**33. Podejścia do zarządzania zespołami realizującymi projekty informatyczne.**

## 1. Wstęp

* Większość profesjonalnego oprogramowania jest tworzona przez zespoły składające się od dwustu do kilkuset osób (Sommervile)
* Nie ma możliwości, aby wszyscy członkowie tak wielkiego zespołu pracowali razem na jednym problemem
* Duże zespoły dzieli się na kilka grup. Każda grupa odpowiada za budowę jakiegoś podsystemu
* Przyjmuje się zasadę, że grupy nie powinny liczyć więcej aniżeli ośmiu członków
* Stworzenie małych grup umożliwia ograniczenie problemów komunikacyjnych
* Cała grupa może się spotkać przy jednym stole lub zebrać w swoich pokojach, nie są potrzebne skomplikowane struktury komunikacyjne
* Kiedyś celem zarządzania było kontrolowanie siły roboczej, wskazywanie co trzeba zrobić i pilnowanie, aby było to zrobione.
* Obecnie uznaje się, że takie podejście nie daje przewagi konkurencyjnej, a raczej odwrotnie.
* Obecnie pracownicy są dobrze wykształceni i oczekują, że będą włączani w przedsięwzięcie.
* A zatem, gdy kierownictwo nadal musi koordynować pracę ludzi i grup, to pomału odchodzi się od podejścia opartego na władzy i ścisłej kontroli.
* Nowoczesne podejście polega na wsparciu siły roboczej, umożliwianiu wykonania pracy oraz dbaniu o rozwój poprzez szkolenie i zachęcanie do samorozwoju.
* Utworzenie grupy, która będzie wydajnie pracować, jest krytycznym zadaniem menadżera
* W grupie powinna panować równowaga umiejętności technicznych, doświadczenia i osobowości
* W dobrej grupie panuje duch zespołu, tzn. jej członkowie są zmotywowani sukcesem grupy na równi z realizacją własnych celów
* Menadżerowie powinni zachęcać do jawnych czynności „budowania zespołu”, aby wypracować poczucie lojalności grupowej
* Kierownik musi motywować zespół jako całość oraz motywować osobno każdą osobę.
* Motywacja zespołu ma swoje źródło w osobistym zaangażowaniu kierownika, w sposobie przydziału i podziału pracy, wyraźnej wizji celu i sposobu jego osiągnięcia.
* Kierownik daje przykład przez własne zaangażowanie i zachowanie oraz tworzy klimat postępu i akceptacji zmiany.
* Motywację osobistą osiąga się poprzez własny stosunek i „niepisaną umowę”, czego dana osoba i kierownik wzajemnie od siebie oczekują.
* Kluczowym elementem motywacji jest projektowanie pracy poszczególnych osób; praca powinna zawierać odpowiednią ilość wyzwań i różnorodności oraz zmierzać do znaczącego wyniku.
* Każdy potrzebuje uzgodnionych celów, które są zrozumiałe i splatają się z osobistym i zawodowym rozwojem kariery, wynikającym z ambitnej pracy, profesjonalnych standardów, właściwych relacji i szkolenia.

## 2. Motywowanie

* Kierownik musi motywować zespół jako całość oraz motywować osobno każdą osobę.
* Motywacja zespołu ma swoje źródło w osobistym zaangażowaniu kierownika, w sposobie przydziału i podziału pracy, wyraźnej wizji celu i sposobu jego osiągnięcia.
* Kierownik daje przykład przez własne zaangażowanie i zachowanie oraz tworzy klimat postępu i akceptacji zmiany.
* Motywację osobistą osiąga się poprzez własny stosunek i „niepisaną umowę”, czego dana osoba i kierownik wzajemnie od siebie oczekują.
* Kluczowym elementem motywacji jest projektowanie pracy poszczególnych osób; praca powinna zawierać odpowiednią ilość wyzwań i różnorodności oraz zmierzać do znaczącego wyniku.
* Każdy potrzebuje uzgodnionych celów, które są zrozumiałe i splatają się z osobistym i zawodowym rozwojem kariery, wynikającym z ambitnej pracy, profesjonalnych standardów, właściwych relacji i szkolenia.

