Проектиране и Тестиране на Софтуер
ТУ, кат. КС, летен семестър 2012

Лекция 21

Тема:

Инженерен подход за разработка на програмни продукти

Съдържание:

- Грешки в ПО
- Надеждност на ПО
- Жизнен цикъл на програмен продукт
- Етап проектиране – методи
- Етап проектиране – техники

Част 1

Грешки и Надеждност на ПО
Жизнен цикъл на програмен продукт

Инженерни методи и ПО

- Тенденции в развитието на компютърната техника
- Релация хардуер - софтуер

Инженерни методи и ПО

- Отначало компютърните системи бяха наситени с “много” хардуер и “малко” софтуер и определящ фактор за цената бе стойността на апаратурата. Съвременните изчислителни системи са изградени от евтини интегрални схеми и са обезпечени с разнообразно програмно осигуряване на различни езикови нива. Цитират се данни, според които разходите за изграждане на програмно осигуряване надхвърлят 50% и достигат 60-70% от стойността на една изчислителна система.

Софтуерна криза

Разликата в тенденциите при развитието на хардуера и софтуера наложи представата за ***софтуерна криза*** и необходимостта от нейното разрешаване. Развиха се методики за повишаване ефективността на програмисткия труд.

Въведе се концепцията за ***жизнен цикъл на програмните системи***.

Софтуерна криза

- Начини за разрешаване
  - Тотално проектиране
  - Възходящо проектиране
  - Низходящо проектиране
  - Модулно програмиране
  - Структурно програмиране
  - ООП
  - Компонентно, визуално и събитийно програмиране

Софтуерна криза

- Основна задача за разрешаване:
  - Осигуряване на качество
  - Разработка на надеждно ПО

Софтуерна криза

Една от проявите на софтуерната криза се свързва с оплакванията, че ПО е скъпо и ненадеждно.

Експлоатацията му се съпровожда с проява на грешки и това налага допълнителни разходи на средства за тестиране и съпровождане на програмните продукти.

Възложител и разработчик на ПО влагат различен смисъл в това понятие, а те се намират от различни страни в процеса на разработка и експлоатация на ПС.

Скъпо и ненадеждно ПО

- Скъпо, т.к. Е ненадеждно
- Грешки, които се проявяват при изпълнение
- Това налага разходи за
  - тестиране и
  - съпровождане

Грешки в ПО

- Необходими са ясна дефиниция и стриктно описание на:

Що е грешка в ПО?

- За избягване на недоразумения:
  - Потребител:
  - Разработчик:

Пример

USA: Ballistic missiles early warning system

Различните субекти разбират грешките по различен начин.

Грешка в ПО

- Опит за дефиниция

Определение 1: ПО съдържа грешка, ако поведението му не съответства на спецификациите за реализацията му.

Това определение не е изрядно, тъй като неявно се предполага коректност на спецификацията.

Грешка в ПО

- Опит за дефиниция

Определение 2: ПО съдържа грешка, ако поведението му не съответства на спецификациите при експлоатацията му за изменение стойностите на определени параметри в зададени граници.

Недостатък на тази дефиниция е, че ПО трябва да има адекватно поведение дори и в случаи, когато стойността на даден параметър при изпълнение излезе извън зададените граници.

Грешка в ПО

- Опит за дефиниция

Определение 3: ПО съдържа грешка, ако поведението му не съответства на официалната документация, предоставена на потребителя.

Това определение също е некоректно. Възможна поява на грешки както в ПО, така и в съпровождащата документация.

Грешка в ПО

- Коректната дефиниция

Определение 4: **ПО съдържа грешка, ако то не изпълнява това, което потребителят разумно очаква от него.**

Наличието на грешки в ПО се определя като неспособност на програмния продукт да действа в съответствие с изискванията на потребителя.

Отказите при експлоатацията на ПО са проява на грешки при работата на програмите.

Надеждност на ПО

- Определение: ***Вероятността за безотказна работа в течение на определен период от време.***
  Надеждността на ПО не е само вътрешно свойство на програмите. Тя е свързана повече с това как се експлоатира програмата или каква е вероятността потребителят да подаде набор входни данни, които да предизвикват отказ и авариен изход на програмата при изпълнение.

