody Agile dla projektów BI

Załóżmy, że jesteś liderem projektu BI. Załóżmy również, że stosujesz typowe podejście do wdrażania projektu hurtowni danych z powiązanym narzędziem raportowania, procesem ETL odczytującym ze źródłowej transakcyjnej bazy danych i wstawiającym dane do swojego DWH. Zgodnie z typowym podejściem powinieneś zebrać specyfikacje od kluczowych użytkowników, pomyśleć o bardzo solidnym modelu danych, który służy do osiągnięcia tych specyfikacji, zainstalować wszystkie komponenty, wyodrębnić wszystkie potrzebne dane z różnych źródeł danych, zweryfikować integralność wszystkich tych danych dla wszystkich pól zdefiniuj wymagane raportowanie, a następnie będziesz mógł pokazać użytkownikowi kluczowemu wynik. Cały proces mógł trwać miesiące, a może nawet lata. Kiedy sprawdzasz z kluczowym użytkownikiem, jaki był wynik, Twój użytkownik mógł zmienić zdanie co do tego, czego potrzebuje, lub może Twój kluczowy użytkownik zmienił zdanie i ma zupełnie inne pomysły na to, co wykorzystać w swoich raportach. Możesz stracić dużo czasu i wysiłku na rozwój, nie osiągając żadnych rezultatów, a to może stanowić realne ryzyko przy tradycyjnym podejściu do zarządzania projektami hurtowni danych. Aby uniknąć tego ryzyka, możesz spróbować użyć tego, co uważamy za najlepsze metodyki Agile stosowane w projektach BI. W każdym razie, nie zawsze stosowanie metodologii Agile jest najlepszą strategią kontynuowania projektu. Tradycyjne metodologie opierają się na sekwencyjnych procesach, intensywnym planowaniu i dużych kosztach zarządzania projektami, więc mogą być przydatne, gdy masz zamkniętą definicję wymagań projektowych, jasną strategię rozwoju projektu i długoterminowe projekty do opracowania. Tłumaczymy, jak pracować z metodologiami Agile, ponieważ jesteśmy przekonani, że jeśli zamierzasz realizować projekt BI od zera, z użytkownikami, którzy nigdy nie korzystali z narzędzia BI, całkiem możliwe, że będziesz musiał wprowadzić zmiany w trwającego projektu, współpracować z użytkownikami końcowymi, aby sprawdzić, czy wynik projektu jest zgodny z ich oczekiwaniami, stworzyć wstępne rozwiązanie, a następnie dodawać nowe funkcjonalności w kolejnych wersjach. W tych warunkach użycie Agile może pomóc w realizacji projektu. Istnieje również podejście pośrednie, które łączy niektóre tradycyjne cechy z Agile, wykorzystując je do wykonywania niektórych tradycyjnych zadań. Oczywiście Agile działa również wtedy, gdy masz zamkniętą definicję wymagań projektowych, więc możesz używać Agile do rozwiązywania zarówno dynamicznych, jak i statycznych wymagań projektowych. Chcielibyśmy również zauważyć, że Agile nie jest metodologią samą w sobie, jest modelem do stworzenia Twojej metodologii; to zestaw reguł, które musisz dostosować do swojego środowiska i technologii, aby dokładnie określić parametry procesów roboczych. Ważne jest, aby powiedzieć, że być może, jeśli jesteś w małej firmie, niektóre zalecenia wewnątrz metodologii nie mają zastosowania. Istnieje pewna metodologia, która obejmuje codzienne spotkania, jeśli jesteś jedynym facetem w IT i będziesz odpowiedzialny za opracowanie całego rozwiązania, nie będziesz potrzebować wewnętrznych spotkań programistów w celu śledzenia statusu zadań, o ile będziesz być jedyną osobą na tym spotkaniu, chyba że masz wiele osobowości i tak czy inaczej w tym przypadku trudno byłoby spotkać wszystkich razem. W przypadku, gdy Twój zespół nie uzasadnia wdrażania pewnych rzeczy z metodologii, po prostu je zignoruj i weź stąd metody i działania, które uważasz za interesujące, aby poprawić swoją produktywność i odnieść sukces w swoim projekcie. Aby umożliwić Ci podjęcie decyzji, czy jesteś zainteresowany stosowaniem metodologii Agile, zróbmy krótkie wprowadzenie do tego, czym są metodologie Agile i jak można je wykorzystać do poprawy zadowolenia klienta w projekcie BI.

Uwaga: ta część to zbiór rekomendacji, które uważamy za interesujące, wyciągniętych z naszego doświadczenia z metodologiami Agile, zwłaszcza Scrum i Kanban, więc nie traktuj ich

jako czysto Agile, ponieważ to, co przeczytasz, to osobiste przemyślenia na ich temat.

Wprowadzenie do metodyk Agile

W niektórych przypadkach nazwa pojęcia, które próbujesz zdefiniować, jest na tyle znacząca, że może być autoopisowa. Opisujemy teraz jedną z takich sytuacji. Z naszego punktu widzenia główną cechą metodologii Agile jest to, że są one zwinne. Ale co to znaczy - zwinny pod względem projektów programistycznych? Zasadniczo istnieją cztery obszary, na których musi koncentrować się metodologia Agile . jeśli spojrzymy na zasady Manifestu Agile. Oto cztery zasady manifesty Agile, które doceniliśmy:

- Osoby i interakcje ponad procesami i narzędziami
- Działające oprogramowanie ponad obszerną dokumentację
- Współpraca z klientem ponad negocjacje kontraktów
- Reagowanie na zmiany zamiast podążania za planem

Z tych wartości manifestu Agile, dwie są dla nas szczególnie istotne: skupienie się na zadowoleniu klienta i dostosowywaniu się do zmian; ta ostatnia koncepcja sama w sobie jest definicją zwinności. Wewnątrz projektu Agile nie martwimy się zmianami, które mogą pojawić się w specyfikacjach, uważamy je za normalne; w rzeczywistości są zdrowe, o ile użytkownicy mogą dostosować swoje potrzeby, gdy zobaczą, co może zrobić narzędzie BI. Powinniśmy mieć w naszej firmie użytkowników o zdolnościach widzących, jeśli oczekujemy od nich, że są w stanie wiedzieć, czego chcą, nie wiedząc o możliwościach, jakie oferuje Twoje narzędzie. Aby postępować zgodnie z metodykami Agile, powinieneś skupić się na członkach zespołu i ich wiedzy, a nie na zdefiniowanym procesie rozwoju. Wolisz, aby wszyscy członkowie zespołu byli świadomi tego, co robi reszta, jak to zrobić i czy jest jakaś blokada, którą mogą rozwiązać inni członkowie zespołu, dzieląc się doświadczeniem w celu ulepszenia technik zamiast silnych zasad rozwoju lub bezużyteczne narzędzia kontrolne. Będziesz także zainteresowany posiadaniem produktu do pokazania klientom, a nie wymaganą dokumentacją. Dokumentacja to drugorzędne zadanie, które należy wykonać, ale priorytetem jest posiadanie działającego systemu bez awarii. Lepiej mieć solidny mały projekt niż niespójny duży system. Lepiej mieć kilka niezawodnych funkcji niż setki funkcji, które od czasu do czasu zawodzą, ale powinny działać, jeśli sprawdzisz w dokumentacji. Tutaj możemy przyjrzeć się koncepcji Incremental Delivery, o ile będziemy zainteresowani posiadaniem małego projektu początkowego, ale z dodawanymi funkcjonalnościami co kilka tygodni. Zależy nam na tym, aby rozwiązanie było dostępne jak najszybciej, aby uzyskać opinie klientów na początkowym etapie całego projektu, aby móc korygować nieporozumienia i zmieniać funkcjonalności w oparciu o oczekiwania klientów. Aby to osiągnąć, będziemy potrzebować ścisłej współpracy z naszymi klientami. Jak zobaczysz w kolejnych sekcjach, klient jest włączony w cykl życia każdej iteracji rozwoju, który jest podstawą rozwoju Agile. Życie to zmiana. Nic nie jest statyczne. Zwinne dostosowywanie się do zmian to główna zaleta, jaką metodologie Agile mogą zaoferować, aby osiągnąć sukces w projekcie BI. Dobrze jest mieć plan, ale dostosowanie się do zmiany da klientowi większą satysfakcję niż podążanie za planem. Ostatecznie to Twój klient będzie korzystał z Twojego systemu, a korzystanie z systemu jest głównym wskaźnikiem sukcesu Twojego projektu. Agile to także optymalizacja. Za Agile kryje się intencja optymalizacji naszych zasobów, aby zmaksymalizować postrzeganą wartość dla naszych klientów; mogą to być wewnętrzni lub zewnętrzni użytkownicy końcowi platformy BI. Postaramy się ograniczyć lub bezpośrednio uniknąć wszystkich tych zadań, które nie dodają wartości dla klienta. Z naszego doświadczenia wiemy, że wysoki odsetek funkcjonalności

wymaganych w początkowych wymaganiach dużego projektu nigdy nie jest wykorzystywany przez klientów, wiele wymiarów lub metryk nie jest używanych w raportach, wiele raportów nie jest wykonywanych po wstępnej walidacji, a także wielokrotnie wykryliśmy, że wielu żądanych użytkowników, którzy mają dostęp do platformy, nigdy nie logował się na naszych platformach. Z Agile postaramy się uniknąć wszystkich tych bezużytecznych wysiłków, aby spróbować skupić się na tych funkcjonalnościach, których naprawdę chce prawdziwy użytkownik. Wprowadzam tutaj pojęcie rzeczywistego użytkownika, ponieważ znaleźliśmy również wymagania projektowe definiowane przez informatyków, przez działy, które będą inne niż te, które będą korzystać z narzędzia lub przez menedżerów bardzo wysokiego szczebla, którzy czasami nie są świadomi potrzeby wykonywać codzienne zadania pracowników, więc rzeczywisty użytkownik musi mieć bezpośrednią wiedzę na temat tych codziennych potrzeb, aby zdefiniować projekt, który od początku będzie w stanie objąć najczęściej używane funkcjonalności. Gile to także synonim współpracy. Nie ma lidera zespołu programistów (może się to wydawać pośrednio na podstawie doświadczenia niektórych członków zespołu), ale chodzi o to, że wszystko jest omawiane, oceniane i analizowane przez cały zespół, a następnie poszczególne osoby wykonują zadania rozwojowe, ale reszta jest zrobiona współpraca z całym zespołem. Kolejną zaletą Agile, jeśli jesteś w stanie wdrożyć ją we właściwy sposób, jest to, że spróbujesz dostosować wymagania dotyczące obciążenia pracą z istniejącą grupą zadaniową, aby nie stresować swojego zespołu programistów, unikając również szczytów obciążenia pracą. Spróbuj zmniejszyć wielozadaniowość w Twoim zespole, aby umożliwić programistom skupienie się na przydzielonym zadaniu. Pomysł polega na tym, że każdy członek zespołu wykonuje po prostu szeregowane zadania, jedno po drugim.

Zwinne podejście

Istnieje wiele metodologii Agile, które są zgodne z zasadami Agile, które właśnie widzieliśmy w poprzedniej sekcji, takie jak Scrum, Kanban, Lean Software Development, Adaptive Software Development, metody Crystal Clear, Extreme Programming, Feature-Driven Development i inne, ale ta rozdział nie pretenduje do miana wyczerpującej analizy Agile, bo treści starczyłoby na jeszcze jedną czy dwie książki; więc skupimy się na dwóch głównych metodach, Scrum do zarządzania projektami i Kanban do utrzymania rozwiązania. Istnieje również mieszanka znana jako Scrumban, która ma pewne cechy obu metodologii.