**Ludzi motywuje się poprzez spełnienie ich potrzeb:**

**Ludzie pracujący w firmach softwarowych nie są ani głodni, ani spragnieni, ani nie czują się fizycznie zagrożeni. Z menadżerskiego punktu widzenia najistotniejsze jest spełnienie potrzeb społecznych, szacunku i samorealizacji:**

* **Spełnienie potrzeb społecznych** oznacza zgodę na spotykanie się ze współpracownikami i zapewnienie miejsc na takie spotkania. Nieformalne i łatwe w użyciu kanały komunikacyjne, takie jak poczta elektroniczna, są ważne.
* **Spełnianie potrzeby szacunku** wymaga pokazania ludziom, że są wysoko oceniani przez firmę. Publiczne doceniania osiągnięć jest prostym i skutecznym środkiem do tego celu.
* **Spełnienie potrzeby samorealizacji** wymaga przekazania ludziom odpowiedzialności za ich pracę, przydzielanie im trudnych, ale nie niemożliwych do wykonania zadań i zaoferowanie programu szkoleń, który umożliwi im rozwijanie swoich umiejętności.

Wg Herzberga na pracowników wpływają dwa rodzaje czynników: tzw. czynniki motywujące i czynniki higieny:

* **Czynniki higieny** to elementy, których ludzie oczekują w już w momencie podejmowania pracy w wybranej firmie, a więc wypłata, ubezpieczenie, bezpieczeństwo pracy, prawo do urlopu i poczucie wspólnoty.
* **Czynniki motywujące** to możliwość zdobycia nowych umiejętności, otrzymania awansu i nagród za wytrwałą pracę.
* Czynniki higieny nie motywują pracowników, robią to tylko czynniki motywujące, jednakże ich brak ma negatywny wpływ na motywcję.

Wg Herzberga istnieją zarówno osoby szukające motywacji, jak i osoby szukające higieny:

* **Szukający higieny** – oczekują od pracodawcy:
  + spójnej polityki i administracji firmy
  + właściwego nadzoru
  + odpowiedniego wynagrodzenia
  + prawidłowych relacji międzyludzkich
  + godnych warunków pracy.
* **Szukający motywacji** – ich zadowalają następujące czynniki
  + osiągnięcia
  + uznanie
  + samodzielna praca
  + odpowiedzialność
  + rozwój.
* Zamiast dużej hali, lepsze wyniki daje umieszczenie dwóch-trzech stanowisk pracy w wielu mniejszych pomieszczeniach.
* Personalizacja stanowiska pracy.
* Pokój zebrań dla organizowania formalnych spotkań pracowników.
* Miejsce dla spotkań nieformalnych (np. omówienie spraw przy kawie).
* Poczucie pracy na nowoczesnym sprzęcie. Wydajność i chęć ludzi do pracy gwałtownie spada, jeżeli odczuwają oni, że pracują na przestarzałym sprzęcie - nawet wtedy, gdy wymiana sprzętu jest merytorycznie nieuzasadniona.
* Komfort psychiczny, właściwa atmosfera w pracy, eliminacja napięć i zadrażnień, nie dopuszczanie do rozmywania odpowiedzialności, sprawiedliwa ocena wyników pracy poszczególnych członków zespołu, równomierny rozkład zadań.

Są różne typy osobowości ludzi, do każdego trzeba mieć inne podejście. Np. ktoś, kto nie zrobi nic bez wyraźnego polecenia, potrzebuje zachęty do wzięcia spraw w swoje ręce. Ktoś, kto np. jest zabawny, rozśmiesza grupę, ale marnuje dużo czasu, zazwyczaj ‘potrzebuje’ większej liczby obowiązków.