Диаграми

- Надеждност на хардуера, честота на отказите
- Надеждност на софтуера, честота на отказите

Надеждността на хардуера и софтуера се проявяват по различен начин като функция на времето.

Диаграми

- Експлоатацията на хардуера в началния период се характеризира с висока честота на отказите. Следва дълъг период на експлоатация с ниско ниво на отказите, след което тяхната честота отново се повишава. Честотата на отказите в хардуера зависи от времето и не зависи от входните данни, с които се изпълняват програмните продукти.

Диаграми

- Честотата на отказите в софтуера непрекъснато намалява. Това е вярно, ако изчистването на една грешка не води до вмъкване на нови грешки. Честотата на отказите в софтуера не зависи от времето в горния смисъл. Проявлението на грешките в една програмна система съществено се определя от входните данни.

Скъпо, ненадеждно ПО

- Стойност, разпределение на разходите
  - Програмиране - 5%
  - Проектиране - 20%
  - Тестиране - 25%
  - Съпровождане - 50%

Грешки

- Компилация
- Изпълнение
- Определение и формулиране на задача

- Грешки със значими последици
- Грешки с незначителни последствия

Примери за значими грешки

- Фортран програма на САЩ полет до Венера
  - Резултат
  - Цена
  - Отговорност
  - Причинител на грешката
  - Причина за грешката

Примери за значими грешки

- СССР полет до Марс

Примери за значими грешки

- Европейски проект Ариана

Примери за значими грешки

- Франция, научен експеримент по метеорология

Примери за значими грешки

- Банково дело и банков софтуер

Извод:

- Наличието на грешки поставя въпроса за отстраняването им и създаване на надеждни програмни продукти.

Усилията следва първо да се насочват към анализ и отстраняване на причините за грешките и след това към разработка на ефективни методи за локализиране и отстраняване на грешки.

Единствената важна причина за грешките в ПО е ***неправилният превод на данни от едно представяне в друго***.

Извод:

- Преди да се разработват методи за избягване на грешките, трябва да се разберат причините за грешките.
- Процесът на създаване на ПО да се обвърже с борба срещу причините за грешките.
- В този смисъл: единствената най-важна причина за грешките в ПО – това е неправилният превод (трансформация) на данни от едно представяне в друго представяне.

Данни и Информация

Данните са сурови факти, изолирани и независими. Те носят смисъл и значение, но сами по себе си не са полезни.

Информацията са обработени и трансформирани данни с цел те да бъдат полезни на потребителя. Информацията казва на хората нещо, което те не знаят или потвърждава нещо, което те подозират.

Дни на Майкрософт’ 2012

Explosion of data

60% increase per year

15% out of those 60% - structured data

85% out of those 60% - unstructured data

Данни и Информация

Изобщо, информация се получава като резултат или функция от обработката на данни.

