Тема 2 Б

Съвременни софтуерни технологии

① Софтуерен продукт и процес

Софтуерен продукт е програма или съвкупност от взаимодействащи програми, записани във технически носител и придружени от съответната документация.

Програмата е последователност от инструкции, които когато бъдат декодирани от компютъра, водят до решаването на зададена задача от компютъра

Софтуерният процес е структурирано мн-во от дейности, нужни за разработката на софтуерна система.

② Модел на софтуерен процес

Това е абстрактно представяне на процеса. Той представлява описание на процеса от определена гледна точка. Примери за такива модели са Waterfall, еволюционна разработка, разработка по спирала и д.р.

③ Софтуерни технологии (СТ)

СТ представляват систематичен подход или разработването на качествен софтуер, както и цялостното предлагане на пазара, експлоатация и сопровождане.

④ Управление на софтуерен проект и ресурси

Управление на съвкупност от разнородни дейности с цялостно осъществяване за решаване на сложен, нестандартен проблем, при наложени ограничения относно време, разходи, качество и специфични изисквания или организацията на работата.

Ключови идеи: сложна и неструктурирана до момента работа за постигане на промяна или по-добро.

Предварително определени цени и изисквания за продължителност, качество и разходи. Може да се наложи преструктуриране на организацията - изпълнителят, приет в сила да работи и управлението на външните ресурси.

⑤ Функционални и нефункционални изисквания

ФИ описват системни услуги или функции.
НФИ са ограничения в/у системата или в/у процеса на разработка. НФИ може да са по-важни от ФИ ако те не бъдат постигнати, то системата е безполезна.
Пример за тях са бързина, условия по имплементацията, операция и т.н.

⑥ Анализ и формулиране на софтуерните изисквания

Управление на изискванията:
1) Идентифициране на изискванията
2) Анализ на изискванията
3) Спецификиране на изискванията
4) Валидиране на изискванията
5) Управление на изискванията - проследяване, промяна

⑦ Проектиране на софтуера

Обхваща всички дейности за превръщане на утвърдените изисквания в описание, определящо тяхното съдържанието и функционалните функциите на програмите. Основен принцип на проектирането е изследване и разбиране на проблема и последователно му декомпозиране на подпроблеми.
Два основни подхода:
- функционален — състоянието на системата е централизирано и се разделя м/у функции, опериращи в/у това с-е.
- обектно-ориентиран - състоянието на системата е децентрализирано и всеки обект управлява собственото си състояние. Обектите си комуникират помежду си чрез съобщения

(8) Обектно-ориентиран дизайн

Обект – същност, имаща важно функционално значение в разглежданата приложна област. Всеки обект има състояние, поведение и индивидуалност. Сходни обекти определят общ за тях клас. Състоянието на обекта се характеризира с изброяване на всичките му възможни атрибути. С всеки обект могат да се свършат операции. Понятието клас е основно в ОО-подхода. Той обхваща данните и алгоритмите, описващи структурата и поведението на "нещата от реалния свят.

Суперклас е съвкупност от класове, а подклас е екземпляр на клас. Съществуват и йерархии от класове като подкласовете наследяват атрибутите и операциите на суперкласовете, но могат да имат и специфични за тях.

Взаимодействие чрез съобщение – изпълнява се посочената в съобщението операция и следователно в такъв случай управление на обекта получател. Така се описва поведението на обектите и на ОО-системите.

Четири важни характеристики: Абстракция, Енкапсулация, Наследяване, Полиморфизъм.

(9) Езици за описание UML

Езиците за описание могат да се разделят на две основни групи – GPL – език с общо предназначение и DSL – език, специфичен за домейн. Отделно спрямо степента на формалност се разделят на формални и неформални.

DSL е специализиран език, създаден за конкретен тип задачи. GPL са създадени, за да се решат най-различни програми с нашата и да е логика.

ADL е език за описание на софтуерни архитектури и е подгрупа на DSL.