## 3. Dobór personelu

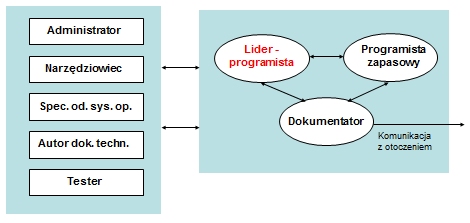
**Pożądane cechy członków zespołu:**

* podatność na oddziaływanie kierownictwa projektu,
* umiejętność pracy zespołowej,
* wysokiej klasy umiejętności techniczne,
* silna orientacja na rozwiązywanie problemów,
* silne nastawienie na osiąganie rezultatów,
* wysoka samoocena,
* konstruktywna krytyka
* Umiejętność pracy w stresie: Dla większości osób niewielki stres działa mobilizująco. Po przekroczeniu jednak pewnego progu następuje spadek możliwości danej osoby. Próg ten jest różny dla różnych osób.
* Zdolności adaptacyjne: Informatyka jest jedną z najszybciej zmieniających się dziedzin. Ocenia się, że 7-9 miesięcy przynosi w informatyce zmiany, które w innych dziedzinach zajmują 5-7 lat. Uśpienie, zajmowanie się jednym problemem w jednym środowisku przez lata jest w informatyce bardzo groźne!
* Doświadczenie w dziedzinie zastosowania: Jeżeli projekt ma zakończyć się sukcesem, to jego twórcy muszą rozumieć dziedzinę zastosowania
* Doświadczenie w pracy z platformą: Może być istotne, jeżeli prace obejmują programowanie na niskim poziomie. W przeciwnym wypadku zwykle nie jest krytycznym atrybutem
* Doświadczenie w pracy z językiem programowania: Zwykle jest istotne jedynie w wypadku krótkotrwałych projektów, przy których nie ma czasu na naukę nowego języka
* Zdobyte doświadczenie: Może być podstawową wskazówką co do tego, co kandydat powinien umieć, i o jego zdolności do uczenia się.
* Zdolności komunikacyjne: Są ważne, ponieważ członkowie zespołu muszą porozumiewać się pisemnie i ustnie z innymi współpracownikami, menedżerami i klientami
* Zdolność do przystosowywania się: Tę zdolność można ocenić przyglądając się różnym rodzajom doświadczeń zdobytych przez kandydata. To ważny atrybut, bo z niego wynika zdolność do uczenia się.
* Nastawienie: Członkowie zespołu powinni być pozytywnie nastawieni do wykonywanej pracy i chcieć zdobywać nowe umiejętności.
* Osobowość: To bardzo ważny atrybut, ale zwykle bardzo trudny do oceny. Kandydaci muszą być racjonalnie zgodni z innymi członkami zespołu. Żaden konkretny typ osobowości nie jest mniej lub bardziej odpowiedni do inżynierii oprogramowania

**Celem** stojącym przed kierownikiem projektu jest dostarczenie produktu w wymaganym czasie, w ramach danego budżetu i posiadającego odpowiednią jakość.

**Główne funkcje kierownika projektu:**

* + planowanie,
  + organizowanie,
  + motywowanie,
  + kontrolowanie.

**4. Budowanie zespołu**

Wg Bakera, Arona i Brooksa najskuteczniejsze wykorzystanie dobrych programistów osiąga się przez zbudowanie zespołu wokół jednego wysoko wykwalifikowanego lidera-programisty.

Głowni członkowie zespołu lidera-programisty:

* **Lider - programista** – bierze całkowitą odpowiedzialność za zaprojektowanie, zaprogramowanie, przetestowanie i instalację systemu.
* **Doświadczony programista zapasowy** – wspiera lidera-programistę, bierze odpowiedzialność za zatwierdzanie oprogramowania.
* **Dokumentator** – przejmuje wszystkie funkcje urzędnicze projektu, takie jak zarządzanie konfiguracjami, redagowanie dokumentów, opracowywanie dokumentacji.
* Zależnie od rozmiarów i rodzaju tworzonego oprogramowania, mogą być potrzebni inni specjaliści do czasowej lub stałej pracy w zespole. Mogą to być administratorzy, specjaliści od systemów operacyjnych i języków, testerzy itp.

**Uzasadnienie**: najlepsi programiści są do 25 razy bardziej wydajni od najgorszych. Pomysł ma już 30 lat, a ciągle jest skutecznym sposobem organizacji małych grup tworzących oprogramowanie.