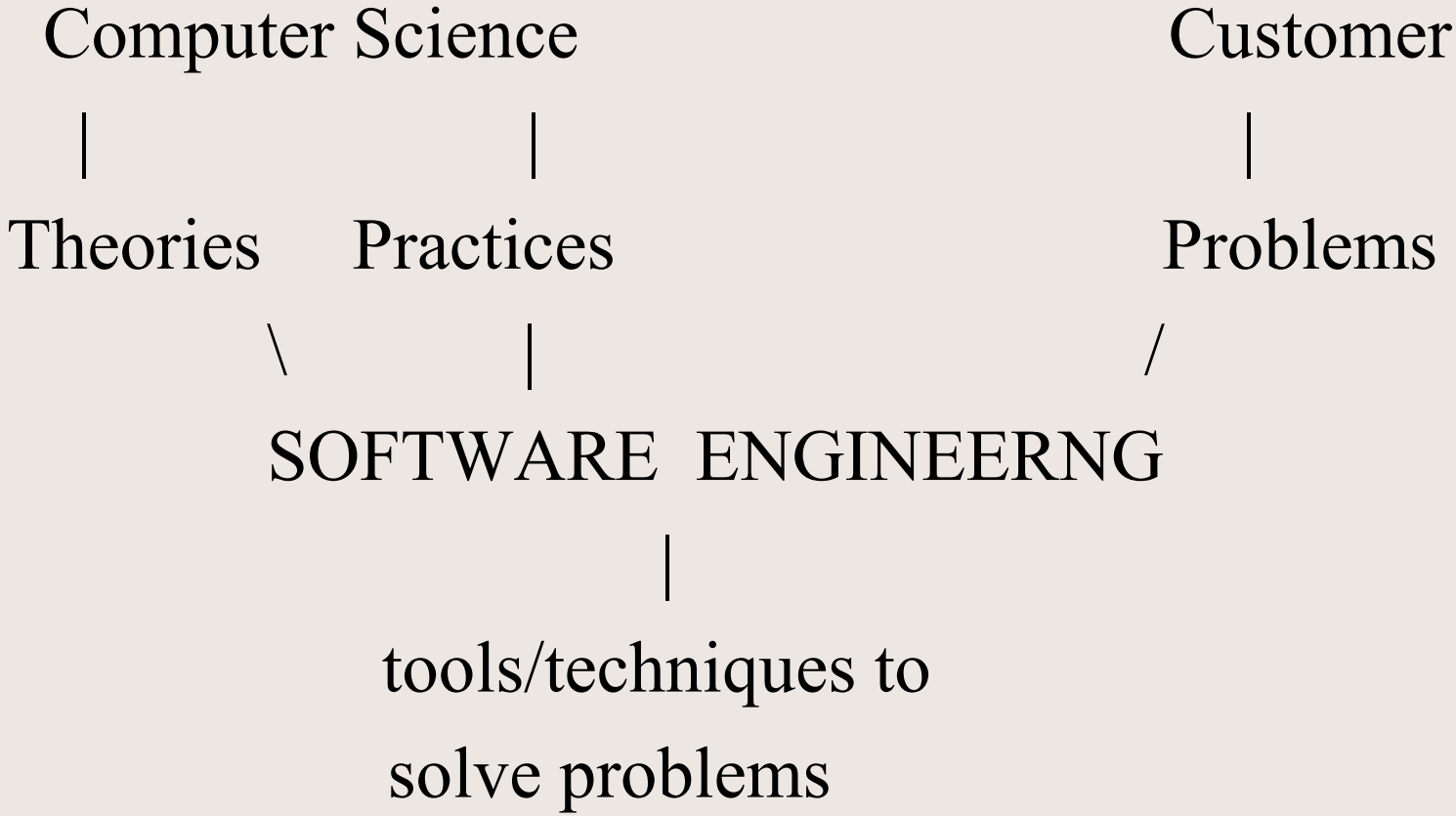
INF = function(Data, Processing)

Възможно е информацията от един процес да послужи като данни за обработка при втори, следващ процес.

Софтуерно Инженерство

- СИ е дейност, при която хората прилагат своите знания по компютри и ИТ за решаване на проблеми
- СИ в два етапа:
  - Анализ
  - Синтез

Relation COS & SE



Жизнен цикъл на ПС

От гледна точка на концепцията за преобразуване на данни (информация) от едно представяне в друго жизненият цикъл на една програмна система (от идеята за нейното създаване през реализацията и експлоатацията й до изхвърлянето й от употреба) се описва със следните етапи:

Жизнен цикъл на ПС - 1

1. Дефиниране (специфициране) на проблем за решаване;
2. Изготвяне на задание;
3. Проектиране;
4. Програмиране;
5. Тестване и настройка;
6. Демонстрация и предаване на клиента (краен потребител);
7. Експлоатация и съпровождане.

ЖЦПС

Създаването на ПО се описва като верига от процеси за преобразуване на данни, започващи от описанието (спецификацията) на задачата и завършващи с работоспособна програма (инструкции за компютър) за решаването на задачата.

Програмирането е само етап при решаването на задачата.

ПО се явява съвкупност от информационни елементи (данни и команди), описващи решението на проблема.

