2.Софтуерни процеси(СП) и модели(М) на софт. п-си(МСП)- Резюме

Софтуерният процес(П): Структуриран набор от **дейности**, **необходими за разработване на софтуерна система(С)**. Има много различни СП, но всички те включват:

▪ **Спецификация** – определяне това, което С трябва да прави;

▪ **Дизайн и имплементация** – определяне организацията и имплементиране на С;

▪ **Валидация**– проверка дали С прави това, което клиентът иска;

▪ **Еволюция** – променя С според променящ. се нужди на клиента

МСП е абстрактно представяне- описание на п-са от определена перспектива.

Описания на СП – чрез дейностите в тези процеси- специфициране на М на данни, проектиране на потребителски интерфейс и т.н., както и за последователността на тези дейности. Описание на п-сите вкл.:

▪**Продукти** – резултатите от дадена дейност в П;
▪**Роли** – отразяват отговорностите на хората, участващи в П;
▪**Пред- и постусловия** – изявления, които са верни преди и след изпълнението на дадена дейност или създаването на продукт.

Планово-ориентирани СП– всички дейности се планират **предварително**, а **напредъкът се измерва спрямо този план**.

Гъвкави (Agile) СП – планирането е итеративно и П лесно може да бъде променен**,** за да отразява променящ. се изисквания на клиента.

Повечето реални СП включват елементи както от планово-ориентирания, така и от гъвкавия. Няма правилни или грешни СП.

Модели на софтуерния процес:

Водопаден: Планово-ориентиран. Отделни и последователни фази на спецификация и разработка.

Инкрементално разработване: Спецификация, разработка и валидация се извършват паралелно. Планово-ориентиран или гъвкав.

Програмиране чрез повторна употреба: преизползване на съществуващи компоненти. Планово-ориентиран или гъвкав.

Дейности в процеса: технически, съвместни и управленски дейности с цел да се специфицира, проектира, имплементира и тества С. 4 осн. дейности на СП – спецификация, разработка, валидация и еволюция – организирани по различен начин в различни процеси на разработка. **В водопадния - организирани последователно, в инкременталния- преплетени.**

Софтуерна спецификация - установяване какви **услуги са необходими и какви са ограниченията** върху работата и разв. на С.

Вкл. Процес на **инженеринг на изискванията**:

▪**Проучване на възможността**- Технически и финансово възможно ли е да се изгради С?

▪**Извличане и анализ на изискванията(И)**-Какво изискват или очакват заинтересованите страни от С?

▪**Спецификация на изискванията**-Детайлно дефиниране на И

▪**Валидация на изискванията**-Проверка на валидността на И

Софтуерен дизайн и имплементация - П на преобразуване на спецификацията на С в изпълнима С.

Софтуерен дизайн- Проектиране на софтуерна структура, която реализира спецификацията;

Имплементация: Превод на тази структура в изпъл. програмен код;

Дейностите по дизайн и имплементация са тясно свързани и могат да бъдат преплетени.

Дизайнерски дейности:

-**Архитектурен дизайн**- определя цялостната структура на С, осн. компоненти, техн. взаимоотношения и как те ще бъдат разпределени.

-**Дизайн на интерфейси**- интерфейсите между компонентите на С.

-**Дизайн на компоненти**- всеки компонент на С се проектира как ще функционира.

-**Дизайн на бази данни**-проектират се структурите на данни на С и как те ще бъдат представени в база данни.

Софтуерна валидация

-**Проверка и валидация (V & V)** - да покаже, че С отговаря на спецификацията си и удовлетворява И на клиента. Вкл. проверки, прегледи на процесите и тестване.

-**Тестване на системата** - изпълнение на системата с тестови случаи, които са изведени от спецификацията на реалните данни, които ще бъдат обработвани от С. Тестването използваната дейност в V & V.

Етапи на тестване

Тестване на разработка или компоненти: Индивидуалните компоненти се тестват независимо; Компонентите могат да бъдат функции или обекти, или свързани групи от тези елементи.

Тестване на системата: Тестване на С като цяло. Тестването на възникващите свойства е особено важно.

Приемателно тестване: Тестване с данни от клиента, за да се провери дали С отговаря на нуждите на клиента.

Еволюция на софтуера:

Софтуерът е гъвкав и може да се променя. Ако И се променят поради промени в бизнеса, С също трябва да еволюира и да се променя. Разделение между разработката и еволюцията (поддръжката) става все по-незначително, тъй като все по-малко С са напълно нови.