Nasza rekomendowana mieszanka Scruma i Kanbana

Opierając się na zasadach Agile, istnieje kilka elementów, które naszym zdaniem powinny być wspólne dla metodologii Scrum i Kanban. Pierwszym z nich jest panel wizualizacji, o ile ważne jest, aby każdy w zespole miał dostęp do statusu każdego zadania. Panel wizualizacji jest obowiązkowy dla Kanbana ale opcjonalne dla Scruma, który uwzględnia tylko zaległości zadań, ale zdajemy sobie sprawę, że posiadanie go na fizycznej tablicy jest bardzo pozytywne. Uważamy również, że bardzo interesującą funkcją są regularne, codzienne spotkania. Są one obowiązkowe dla Scruma i opcjonalne dla Kanbana, ale i tak uważamy to za bardzo interesującą funkcję. Oba podobieństwa wynikają z pierwszej wartości z manifestu Agile, aby nadać priorytet indywidualnej ewolucji i interakcji między nimi w stosunku do narzędzi, których używamy, na końcu priorytetem jest wiedza, którą zdobywają. Spotkania muszą mieć dużą cykliczność, jeśli chodzi o przepływ wiedzy między członkami zespołu i ważne, żeby były krótkie, ale częste. Idealnym okresem jest codzienne 15-minutowe spotkanie na początku dnia i powinno to być obowiązkowe. Również w Scrumie będziesz mieć różne okresowe spotkania, jak zobaczymy w następnej sekcji. Te codzienne spotkania można doskonale przeprowadzić stojąc w kącie biura, a szczególnie interesujące jest to, że odbywają się przed drugim wspólnym elementem, panelem wizualizacyjnym. Ważne jest, aby mieć tablicę lub ścianę z uporządkowanymi wszystkimi zadaniami,

które się tam znajdują aby zobaczyć w bardzo graficzny sposób, które tematy są w toku, które z nich z backlogu są najważniejsze, aby je podnieść i czy istnieje jakakolwiek zależność między zadaniami itp. Standardowym sposobem lokalizowania zadań jest uporządkowanie według kolumn na podstawie statusu zadań i grupowania według wierszy w zależności od tematu lub obszaru. Istnieją również narzędzia, które pozwalają uporządkować wszystkie zadania, więc ta graficzna wizualizacja może być na tablicy lub przy użyciu projektora lub ekranu. Na codziennych spotkaniach cały zespół przegląda status zadań i przesuwa zadania wzdłuż kolumn, gdy stan się zwiększa. Liczba kolumn i wierszy zostanie dostosowana w zależności od potrzeb projektu, ale powinny mieć co najmniej trzy kolumny: Backlog, In Progress i Done. W Backlogu zlokalizujemy wszystkie zadania oczekujące na wykonanie, w kolumnie W toku zlokalizujemy wszystkie zadania, w które zaangażowany jest jeden z członków zespołu lub klient, a na końcu przejdziemy do Gotowe wszystkie zakończone zadania . Gdy pojawi się nowa wersja oprogramowania/projektu, czyścimy kolumnę Gotowe i zaczynamy od nowa. Przyjrzymy się szczegółowo procesowi w każdym podejściu Agile.

Tworzenie projektów w Scrumie

Scrum był jedną z pierwszych metodyk, które pojawiły się jako próby łagodzenia problemów typowego zarządzania projektami z podejściem sekwencyjnym i wysoce zorganizowanymi projektami w zmieniającym się środowisku. Została zdefiniowana w 1986 roku przez Hirotakę Takeuchi i Ikujiro Nonakę. Chcieli wdrożyć elastyczną strategię rozwoju skupiając się na zdefiniowaniu wspólnego celu, a więc skupiając się na zespole deweloperskim. Istnieje kilka koncepcji wewnątrz Scruma, niektóre z nich są wspólne z innymi podejściami, ale zgodnie z komentarzem, skupimy się na tej metodologii. Aby wyjaśnić opis metodologii, która może być bardzo abstrakcyjna, zilustrujemy cały opis metodologii na podstawie tego samego fikcyjnego przykładu, projektu opracowanego dla firmy zajmującej się sklepami z narzędziami (HSC), która chce wdrożyć system BI do analizy spostrzeżeń firmy. Jest to średnia firma z 5 sklepami i 70 pracownikami, 20 z nich pracuje w centrali, 50 z nich pracuje w sklepach, w zespole sprzedaży. Każdy sklep ma kierownika sklepu, który koordynuje wszystkie wymagane przez sklep czynności, takie jak uzupełnianie zapasów, zarządzanie kasami, koordynacja z zespołami centralnymi do zarządzania zasobami ludzkimi, inicjatywami handlowymi i ofertami itp. Dział IT składa się z 3 pracowników:

* Menedżer IT: odpowiedzialny za koordynację zadań IT, kontakt z działem zakupów w celu zakupu urządzeń technicznych i zakupów oprogramowania oraz kontakt z dostawcami zewnętrznymi w razie potrzeby.

* Technik terenowy: odpowiedzialny za konserwację i naprawę komputerów i sieci w centrali i sklepach.

* Technik danych: odpowiedzialny za administrowanie serwerami centralnymi, takimi jak system ERP, poczta e-mail, współdzielone dyski sieciowe itp.

W tej firmie użyjemy Scruma do wdrożenia systemu BI, który pomoże firmie analizować dane, zaczynając od analizy sprzedaży, ale po zakończeniu będzie chciał uwzględnić zarządzanie zapasami, dane dotyczące zamówień i dane dotyczące zasobów ludzkich.

Role

Aby zdefiniować, co kto robi, zacznijmy od zdefiniowania ról, które mamy w Scrumie. Powiążemy je również z naszą przykładową strukturą.

Właściciel Produktu

Jest to podstawowa rola ze strony klienta. Będzie odpowiedzialny za zdefiniowanie, co chcą zawrzeć w projekcie, co jest priorytetem dla każdej funkcjonalności, jakie są warunki uznania funkcjonalności za w pełni działającą i akceptację rozwoju. Będzie współpracował z zespołem programistów w celu zdefiniowania, przeanalizowania i uszczegółowienia każdego wymagania. W naszym przykładzie rolę tę przejmie kierownik Kontrolingu Sprzedaży firmy.

Mistrz Scruma

Scrum Master będzie osobą odpowiedzialną za koordynację wszystkich opracowań, upewniając się, że zespół programistów rozumie i przestrzega zasad Agile; przeszkoli zespół, aby pomóc w jego autoorganizacji; pomóc usunąć blokady i przeszkody; i staraj się unikać ryzykownych przerw, które mogą przeszkadzać zespołowi w jego głównym zadaniu. Na przykładzie będzie reprezentowany przez kierownika IT.

Zespół deweloperski

Zespół programistów będzie siłą roboczą do wykonania wszystkich działań wymaganych w rozwoju. Wycenią każde wymaganie, analizując, ile może kosztować i jak je rozwinąć. Zorganizują wewnętrznie, kto w zespole podejmie się każdego zadania, a także przeanalizują blokady, które mogą pojawić się podczas procesu rozwoju, aby spróbować ich uniknąć. Pod koniec rozwoju przeanalizują, jak ulepszyć proces rozwoju dla kolejnych wymagań. Wewnątrz tego zespołu możemy zdefiniować różne role podrzędne, takie jak architekt rozwiązania, programiści, testerzy, analitycy funkcjonalni, grafik itp. Zapewnią one wewnętrzną jakość realizowanych rozwiązań, aby uniknąć skutków w przyszłości. Idealnie byłoby, gdyby ich profil był ogólny, co oznacza, że każdy członek zespołu może wykonać dowolne zadanie, ale nie zawsze jest to możliwe. W tym przykładzie rola ta zostanie przejęta przez kierownika IT, technika danych i dwóch programistów wynajętych z firmy zewnętrznej do pomocy we wstępnym rozwoju.

Interesariusze

Jest to bardzo ogólna koncepcja, ponieważ będzie wielu interesariuszy zainteresowanych tym, co robimy. Nie są zaangażowani w proces rozwoju, ale muszą być informowani o decyzjach projektowych io postępach projektu. W naszym przykładzie za interesariuszy uznamy cały zespół zarządzający, w tym dyrektora generalnego.

Użytkownicy

Będą oni końcowymi konsumentami informacji, które dostarczamy. W naszym przykładzie liczba użytkowników będzie rosła, zaczynając od zespołu wsparcia sprzedaży i kierowników sklepów, a po dodaniu nowych funkcjonalności będziemy obejmować jako użytkowników zespół HR, zespół finansowy, magazynierów odpowiedzialnych za wszystkie sklepy oraz zespół zakupów.

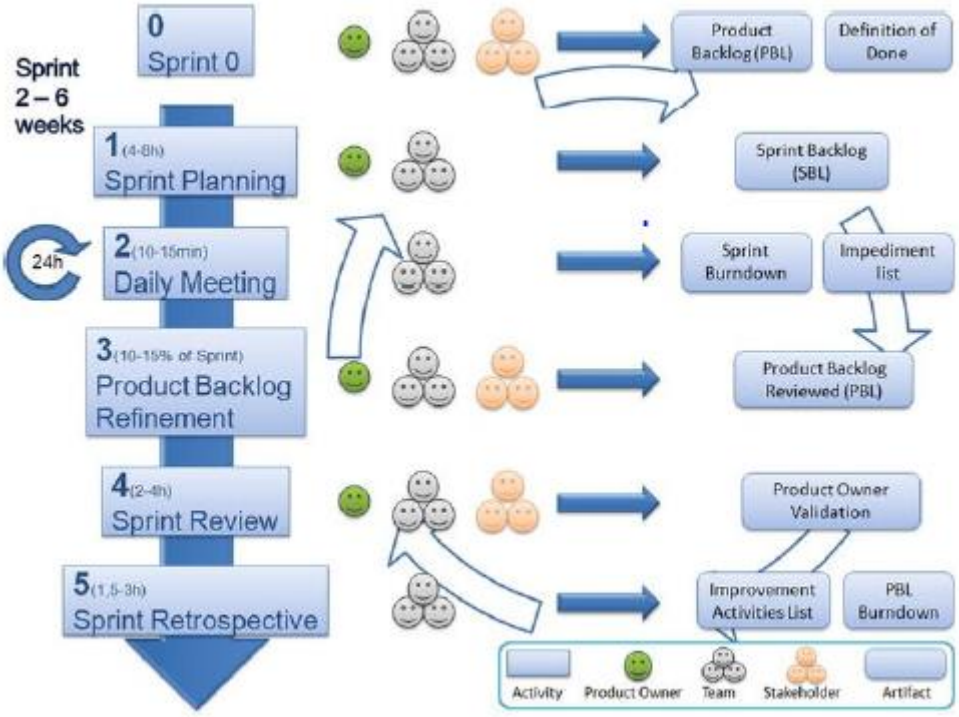
Asystent Właściciela Produktu

Ta rola jest opcjonalna; będzie to zależało od wiedzy Agile Właściciela Produktu. Będzie odpowiedzialny za ocenę Właściciela Produktu pod kątem zasad i zasad Agile. W naszym przykładzie nie rozważymy tego.