ЖЦПС

Постановката с етапи в ЖЦПС позволява да се посочат източниците за грешките в ПО. Приема се, че създаването на ПО е съвкупност от процеси на транслация (трансформация) на данни от едно представяне в друго. Става дума за превод на първоначално формулирана задача в различни междинни решения до получаването на програма като набор машинни команди. Тогава причините за грешките, които се локализират в окончателния продукт се дължат на ***непълен и/или неточен превод на данни от едно междинно представяне в друго междинно представяне***.

ЖЦПС

Ето защо грешките не винаги са вътрешно присъщи на програмите. Причините за грешките могат да се формулират като:

1. Неточности при преход от описание на проблема към формулиране на задание;
2. Неточности при преход от задание към изготвяне на проект;
3. Неточности при преход от готов проект към реална програмна система.

ЖЦПС - 2

1. Системен анализ (Systems Analysis);
2. Системно проектиране (Systems Design);
3. Системна реализация (Systems Implementation);
4. Системна поддръжка (Systems Support).

Системният анализ е началният етап от жизнения цикъл. Той включва изучаване на текущото състояние на проблема, определяне на специфични потребности и изисквания, начална оценка на алтернативни решения. Този етап завършва с изготвяне на задание за проектиране, което включва уточнените изисквания на потребителя към системата.

Системното проектиране завършва със спецификация за реализация. Тази спецификация съдържа общо и детайлно описание на компютърното решение, избрано при анализа. Формулираните спецификации се дават на програмистите за реализация.

Системната реализация включва създаването на работеща програмна система. Програми се пишат, тестват и настройват. Бъдещите потребители се обучават за работа с новата програмна система.

Ситемната поддръжка включва експлоатация и съпровождане след внедряване на системата. Възможно е чрез обратна връзка с програмистите да се ре-програмират отделни модули за подобряване характеристиките на системата. Допуска се този етап да завърши с формулиране на изисквания за разработка на нова система, с което цикълът анализ, проектиране, реализация и поддръжка се затваря.

ЖЦПС - 3

1. Формулиране на изисквания (Requirements phase) – 3%;
2. Описание на проблема (Specification phase) – 4%;
3. Планиране (Planning phase) – 2%;
4. Проектиране (Design phase) – 6%;
5. Реализация (Implementation phase) – 12%;
6. Интегриране (Integration phase) – 6%;
7. Поддържане/съпровождане (Maintenance phase) – 67%.

Относителният дял на всяка една фаза е представен в %.

Цел на фаза формулиране на изисквания е в диалог (интервю) с потребителя да се определят възможно най-точно неговите потребности – от каква програмна система се нуждае той. Фазата се нарича още системен анализ.

Цел на фаза описание на проблема е след определяне на потребностите да се създаде документ, описващ проблема, който точно отразява какво следва да прави програмният продукт.

Цел на фаза планиране е оценка на необходимото време и ресурси за изпълнение на проекта – каква продължителност, колко проектанти и програмисти, какви разходи ще изисква проектът.

Цел на фаза проектиране е следваща детайлизация. Ако във фаза описание се казва ***какво*** трябва да прави продуктът, сега при проектирането се създава документ, който предписва ***как*** продуктът ще постигне целта си. Продуктът се разбива на съставни компоненти, всеки от които се проектира.

Цел на фаза реализация е програмиране на продукта. Изборът на инструментален програмен език се фиксира в документа за описание на проблема. Всеки модул се кодира и тества.

Цел на фаза интегриране е свързване на отделните модули в завършен продукт. Тази фаза завършва с успешно провеждане на тест за приемане, с който клиентът потвърждава, че продуктът отговаря на изискванията в документа за описание.

Цел на фаза поддържане/съпровождане е отразяване на промени в продукта, след като е предаден на потребителя. Промените могат да се предизвикват поради отстраняване на възникнали грешки (bugs, corrective maintenance), или поради необходимост от добавяне на нови функционални възможности на продукта (enhancement maintenance).

