Агилни развој софтвера

Сенад Ибраимоски
mi08247@alas.matf.bg.ac.rs

Математички Факултет Универзитета у Београду

Покривене теме

* Агилни методи
* Развој вођен планирањем и агилни развој
* Екстремно програмирање
* Агилно управљање пројектима
* Скалирање агилних метода

Брзи развој софвера

* **Брз развој и достава су често најбитнији захтеви у пројектима.**
  - Предузећа послују у изузетно динамичним околностима и стога је немогуће конструсати поуздани скуп корисниичких захтева.
  - Еволуција софтвера мора да рефлектује промене у пословању.

* **Брз развој софтвера**
  1. Спецификација, план и имплементација се преплићу.
  2. Систем се развија у серији верзија заједно са странкама које деле интерес (енг. stakeholder)
  3. Кориснички интерфејси се често развијају уз помоћ графичких алата.

Принципи агилних метода

Укључење клијената	Клијети би требало да буду укључени у целокупан развојни процес. Њихова улога је да приоритизују нове захтеве и да оцене итерације у развоју.
Инкрементална достава	Софтвер се развија у серији инкремената где клијент диктира који захтеви ће бити покривени у инкременту.
Људи! Не процес	Способности чланова тима је потребно препознати и искористити што је то боље могуће. Чланови тима требају радити на начин за који они сматрају да је најбољи. Никакви процеси развоја им се не смеју наметати.
Промене су добродошле	Потребно је очекивати промене у корисничким захтевима и с тим на уму развијати систем који у сваком тренутку може рефлектовати исте.
Једноставност	Кад је год могуће активно радити на елиминацији комплексности система и развојног процеса.

Примењивост агилних метода

* Мали или средње велики пројекти намењени продаји.
* Пројекти унутар предузећа где су клијенти наклоњени прикључењу у развојни процес и где не постоји много спољашњих правила која се односе на софтвер.
* Због фокуса на мале пројекте скалирање агилних метода на комплексније системе представља изузетан проблем.

Проблеми

* Показује се тешким одржити заинтересованост клијената који су укључени у развојни процес.
* Чланови развојног тима могу бити неподобни посвећености коју захтевају агилни методи.
* Одређивање приорита постаје скоро немогуће када је укључен већи број интересних странака.
* Одржавање једноставности захтева додатан напор.

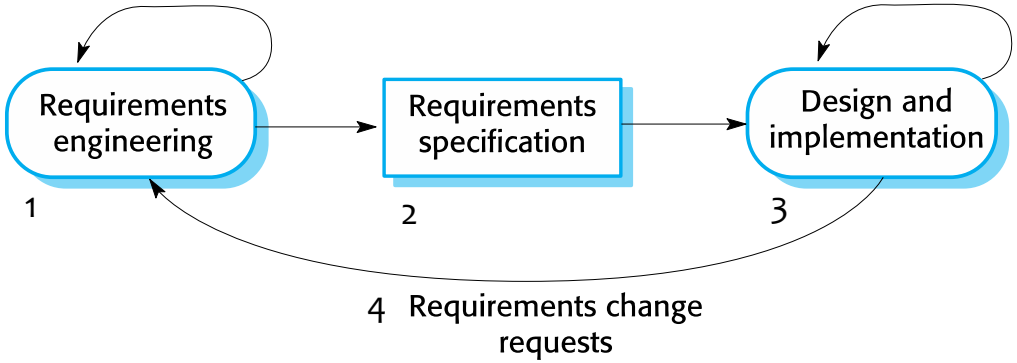
Развој вођен планирањем и агилни развој

* Развој вођен планирањем
  - Подразумева одвојене развојне фазе. Резултати ових фаза се морају доставити по унапред припремљеном плану.
  - Иако подсећа на модел водопада – инкрементални развој је могућ унутар активности које припадају одређеној фази.

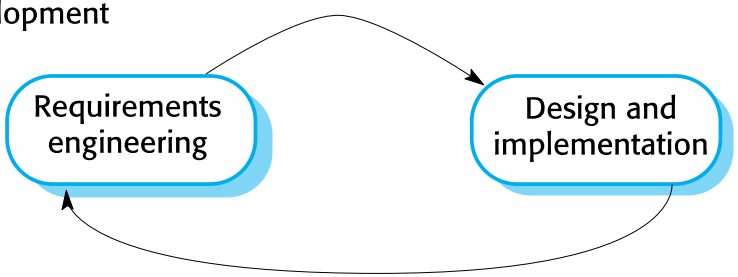
* Агилни развој
  - Спецификација, дизајн, имплементација и тестирање се преплићу.