UML е графичен език за визуализация, специфициране, конструиране и документиране на елементите на една софтуерна система. Има много различни видове UML диаграми, най-популярните, от които са: Use case, class, Activity, State Machine и др.

(10) Верификация и валидация на софтуера. Тестване на софтуера. Управление на процеса на тестване.

Тестването е изследване на програмите за установяване на съответствието им с различни по степен на формализираност характеристики, правила и изисквания.

Дейностите настройване и тестване не са една и съща. След настройването е ясно, че програмата решава някакъв проблем, но тестването доказва, че програмата решава точно определен проблем (емпиричен). Тестването се осъществява на няколко етапа:

- Планиране - определя се целта на тестването, какво да се тества, кога, с какви данни, как и кой да го изпълни.
- Реализация на тестването - описват се предусловия за тестване.
- Отчитането се изпълнява след документиране на резултатите и сравняване на критериите за изчерпателността и обхвата на тестване.

Има два подхода за тестване: модулярен структурен и функционален. Целта на структурното тестване е да се изберат такива тестови данни, че да се осигури преминаване през всички програмни части на системата. При функционалното тестване се проверява правилността на реализираните основни функции.

(11) Управление на качеството на създаване на софтуер

Качеството е съвкупността от средства и характеристики на даден продукт или услуга, носители на способността му да отговори на явно или неявно указани нужди.

Има различни модели за управление на качеството. Моделът Боем се концентрира главно в/у поддръжката на софтуера и възможността за лесно възпроизвеждане.

(12) Съвременни софтуерни технологии. Гъвкави (agile) софтуерни технологии.

Причини: Бързо променящи се пазарни условия. ~~Пълно отно~~ Променено отношение на разработчиците към стила на работа.

Принципи:

- Активно включване на потребителите в процеса на разработка на софтуера.
- Реализация на инкрементално разработване - реализация на разработването се разработване, като изискванията за вс. следваща итерация се преформулират въз основа на оценяване на текущата.
- Регламентиране на стил на работа, основан на компетентност и екипност.
- Приемане на неизбежността на промените.
- Придържане към опростени конструкции и схеми на работа.

Извод: ГМ не предлагат универсално решение на проблемите в софтуерното производство.

(13) Extreme Programming (XP)

Същност - лека методология за малки и средни колективи разработващи софтуер при нечести или бързо променящи се изисквания.

Ангажименти:

- Чрез програмистите – няма да им се налага да се справят с тежките ситуации сами, ще имат възможност да допринасят максимално за успеха на проекта, ще им се налага да вземат решения от своята компетентност и никога повече, за които нямат квалификация.
- Чрез клиентите – на всеки няколко седмици ще получават конкретни използваеми резултати, колкото предварително са дефинирани (т.е. release), ще имат възможност при необходимост да променят посоката на проекта по време на реализацията му, без това да бъде свързано с прекалено големи разходи.

(14) Test Driven Development

TDD е метод за разработка на софтуер, при който се спазва следния работен цикъл:

- първо се пишат тестови варианти, които да покриват изискванията за новия изходен код.
- след това се пише програмния код така, че да покрива тези тестове.

(15) Feature-driven development

FDD е итеративен и инкрементален процес на разработка на софтуер. Това е лек или гъвкав метод за разработка. Съчетава редица признати от индустрията най-добри практики в един метод. Тези практики имат като отправна точка перспективата за функционалността от гледна точка на потребителя/клиента. Неговата главна цел е да доставя работещ софтуер през уговорени времеви интервали.

(16) SCRUM

SCRUM е итеративна, инкрементална рамка за управление на проекти, често при Agile методология. Той се фокусира в/у управлението на софтуерни проекти и може да бъде използван, за да задвижва екипи за софтуерна поддръжка или като общ подход за проектов/програмен мениджмънт.

SCRUM се състои от отделни итерации наречени спринтове. Те могат да имат продължителност от 1 до 4 седмици. В края на всеки спринт, екипът разполага с работеща версия на продукта, която включва всички готови задачи от backlog-а