Sprint

Koncepcja sprintu lub iteracja to jedno z najważniejszych pojęć w Scrumie, także w większości metodologii Agile. Można to również potraktować jako iterację. Sprint to pojedynczy cykl rozwoju wewnątrz projektu, który zapewni w pełni użyteczną wersję projektu dostępną dla klientów do

przetestowania. Jest to okres próbowania od 2 do 6 tygodni, w zależności od wielkości zadania projektu, z zamiarem ograniczenia go do minimum. Docelowy rozmiar zadania powinien wynosić od 2 do 8 godzin obciążenia pracą, aby łatwo wykrywać zadania, które miały status „W toku" dłużej niż jeden dzień. Aby osiągnąć ten cel, jest całkiem możliwe, że musimy podzielić zadania na podzadania, aby zmieścić się w tym maksymalnym rozmiarze zadania wynoszącym 8 godzin. Wyjaśnimy, jak to zrobić w następnej sekcji Techniki segmentacji projektów. Wybierzemy wystarczającą liczbę zadań, aby uwzględnić je w sprincie, starając się, aby zespół był zajęty przez cały sprint, ale upewniając się, że zespół nie zostanie przeciążony. Sprint jest podzielony na trzy części: inicjalizacja sprintu w celu określenia, jakie zadania mają zostać uwzględnione, opracowanie tych zadań, które zapewnią następną wersję projektu oraz zakończenie sprintu w celu opublikowania wyników klientowi i oceny występ zespołu. Z tej organizacji sprintu wynikają spotkania wyjaśnione w następnej sekcji. Na rysunku



możesz zobaczyć diagram przedstawiający organizację sprintu, jakie są jego spotkania, kto powinien asystować oraz oczekiwany rezultat każdego spotkania. W przykładzie, który śledzimy wstępnie zdefiniujemy czas trwania sprintu na 4 tygodnie i postaramy się go skrócić podczas kolejnych sprintów, więc początkowo będziemy mieć siłę roboczą 144 godzin na osobę aż do pierwszego i ostatniego dnia każdego sprintu jest przeznaczona na rozpoczęcie i zakończenie sprintu. Jeśli mamy 3 osoby w zespole, będziemy mieli do dyspozycji do 432 godzin programistycznych do zarządzania w sprincie.

Konkretne spotkania Scrumowe

W odniesieniu do sprintu znajdują się cztery powiązane spotkania, które są specyficzne dla Scruma, również współdzielone z inną metodologią, a także kilka wstępnych spotkań w celu zdefiniowania zakresu projektu i wspólnego codziennego spotkania. Sprint zero meeting : Celem tego spotkania lub spotkań jest określenie zakresu projektu i jakich funkcjonalności oczekuje użytkownik, aby zdefiniować przegląd wymagań projektowych. Z tego spotkania powinniśmy wydobyć Backlog Produktu, czyli listę wymagań, które należy opracować, aby móc dostarczyć działającą wersję projektu. Ta lista wymagań musi zawierać opis każdego elementu Backlogu Produktu, czyli znaczenie tego PBI dla klienta, jaki jest koszt wdrożenia go przez zespół programistów i czy istnieje jakiekolwiek ryzyko związane z

wymaganiem. Również na tym spotkaniu powinniśmy otrzymać Definicję Ukończenia, czyli listę kontrolną warunków wynikających z umowy między klientem a deweloperem, której rezultatem jest oczekiwany wynik dla wszystkich wymagań. W spotkaniu tym powinien wziąć udział również właściciel produktu, zespół deweloperski i przynajmniej ktoś reprezentujący wszystkich interesariuszy. W naszym przykładzie zdefiniujemy na spotkaniu zerowym Sprintu, że chcemy uwzględnić następujący Backlog Produktu zawarty w Tabeli 1

Product Backlog Item	Business value (1-10)	Development cost	Risks
Sales data coming from ERP system	10	200 hours	Unknown source structure
Master data hierarchies from ERP system	9	240 hours	Lack of definition of the required levels
Customized masterdata groups	7	160 hours	Extra technology to connect
Warehouse Stock information	8	320 hours	Different masterdata values
Financial information	9	400 hours	N/A
HR information	7	320 hours	Confidential restrictions

* Spotkanie planowania sprintu: Na tym spotkaniu wybieramy umowę z Właścicielem Produktu, listę Elementów Backlogu Produktu, które będziemy rozwijać w sprincie, traktowanych również jako Backlog Sprintu. Elementy te można podzielić na zadania, aby zespół mógł zarządzać rozmiarem zadania. To spotkanie jest podzielone na dwie części: pierwsza obejmuje właściciela produktu i zespół programistów, a druga obejmuje tylko zespół programistów.

- Spotkanie „Co”: Pierwsza część tego spotkania powinna zająć od dwóch do czterech godzin, aby zdefiniować wszystkie uwzględnione PBI i uzyskać pełne zrozumienie od zespołu programistycznego, do czego odnosi się PBI, wprowadzając szczegóły dotyczące dokładnego wyniku wymagania lub funkcjonalność.

- Spotkanie „How to”: Druga część tego spotkania powinna zająć również od dwóch do czterech godzin i powinniśmy ocenić, jak spełnić wymagania właściciela produktu, które należy podzielić na zadania każdego PBI, który z zespołu deweloperskiego jest w za każde zadanie i jak długo spodziewamy się, że to zadanie może zająć. Jeśli niektóre zadania wyjściowe wymagają wyceny wymagającej dużego nakładu pracy, musimy spróbować ponownie je podzielić. Oszacowanie nakładu pracy musi być wykonane wspólnie przez cały zespół starający się osiągnąć jak największy postęp, co może wymagać rozwiązania problemów związanych z rozwiązywaniem problemów.

Mamy zaległe obciążenie szacowane na 432 godziny. Nasza wycena pracochłonności dla dwóch pierwszych zadań to 440 godzin, więc w ramach pierwszego sprintu zespół spróbuje zaimplementować pierwszy element z Backlogu Produktu i spróbuje ukończyć drugi z pomocą kierownika IT. W tym celu postaramy się podzielić każdy element na zadania, aż do osiągnięcia rozsądnej wielkości zadania. W Tabeli 2 pokazujemy tylko początkowy podział pierwszego zadania. Powinniśmy podążać za podziałem próbując zdefiniować pojedyncze podzadania z rozmiaru od 2 do 8 godzin.

Product Backlog Item	Task	Development cost	Who
Sales data coming from ERP system	Analyze information source	16 hours	Data technician
	Define database model	24 hours	Data technician
	Implement ETL process	40 hours	External 1
	Define data views	16 hours	External 1
	Create logical model	16 hours	External 2
	Define BI objects	32 hours	External 2
	Create BI reports	24 hours	External 2
	Technical validation	16 hours	Data technician
	Process automation	16 hours	Data technician

* Codzienne spotkanie: Zgodnie z komentarzem w podejściu ogólnym, zespół programistów będzie miał 15-minutowe codzienne spotkanie rano, aby skomentować bieżące zadania. Na tym spotkaniu każdy w zespole powinien odpowiedzieć sobie na trzy pytania:

- Co zrobiłeś od wczorajszego spotkania?

- Co dzisiaj robisz?

- Jakie punkty blokujące znalazłem lub mogą się pojawić?

Idea maksymalnego rozmiaru zadania wynoszącego osiem godzin polega na tym, że podczas codziennego spotkania można wykryć, czy jakieś zadanie zostało zawieszone, o ile nie powinno się wykonywać tego samego zadania na dwóch kolejnych codziennych spotkaniach. Z tego spotkania powinniśmy prowadzić listę przeszkód, które mogą się pojawić. Wracając ponownie do naszego przykładu, możemy wykryć, że nie mamy wymaganych poświadczeń, aby uzyskać dostęp do danych ERP, lub że potrzebujemy wsparcia od dostawcy ERP, aby zrozumieć strukturę tabeli. Być może jakiś składnik rozwiązania BI, taki jak narzędzie do raportowania BI, zwłaszcza w początkowej fazie rozwoju, oczekuje na dostarczenie; zabrakło nam miejsca w naszym rozwojowym systemie bazodanowym lub wielu innych przykładowych blokad, które mogą się pojawić.

* Udoskonalanie Rejestru Produktu: To spotkanie można zaplanować w dowolnym momencie sprintu, ale należy je przeprowadzić dla wszystkich sprintów, aby przejrzeć pozostały Backlog Produktu, czy są jakieś dodane zadania, które należy uwzględnić na liście, jakiś priorytet zmiana itp. Podobnie jak w przypadku spotkania zerowego sprintu, wymagana jest asysta Właściciela Produktu i zespołu deweloperskiego na spotkaniu, a oczekiwanym wynikiem jest przegląd i aktualizacja Backlogu Produktu.

* Przegląd sprintu: Po zakończeniu sprintu dokonamy wraz z właścicielem produktu i interesariuszami przeglądu wszystkich wdrożonych PBI. Da to naszemu klientowi możliwość przeglądu swoich oczekiwań, zmiany priorytetów zadań, zmiany wszystkiego, co uważa, lub dodania nowych wymagań w oparciu o rzeczywisty wynik dotyczący tego, co może zrobić nasze narzędzie. Wtedy podczas spotkania dopracowującego backlog produktu będzie mógł przełożyć naszą pracę. W przykładzie HSC możemy stwierdzić, że kierownik ds. wsparcia sprzedaży wykrył, że hierarchia produktów z transakcji nie pasuje do grup finansowych wymaganych do wyodrębnienia danych o przychodach netto, ponieważ muszą one grupować według produktów z różnymi stawkami podatkowymi i jest to nie ustawiono w wyodrębnionych przez nas tabelach ERP. Więc kiedy część sprzedażowa działa, kiedy próbujemy dodać informacje finansowe, musimy zmodyfikować obciążenie masterdat dodaj kilka dodanych pól.

* Retrospektywa sprintu: Jest to wewnętrzne spotkanie zespołu deweloperskiego, które pozwoli zespołowi przejrzeć to, co zrobiliśmy, jak rozwiązaliśmy problemy, które się pojawiły, co możemy zrobić, aby uniknąć ponownego pojawienia się problemów i jak uniknąć blokad, które może zamrozić nasz rozwój. To spotkanie pozwoli na ciągłe doskonalenie całego zespołu poprawiając produktywność każdego z nas. Wynikiem tego spotkania powinna być lista ulepszeń, które powinniśmy zastosować, aby uzyskać lepszą wydajność naszej fazy rozwoju. Z tego spotkania mogliśmy zobaczyć, że wolelibyśmy mieć rozmiar sprintu wynoszący trzy tygodnie i zmniejszyć rozmiar zadania, ponieważ chcemy mieć większą interakcję z właścicielem produktu. Lub możemy ulepszyć część metody, na przykład dodać nowy status do walidacji zadania, lub zdecydować, że spotkania będą odbywać się na projektorze zamiast fizycznej tablicy.

Wydanie

Zwykle będziemy pracować z co najmniej dwoma środowiskami: jednym do programowania i drugim do produkcji, a może trzema, dodatkowym środowiskiem, w którym użytkownik sprawdzi poprawność wykonanego rozwoju za pomocą ekstrakcji danych produkcyjnych. Po wykonaniu i zatwierdzeniu sprintu musimy promować wszystkie zmiany w środowisku produkcyjnym, aby móc czerpać korzyści z opracowanych nowych wymagań. Wiemy, że wynikiem wszystkich sprintów jest zestaw wymagań składających się na w pełni działające rozwiązanie dostarczane Właścicielowi Produktu do walidacji, ale przejście do produkcji może wymagać pewnych wysiłków, takich jak ponowna kompilacja, pobieranie klienta przez wszystkich użytkowników, ponowne ładowanie danych itp. ., i możliwe, że zdecydujemy się poczekać i zamrozić część rozwoju, dopóki nie będziemy mieli większej ilości wymagań, aby przejść do produkcji. Koncepcja wydania polega na przeniesieniu do produkcji zestawu rozwiązań, które mogą pochodzić z jednego lub wielu sprintów. Dostarczymy wersję, gdy będziemy w stanie zgromadzić wystarczającą liczbę funkcji o wysokiej wartości dla użytkowników końcowych, aby zaakceptowali zmianę swojego obecnego narzędzia. Ponownie w naszym przykładzie wprowadzimy do produkcji po opracowaniu pierwszych trzech elementów: danych sprzedaży, danych podstawowych z ERP i dostosowanych danych podstawowych, o ile jest to obowiązkowe dla kierownika ds. wsparcia sprzedaży.