Развој вођен планирањем и агилни развој

Plan-based development



Agile development



Речник појмова:

1. Генерисање захтева
2. Спецификација захтева
3. Дизајн и имплементација
4. Захтев за измену корисничких захтева

Проблеми техничке и организационе природе

* Већина пројеката укључује елементе развоја вођеног планирањем и (или) агилних метода. При балансирању у примени ових елемента води се рачуна о:

  - Битности поседовања детаљних спецификација и дизајна пре него што се започне имплементација. Ако вам је поменуто битно потребан вам је развој вођен планирањем.
  - Битности брзе испоруке инкремената и благовремене оцене истих од стране клијената. - Агилни методи

Проблеми техничке и организационе природе

* На који начин се развија систем
  Развој вођен планирањем је потребан за системе који захтевају обимну анализу пре имплементације.
* Дуговечност имплементације
  Ако се захтева дуговечност система потребно је направити добар дизајн и документацију о систему.
* Доступност различитих алата
  Агилни методи се ослањају на добре алате да би се успешно пратила еволуциона природа софтвера.
* Организација развојног тима
  На који начин је организован тим. У малу изузетно способну екипу или је дисрибуиране природе где се одређени део развоја outsource – ује.

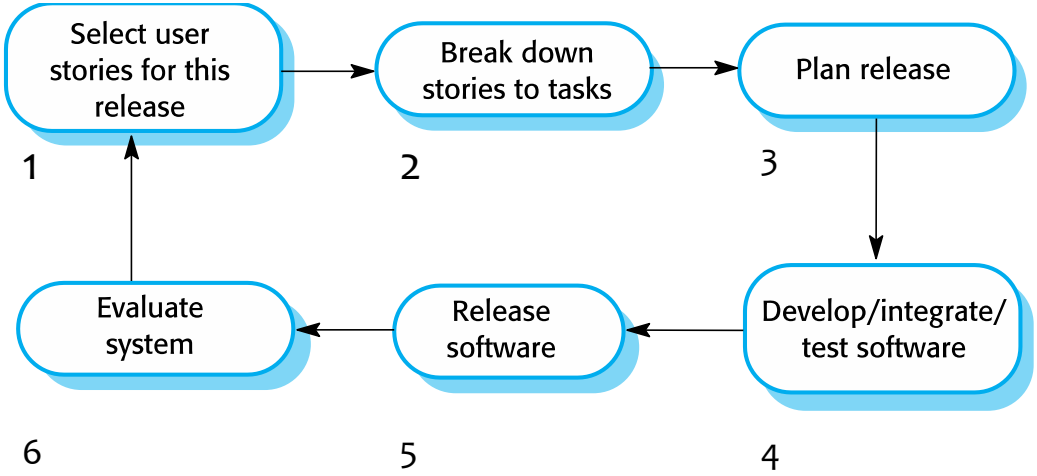
Екстремно програмирање

* Потенцијално најпознатија и највише коришћена агилна техника.
* Екстремно програмирање заузима „екстреман приступ“ при итеративном развоју.
  - Нове верзије се генеришу неколико пута на дан.
  - Инкременти се достављају клијентима сваке две недеље.
  - Сви тестови се морају успешно извршити за сваку верзију пре него што исту прихватимо.

Екстремно програмирање и агилни принципи

* Инкрементални развој је подржан кроз мале и честе извршне верзије.
* Клијенти су укључени у целокупан развој заједно са тимом.
* Људи! Не процес – Кроз програмирање у паровима.
* Промене су подржане због генерисања честих верзија.
* Једноставност је подржана кроз констатно рефакторисање кода.

Екстремно програмирање – извршни круг



1. Одабир корисничких прича за тренутну извршну верзију.
2. Приче се фрагментују на једноставније задатке.
3. Планирање извршне верзије.
4. Имплементација, Интеграција и тестирање
5. Генерише се извршна верзија
6. Евалуација, оцена система.

Екстремно програмирање - принципи

Инкрементално планирање	Захтеви се чувају на тзв. картицама прича. Картице тј. захтеви који ће бити укључени у извршну верзију се приоритизују по битности и времену које је потребно за њихову ипмплементацију.
Мале извршне верзије	Минимални користан скуп функционалности поседује приоритет. Верзије су честе тако да инкременти увек додају нове на постојеће функционалности или их мењају ако постоји потреба.
Једноставан дизајн	Дизајнира се само толико да се покрију тренутни захтеви и ништа више.
Развој вођен тестирањем	Користи се аутоматизована платформа за тестирање при писању тестова за функционалности које се имплементирају.
Рефакторисање	Констатно. Поготову при програмирању у пару.

