

16 Софтуерно инженерство и неговото място като дел от знанието. Софтуерен процес и модели на софтуерни процеси. Концепция за многократна употреба.

1. Софтуерно икономико.

Обобщение

→ Какво е софтуер

- инструкции - осигуряват желан резултат
- структури от данни - удовлетворяват харект. функционалности, производител
- документи - описание работата и използването

Характеристики:

- разработва се, а не се произвежда
- не се използва, но оставява (инвентар)
- адстрактен и логичен.

Софтуер, който се продава - продукт

Изключки и попълнителни

Два типа продукти:

- общи (generic) - предназначени за всички потребители, които имат нужда
 - наредени (customized) - персонализирани за нуждите на клиент
- generic - използванията и определя организациите
customized - използванията и определя клиентите

→ Видове софтуер:

- системен - общиност от програми, обслугиващи др. програми
 - интегрирана комуникация с хардуер

Групират се: ОС, драйвери, компилатори и др.

- Грийзен - самостоятелни програми, решаващи специални задачи на даден компютър

Грийзър: - тематични редактори, комуникационни приложения, ...

- Научни - обслугива нуждите на конкретна наука

- свръзки със симулации, автоматизирано проектиране, анализ

- Вграден (embedded) - съхранява се в телефони, лаптопи, фотоапарати и т.н.

- изпълнява се за реализация и управление на функции. Всички хардуер

- Други приложения - свръзки със хипертекст

- предоставят информация чрез текст и графики.

- Многолетни икономии, работят в личева среда, отговарят на

многолетни икономии - изпълняват числени или за решаване на сложни

задачи, които не подлежат на директен анализ

Грийзър - приложения за разпознаване на звуци

И. Групи за технологии:

- Отворен код (open source) - популярният начин да го използват, приложват, разработват, разпространяват
- Наследи (legacy) - софтуер, разработен преди началото на съвременния, който предвидава да се използва и непрекъснато бива изменян
 - критично важно за бизнеса
 - дава пивот

Същност и обхват на СИ

СИ е дисциплината, която се занимава с проектирането, разработката, поддръжката и управлението на софтуерни системи с високо качество, при създаване на системи, дисциплинирани и изцелени подходи. Включва в себе си и изучаването на тези подходи.

Основи:

- процес на разработка - методи и модели за управление на проект
- технологии и инструменти
- управление на проекти
- качество на софтуера

2. Софтуерен процес

Представяне:

- последователност от предварителни стапки - план, когто избира, за да създаде продукт/система в срок и с високо качество.
- рамка на задачите, които е необходимо да се извършат, за да се изгради функционалният софтуер

Фази и основни действия

Етапи:

- Анализ и дифигуриране на използваната
- Проектиране на системата
- Проектиране на програмата ???
- Написане на програмата
- Тестване на единичните
- Интеграционно тестване } може на едно
- Тестване на системата
- Интеграция и внедряване
- Сървъроподдържане и поддръжка

Основни дейности:

- Кашутичане - обзор и разбиране на изискванията за функционалността на софтуера и на ограниченията върху работата и разработването requirements engineering
 - Планиране - създаване на план за работата по разработката. Описват и рискове, ресурси, работни продукти, които ще са произведени, графики
 - Моделиране - създават с модели с цел по-добро разбиране
 - дизайн
 - софтуерна архитектура
 - софтуерен дизайн - данни, интерфейси
 - Конструиране - писане/генериране на код и тестване (Bugobe)
 - Внедряване - софтуерът се предоставя на клиенти и той го тества + потребителско тестване
- Bugobe модели и език за моделиране

Модел на софтуерен процес представлява опростено описание на софтуерен проект, което е представено от определена перспектива и е адекватен на дадените във времето и място.

- описание - описват как се е случило
- приложение - предвидват как да се случи
- Workflow model
- Dataflow model
- Role/action model

Език за моделиране на процес е език, адаптиран към изходен със представянето на процеси. Използвам се за представяне на

- дейности, които трябва да се извършат
- роляте на хората
- структурата и логиката на артифактите, които се създават
- средствата, които се използват

Страна организациите си, се разделят на:

- entity-relation - сущности и връзки между тях
- role-interaction - роли и тяхните взаимодействия
- object-oriented - обекти и данни

Страна формалност:

1 - формални
2 - формални
3 - формални
4 - формални

- полуформални
- правилно пред
- информатични
- национални

→ Методи за описание:

Ще разгледаме 3 метода за описание на софтуерен процес:

- **Модел на водопада**
- **Модел на спирала**
- **Модел на въртящ се кръст**

И ще разгледаме какво определя доколко, учен или подход и/или последователността от действия за разработване на софтуер.