ЖЦПС - 4

- SDM
  - Specify the problem requirements
  - Analyze the problem
  - Design the algorithm to solve the problem
  - Implement the algorithm
  - Test and verify the completed program
  - Maintain and update the program

ЖЦПС - 5

- SDM
  - Requirements analysis and definition
  - System design
  - Program design
  - Program implementation
  - Unit testing
  - System delivery
  - Maintenance

Опростен ЖЦПС - 6

- SDM (според E.Petroutsos, pp28):
  - Decide what the application will do and how it will interact with the user.
  - Design the application's user interface according to requirements of step 1.
  - Write the actual code behind the events you want to handle.

ЖЦПС - 7

- **Al Aho, aho@cs.columbia.edu**

- **The Quintessential Questions of Computer Science**
- **November 14, 2007**

ЖЦПС - 7

The Software Development Process

- **Specification**
  - **Define system functionality and constraints**
- **Validation**
  - **Ensure specification meets customer needs**
  - **“Are we building the right product?’’**
- **Development**
  - **Produce software**
- **Verification and testing**
  - **Ensure the software does what the specification calls for**
  - **“Are we building the product right?’’**
- **Maintenance**
  - **Evolve the software to meet changing customer needs**
- **Quality plan**
  - **Ensure product meets user needs**

ЖЦПС - 7

Where is the Time Spent?

- **1/3 planning**
- **1/6 coding**
- **1/4 component test and early system test**
- **1/4 system test, all components in hand**

  - **“In examining conventionally scheduled projects, I have found that few allowed one-half of the projected schedule for testing, but that most did indeed spend half of the actual schedule for that purpose.”**
    - **F. B. Brooks,The Mythical Man-Month, 1995.**

ЖЦПС - 7

Why Do Software Projects Fail?

- **Unrealistic or unarticulated project goals**
- **Inaccurate estimates of needed resources**
- **Badly defined system requirements**
- **Poor reporting of the project's status**
- **Unmanaged risks**
- **Poor communication among customers, developers, and users**
- **Use of immature technology**
- **Inability to handle the project's complexity**
- **Sloppy development practices**
- **Poor project management**
- **Stakeholder politics**
- **Commercial pressures**

ЖЦПС - 7

Ingredients for Making Reliable Software

- **Good people/management/communication**
- **Good requirements/modeling/prototyping**
- **Sound software engineering practices**
- **Use of mature technology**
- **Thorough testing**
- **Verification tools**
  - **–model checkers**
  - **–theorem-proving static analyzers**

ЖЦПС

- Обобщение
  - ЖЦПС е процес
  - Процесът ЖЦПС се илюстрира с редица модели на неговото изпълнение:
- Build and Fix model
- Waterfall model
- Rapid Prototyping model
- Spiral model

Създай и фиксирай (Build and Fix model)

Продуктът се създава и предава на клиента без да се провеждат фази описание на проблема, планиране и проектиране. Клиентът посочва какви евентуални промени да се направят в продукта. Промените се правят, след което продуктът се предоставя на клиента. Този процес продължава, докато изискванията на клиента се удовлетворят. Моделът е приложим за малки по обем проекти като студентски задачи и учебни демо примери.

Водопад (Waterfall model)

Този модел се нарича още лавинообразен или каскаден. Последователно се изпълняват фазите от жизнения цикъл. Резултатът от всяка фаза се подлага на проверка за валидност (verify and test) и само след положителен резултат се пристъпва напред към следваща фаза. Особено внимание се отделя на диалога с клиента за получаване на необходимата информация за определяне на изискванията и формулиране на документа за описание на проблема.

Предимство на модела е наличие на документация, съпътстваща всяка фаза.

Недостатък на модела е, че по време на отделните фази клиентът не разполага с работеща макар и неокончателна версия на продукта, а разполага само с документация, от която слабо се информира за потенциалните възможности на разработвания продукт.