Процес- последователност от стъпки вкл. дейности, ограничения и ресурси, които осигуряват постигането на някакъв вид резултат. Процесът е повече от процедура – Процесът е съвкупност от процедури, организирани така, че да се изграждат продукти, задоволяващи определени цели и стандарти

Характеристики на процес(П):

• описва всички важни основни дейности

• използва ресурси, които най-често са ограничени

• създава междинни и крайни работни продукти

• може да е съставен от подпроцеси, с йерархии и връзки м/у тях

• представлява **последователност от дейности**:

• съществуват входни и изходни критерии за всяка дейност и по този начин е ясно кога започва и кога свършва отделна дейност

• съществуват ръководни принципи, вкл. цели на всяка дейност

• съществуват ограничения за всяка дейност, били те по отношение на ресурсите или на работния продукт, очакван от дейността.

Значение на процеса: Две крайни становища:1/ П е нищо (Hero tendency) 2/ П е всичко

Рамка на процеса - общ модел за организиране и изпълнение на софтуерния процес, като включва различни дейности и задачи, които спомагат за успешното разработване на софтуерни продукти. Вкл:

-Основни дейности (Framework activities) – малък брой, които са приложими за всички софтуерни проекти. **Основни дейности**: Комуникация; Планиране; Моделиране; Конструиране; Внедряване

-**Действия**– съвкупност от свързани задачи, които водят до значим работен продукт. Те могат да се разглеждат като подетапи в рамките на дад. основна дейност:

-**Работни задачи (Work tasks)**– изпълняване на част от работата, която е дефинирана от **действието**

-Допълнителни дейности (Umbrella activities) – прилагат се по време на целия СП и не са ограничени до определен етап. Те осигуряват допълнителна подкрепа и управление на процеса. Примери за такива дейности включват: Управление на проекти; Контрол на качеството; Управление на конфигурацията; Управление на рисковете; Документиране

Основни дейности:

Комуникация: събиране и разбиране както на **изискванията** за функционалността на софтуера, така и на **ограниченията**

Планиране: **план** за бъдещата работа по разработка на софтуера; Описват се: техническите рискове; потенциалните рискове; необходимите ресурси; работните продукти; врем. график на работа

Моделиране:

• **Действие Анализ**

Работни задачи: Събиране на изисквания; Уточняване; Договаряне; Специфициране; Валидиране;

Работни продукти: М на анализа; Спецификация на И

•**Действие Проектиране**

Работни задачи: Дизайн на данните, Дизайн на архитектурата, Дизайн на интерфейс, Дизайн на ниво компоненти

Работни продукти: М на дизайна; Спецификация на дизайна

•**Конструиране**

Генериране на код – ръчно или автоматично

Тестване: на самостоятелни компоненти, на интегрираната система от компоненти (модули, подсистема); потребителско (бета-тестване)

•**Внедряване**: Софтуерът се предоставя на клиента; Клиентът оценява продукта: Забележки; Препоръки

Допълнителни дейности:

• Следене и управление на софтуерния продукт

• Управление на риска (Risk management)

• Осигуряване на качеството (Software quality assurance)

• Формални технически прегледи (Formal technical reviews)

• Измерване (Measurement)

• Управление на софтуерната конфигурация

• Управление на повторното използване

• Подготовка и генериране на работни продукти

МСП: Опростено описание на начина на разработване на софтуера, представено от определена гледна точка

Цели на МСП:

• Формиране на общо разбиране у участниците в разработването на софтуер за дейностите, ресурсите и ограниченията;

• Намиране на несъответствия, излишества и пропуски в П от разработващия екип- подобряване П;

• Намиране и оценяване на подходящи дейности за постигане на целите на П;

• Адаптиране на общ. П към отделна ситуация, в която ще се приложи

Разлики между МСП

• Общият поток от дейности и задачи и зависимостите между тях

• Степента до която са дефинирани работните задачи в рамките на всяка основна дейност

• Степента до която са дефинирани и изисквани работни продукти

• Степен на детайлност и строгост, с които П е описан

• Степен до която купувачът и другите заинтересовани лица са включени в проекта

• Степен на автономност, която се дава на софтуерния екип в проекта

• Степента на описание структурата на екипа и отделните роли

Шаблони за описание на процес

Шаблон(Ш) е описание на общо решение на общ проблем или въпрос, на базата на което може да се извлече детайлно решение на

специфичен проблем. Ш, свързани с разработката на софтуер: Ш на анализа, Ш на проектирането, организационни Ш, Ш на процес и др.