Artefakty używane w Scrumie

Z objaśnienia sprintu widzieliśmy kilka narzędzi, które pomogą nam iść do przodu z naszą metodologią Scrum. Istnieją również inne narzędzia, które pomogą nam monitorować i śledzić rozwój projektu. Wszystkie wymienione i wyjaśnione narzędzia są uważane za artefakty w nomenklaturze Scruma.

Historia użytkownika

Ważne jest, aby mieć techniczną definicję tego, co jest wymagane do opracowania, ale ważne jest również, aby użytkownik wyjaśnił, co chce uzyskać, aby móc poprawnie zrozumieć, jakie są prawdziwe wymagania techniczne. Historyjka użytkownika jest krótką definicją wymagania i oczekuje się, że zostanie napisana zgodnie z szablonem historii użytkownika, który obejmuje rolę osoby żądającej, jakie jest wymaganie i jakie są korzyści z jego wdrożenia. Przykładem szablonu może być to:

As < role >,

I want < requirement >

so that < benefit >

Przechodząc ponownie do naszego przykładu firmy sprzętowej, moglibyśmy postrzegać je jako historie użytkowników:

Jako właściciel produktu chcę mieć możliwość drążenia hierarchii klientów, aby użytkownicy mogli analizować szczegóły sprzedaży poprzez organizację klientów ERP. Jako HR Manager chcę mieć dostępne dane o wynagrodzeniach i premiach, aby nasz dział mógł analizować informacje w podziale na hierarchię firmy, analizując koszty HR, urlopy i szkolenia. Jako kierownik finansowy chcę mieć informacje o zyskach i stratach, aby nasz dział mógł analizować PNL poprzez główną hierarchię produktów. Jako Sales Force Manager chcę analizować wyniki sprzedaży w poszczególnych sklepach, aby móc analizować sprzedaż według kategorii, grup klientów i kampanii promocyjnych.

Właściciel produktu i zespół programistów przeanalizują i posortują według priorytetów wszystkie te historie użytkowników, włączając je do Backlogu Produktu, a następnie zespół programistów przeanalizuje, jak technicznie postępować, aby spełnić to wymaganie. Istnieje procedura o nazwie INVEST, która sprawdza, czy historia użytkownika jest wystarczająco dobra, aby potraktować ją jako punkt wyjścia do rozwoju. Ta procedura odpowiada inicjałom słów:

Niezależne: Powinniśmy być w stanie opracować historyjkę użytkownika w oderwaniu od reszty oczekujących historyjek użytkownika.

Do negocjacji: Ponieważ historia użytkownika nie jest ściśle zdefiniowana, możemy negocjować z właścicielem produktu, jak postępować z rozwojem historii użytkownika.

Wartościowe: historia użytkownika zapewni użytkownikowi końcowemu pewną wartość dodaną. Szacunkowe: Z definicją, którą wykonał użytkownik, możemy określić, ile wysiłku będziemy potrzebować, aby wdrożyć tę historyjkę użytkownika.

Mały: Wynik tej oceny powinien być możliwy do zarządzania w sprincie.

Testowalne: istnieje sposób sprawdzenia, czy osiągnęliśmy oczekiwaną długość użytkownika dla tej historii użytkownika.

Historia dewelopera

Historia programisty jest powiązana z historią użytkownika i jest bardziej szczegółowym wyjaśnieniem w krótkich zdaniach, co należy zrobić, aby spełnić wymagania historii użytkownika. Jest to pierwszy krok szczegółowej analizy, której będziemy wymagać dla każdego elementu Backlogu Produktu, aby był on wystarczająco jasny, aby rozpocząć rozwój (aby włączyć go do Backlogu Sprintu). Zgodnie z nazwą, historia programisty jest dostarczana przez programistów i jest napisana językiem, który zarówno programiści, jak i Właściciel Produktu mogą w pełni zrozumieć. Wyjaśnimy to jaśniej na przykładzie naszej ukochanej firmy HSC. W tym przykładzie przeanalizujemy bardziej szczegółowo, jaka jest historia programisty dla historii użytkownika związanej z menedżerem HR w poprzednim przykładzie. Najpierw, podobnie jak w historii użytkownika, zdefiniujemy szablon historii dewelopera.

The < data module >

will < Action/feature >

that will allow < requirement >

related to the < related user story >

Za pomocą tego szablonu zdefiniujemy nasze historie programistów, aby wspierać historię użytkowników HR

Model HR otrzyma dane z systemu META4, które pozwolą na analizę dostępności danych HR firmy związanych z HR user story. Model HR będzie zawierał hierarchię pracowników które pozwolą drążyć

poziomy hierarchii HR związane z historią użytkownika HR. Model HR zapewni raport podzielony na strony według sklepu, który umożliwi analizę kosztów HR, szkoleń i urlopów na sklep w odniesieniu do historii użytkownika HR.

Również w tym przypadku fani metodologii, którzy lubią definiować akronimy, zdefiniowali jeden, aby sprawdzić, czy historia programisty jest wystarczająco dobra, aby rozpocząć rozwój, a jego nazwa to DILBERT'S:

Możliwy do wykazania: Po opracowaniu historii programisty powinniśmy być w stanie pokazać Właścicielowi Produktu, że działa ona zgodnie z oczekiwaniami.

Niezależne: podobnie jak w przypadku historii użytkownika, musimy zdefiniować historie programisty, które można opracować bez wpływu na inną historię programisty, a programowanie powinno zapewniać funkcjonalność działającą samodzielnie.

Warstwowa: Ta cecha jest bardzo skoncentrowana w rozwoju Business Intelligence, ponieważ większość zmian po stronie BI musi uwzględniać rozwój ETL, bazy danych i interfejsu użytkownika.

Wartość biznesowa: użytkownik musi docenić nasz rozwój , wynikającye z wartości user story.

Szacunkowa: ponownie wychodząc z charakterystyki historii użytkownika, historia programisty musi być wyraźnie możliwa do oszacowania. W tym przypadku różnica polega na tym, że należy to zrobić z dokładniejszym oszacowaniem, o ile deweloper pisze, co zamierza opracować.

Możliwość dopracowania: Historia programisty musi być napisana w bardziej konkretny sposób, niż historia użytkownika, ale znowu można ją przejrzeć i dostosować podczas opracowywania.

Testowalny i mały: te dwie cechy są dokładnie takie same jak w historii użytkownika.

Rejestr Produktu

Backlog produktu to lista wymagań na wysokim poziomie, bez szczegółowych specyfikacji kwalifikowanych według priorytetu użytkownika i kosztu rozwoju, które muszą być zdefiniowane na początku projektu i przeglądane po każdym sprincie, więc nie jest to stała lista; może ewoluować i zmieniać się w trakcie projektu. Właściciel produktu jest odpowiedzialny za tworzenie i aktualizowanie tej listy. W tym przypadku odwołujemy się do przykładu wyjaśnionego na konkretnych spotkaniach Scrumowych, aby zrozumieć, czym jest Backlog Produktu oraz w jakim momencie jest tworzony i utrzymywany.

Definicja Wykonania

Definicja wykonania to lista kontrolna, którą musi spełnić każdy element, aby został uznany za wystarczająco dobry, aby można go było przenieść do środowiska produkcyjnego. Jest to walidacja jakości, która ma na celu upewnienie się, że wszystkie wymagania zostały opracowane z zachowaniem pożądanych standardów jakości. Definicja wykonanej listy kontrolnej zawiera pozycje takie jak: opracowany kod odpowiada standardom kodyfikacyjnym, testy zostały pomyślnie zakończone, testy wydajnościowe zakończyły się sukcesem, uzyskaliśmy akceptację klienta, czy dokumentacja jest zakończona.

Kiedy zacząć - gotowa definicja

Ciekawą ideą dodaną do oryginalnej metodologii Scrum jest definicja gotowości lub definicja gotowości. Jest to analogia do Definicji Ukończenia, czyli kiedy nasz rozwój można uznać za zakończony, ale związana z definicją, którą Właściciel Produktu musi zrobić dla PBI, które chce opracować. Kiedy

definiujemy Backlog Produktu, PBI jest definiowane na bardzo wysokim poziomie, więc nie jest to coś, co można zacząć rozwijać. Aby być kandydatem do włączenia do rejestru sprintu, ten PBI musi być w pełni zdefiniowany i określony zgodnie z parametrami listy Definicja lub Gotowy. Niektóre przykłady listy kontroli, które mogą zawierać Definicja lub Gotowe, mogą obejmować pytania takie jak:

* Czy przedmiot może zostać ukończony w sprincie, który zamierzamy rozpocząć?

* Czy ten element został oszacowany przez zespół programistów?

* Czy ten przedmiot sam w sobie stanowi wartość dodaną? A może trzeba to zrobić z innymi pozycjami z listy?

Musimy uzgodnić z właścicielem produktu, jakie wymagania muszą spełniać definicję funkcjonalności, aby zostać zaakceptowanym do rozpoczęcia rozwoju.

Wykres wypalania Backlogu Produktu

Jest to wykres, który pokazuje na osi X liczbę sprintów, a na osi Y liczbę elementów Backlogu Produktu oczekujących na wdrożenie lub oczekiwany wysiłek włożony w ich opracowanie. Trend powinien być spadkowy, ale może ulec pewnemu wzrostowi z powodu pewnych zmian w PB, które obejmują nowe PBI, aby spełnić nową specyfikację zdefiniowaną przez Właściciela Produktu.

Backlog sprintu

Na początku sprintu wybieramy listę PBI, które zostaną opracowane w ramach sprintu. Ten wybór to backlog sprintu. Lista elementów do opracowania musi być podzielona na zadania, które powinny zostać opracowane w maksymalnym okresie 2 dni z zaleceniem, aby zadanie mieściło się w przedziale od 2 do 8 godzin prac rozwojowych. Rejestr sprintu nie powinien być modyfikowany, chyba że istnieje poważna blokada, która utrudnia ukończenie niektórych PBI. To, co musimy na bieżąco aktualizować, to informacje dotyczące każdego zadania, czy jest jakaś blokada, ile godzin pozostało do zakończenia rozwoju i kto ma przydzielone to zadanie. Również na tej liście powinniśmy unikać zbytniego dzielenia zadań, które nie wchodzą w zakres mikrozarządzania.

Wykres wypalania backlogu sprintu

Ten wykres liniowy jest oszacowaniem pozostałego wysiłku do ukończenia sprintu. Teoretycznie powinna to być linia malejąca, która przecina linię zerową na końcu sprintu. Ale o ile pozostały wysiłek jest weryfikowany każdego dnia, może się różnić, jeśli rozpoczynając zadanie wykryjemy, że nie zostało ono poprawnie oszacowane. Zwykle wyznaczamy dwie linie, jedną z teoretycznym trendem, a drugą z rzeczywistym postępem.

Lista przeszkód

Podczas codziennych spotkań wykryjemy przeszkody lub blokady, które mogą zagrozić realizacji danego zadania lub pełnego PBI. Mistrz Scrum musi przechowywać te blokady na liście i śledzić wszystkie tematy, aby móc je odblokować. Na tej liście musimy poinformować o blokadzie, powiązanym PBI, statusie blokady, planowanej dacie rozwiązania i osobie odpowiedzialnej za jej rozwiązanie.