Моделът определя какво трябва да се направи, а не този начин как трябва да се направи. Модели са дефинирани на различни нива на абстракция.

3. Сравнителен анализ на описателни модели на софтуерен процес

3.1 Модел на водопада

- най-старият модел за структуриране на разработване
 - едноставна дейностност:
 - обзор на изисквания
 - събиране, изгответе на график
 - анализ и проектиране
 - итериране на код и тестване
 - поддръжка - всяка стапка трябва да бъде напълно завършена преди да се премине към следващата. (всяка стапка завърши със своя документ)
 - не се връщате към завършена дейност
 - линеен последователен подход
 - поддържането като всяки изисквания са ясни откъм началото
- Проблеми:
- трудно реагира на промени при съществуващи
 - не е удобно разделянето на проекта на етапи
 - когато се правят промени, започнато към края не винаги истицате преди края на етап
 - може да се приложи при по-големи проекти, при които времето и бюджета не са ограничени.

3.2. Модел на вързана разработка (RAD-Rapid App Development)

- чрез-краток цикъл на разработка (60-90 дни)
- високоскоростна адаптация на водопада
- дейности:
 - разликата с водопада е, че при планиране проектират се разделя на модули, така че да може да се работи паралелно от различни екипи по тях
 - използват се предсъществуващи компоненти и итериране на код

Грубдели:

- за голями приложения, разделянето на модули, води до значителни ^{човешки} ресурс
- рисков модул при използване на гипотетична технология ^{навсяки и да са добре усвоени}

Подходящ за малки проекти или проектни задачи със скромни модули

3.3. Прототипен модел

представя базов вариант и постепенно уточнява

- Еволюционен прототип - цел да достави работеща система
- Чуквъртил (throw-away) - цел да създават прототип, предлагащ обратно подобряване от функциите на изпитната система
- цел да подпомага спецификацията на изискванията

Грубдели:

- може да изразява значителен ресурс, но да не удовлетвори изисквания
- може да доведе до лоши проектирани системи, ако става част от
- Не е подходящ, ако проблемът е добре разбран и измерен и е ясен.

Подходящ

- да се прилага в комбиниране с други модели
- когато се създават нови популационни изисквания ^{изградени} ^{на системата}

3.4. Спираловиден

- еволюционен модел, представя прототипични и водеща
- анализира вътрешната нарица
- итеративен/циклически
- представя със спирала и всяко път завршва с една отвън от разработвател

Сърцевини:

- установяване на цел
- оценка на риска и нападяването му
- разработка и валидизация
- планиране на следващо завртане

Грубдели:

- сложен за управление и изпълнение
- тръгва разработчици към неизвестни в определянето на рисков
- риск за прилагател

Грижат се:

- за голями проекти
- критични системи
- където има пречки в изпълнението

3.5 Други

- Фазови - постепенен и имплементиран (евалуирателен)
- Agile - въвежда
 - накланява едините разходи
 - бързо реагира на промана на използванието
 - постепенно въвеждане на функциите / ЗЛ
 - инкрементално разработване
 - готовност за практика

Ч. Концепция за многократна употреба

Техника за прилагане на съществуващи компоненти, код, библиотеки и ресурси в нови разработки, с цел спестяване на време и ресурси.

Преимущества:

- спестява време и усилия
- накланява времето на разработка и тестване
- подобрява качеството - пазъв с него проверено
- накланява риска
- консистентност

Минуси:

- зависимата на поддръжника може да се увеличи
- спадаща отговорност
- трудност за напомняне на поддръжчи компоненти

Групирани:

- COTS - Commercial Off-The-Shelf - чужди приложения прилагани API
 - приложени в мн. области
 - посоката предоставят повече откаченото им творчество
 - евентуално неизвестно

Градежни:

- компактна може да има своя определена функционалност
- компактна труда на интеграция
- компактна контрол върху развиците и продължите на API-та разработки
- компонентно базиране - изграждане на софтуер от многократно използвани компоненти
 - използват се чрез интерфейси
 - възможности (require) - какво давам на компонента
 - изподават (provide) - какво предоставя компонента
- Архитектурни принципи като модули . . .
- Design patterns . . .
- Програмна генерация . . . - алгоритми и шаблони