Водопад (Waterfall model)

- Идеален водопаден модел

Водопад (Waterfall model)

- Реален водопаден модел

Водопад (Waterfall model)

- Модифициран водопаден модел

Бърз прототип (Rapid Prototyping model)

При този модел първата фаза за определяне на изискванията освен с данни за съставяне на документ за описание на проблема, завършва и с предварителна груба и примитивна реализация на продукта, наречена бърз прототип (rapid prototype). Клиентът получава възможност за визулана представа относно бъдещия продукт и реална оценка на функционалните му възможности. Така той преценява дали се удовлетворяват или не съответните изисквания. Клиентът експериментира с прототипа и прави забележки, които се отразяват непосредствено в прототипа. Този процес трае, докато клиентът бъде удовлетворен. Едва сега се създава описанието на проблема, който става спецификация за реализация на продукта.

Спирала (Spiral model)

Основно внимание се отделя за оценка на рисковете, свързани с реализацията на продукта и извършване на действия, които разрешават проблемите и водят до минимизиране на рисковете.

Две условия обуславят прилагане на спирален модел:

Наличие на голям по обем проект,

Клиент и разработчик на проекта да са от една организация.

Спирала (Spiral model)

Възможни рискове:

Опасение, че крайният продукт няма да удовлетвори изискванията на клиента. Удачно решение за разрешаване на този риск е да се създаде и предостави своевремнно на клиента работещ прототип на бъдещия продукт.

Опасение, че разходите за реализация на продукта няма да се възстановят. Удачно решение за разрешаване на този риск е да се провежда преди началото на всяка нова фаза прецизен икономически анализ и по-нататъшната реализация на продукта да продължава само след положителна икономическа прогноза.

Период на ЖЦПС

Времето от

- Момента Създаване на документ с изисквания до
- Момента Предоставяне на програмния продукт

- Цел: минимизация на периода
- Системата се проектира така, че се реализира на части с определена функционалност, които се предоставят на потребителя, а разработчиците работят по следваща версия. Така, ако се развива версия *n+1*, потребителят борави с версия *n*.

ЖЦПС

- Подходи:
  - Инкрементален
  - Итеративен

Инкрементален подход

- Системата е описана в изискванията като разделена на подситеми по фукционалност. Реализациите започват с имплементация само на една подсистема, след което всяка нова версия се допълва с **инкремент** на нова подситема. Пример: 3 подсистеми 1,2,3
- А: 1
- Б: 1 + 2
- В: 1 + 2 + 3

Итеративен подход

- В самото начало се доставя система с всички функционални подсистеми, а всяка следваща версия итеративно сменя една или друга подситема (например по-добър алгоритъм)
- А: 1 + 2 + 3 (прости, примитивни версии)
- Б: 1 + 2 + 3 (подобрени версии)
- В: 1 + 2 + 3 (подобрени версии)

ЖЦПС - принципи

- Програмната система е предназначена за крайния потребител.

ЖЦПС - принципи

- Жизненият цикъл се състои от фази (етапи). При големи проекти тези фази се разбиват на подетапи – задачи.

ЖЦПС - принципи

- Създаването на програмна система не е строго последователен процес. Възможно е отделни фази да се застъпват и/или препокриват.

ЖЦПС - принципи

- Създаването на програмна система струва пари – капитално вложение.
- Това налага анализ на алтернативни решения и по критерий икономческа ефективност.

ЖЦПС - принципи

- Без колебание канцелирайте неудачен проект!

ЖЦПС - принципи

- Документацията на програмния продукт се съставя по време на всички етапи.

ЖЦПС – действащи лица

Класическа, наложила се с годините схема

- Възложител
- **Проектант**
- **Разработчик**
- Краен потребител

ЖЦПС – действащи лица

- Проектант – следните синоними на англ.
  - Systems analyst, Systems designer
  - Systems consultant, Management consultant
  - Operations analyst, Information analyst
  - Data analyst, Business analyst
- Разработчик (developer)
  - Системен програмист
  - Приложен програмист

ЖЦПС – действащи лица

Програмистите и проектантите са специалисти с различна квалификация.

Какви са изискванията към програмиста?

Той/Тя трябва да:

- владее програмен език или програмни езици,
- конкретна операционна система,
- набор ситемни помощни програми.

ЖЦПС – действащи лица

Каква квалификация е нужна на проектанта?