Lista ulepszeń

Po każdym sprincie dokonamy przeglądu naszej aktywności na spotkaniu retrospektywnym. Tam cały zespół powinien zasugerować wszelkie ulepszenia, które jego zdaniem są istotne dla wyników zespołu.

Ta lista jest również prowadzona przez Scrum Mastera i powinna zawierać element akcji, status, zaplanowaną datę zakończenia oraz osobę odpowiedzialną za jego rozwiązanie.

Ograniczanie czasu spędzonego za pomocą Timeboxingu

Czas w naszym życiu jest ograniczony. Jest również ograniczona w naszej pracy, a także w naszych projektach. Musimy ograniczyć czas, który poświęcamy na każde zadanie, aby nie tracić zbyt wiele czasu poświęconego na to samo zadanie. Praktyka Timebox określa granice każdego rodzaju zadań, aby zapewnić, że realizacja projektu nie zawiesza się. Ta technika timeboxingu jest stosowana na wszystkich poziomach, więc staramy się określić, ile czasu ma zająć cały projekt, ile czasu potrzebujemy na wydanie wydania, czyli nasz okres sprintu, a następnie ustalamy, ile czasu musimy spędzić w jednym zadania takie jak przyjmowanie wymagań, opracowywanie kodu, czyli każdy rodzaj czasu trwania spotkania, opracowywanie dokumentacji itp. Ta metodologia jest wysoce oparta na czasie, ponieważ wszystkie działania mają ograniczenia czasowe. Za tą strategią stoi teoria równoważenia trzech elementów: wysiłku, wymagań i harmonogramu. Zawsze możemy ulepszyć dwa z nich, równoważąc drugi. Możemy użyć mniej wysiłku, jeśli zmniejszymy wymagania lub zwiększymy harmonogram. Jeśli chcesz mieć więcej wymagań, możemy to zrobić, poświęcając więcej czasu lub zasobów. Jeśli chcesz skrócić harmonogram, możemy to zrobić, dodając zasoby lub zmniejszając wymagania. To, co Scrum proponuje w ramach tego ograniczenia czasowego, to próba ustalenia jednego z parametrów i maksymalizacji pozostałych w oparciu o priorytety klienta. Ustalamy, że w sprincie spędzimy 4 tygodnie. Jeżeli mamy 5 osób oddelegowanych do rozwoju to możemy zainwestować ustaloną ilość godzin. Dlatego nasz klient musi nadać priorytet zadaniom, które chce mieć dostępne, a my skupimy się na nich. Zwiększymy satysfakcję klienta, bo pomimo braku niektórych funkcjonalności, nasz klient może zacząć walidować i wykorzystywać najważniejsze dla niego zadania. W ten sam sposób ustalamy upływający czas na spotkania. Musimy zacząć od najważniejszych tematów do omówienia i nie spędzać na spotkaniu więcej czasu niż jest to wymagane. Kiedy szukamy informacji do wdrożenia nowej funkcjonalności, robimy dokumentację lub cokolwiek innego, mamy na to maksymalną ilość czasu i musimy zrobić wszystko, co w naszej mocy, aby rozwinąć zadanie. Ideą takiego podejścia jest to, że musimy uzyskać wyniki wystarczająco dobre, aby były zgodne ze standardami jakości, ale nie doskonałe. Wszystko, co doskonałe, kosztowałoby zbyt wiele czasu i wysiłku, więc byłoby sprzeczne z naszymi celami szybkiego dostarczania rzeczy.

Spike

Czasami jest jakieś zadanie, które trzeba wykonać, ale nie dodaje bezpośredniej wartości dla klienta. Opierając się na standardach Scrum, wszystkie nasze działania lub PBI same w sobie muszą zapewniać wartość dodaną, ale tworzenie oprogramowania, a w tym przypadku rozwój BI, może wymagać zrobienia czegoś, na czym tak naprawdę nie zależy naszym klientom. Aby móc włączyć je do Backlogu Produktu/Backlogu Sprintu, Scrum umożliwia zarządzanie nim przez koncepcję Spike. Pokażmy przykład. Wyobraź sobie, że użytkownik poprosił Cię o możliwość drążenia z jednego raportu do innego z bardziej szczegółowym filtrowaniem dynamicznym wybranego przez siebie klienta. Opracowałeś rozwiązanie BI przy użyciu narzędzia BI, które może wykonać tego rodzaju drążenie, ale nie w obecnej wersji, którą zainstalowałeś na swojej platformie, ale w nowej. Aktualizacja platformy do nowej wersji może być uznana za skok, jako coś, co jest wymagane, ale użytkownik nie prosi o to bezpośrednio. Innym przykładem może być sytuacja, w której w celu spełnienia danego wymagania zespół musi użyć funkcji, z której nigdy wcześniej nie korzystał. Jeśli chcesz odnieść sukces w spełnieniu tego wymogu, możliwe, że Twój zespół wymaga przeszkolenia w zakresie tej funkcji lub weryfikacji koncepcji, aby mieć pewność, że wiedzą, jak zastosować ją do potrzeb klienta.

Konserwacja z Kanbanem

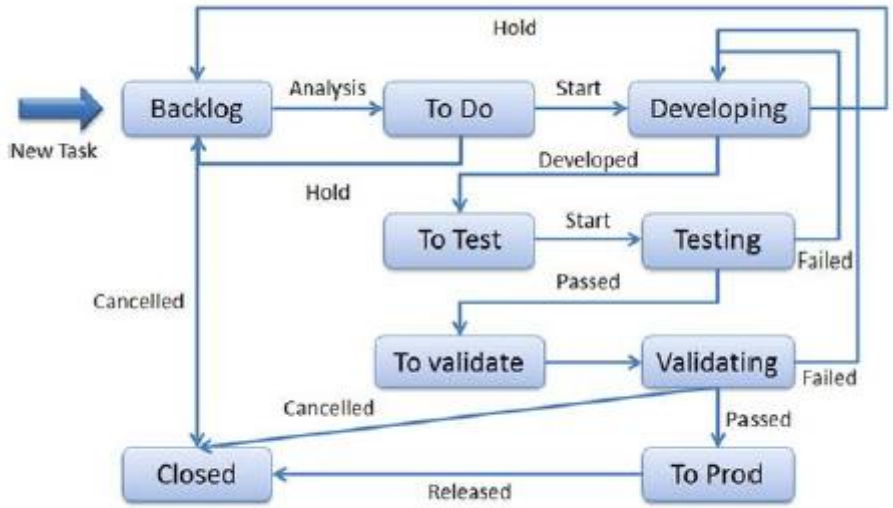
Po opracowaniu głównych funkcjonalności wymaganych przez klientów można śmiało powiedzieć, że Twój system BI wymaga konserwacji, poprawek, drobnych usprawnień i administracji. Aby przeprowadzić tę konserwację, naszym preferowanym podejściem z metodologii Agile jest Kanban, który pozwala na utrzymanie aplikacji z podejściem ciągłego doskonalenia, koncentrując się na małych zmianach w rozwiązaniu, aby dostosować je do nadchodzących wymagań.

Koncepcje Kanbana

Istnieje kilka interesujących koncepcji z teorii Kanbana, które musisz znać, aby zwinnie zarządzać swoim projektem.

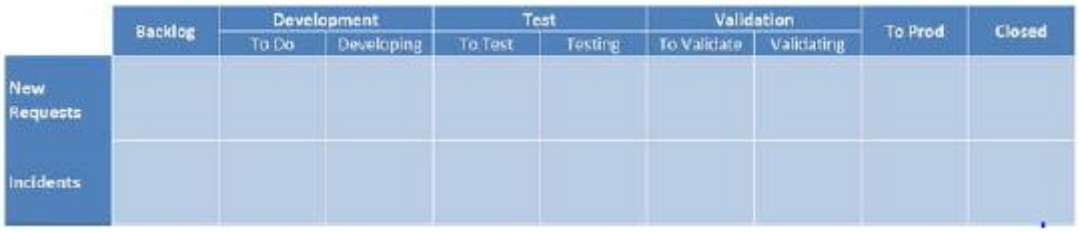
Przepływ pracy zadania

Kanban definiuje przepływ pracy zadania, definiując różne statusy, jakie zadanie powinno przyjąć, od początkowego zaległości do rozwoju, testowania, walidacji i zakończenia. Ta lista statusów może się różnić w zależności od naszego środowiska i być może w zależności od typu zadania, ale ważne jest, aby były one poprawnie zdefiniowane, aby zmaksymalizować wydajność zespołu. Na rysunku możesz znaleźć przykład przepływu pracy dla Kanbana.



Tablica zadań i zarządzanie wizualne

Aby mieć kontrolę nad wszystkimi zadaniami, wykrywać wąskie gardła, wykrywać blokady i łatwo komunikować się wewnętrznie z zespołem, czyli statusem wszystkich zadań i postępem, jaki zespół wykonał na co dzień, ważne jest, aby mieć tablica, która zawiera wszystkie zadania wewnątrz, używając karty do każdego zadania i przesuwając ją do przodu, o ile zadanie przechodzi przez inny status zdefiniowane w przepływie pracy. Tablica będzie zorganizowana z kolumną dla każdego statusu i jednym lub kilkoma wierszami w zależności od typów zadań, priorytetów, incydentów vs. próśb lub jakiejkolwiek potrzeby, którą mamy w naszym projekcie. Rysunek przedstawia przykład tablicy zadań Kanban.



Zarządzanie wizualne jest bardzo ważną cechą Kanbana. Ważne jest, aby mieć fizyczną tablicę, a przynajmniej projektor lub ekran, który jest stale widoczny, aby cały zespół mógł w każdej chwili zobaczyć, w jakim jesteśmy stanie i jakie dalsze kroki są wymagane. Jako metodologia Agile, Kanban udaje, że codziennie odbywa 15-minutowe spotkanie z całym zespołem, aby skomentować status zadania i jakie są zmiany statusu każdego zadania.

Praca w toku

Jednym z najważniejszych parametrów w naszym systemie Kanban będzie Work In Progress czyli WIP. Dla każdego statusu ustawimy WIP, określając maksymalną liczbę zadań, które możemy mieć w określonym statusie. Możliwe, że nie wszystkie kolumny wymagają WIP i trzeba go dostosować do projektu, nad którym pracujemy; posiadanie wysokiego WIP spowoduje, że będziemy przeciążeni, dlatego zaczynamy od wielozadaniowości; a jedną z idei rozwoju Agile jest to, że musimy skoncentrować się na pojedynczym zadaniu w danym momencie, aby być bardziej wydajnym w naszym rozwoju. Moglibyśmy również cierpieć z powodu długich czasów oczekiwania: długich kolejek w statusie pośrednim i utraty jakości. Jeśli zdefiniujemy niski WIP, być może mamy osoby, które nie mają nic przypisanego.

Czas realizacji

W systemie Kanban czas realizacji to ilość czasu, jaką potrzebujemy na opracowanie zadania, ponieważ przesuwamy je do stanu początkowego, aż do jego zamknięcia. Nie uwzględnia czasu zaległości. Mierząc średni czas realizacji, będziemy mogli zobaczyć, jak skuteczna jest strategia Kanban. Chodzi o to, że zadania nie pozostają w statusie w nieskończoność, więc jeśli wykryjemy, że kolumna ma więcej zadań niż powinna, musimy przenieść zasoby, aby pomóc tym, które mają blokady, aby uniknąć wąskich gardeł.