Problemy:

* Liczba utalentowanych projektantów i programistów jest niewielka. jeżeli oni popełnią błędy, to nikt nie będzie kwestionował ich decyzji.
* Lider-programista bierze całą odpowiedzialność i może przypisywać sobie wszystkie zasługi w wypadku sukcesu. Członkowie grupy mogą być niezadowoleni, jeżeli ich rola w przedsięwzięciu nie jest doceniana. Ich potrzeba szacunku nie będzie zaspokojona.
* Przedsięwzięcie będzie zagrożone, gdy lider-programista zachoruje lub odejdzie z firmy. Zarząd firmy może nie chcieć zaakceptować takiego ryzyka.
* Struktury organizacyjne mogą nie być zdolne do przyjęcia takiej grupy. Wielkie firmy mają starannie zdefiniowaną strukturę.

**5 błędów zarządzania, które owocują utratą zaufania pracowników:**

Każdy szef chciałby, by jego zespół był zaangażowany w pracę, lojalny i efektywny. Niestety nie da się tego osiągnąć, jeśli lider zespołu nie zdobędzie zaufania swoich pracowników. To zaufanie, poprzez błędy managera, utracić jest bardzo łatwo, a jego odzyskanie, to niezwykle trudna sprawa. Co powoduje, że pracownicy tracą zaufanie do swojego szefa?

**1. Niedotrzymane obietnice**

Najpoważniejszym wykroczeniem przełożonego jest niedotrzymywanie obietnic składanych zespołowi. Nie ma tu znaczenia to, czy chodzi tu o podwyżkę, nowy gadżet do biura, czy przyrzeczenie zajęcia się określoną sprawą priorytetowo. Jeśli szef nie dotrzyma słowa i nie wyjaśni powodów jego niedotrzymania, to w bardzo krótkim czasie, utraci zaufanie zespołu, atmosfera stanie się napięta, a pracownicy zaczną podważać każdą obietnicę składaną przez przełożonego. Efekt – mniejsza efektywność pracy i większa rotacja, a także spadek szacunku do szefa.

**2. Nierówne traktowanie pracowników**

Każdy pracownik oczekuje sprawiedliwego traktowania w swoim miejscu pracy. Preferowanie niektórych pracowników prowadzi nie tylko do spadku zaufania u nie uprzywilejowanej grupy, ale również wprowadza rozłam w zespole.

**3. Zwracanie uwagi pracownikowi przy reszcie zespołu**

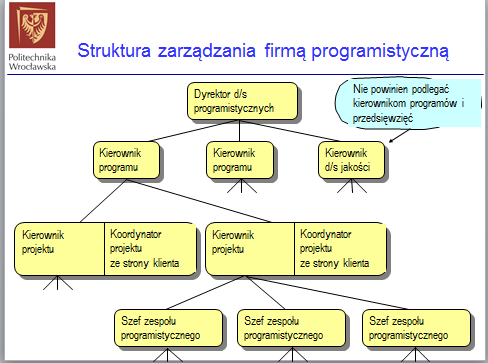
Publiczne zwracanie uwagi owocuje dodatkowym upokorzeniem dla pracownika, który ma poczucie, że jego status w zespole, poprzez reprymendę maleje. Pracownik, który zostanie publicznie upomniany traci poczucie tego, że może popełnić w pracy błąd. Efektem tego bywa często markowanie działań i „zamiatanie problemów pod dywan”. Częstą reakcją wynikającą z mechanizmów obronnych pracownika jest też próba późniejszego zdyskredytowania szefa (np. poprzez plotki na jego temat).

**4. Częste zmiany decyzji**

Szef, który jednego dnia oczekuje od pracowników działania w jeden, a na drugi dzień w inny sposób szybko doprowadza do tego, że w zespole zaczyna panować chaos. Nikt nie wie jak właściwie działać, a pracownicy zaczynają mieć przekonanie, że nie warto rozpoczynać działania, bo zdanie szefa w tym temacie zmieni się jeszcze kilka razy. Jeśli już musisz zmienić decyzję, to umotywuj ją całemu zespołowi.