Екстремно програмирање - принципи

Програмирање у пару	Програмери раде у пару и констатно проверавају шта раде. Док један програмира други служи као подршка и предлаже идеје, примећује грешке и савете при рефакторисању.
Колективно власништво	Свако може да мења шта жели. Сви програмери одговарају за било који део кода.
Континуелна интеграција	Чим се посао или задатак заврши, убацује се у целокупан систем. Систем се тестира након сваке интеграције.
Одржив корак	Прековремен рад је неприхватљив јер често долази до смањења квалитета кода и продуктивности.
Укључење клијента	Констатно. Посматра се као део развојног тима и задужен је за корисничке захтеве.

Картице прича и задаци

Prescribing medication

The record of the patient must be open for input. Click on the medication field and select either 'current medication', 'new medication' or 'formulary'.

If you select 'current medication', you will be asked to check the dose; If you wish to change the dose, enter the new dose then confirm the prescription.

If you choose, 'new medication', the system assumes that you know which medication you wish to prescribe. Type the first few letters of the drug name. You will then see a list of possible drugs starting with these letters. Choose the required medication. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

If you choose 'formulary', you will be presented with a search box for the approved formulary. Search for the drug required then select it. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

In all cases, the system will check that the dose is within the approved range and will ask you to change it if it is outside the range of recommended doses.

After you have confirmed the prescription, it will be displayed for checking. Either click 'OK' or 'Change'. If you click 'OK', your prescription will be recorded on the audit database. If you click 'Change', you reenter the 'Prescribing medication' process.

Task 1: Change dose of prescribed drug

Task 2: Formulary selection

Task 3: Dose checking

Dose checking is a safety precaution to check that the doctor has not prescribed a dangerously small or large dose.
Using the formulary id for the generic drug name, lookup the formulary and retrieve the recommended maximum and minimum dose.
Check the prescribed dose against the minimum and maximum. If outside the range, issue an error message saying that the dose is too high or too low.
If within the range, enable the 'Confirm' button.

Екстремно програмирање – промене и рефакторисање

Дизајнира се за промену. Сматра се вредним уложити додатан напор и време за овакав дизајн. Последица је смањење трошкова у касинијм фазама развоја.

* Ово није случај код екстремног програмирања с обзиром да се сматра да се промене не могу поуздано предпоставити.

* Сматра се да је рефакторисање кода довољно.
  - Констано рефакторисање побољшава читљивост кода и смањује потребни обим документације.
  - Промене се једноставно уносе јер је код добро структуиран и чист.

Екстремно програмирање – тестирање

* Тестирање је једна од главних активности у екстремном програмирању.
* Представља развој вођен тестирањем.
* Инкрементални развој тестова из сценарија.
* Укључење клијената при развоју тестова.
* Аутоматизација.

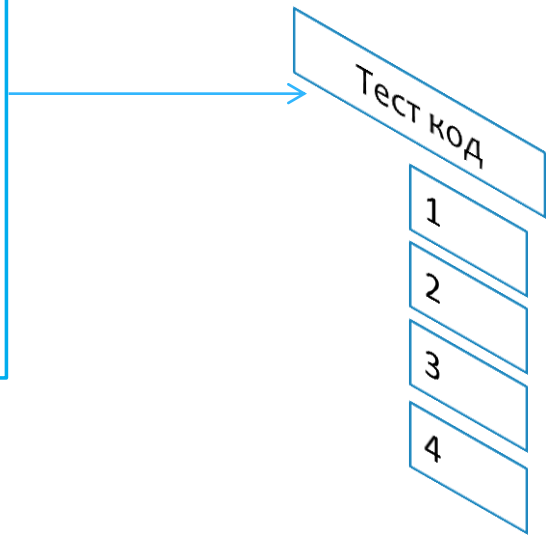
Опис тест случаја

Test 4: Dose checking

Input:
1. A number in mg representing a single dose of the drug.
2. A number representing the number of single doses per day.

Tests:
1. Test for inputs where the single dose is correct but the frequency is too high.
2. Test for inputs where the single dose is too high and too low.
3. Test for inputs where the single dose * frequency is too high and too low.
4. Test for inputs where single dose * frequency is in the permitted range.

Output:
OK or error message indicating that the dose is outside the safe range.