Uwaga: Bardzo ważne jest, aby zauważyć, że Kanban ustala pewne podstawowe zasady, ale musi być dostosowany do każdego środowiska, kiedy próbujemy skonfigurować system, oraz że musi być elastyczny i stale dostosowywany, aby poprawić wydajność zespołu. Będziemy musieli dostroić każdy WIP o każdym statusie, zdefiniować różne tablice zadań Kanban dla różnych typów działań, dostosować status, jaki ma dany typ działania, a wszystkie te adaptacje powinny być dokonywane za pomocą okresowych spotkań retrospektywnych z całym zespołem, który pozwoli nam udoskonalić sposób, w jaki pracujemy.

Mieszanka obu metodologii, Scrumban

Nasze początkowe rozważania dotyczące mieszania pewnych cech obu metodologii wcale nie są czymś, co wymyśliliśmy; przeszukując teorię Agile, można znaleźć metodologię Scrumban. Scrumban stosuje obie metody, ale jest mniej rygorystyczny niż Scrum, który udaje, że ma uniwersalnie wymiennych członków zespołu, więc dopuszcza pewną wyspecjalizowaną rolę, która, jak zobaczysz w następnej sekcji, lepiej pasuje do rozwoju BI; i jest trochę bardziej zorganizowany niż Kanban, używając iteracji do zorganizowania pracy, jak w Scrumie. Scrumban pasuje lepiej niż Scrum w projektach o dużej zmienności, a także w projektach utrzymaniowych lub projektach, w których możemy spodziewać się

wielu błędów ze względu na zmienność informacji źródłowych: jak na przykład projekt oparty na narzędziach Big Data. Jak wspomniano we wstępie do tego rozdziału, nie pretenduje do miana wyczerpującej analizy metodologii Agile, dlatego zachęcamy do ich zbadania; będziesz mógł znaleźć setki stron w Internecie z informacjami na ich temat. Rozważamy bardziej interesujące skupienia na osobliwościach Agile dla BI, co, mam nadzieję, zobaczysz, jeśli będziesz kontynuować czytanie.

Specyfika Scruma dla BI

Zasady i metodologie omówione powyżej są ogólne dla SCRUM w ramach każdego procesu tworzenia oprogramowania, ale istnieją pewne cechy szczególne, które mają wpływ na projekty BI, które są specyficzne dla tego środowiska; łatwo zrozumiesz, co mamy na myśli, przechodząc do tej sekcji.

Sprint 0 - Dłuższa analiza początkowa

Ok, podjęliśmy decyzję o rozpoczęciu rozwoju naszej hurtowni danych, mamy zgodę najwyższego kierownictwa i chcemy zastosować metodologię Scrum. W tym celu zbierzemy wszystkie historie użytkowników; stworzymy nasz Product Backlog, szacując na bardzo wysokim poziomie potrzeby i koszt każdego zadania; niektóre z nich wybierzemy na podstawie priorytetu Właściciela Produktu; uzupełnimy Backlog Sprintu, dokładniej analizując potrzeby każdego PBI; i zaczniemy się rozwijać. Po czterech sprintach dochodzimy do analizy jednego PBI o niskim priorytecie, który był na liście od początku i miał niski priorytet, ale duży wpływ, który spowoduje przebudowę całej wykonanej do tej pory pracy. Dzieje się tak, ponieważ wdrażając bazę danych hurtowni danych, należy zdefiniować model logiczny oparty na jednostkach, które muszą mieć jasno określone relacje; które pojęcia lub atrybuty mają relacje; jakie są rodzaje relacji: relacje jeden-do-wielu, wiele-do-wielu; który jest pojedynczym kluczem do powiązania tabel; a na koniec musimy zdefiniować klucze, aby móc uruchamiać poprawne instrukcje SQL, które powinny zwracać spójne informacje. Musisz również wziąć pod uwagę, że spodziewasz się tam zapisanej długiej historii, dużych tabel z milionami wierszy informacji, które są sprawdzane, formatowane, przekształcane lub cokolwiek innego, czego wymaga Twój proces, aby spełnić Twoje potrzeby. W tej sytuacji za każdym razem, gdy przenosisz zmianę do produkcji, może wymagać przetworzenia dużej ilości informacji, jeśli nie masz solidnego rozwiązania dla swojego schematu tabeli. Tak więc jedną z głównych cech charakterystycznych zastosowania Scruma w projekcie Business Intelligence jest konieczność wcześniejszego myślenia i głębszej analizy implikacji historyjki użytkownika w rozwiązaniu technicznym, które próbujesz dostarczyć. Ponownie, aby zilustrować, jaki może być problem nieprzestrzegania tych zaleceń, pokażmy przykład z firmy HSC. Jest to przykład pochodny od rzeczywistego, którym musieliśmy zarządzać w naszym doświadczeniu w innej firmie, z tą różnicą, że w tamtym momencie nie używaliśmy Scruma. W naszych przykładowych historiach użytkowników znajdujemy cztery różne prośby o dodanie nowych danych do systemu, ale wyobraźmy sobie, że masz ich 40 lub 50. Podążamy za priorytetem zdefiniowanym przez użytkownika i zaczynamy wdrażać pierwsze:

Jako właściciel produktu chcę mieć możliwość drążenia hierarchii klientów, aby użytkownicy mogli analizować szczegóły sprzedaży poprzez organizację klientów ERP.

Moglibyśmy pomyśleć o prostym modelu logicznym obsługującym to wymaganie w oparciu o schemat gwiaździsty z zaledwie czterema tabelami: fakty dotyczące sprzedaży oraz wymiary klienta, produktu i czasu. Podobnie jak w modelu tabelarycznym, powiązane wymiary w narzędziu BI mają swój odpowiedni identyfikator, który określa klucz do łączenia między tabelami. Podążamy za innymi historiami użytkowników, które dodają poziomy do każdej hierarchii; istnieje inna historyjka użytkownika, która wymaga poprawy szybkości, dlatego do naszego modelu dodajemy zagregowane tabele, nowe wymiary analizy oraz coraz więcej tabel i atrybutów. Po kilku sprintach postanawiamy uwzględnić w naszej analizie tę historyjkę użytkownika:

Jako Sales Force Manager chcę analizować wyniki sprzedaży w poszczególnych sklepach, aby móc analizować sprzedaż według kategorii, grup klientów i kampanii promocyjnych.

Nasze początkowe oszacowanie na wysokim poziomie było, ok, przejdźmy do dodania sklepu do tabeli faktów dotyczących sprzedaży i dodajmy tabelę faktów promocyjnych z wyszukiwaniem sklepów. Ale potem zaczynamy szczegółową analizę:

* Sklepy nie mają tej samej liczby produktów; to zależy od wielkości sklepu i lokalizacji. Z tego powodu kategoria niektórych produktów może się różnić w zależności od sklepu, ponieważ niektóre narzędzia są uważane za Ogród w przypadku dużego sklepu z naciskiem na rolnictwo, a za Inne narzędzia w innym sklepie w centrum miasta. Więc kategoria zależy od sklepu. Aby rozwiązać ten problem, musimy dodać sklep do wyszukiwania produktów i ponownie przetworzyć cały wymiar.

* Grupy klientów są również definiowane przez sklep na podstawie rodzaju klientów mieszkających w pobliżu sklepu. Również kod klienta może się powtarzać wzdłuż sklepów, więc aby mieć unikalny identyfikator klienta, musimy dodać sklep do wyszukiwania klientów i ponownie przetworzyć cały wymiar.

* Myślisz na czas … czas nie może się zmieniać. Relacje w hierarchii czasowej są takie same dla wszystkich sklepów… Znowu się mylisz. Aby przeanalizować liczbę dni, które mamy na daną kampanię, musimy policzyć dni robocze. Ponieważ sklepy znajdują się w różnych miastach i regionach, mają różne lokalne święta. Więc znowu musimy dodać sklep do wymiaru czasu i ponownie przetworzyć cały wymiar czasu.

* Twój model logiczny również wymaga przeglądu, aby uwzględnić magazyn terenowy w definicji klucza dla klienta, grupy klientów, kategorii produktu, dnia, typu i wielu innych koncepcji utworzonych do analizy.

Uwaga: Podsumowując tę sekcję, będziemy musieli poświęcić więcej czasu na szczegółowe przeanalizowanie, które historie użytkowników mogą wymagać dużych modyfikacji w naszym modelu. Na koniec dnia powinniśmy postarać się o jak najpełniejszą definicję schematu przed rozpoczęciem programowania. Na pewno pojawią się wymagania, które będą wymagały dostosowań i zmian, ale powinniśmy starać się zapobiegać im w jak największym stopniu.

Segmentacja projektów BI

Po spotkaniach Sprint 0 będziesz mieć wiele próśb do rozważenia od różnych interesariuszy, dla różnych działów, przy użyciu różnych narzędzi do programowania na różnych warstwach BI przez różne role programistów lub przynajmniej kogoś z różnymi obszarami wiedzy programistycznej. Tak więc pod koniec oceny wymagań przeprowadzonej na tych spotkaniach Sprintu 0 prawdopodobnie będziesz miał dużo pracy do wykonania dla wielu zespołów, które o to proszą. W tym momencie masz dwie możliwości: uciec daleko i zostać ogrodnikiem w małym miasteczku lub spróbować zacząć od tego niesamowicie dużego nakładu pracy. Jeśli już doszedłeś do tego miejsca w książce, jestem prawie pewien, że wybierzesz drugą opcję. Jak więc zacząć? Jak omówiono w poprzedniej sekcji, powinieneś zainwestować dużo czasu w analizę wszystkich przychodzących historyjek użytkowników. Z tej analizy powinieneś otrzymać schemat logiczny bazy danych, szkic raportów, które użytkownicy chcą otrzymywać z Twojego systemu BI, a które powinny być główną funkcjonalnością Twojej platformy. Dzięki temu przeglądowi wszystkich oczekujących historii użytkowników do wdrożenia powinieneś być w stanie podzielić je na mniejsze powiązane grupy, aby móc zapewnić działające rozwiązanie po

każdym sprincie. Aby poprawnie zdecydować, jak pogrupować wszystkie powiązane zadania, należy przeanalizować wymagania pod trzema różnymi punktami widzenia:

Obszar funkcjonalny : Musisz podzielić zadania na mniejsze grupy powiązane powiązanym obszarem funkcjonalnym, do którego należą. Możesz więc sklasyfikować oczekujące zmiany jako związane z finansami, związane ze sprzedażą, związane z operacjami, a także komponenty obejmujące różne środowiska, takie jak dane klientów lub dane produktów. W celu przeprowadzenia prawidłowej analizy obszarów funkcjonalnych pomocnym narzędziem będzie analiza modelu logicznego.

Analiza techniczna: mając przegląd wszystkich wymaganych pól do analizy informacji, możesz stwierdzić, że istnieją początkowo niepowiązane historie użytkowników, które mają to samo techniczne źródło i/lub cel. Wyobraź sobie, że Twój kierownik ds. sprzedaży poprosił o wyświetlenie informacji o produktach sklasyfikowanych według rodziny produktów, a Twój kierownik finansowy chce zobaczyć, jaką wagę dajesz swoim klientom na promocję danego produktu. Oba pola informacyjne wydają się być całkowicie niezależne, ale możesz zdać sobie sprawę, że oba pochodzą z tej samej tabeli w Twoim systemie ERP i zamierzasz umieścić je w tej samej tabeli w miejscu docelowym, więc połączenie obu historyjek użytkownika obniży całkowity koszt co do kosztu wykonania ich osobno. Aby prawidłowo przeprowadzić analizę techniczną należy skupić się na fizycznym modelu systemu.