**5. Komentowanie pracy pracownika z innymi członkami zespołu**

O ile chwalenie pracownika do jego współpracowników niesie ze sobą pozytywne wzmocnienia (w tych zespołach, które nie są oparte na rywalizacji lecz współpracy). O tyle wytykanie pracownikowi błędów w rozmowie z jego kolegami z pracy owocuje nie tylko rozbiciem spójności zespołu, ale również utratą poczucia wsparcia ze strony przełożonego.



### Modele organizacyjne zespołów

Organizacja jest wskaźnikiem stopnia dojrzałości organizacji oraz jej podejścia do tego zagadnienia. Wyróżnia się następujące modele organizacyjne pracy zespołów:

* **sieciowy** (uczestnicy komunikują się każdy z każdym)
* **gwiaździsty** (charakteryzuje się silną, centralną pozycją lidera, który pośredniczy w wymianie wszystkich informacji między członkami zespołu)
* **izomorficzny** (oddaje strukturę projektu w strukturę zespołu)
* **specjalizacyjny**,
* **nieegoistyczny**,
* **macierzowy**
  + zadaniowy,
  + funkcjonalny.

|  |  |
| --- | --- |
| **Metodyki wytwórcze** | **Metodyki zarządcze** |
| |  |  | | --- | --- | | Lekkie (Zwinne) | Ciężkie | | * XP * Agile RUP * Crystal | * USDP * RUP | | |  |  | | --- | --- | | Lekkie (Zwinne) | Ciężkie | | * SCRUM * FDD | * PRINCE2 * CMMI | |

### Programowanie zwinne (Agile software development)

Grupa metodyk wytwarzania oprogramowania opartego o programowanie iteracyjne (model przyrostowy). Wymagania oraz rozwiązania ewoluują przy współpracy samozarządzalnych zespołów, których celem jest przeprowadzanie procesów wytwarzania oprogramowania. Pojęcie zwinnego programowania zostało zaproponowane w 2001w Agile Manifesto.

Generalnie metodyka oparta jest o zdyscyplinowane zarządzanie projektem, które zakłada częste inspekcje wymagań i rozwiązań wraz z procesami adaptacji (zarówno specyfikacji jak i oprogramowania). Metodyka ta najczęściej znajduje zastosowanie w małych zespołach programistycznych, w których nie występuje problem komunikacji, przez co nie trzeba tworzyć rozbudowanej dokumentacji kodu. Kolejne etapy wytwarzania oprogramowania zamknięte są w iteracjach, w których za każdym razem przeprowadza się testowanie wytworzonego kodu, zebranie wymagań, planowanie rozwiązań itd. Metoda nastawiona jest na szybkie wytwarzanie oprogramowania wysokiej jakości.

Skład zespołów jest zazwyczaj wielofunkcyjny oraz samoorganizowalny, bez zastosowania jakiejkolwiek hierarchii korporacyjnej. Członkowie zespołu biorą odpowiedzialność za zadania postawione w każdej iteracji. Sami decydują jak osiągnąć postawione cele.

Metoda nastawiona jest na bezpośrednią komunikację pomiędzy członkami zespołu, minimalizując potrzebę tworzenia dokumentacji. Jeśli członkowie zespołu są w różnych lokalizacjach, to planuje się codzienne kontakty za pośrednictwem dostępnych kanałów komunikacji (wideokonferencja, e-mail itp.).

**Manifest zwinności (Agile Manifesto)**

Ważniejsze są:

* **Jednostki i interakcje** niż procesy i narzędzia
* **Działające oprogramowanie** niż obszerna dokumentacja
* **Współpraca klienta** niż negocjacja kontraktu
* **Nadążanie za zmianami** niż trzymanie się planu

### Rational Unified Process (RUP)

Proces iteracyjnego wytwarzania oprogramowania opracowany przez firmę Rational Software Corporation (firma została przejęta przez IBM).

Proces RUP nie jest pojedynczym, ściśle określonym procesem, ale raczej szablonem procesu. Został on zaprojektowany w celu przystosowania do charakteru konkretnej organizacji (przedsiębiorstwa), konkretnego zespołu projektowego lub nawet charakteru konkretnego projektu. Z szablonu RUP można wybrać elementy w zależności od konkretnych potrzeb.