- Наред с уменията на квалифициран програмист Той/Тя следва да е с богата професионална култура:
  - БД,
  - Компютри, Компютърни мрежи и предаване на данни,
  - Софтуерни архитектури,
  - Софтуер за компютри – ПЕВН и ОС,
  - Машинна графика,
  - Сигурност на данни,
  - ИИ,
  - Експертни системи
  - . . .

ЖЦПС – действащи лица

Каква квалификация е нужна на проектанта?

- Проектантът следва да бъде комуникативен. Много важно е да бъдат разбрани и правилно формулирани изискванията на възложителя/потребителя, а това зависи от умението на проектанта да общува.

ЖЦПС – действащи лица

Схеми за обединяване на специалистите в екипи (колективи):

1. Бригада на водещия програмист (chief programmer team). Това е екип с главен програмист и редови програмисти.
2. Екип с двама ръководители. Той включва главен програмист (team leader) и управител (team manager);
3. Екип с йерархична подчиненост на три нива. Тези нива включват ръководител на проект (project leader), един или няколко ръководители на екипи (team leader) и редови програмисти (programmer).

ЖЦПС – действащи лица

Съвременна схема (Ken Lunn, pp73)

- Възложител
- **Steering Group(s) – управителен съвет**
- **Project Manager - Проектант**
- **Other Roles – специфични длъжности**
- Краен потребител

ЖЦПС – действащи лица

- **Steering Group(s) – управителен съвет**

Функции:

- Установява генералните цели на проекта;
- Управлява финансите/бюджета на проекта
- Следи рисковете за проекта (най-високо ниво)

ЖЦПС – действащи лица

- **Project Manager – Проектант**
  - Докладва на управителния съвет,
  - Отговорен за координацията на всички елементи на проекта, например
    - Проектът е планиран,
    - Проктът е ресурсно осигурен,
    - Оценени и адресирани са рисковете на проекта,
    - Други.

ЖЦПС – действащи лица

- **Other Roles – специфични длъжности**
  - Programmers
  - Database architects
  - System designers
  - Testers
  - Help Desk Operator
  - Middle management
  - Architects
  - Business analysts
  - Systems analysts
  - Designers

ЖЦПС – eXtreme Programming

XP - eXtreme Programming е подход за създаване на софтуер, основан на принцпите на простота, общуване, обратна връзка и поощрение.

XP е приложим при малки екипи, които създават софтуер бързо в обкръжение на постоянно изменящи се изисквания (непълнота, изменчивост, противоречивост)

XP – 12 свойства

- Planning Process
- Small Releases
- Metaphor
- Simple Design
- Testing
- Refactoring
- Pair programming
- Collective Ownership
- Continuous Integration
- 40-hour Week
- On-site Customer
- Coding Standard

XP – 12 свойства

- Planning Process

  Възложителят, по данни на програмистите прави бизнес оценка на проекта и фиксира кое да се прави и кое да се отложи
- Small Releases

  Тимът прави работеща версия бързо и в кратки срокове актуализира с нови версии
- Metaphor

  Ползват се конвенции за общи имена и стандарти за описание на ПС

XP – 12 свойства

- Simple Design

  Прави се най-простата възможна версия без предпоставки за бъдещи разширения
- Testing

  Валидация на ПС по всяко време

  Test-driven software development

  Acceptance test, provided by customer
- Refactoring

  Подобрения в кода без нарушаване на функционалността

XP – 12 свойства

- Pair programming

  Работа на групи по двама
- Collective Ownership

  Създаденият код е с равни права на собственост от целия екип
- Continuous Integration

  Интегриране на системата от нейните компоненти многократно всеки ден. Всеки от групата е в курса на делата актуално

XP – 12 свойства

- 40-hour Week

  Свежи, в добро здраве и ефективни
- On-site Customer

  Налице е възложител, който определя изисквания, приоритети и отговря на програмистки въпроси with less hard-copy documentation
- Coding Standard

  Всички програмисти пишат кода в един стил, по едни и същи правила

Част 2

Жизнен цикъл на програмна система

Етап проектиране – методи

Етап проектиране – техники

Съдържание:

- Виж следваща лекция 22

Благодаря
За
Вниманието