• **Шаблон на процес(ШП)** - описва доказан, успешен подход и/или последователност от действия за разработване на софтуер. Представлява структурирано описание на П, което е метод за описание на важните х-ки на СП. Важна х-ка на ШП е, че той **описва какво трябва да се направи, а не точни детайли как трябва да се направи**. Ш се дефинират на разл. нива на абстракция

Пример за шаблон(Ш): • Име на Ш, • Цел • Тип:
– **Task patterns**- Ш за задачи- дефинират действие или работна задача в СИ, която е част от процеса
– **Stage patterns**- Ш за етапи дефинират рамкова дейност за П
– **Phase patterns** -Ш за фази дефинират последователността от рамкови дейности, които се извършват в процеса
• Начален контекст – условията, при които Ш се прилага
• Проблем – проблемът, който трябва да бъде решен от Ш
• Решение – имплементацията на Ш
• Резултиращ контекст – условията, които ще настъпят, след като Ш бъде успешно имплементиран
• Свързани Ш – списък на всички процесни Ш, свързани с този
• Известни употреби/Примери

МСП дефинира: Множество от основни дейности; Съвкупност от задачи, които водят до завършване на всяка дейност; Работни продукти, като следствие от задачите; Множество от допълнителни дейности, които обхващат целия П

Разграничаване на МСП: По обратната връзка (feedback); Използваните методи за управление/контрол по време на разработването; Времетраенето на

Ad-hoc development(when necessary or needed): Възможностите на П са непредвидими. Обикновено графиците, бюджетите, функционалността и качеството на продукта не са съгласувани. Производителността зависи от способностите на отделните хора и се променя с промяната на техните умения, знания и мотивация

Моделите на процесите могат да бъдат **описателни** и **предписателни**. Описателните модели описват **историята** на разработването на програмните системи. Те са специфични за конкретните системи. **Предписателни модели** показват как трябва да се разработи нова програмна система:

М на водопада - Най-старият метод на структурирано разработване на софтуер. Предлага систематизиран, последователен подход към разработването на софтуер, който включва основни дейности: Събиране на софтуерните изисквания; Оценяване, изготвяне на график, проследяване; Анализ и Проектиране; Генериране на код и Тестване; Доставяне и Поддръжка;

Характеристики на М на водопада:

• **ясно разграничен** П, който е **лесен за разбиране**;

• всяка стъпка в М завършва със създаване множество **документи**

• всяка **дейност** трябва да бъде напълно **завършена**, преди да се премине към следваща, като се **одобри множеството от документи**;

• **ясно са дефинирани входовете и изходите на дейностите**, както и **интерфейсите** между отделните стъпки;

• **ясно са дефинирани ролите** на разработчиците на софтуер.

Проблеми на М на водопада- Реалните проекти рядко следват последователния поток на разработване, който се предлага от М.
Критики:

– М налага по-скоро структура на управление на проект за разработване на софтуер, отколкото да дава насоки как да се извършват отделните дейности;

–М е произлязъл от областта на хардуера и не отчита същността на софтуера като творчески процес на решаване на проблем (с итерации и връщане назад)

-Трудно е за потребителя да формулира всички изисквания в начало

- Клиентът трябва да е търпелив

-Разделянето на проекта на отделни етапи не е гъвкаво

-Трудно е да се реагира на променящите се изисквания на клиента

Прилагане на М на водопада

• Когато изискванията са осъзнати и ясно формулирани в началото

• Когато проектите са ясно организирани - ясно дефинирани роли

• При повторяеми проекти и/или големи проекти, за които времето и бюджетът не са критични

Модел на бързата разработка (RAD-Rapid Application Development)

• Основна цел - кратък цикъл на разработка

• “Висококоростна” адаптация на М на водопада

• разчита на използването на различни средства за бърза разработка

Дейности на RAD (Rapid Application Development: **Комуникация**; **Планиране** – Няколко софтуерни екипа работят паралелно; **Моделиране** – паралелно моделиране: бизнес моделиране; моделиране на данните; моделиране на процес; **Конструиране**: Reuse; средства за автоматично генериране на код; **Внедряване**

Недостатъци на RAD

• За големи приложения, подлежащи на разделяне на модули - значителни човешки ресурси

• Когато функционалността на софтуерната система не може да бъде подходящо разделена в отделни модули

• Когато е важна високата производителност на софтуер. прилож.

• Когато за разработката на приложението се разчита на все още нови и недостатъчно усвоени технологии

Фазови (еволюционни): Постъпков (инкрементален); Итеративен

Постъпков: **С не се доставя като едно цяло, а разработката и доставянето са разделени на стъпки и всяка стъпка доставя част от цялата С**. На потребителски И се присвояват приоритети и тези с по-висок приоритет се реализират в първите стъпки. След като започне разработката на една стъпка, И не се променят. **Постъпковият (инкрементален) М** комбинира елементи на **модела на водопада, но приложени на отделни стъпки**

Итеративен: В самото начало доставя цялостната С, макар и част от функционалността да е в примитивна форма. При всяка следваща итерация не се добавя нова функционалност, а само се усъвършенства съществуващата

Проблеми на фазовите (еволюционни)

• Необходимостта от **активно участие на клиентите** по време на изпълнение на проекта може да доведе до закъснения.

• К**омуникация и координация** са от особено значение при разработката и ако не са на добро ниво, водят до проблеми.