Rational Unified Process (RUP) to także nazwa oprogramowania, opracowanego przez Rational Software (obecnie dostępnego w IBM). Produkt ten zawiera hipertekstową bazę wiedzy z przykładowymi artefaktami oraz szczegółowymi opisami wielu typów czynności. Process RUP definiowany jest także w produkcie Rational Method Composer (RMC), który pozwala na tworzenie spersonalizowanych wersji RUP.

**RUP bazuje na zbiorze zasad inżynierii programowania oraz najlepszych praktykach, na przykład:**

1. Iteracyjnym wytwarzaniu oprogramowania (Iterative Development)
2. Zarządzaniu wymaganiami (Requirement Management)
3. Używaniu architektury bazującej na komponentach (Component-based architecture)
4. Graficznym projektowaniu oprogramowania
5. Kontroli jakości oprogramowania (Quality Assurance)
6. Procesu kontroli zmian w oprogramowaniu (Change Management)

**RUP używa podejścia iteracyjnego i przyrostowego z następujących powodów:**

* Integracja oprogramowania robiona krok po kroku podczas wytwarzania oprogramowania, ograniczając go do mniejszej liczby elementów
* Integracja jest prostsza i mniej kosztowna
* Składowe oprogramowania są projektowane oddzielnie i łatwiej poddają się reużywalności
* Łatwiej wykrywać zmiany wymagań i łatwiej nimi zarządzać
* Ryzyka identyfikowane i atakowane są wcześnie ponieważ każda iteracja pozwala wykryć kolejne ryzyka
* W iteracjach ulepszana jest     architektura oprogramowania

### Programowanie ekstremalne

Metodyka programowania mająca na celu wydajne tworzenie małych i średnich "projektów wysokiego ryzyka". Przyświeca temu koncepcja prowadzenia projektu informatycznego, wywodząca się z obserwacji innych projektów, które odniosły sukces.

Podstawą ekstremalnego programowania jest synergia wynikająca ze stosowania rozmaitych praktyk, które same w sobie mają wiele zalet, lecz mogą być trudne w zastosowaniu. Łączne użycie tych praktyk ma zapewniać wyeliminowanie niedogodności każdej z nich.

**Zalecenia**

* **Iteracyjność**
  + Program tworzy się w iteracjach (krótkie, przyrostowe kroki programistyczne) - i co ważniejsze - planuje tylko następną iterację. Efektem każdej iteracji (kilka tygodni) powinna być wersja programu spełniającą założenia dla danej iteracji. Następnie planuje się co zrobić dalej. Odpowiada to zasadzie Open Source: "release early,     release often" (wczesne i częste wydania).
* **Nie projektować z góry**
  + Nie można z góry przewidzieć, jaka architektura będzie najlepsza dla danego problemu. Dlatego należy ją tworzyć w miarę rozszerzania programu.
* **Testy jednostkowe**
  + Testy jednostkowe pisze się zanim w ogóle zacznie się pisać kod - najlepiej na początku iteracji. Potem pisze się kod, który potrafi je wszystkie przejść. Takie testy dają zapewnienie (o ile testy są dobrze napisane), że to, co ważne, zostanie zaprojektowane, na to zaś, co nie jest ważne, programiści nie będą tracić czasu.
* **Ciągłe modyfikacje architektury**
  + Architektura nie jest czymś, czego nie wolno ruszać. Jeśli modyfikacja architektury ułatwi przejście danej iteracji i nie zepsuje wyników testów uzyskanych na poprzednich, należy ją wykonać. Pod tę zasadę podlega także usuwanie wszystkich znanych błędów przed rozszerzeniem funkcjonalności.
* **Programowanie parami**
  + Programiści piszą w parach: jedna osoba pracuje przy klawiaturze i jest głównym koderem, druga obserwuje pierwszą, zgłasza poprawki, zadaje pytania wyjaśniające. Programiści programujący w parze zamieniają się rolami co kilkadziesiąt minut. Ta technika umożliwia wyłapanie wielu błędów oraz wzajemną naukę. Kod, którym zajmuje się tylko jedna osoba, ma tendencje do stawania się całkowicie niezrozumiałym dla kogokolwiek innego niż autor, więc dodatkowy programista zwiększa jakość kodu.