Doświadczenie użytkownika końcowego: w narzędziu BI możesz mieć złożone raporty, które mieszają informacje z różnych obszarów funkcjonalnych, dane pochodzące z różnych systemów i umieszczone w naszym magazynie danych w różnych celach. Ale być może Twój kluczowy użytkownik chce mieć wszystkie te informacje razem w jednym raporcie, więc aby dostarczyć w pełni działający ekran, możesz zdecydować, że powinieneś opracować w tym samym czasie osobną analizę, aby móc zapewnić gotowy rozwój.

Po tej drugiej analizie, aby pogrupować wszystkie zadania, możesz zobaczyć, że niektóre z tych podzielonych na segmenty historii użytkowników są zbyt duże, aby można je było uwzględnić w sprincie, a także powiązane zadania mają wycenę nadmiernego nakładu pracy, aby zmieścić się w oczekiwanym maksymalnym rozmiarze 8 godzin pracy. Aby móc podążać za metodologią Agile, będziesz musiał podzielić segmenty na mniejsze części. Aby to zrobić, możesz powtórzyć analizę funkcjonalną, ale skupiając się na wybraniu z modelu logicznego minimum części, które mężczyzna ma ochotę wspólnie opracować, aby uzyskać mniej tabel i pól do zarządzania, a także możesz uzgodnić z właścicielem produktu, że niektóre części raportu będą wyświetlane z komunikatem „W trakcie opracowywania” aż do następnego sprintu. Ostatecznie byłby to sposób na segmentację według tabel, o ile wybierasz niektóre tabele, które zostaną dodane do modelu, wraz z powiązanym rozwojem BI i ETL. Możesz także podzielić cały projekt, dzieląc załadowane informacje na segmenty. Możesz załadować informacje dla pewnego okna czasowego (bieżący rok, bieżący miesiąc) i pozostawić dane historyczne dla następnego sprintu lub możesz załadować informacje kraj po kraju w środowisku wielonarodowym lub firma po firmie w klastrze firm lub klient po kliencie lub, w przykładzie, który śledziliśmy przez cały rozdział, sklep po sklepie. Jeśli te podziały nie wystarczą, możesz pomyśleć o wybraniu tylko niektórych kolumn swoich tabel, specjalnie pogrupowaniu ich według tabel źródłowych lub środowisk. A jeśli powoduje to, że zadania są nadal zbyt długie, możesz pomyśleć o podzieleniu ich według typu, więc najpierw możesz załadować metryki podstawowe, następnie zaimplementować pochodne z tych metryk, a następnie przejść do agregacji itp.

Działania front-end vs. back-end

Inną cechą charakterystyczną projektów BI są różnice w rozwoju działań frontendowych i backendowych. Aby móc wyświetlać określone informacje w narzędziu front-end BI, konieczne będzie

załadowanie informacji do bazy danych back-end. Z drugiej strony ładowanie informacji do bazy danych, ale bez żadnego raportu front-end do ich analizy, nie ma sensu w rozwiązaniu BI. Tak więc działania front-end i back-end są ze sobą ściśle powiązane; w rzeczywistości większość historyjek użytkownika będzie miała historie deweloperskie z obu typów, o ile wymagane są obie czynności, ale mają one zupełnie inny charakter. Działania backendowe to praca w cieniu, brudna robota, której nikt bezpośrednio nie widzi, podczas gdy raporty frontendowe będą dostępne dla wszystkich, narzędzie BI będzie interfejsem użytkownika umożliwiającym dostęp do informacji na backendzie. Z tego powodu wymagania będą zupełnie inne. Narzędzie front-end jest używane bezpośrednio przez klientów końcowych, więc będziesz musiał wziąć to pod uwagę, aby zdefiniować główne zasady. Nie możesz mieć bardzo ścisłych zasad nomenklatury i konwencji nazewnictwa w interfejsie użytkownika, ponieważ Twoi użytkownicy zwykle nie mają wiedzy technicznej, a nazwy raportów, opisy i obrazy muszą być zorientowane na użytkownika, aby umożliwić użytkownikowi łatwe poruszanie się po narzędziu. Twój interfejs użytkownika powinien być intuicyjny i łatwy w użyciu; będzie to ważniejsze niż rozwiązanie techniczne i możliwości, które oferujesz. Zamiast tego komponent zaplecza ma większe wymagania techniczne, więc możesz ustawić konwencję nazewnictwa, która zmusza programistów do przestrzegania nomenklatury w obiektach bazy danych, obiektach ETL, procedurach programistycznych, stosować najlepsze praktyki dostrajania wydajności itp. W tej części będziesz miał silniejszą pozycję, aby zmusić programistów do przestrzegania reguł, a kilku programistów do ich nauki, również z umiejętnościami technicznymi, które ułatwi ten cel. Na froncie Twoi klienci mają moc decyzyjną, więc musisz dostosować się do ich wymagań. Będziesz miał wielu użytkowników, być może kontaktują się z Tobą przez jakiegoś kluczowego użytkownika, na przykład właściciela produktu, ale na końcu będziesz potrzebować rozwiązania zrozumiałego dla wielu różnych osób, o różnych poziomach wiedzy i z różnych funkcjonalnych światów, podczas gdy dostęp do zaplecza powinni mieć tylko ludzie z głębszą wiedzą na temat tego, co robią.

Izolacja roli - Specyficzna wiedza

Zwinne tworzenie oprogramowania udaje, że ma w pełni elastyczne zespoły, w których wszystkie komponenty zespołu mają wiedzę programistyczną, a historie programistów może wymyślać każdy z nich, ale jak można się domyślić, na podstawie poprzedniej sekcji, narzędzia programistyczne i umiejętności dla dowolny komponent platformy BI są zupełnie inne. Będziesz potrzebować umiejętności technicznych do tworzenia struktur baz danych i procesów ETL, z myśleniem matematycznym, zdolnym do definiowania struktur, relacji, procedur ładowania i zrozumienia modelu relacyjnego, gdy mówisz o narzędziach zaplecza, znajomości definicji schematu w celu zdefiniowania BI narzędzia, modele, umiejętności projektowania funkcjonalnego i graficznego w celu zdefiniowania raportów końcowych, a znalezienie kogoś, kto ma wszystkie te różne umiejętności, jest trudne i kosztowne, więc prawdopodobnie będziesz potrzebować różnych podzespołów w swoim zespole skupionych na różnych obszarach rozwoju. Tak więc członkowie Twojego zespołu nie będą w pełni wymienni, aby pracować w różnych historiach użytkowników, co zwiększy złożoność zarządzania zespołem. Aby móc ruszyć z całym projektem będziesz potrzebować w swoim zespole przynajmniej kogoś, kto posiada którąś z poniższych umiejętności:

* Architekt danych: odpowiedzialny za zdefiniowanie modelu fizycznego, struktur tabel, typów pól, modelu bazy danych itp.

* Programista bazy danych: ta rola będzie rozwijać wymagane struktury zdefiniowane przez architekta danych wewnątrz bazy danych.

* Deweloper ETL: będzie odpowiedzialny za zdefiniowanie procesów ładowania z systemów źródłowych do docelowej bazy danych.

* Data modeler: ta rola będzie wymagać wiedzy o tym, jak zdefiniować i zaimplementować w narzędziu BI model logiczny, który umożliwi narzędziu BI wygenerowanie poprawnego kodu SQL do uruchomienia w bazie danych. Te zadania ról będą się różnić w zależności od wybranego narzędzia; istnieją złożone narzędzia BI, które pozwalają tworzyć złożone modele z setkami funkcjonalności oraz prostsze narzędzia, które nie będą wymagały dużych możliwości modelarza danych.

* Programista front-end: Będziesz także potrzebował w swoim zespole kogoś z wiedzą funkcjonalną, aby zrozumieć potrzeby klienta i powielić je w systemie BI. Ta rola będzie wymagała również dobrych umiejętności komunikacyjnych, o ile będzie głównym interaktorem z klientem końcowym.

* Projektant graficzny: W tym przypadku posiadanie grafika w zespole nie jest obowiązkowe, ale ponieważ możliwości raportowania stają się coraz bardziej atrakcyjne wizualnie, zaleca się, aby w zespole był ktoś z umiejętnościami projektowania graficznego, który może współpracować w definicji raportów, do których użytkownik będzie miał dostęp.

Te różne role i umiejętności mogą utrudniać wymianę zespołu programistów z ról, które mają, jeśli chcesz mieć zespoły multidyscyplinarne, będziesz musiał zainwestować w szkolenie zespołu, co z drugiej strony jest zawsze godne polecenia. Zdajemy sobie również sprawę, że jeśli pracujesz w małej firmie próbującej wdrożyć rozwiązania BI, możliwe, że będziesz jedyną osobą odpowiedzialną za robienie wszystkich rzeczy, więc raczej nie zalecamy kontynuacji czytania tej książki o ile postaramy się dać ci podstawę, kontynuuj rozwój wewnątrz wszystkich komponentów BI. Z pewnością przedstawimy przegląd typowych typów obiektów, ale będziesz musiał uzyskać głębszą wiedzę na temat narzędzi, których będziesz używać do każdego komponentu. Możesz znaleźć kilka książek poświęconych w całości każdemu dostawcy oprogramowania, więc w tej książce nie możemy pokazać wszystkich opcji dla wszystkich różnych narzędzi, byłoby to prawie niemożliwe do osiągnięcia.

Typy historii programistów w BI

Ta sekcja jest ściśle powiązana z poprzednią sekcją, ponieważ będziesz potrzebować różnych ról w swoim zespole, aby móc kontynuować opracowywanie różnych historii programistów. Określamy tutaj historie deweloperów, a nie historie użytkowników, ponieważ na koniec dnia użytkownik będzie chciał mieć narzędzie dostępne w narzędziu front-endowym i aby móc zapewnić użytkownikowi tę funkcjonalność, najczęściej będziesz musiał podzielić tę historię użytkownika na wiele historii programistów dotyczące modelowania danych, ETL, schematu narzędzia BI oraz raportów lub pulpitów nawigacyjnych narzędzia BI. Nie wszystkie historie użytkowników będą miały wszystkie komponenty; być może żądanie użytkownika wykonuje historyczne obciążenie dla istniejącej analizy, a Twoim zadaniem będzie ponowne uruchomienie procesów ETL i walidacja wyników, bez żadnych modyfikacji narzędzii BI; być może masz jakieś żądanie, które wymaga tylko modyfikacji istniejącej relacji między wymiarami danych, która dotyczy tylko twojego schematu BI, lub zmiany typu kolumny w bazie danych, aby móc pomieścić dłuższe opisy dla istniejącego pojęcia. Ostatecznie celem wszystkich tych modyfikacji będzie pokazanie wyniku w raporcie BI, ale sama modyfikacja być może nie wymaga tworzenia żadnego raportu.

Historie deweloperów modelowania danych

W przypadku historii programistów zajmujących się modelowaniem danych będziesz musiał skoncentrować się na rozwoju bazy danych. Oznacza to, że na pewno będziesz potrzebować umiejętności bazodanowych i ewentualnie architekta danych. Historie modelowania danych określą, jaka jest pożądana struktura tabeli, i na tej podstawie programiści będą pracować z narzędziami do projektowania baz danych, aby stworzyć całą wymaganą infrastrukturę. Aby zweryfikować proces modelowania danych, najpierw przetestujemy za pomocą zapytań, czy oczekiwane sprzężenia

zwracają poprawne dane, sprawdzimy integralność danych w systemie i musimy upewnić się, że każda kolumna została zdefiniowana z najbardziej odpowiednim typem kolumny i rozmiar. Musimy również wziąć pod uwagę, w jaki sposób zmiana może zostać zastosowana do istniejącego środowiska produkcyjnego, na przykład, jeśli nasz transport obejmuje zmianę typu kolumny w istniejącej tabeli, będziemy musieli sprawdzić, czy potrzebujemy tabeli pomocniczej, aby zachować istniejącą Informacja; lub jeśli chcemy utworzyć nową tabelę w produkcji, musimy sprawdzić, czy musi ona zostać utworzona z informacjami o środowiskach deweloperskich lub testowych użytkowników, czy też musi być utworzona pusta, aby wypełnić ją powiązanym procesem ETL.