• **Неформалните заявки за подобрения** след завършването на всяка стъпка могат да доведат до **объркване**.

• М може да доведе до т. нар. “scope creep” – бавно и постепенно разширяване на обхвата на приложението, без П да е сходящ

Предимства на фазовите (еволюционни):

• Клиентът може да използва С, преди да е готов целият продукт.

• Първите стъпки могат да служат като **прототип**, за да помогнат за извличане и изясняване на изискванията към следващите стъпки.

• **По-малък риск от неуспех** на целия проект.

• Функционалностите от цялата система, които са с най-висок приоритет, са тествани най-много

Прилагане:

• Когато организацията няма достатъчно човешки ресурс за цялостната реализация в определен срок– В разработването на по-ранните версии участват по-малко хора и в зависимост от обратната връзка, получена от клиентите, могат да се присъединят още разработчици на следващите итерации

• Когато с итерациите може да се управляват технологичните рискове – Итерация, която изисква използването на нова технология или продукт може да се планира по-късно с цел да има достатъчно време да се усвои или да се достави новият продукт

Прототипен Два типа прототип:

-Еволюционен - да достави работеща С на крайния потребител

-Изхвърлен (throw-away) – специфициране И към софтуера

Подходи

• Създаване на основните потребителски интерфейси, без да има някакво значително кодиране.

• Разработване на съкратена версия на С с ограничени функции

• Използване на съществуваща С / компоненти, за да се демонстрират функции, които се включат в разработваната С

Проблеми на прототипния

• Прототипният М може да използва значителни ресурси, а като резултат прототипът да не успее да удовлетвори очакванията.

• Прототипът може да доведе до лошо проектирана система, ако самият той стане част от крайния продукт.

• Прототипният М не е подходящ за използване при разработване С, където проблемът е добре разбран и интерфейсът е ясен и прост

Прилагане

• В проекти, където не са ясни потребителските И и дизайнът на С

• Както самостоятелно, така и в комбинация с други М на процеси - М на водопада, спираловиден М, постъпков М и т.н.

Спираловиден-еволюционен - съчетава прототипния и на водопада

• Движещият фактор е **анализ на риска**

• Основни характеристики: – итеративен/цикличен подход – има множество от точки на прогреса (**anchor point milestones**)

При всяко завъртане по спиралата се преминава през 4 сектора:

• Установяване на целите – определят се целите, алтернативите и ограниченията на текущата фаза от разботката;

• Оценка на рисковете и намаляването им – идентифицират се и се анализират потенциалните рискове – предприемат се действия за намаляването или елиминирането им;

• Разработване и валидиране – избира се М за разработване на текущата фаза;

• Планиране – преглежда се и се анализира текущото състояние – планира се следващото завъртане по спиралата

Проблеми със спираловидния

• Може да се окаже трудно да се убедят клиентите, че процесът на разработка е контролируем, а не е безкраен цикъл.

• Изисква се участието на разработчици с **компетентност за оценка на рисковете**.

• Ако не се идентифицира и открие някой основен риск, това може да доведе до неуспех.

Прилагане: При разработване на големи (large-scale) софт. С; Адаптира се и се прилага през целия жизнен цикъл на софтуера

Метод на формална трансформация - Основава се на математическо трансформиране на спецификацията на системните И до изпълнима програма

• При трансформирането трябва да се запази коректността и да се покаже, че изпълнимата програма съответства на спецификацията

• Спецификацията на софтуерните И се усъвършенства в детайлна формална спецификация, изразена с математическа нотация

• Дейностите моделиране и конструиране са заменени с разработване и прилагане на трансформации

Проблеми на М с формални методи

• разработването на формални М е скъп и бавен процес;

• необходими са разработчици със специализирани умения, както и обучение за това, как да се прилага формалната трансформация;

• поради сложността си М трудно могат да се използват;

• някои от аспектите на софтуерна система е трудно е да се специфицират формално – например потребителският интерфейс

Прилагане: критични софтуерни системи

Избор на подходящ М на процес - зависи основно от два фактора: Организационната среда; Същността на приложението

• Видове взаимовръзки между С и нейната организационна среда: Стабилна среда (unchanging) - традиционни; Променяща се среда (turbulent) - **гъвкави методологии** като **Agile** или **DevOps**.; Неопределена среда (uncertain) - **итеративен и експериментален подход**; Адаптивна среда (adaptive, **интелигентно реагираща среда**, която използва обратна връзка, за да се **самокоригира и оптимизира**) - автономни системи, умни градове, адаптивни бизнес процеси

• М на водопада: Линейно развитие; Приложимост – добре дефинирани и стабилни изисквания

• М на бързата разработка: Инкрементални МСП – Series of increment releases;

• Еволюционни МСП – produce incremental work products quickly