Nie jest jasne, czy sumarycznie łączna wydajność pracy przy takiej metodzie jest wyższa, taka sama, czy niższa niż w tradycyjnym programowaniu indywidualnym.

**Kwestie kontrowersyjne**

* Brak dokładnej specyfikacji.
* Konieczna stała dostępność     przedstawiciela klienta.
* Wspólna "własność"     kodu - każdy może zmieniać dowolny fragment systemu.

### Scrum

To metodyka prowadzenia projektów. Zaliczana do tzw. metodyk zwinnych, zgodnych z Agile Manifesto. Najczęściej wykorzystywana jest w projektach informatycznych.Używana jest w skomplikowanych projektach, w których nie można przewidzieć wszystkiego, co może się przydarzyć lub w przypadku przedsięwzięć o wysokim stopniu innowacyjności.

Metodyka skupia się na:

* dostarczaniu kolejnych, coraz bardziej dopracowanych wyników projektu
* włączaniu się przyszłych użytkowników w proces wytwórczy
* samoorganizacji zespołu projektowego.

**Zespół i role**

Zazwyczaj zespół scrum składa się od 5 do 9 osób. Dobrze, gdy ma charakter interdyscyplinarny i składa się z osób reprezentujących różne umiejętności. Osoby uczestniczące w zespole nie mogą uczestniczyć w innych zespołach.

**Główne role w projekcie grają:**

* "Mistrz Scrum" (Scrum Master),
* Właściciel Produktu (Product Owner)
* Członkowie Zespołu (The Team)

**Opis metodyki**

Zespół projektowy pracuje w określonym przedziale czasowym zwanym przebiegiem (ang. sprint). Efektem przebiegu za każdym razem powinno być dostarczenie użytkownikom kolejnego działającego produktu. Zasadą jest to, że zmiany wprowadzane w jednym przebiegu muszą być namacalne dla użytkowników. Muszą wnosić wartość funkcjonalną (addytywną). Przebieg może trwać od 2 do 6 tygodni. Zaleca się stosowanie przebiegów o stałych długościach.

W pierwszym etapie tworzona jest lista wymagań użytkownika, są one gromadzone w postaci "historyjek". Każda historyjka opisuje jedną cechę systemu. Właściciel projektu jest też zobowiązany do przedstawienia priorytetów wymagań oraz głównego celu przebiegu. Po tym formułowany jest rejestr wymagań. Cel przebiegu jest zapisywany w widocznym miejscu w pokoju członków zespołu.

Następnie wybierane są zadania o najwyższym priorytecie, a jednocześnie przyczyniające się do realizacji celu projektu. Szacuje się czas realizacji każdego zadania. Lista zadań wraz z oszacowaną czasochłonnością nosi nazwę rejestru zadań przebiegu (sprint backlog).

Po etapie przygotowawczym zespół przechodzi do realizacji przebiegu. W jego trakcie Właściciel Produktu nie może ingerować w prace zespołu. Nie powinno się także zmieniać zakresu sprintu.

Jako że zespół z założenia jest samoorganizującym się ciałem, nie ma mowy o odgórnym przypisywaniu zadań do poszczególnych członków zespołu, lecz samodzielnie dokonują oni wyboru realizowanych zadań, według wspólnych ustaleń, umiejętności czy innych preferencji.

Naczelną zasadą metodyki jest przeprowadzanie codziennych (około 15-minutowych) spotkań (z ang. scrum meeting), na których omawiane są zadania zrealizowane poprzedniego dnia i problemy występujące przy ich realizacji oraz zadania do wykonania w dniu spotkania.

Sprint kończy się spotkaniem sprint review, na którym prezentowany jest wynik pracy zespołu poprzez prezentowanie 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 spotkania planistycznego do następnego przebiegu.