Historie programistów ETL

Historie programistyczne ETL opierają się na sposobie, w jaki zamierzasz uzupełnić informacje do bazy danych, więc w analizie, aby móc je uruchomić, będziesz potrzebować wiedzy, jakie jest źródło informacji, jak możesz połączyć się z tym źródłem informacji, jaka jest oczekiwana częstotliwość ładowania, które tabele docelowe należy wypełnić, w jaki sposób zdefiniowano obciążenie, przyrostowe lub całkowite, oraz jakie są wymagane kontrole w celu sprawdzenia, czy informacje zostały załadowane poprawnie. Ponownie musimy potwierdzić, że nasz rozwój jest prawidłowy. Aby zweryfikować historie programistów ETL, będziemy wymagać sprawdzenia, czy informacje w źródle są takie same jak w celu, czy nasz proces nie powiela informacji, czy nie tracimy informacji na żadnym etapie ładowania, czy wydajność ładunku i czas ładowania są prawidłowe, że ustawiliśmy ten ładunek we właściwej pozycji w stosunku do pozostałych ładunków, a jeśli takie istnieją, że nasz rozwój wpisuje się w definicję najlepszej praktyki dla ETL w naszej Spółce.

Historie deweloperów modeli BI

W scenariuszu deweloperskim mającym na celu modyfikację modelu BI zostanie zdefiniowany zestaw obiektów narzędzi BI, które będą musiały zostać zaimplementowane, aby mogły być używane w interfejsie raportowania. Typy obiektów różnią się znacznie w różnych narzędziach; istnieją narzędzia, które wymagają dużych nakładów na modelowanie i które są zwykle bardziej niezawodne, przeznaczone do dużych wdrożeń, a inne narzędzia są łatwiejsze w zarządzaniu i mają niewiele wymagań dotyczących modelowania. Tak więc w zależności od narzędzia zdecydujesz, czy wysiłek związany z opowiadaniami modelu BI będzie duży, czy mały. Kiedy modyfikujemy definicję modelu BI, musimy upewnić się, że modyfikacja, której dokonujemy, nie spowoduje niepożądanego efektu w istniejącej strukturze. Musimy więc zweryfikować zmiany, które wprowadzamy, ale także musimy mieć zestaw ogólnych walidacji zdefiniowanych w narzędziu BI, aby upewnić się, że nie zmienią się one, gdy zmodyfikujemy istniejący model. Oczekiwanym rezultatem opracowania modelu BI jest to, że kiedy użyjesz tych obiektów w narzędziu, wygenerują one oczekiwane zapytanie SQL, więc będziesz musiał sprawdzić, czy korzysta z oczekiwanych pól, łącząc tabele przy użyciu odpowiednich pól kluczowych, bez niepożądane użycie stołu.

Historie deweloperów raportów i pulpitów nawigacyjnych

Jako najnowszy rodzaj deweloperskich historii standardowego systemu BI, chcielibyśmy omówić ostatni krok, czyli raporty i pulpity nawigacyjne, z którymi użytkownik będzie miał do czynienia. Aby móc opracować raport, będziemy musieli wiedzieć, jakie informacje nasz użytkownik chce zobaczyć, w jakiej kolejności, w jakim formacie, czy będzie jakiś obraz, ścieżki wiercenia, informacje firmowe, czyli szczegółowość wymaganych danych, czy istnieje jakieś pole grupujące, jakie są filtry, które mają zastosowanie do raportu, czy istnieje jakikolwiek selektor do interakcji z informacjami, ile paneli informacji ma zostać wyświetlonych oraz wszelkie inne szczegóły, które mogą być istotne dla raportu . Ponownie będzie to zależeć od narzędzia, którego zamierzasz użyć i jakie możliwości ci ono oferuje. Będzie to ta część, która zostanie bezpośrednio zweryfikowana przez właściciela produktu i klienta, ale

będzie miała niejawną weryfikację pozostałych komponentów rozwiązania BI, o ile uruchamia raport, który użytkownik może zweryfikować, jeśli baza danych ma został poprawnie zamodelowany, czy ETL poprawnie ładuje informacje i czy nasz model został poprawnie zdefiniowany w narzędziu BI. W ramach walidacji musimy również upewnić się, że wykonanie raportu jest zgodne z oczekiwanym.

Historie programistów MOLAP

MOLAP wykracza poza standardowe rozwiązanie BI, ale może być również włączony do rozwoju Agile, gdy potrzebujemy opracować dowolną bazę danych MOLAP. Historie programistów MOLAP muszą określać źródło informacji; jakie wymiary firmy mają znaleźć się w bazie; w jaki sposób zagreguje wszystkie wymiary; który interfejs będzie używany do ładowania informacji; czy informacje zostaną załadowane ręcznie, automatycznie lub w połączeniu z obydwoma; w jaki sposób zostanie zintegrowany z resztą platformy BI; czy zapiszemy informacje z MOLAP do hurtowni danych; czy będziemy odczytywać rzeczywiste informacje z hurtowni danych; oraz jaki jest okres użytkowania i przewidywana częstotliwość obciążenia narzędzia. Walidacja rozwoju MOLAP powinna uwzględniać, czy dane agregują się poprawnie, czy metryki zostały poprawnie zdefiniowane i czy wydajność ładowania informacji jest akceptowalna. Jako raportowanie BI, to narzędzie będzie również używane przez użytkownika końcowego, więc kluczowi użytkownicy i właściciele produktów będą również współpracować przy walidacji rozwoju MOLAP

Zwinne narzędzia zarządzania

Istnieje kilka narzędzi, które mogą nam pomóc w zarządzaniu wszystkimi rzeczami, o których mówiliśmy, przydzielaniu zadań, posiadaniu tablicy w naszym komputerze, tworzeniu profili użytkowników, grup, ustalaniu czasu realizacji zadań itp. Idziemy aby pokazać Ci, jak korzystać z jednego z nich, a także informacje o innych narzędziach, które mogą Cię zainteresować.

Trello

Trello to narzędzie typu open source, które umożliwia tworzenie własnych paneli roboczych, w których można zlokalizować tablicę zadań zarówno dla metodologii Scrum, jak i Kanban. Na stronie internetowej http://www.trello.com można uzyskać dostęp do narzędzia. Uzyskując dostęp do swojej tablicy, zobaczysz różne karty, które na koniec dnia są kolumnami, które pozwolą ci uporządkować różne statusy, jakie może przyjąć zadanie. Wewnątrz każdego zadania możesz zobaczyć różne pola, takie jak kto przydzielił to zadanie (w Trello jest brane pod uwagę, kto jest członkiem), jaki termin ma zostać dostarczony, opis zadania, komentarze, etykiety, dodaj załączniki i ciekawą funkcję Trello dodaj listy kontrolne, które pomogą Ci zdefiniować kroki, walidacje lub wymagania lub Definicję ukończenia. Gdy zadanie zmienia swój status, możesz po prostu przeciągnąć i upuścić zadanie w Trello, aby przenieść je do następnego stanu. Możesz także subskrybować zadanie, aby otrzymywać wiadomości e-mail za każdym razem, gdy zadanie zostanie zaktualizowane. Z drugiej strony masz możliwość tworzenia użytkowników i zespołów, a następnie możesz przypisać tablicę do zespołu. Uważamy, że to narzędzie jest dość proste w użyciu, ale jednocześnie całkiem kompletne, aby śledzić całą wykonaną pracę. Możesz także uaktualnić do wersji płatnej, która zapewni Ci zaawansowaną funkcjonalność i dodatkowe miejsce do zapisywania załączników.

Oprogramowanie JIRA

Jest to jedno z najpopularniejszych programów do zarządzania rozwojem z perspektywy Agile. Możesz tylko ocenić oprogramowanie za darmo, ale możesz je zdobyć dość tanio (10 USD), jeśli chcesz je zainstalować na swoim serwerze i od 10 USD miesięcznie, jeśli chcesz mieć je w chmurze. Mamy duże doświadczenie w korzystaniu z tego narzędzia do zarządzania i jest ono bardziej kompletne niż Trello,

ale o ile zapewnia więcej opcji, zarządzanie nim może być nieco trudniejsze. Można go znaleźć na stronie dostawcy (Atlassian):

https://www.atlassian.com/software/jira

Możesz dostosować swoją tablicę, aby zdefiniować kolumny i wiersze, aby zlokalizować zadania, ale możesz dodać do 140 pól do zadań, aby móc je następnie grupować i klasyfikować, dzięki czemu będzie to bardziej elastyczne; ale jeśli użyjesz wielu z tych pól, zwiększysz ogólne koszty utrzymania tego oprogramowania. Na rysunku 2.13 możesz zobaczyć niektóre z dostępnych pól, które można skonfigurować dla każdego zadania; możesz skonfigurować ponad 100 pól. Ponieważ JIRA jest zorientowana na użycie Kanbana, możesz zdefiniować WIP każdej kolumny, a tło kolumny jest zaznaczone na czerwono, gdy masz więcej zadań w kolumnie niż zdefiniowany WIP. Tablica działa bardzo podobnie do Trello, ponieważ możesz przeciągnąć i upuścić zadanie, aby zmienić status, przypisać zadania dowolnemu członkowi zespołu, dodać załączniki lub śledzić konkretne zadanie opracowane przez innych członków zespołu. Oprócz tego możesz również zdefiniować swoje dashboardy, aby analizować sposób, w jaki pracujesz, określać średni czas dostawy, śledzić ostatnią aktywność na platformie, zużyty czas, kalendarz dat dostaw przychodzących i wiele innych gadżetów, które można umieścić na pulpicie nawigacyjnym. Istnieje wiele innych opcji w JIRA, ale nie chcielibyśmy spędzać zbyt wiele czasu na tym temacie, ponieważ mamy wiele innych komponentów i narzędzi do analizy. Zachęcamy do oceny JIRA, Trello lub dowolnego innego oprogramowania do zarządzania rozwojem, które znajdziesz i sprawdzenia, które z nich lepiej dostosowuje się do Twoich potrzeb.

Uwaga: W ramach podsumowania tego rozdziału chcielibyśmy zalecić przestrzeganie zasad Agile, ale dostosowanie ich do Twojej organizacji i Twoich potrzeb. Każda metodologia zarządzania powinna być czymś, co ułatwia nam życie, a nie czymś, co tylko zwiększa złożoność i koszty zarządzania naszymi projektami. Zachowaj prostotę, a przynajmniej tak prostą, jak tylko potrafisz.

Wniosek

W tej części zobaczyliśmy, jak wykorzystać metodologie Agile w zarządzaniu projektami, dodając do projektu warstwę zarządzania, która może pomóc w zorganizowaniu całej liczby zadań wymaganych do dostarczenia systemu BI ze wszystkimi komponentami. Jak zauważono w rozdziale, nie lubimy być bardzo surowi w zakresie zasad metodologii, dlatego proponujemy dostosować je do swojego projektu za pomocą tych narzędzi, które mają dla Ciebie sens. W kolejnych rozdziałach zapoznamy się ze szczegółowymi informacjami na temat wszystkich komponentów BI, zaczynając od podstaw języka SQL, które ułatwią interakcję z bazą danych.