Скрам – (енг. Scrum)

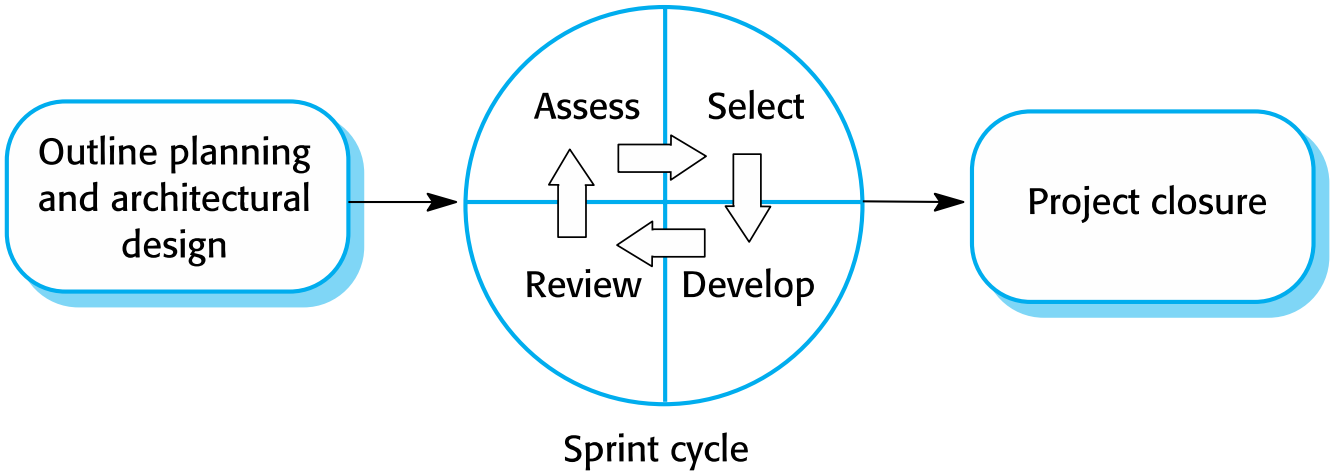
* Скрам – Агилни метод са фокусом на итеративни развој.
* Три фазе скрама:

1. Иницијална планска фаза у којој се успостављају генерални циљеви пројекта и дизајн архитектуре софтвера.

2. Серија спринт циклуса, где се у сваком циклусу развија инкремент система.

3. Зафршна фаза. Документација, туторијали. У овој фази се такође деле лекције које су научене од почетка пројекта

Скрам процес



Спринт

* Спринтови су фиксиране дужине (обично две до четири недеље)
* Почетна тачка је планирање баклога.
* Селекциона фаза захтева присуство читавог тима који ради заједно са клијентом при одабиру функционалности које је потребно развити током спринта.
* Тим се организује и започиње развој. Током ове етапе тим је изолован од клијената и предузећа и сва комуникација се одвија преко члана тима који се зове Скрам мастер.
* На крају спринта, завршен посао се презентује корисницима од интереса и након успешне оцене започиње се нови спринт.

Скрам Тим

* Скрам мастер је посао организационе природе. Скрам мастер има задатак да организује дневне састаке, прати баклог, мери прогрес пројекта и комуницира са клијентима.

* Комплетан тим присуствује кратким дневним састанцима где сви чланови деле информације, описују докле су стигли од прошлог састанка, проблеме који су настали и шта је планирано за дан који следи.
  * Када настану проблеми комплетан тим је упознат на чему се тренутно ради и на тај начин могу брзо да се реорганизују и заједничким напором реше исте.

Корисност скрама

* Подела на једноставне целине.
* Нејасни захтеви не успоравају процес.
* Читав тим „види све“ и на тај начин се побољшава комуникација.
* Клијенти на време добијају инкременте и прослеђују своје сугестије и оцене скрам мастеру.
* Успоставља се поверење међу програмерима и клијената

Скалирање агилних метода

* Велики системи су често колекција одвојених система који комуницирају, где сваки тим развија различите системе.
* Комплексни системи и развојни процеси који се користе се не слажу са агилним принципима мада постоје покушаји да се „агилизују“ одређене развојне фазе.
* Овакви системи често поседују велики број интересних странака тако да је практично немогуће све укључити у развојни процес.
* Такође је тешко задржати тим на окупу. С обзиром да развој оваквих система захтева дужи временски период, неретко чланови тима добију прилике из других предузећа.

Питања?

Крај!

Сенад Ибраимоски