Тема 27

Архитектури на софтуерни

(1) Софтуерни технологични центрирани около СА.

def/ СА

СА на една изчислителна система е съвкупност от структури описващи системата, състояща се от софтуерни елементи, релациите м/у тях и свойствата и на двете. Неформално: описва поведението на софт. с-ма и включва информация и за документационните изисквания и за КА.

Една от мотивациите за създаване на СА е от абстрактни изисквания подвити от естествения език, да създадем конкретна имплементация на формален език.

Обикновено СА се създава като първа стъпка по време на проектирането, като целта е да се гарантира налични на дадени качества в системата.

Предмет на СА е поведението и връзките м/у различните елементи, разглеждани като черни кутии.

(2) Качествени атрибути - дизайн на архитектурата

КА е измерим или тестируем св-во на с-мата, което служи за индикатор колко е добра системата, колко удовлетворява нуждите на stakeholders. Те са характеристиките, които системата трябва да притежава освен функционалностите.

Те засягат гил-ките производителността, дизайна на системата и потребителския опит. Важни са използната използваемост, производителност, надеждност и сигурност на софт. с-ма. Някои КА са повече свързани с цялостната системен дизайн на продукта, докато други засягат по-специфични части на дизайна. като гил-ките поведение, user-centric experience и д.р.

Две с-ми могат да приличават едни и същи функционалности, на разл. структури и в зависимост от структурата притежават различни степени на развитие на даден КА. Постигане на КА зависи от ранните арх. дизайн решения. Принципно не може да се реализира функционалност и след това да мислим дали трябва да е ефективно отнем време или отнем памет.

КА са най-трудни за намиране и документиране и то самите имена на КА не стигат т.к. имат нееднозначна интерпретация (множество).

Чрез Quality Attribute Scenarios ги намираме както ФИ чрез Use Cases ги намираме „част от КА са: Supportability (възможност за поддръжка), Testability (одитаемост), Usability, Availability, Interpretability, Manageability, Performance, Reliability, Scalability, Security, Flexibility, Maintainability, Reusability.

(3) Дизайн за постигане на ефективност, сложност, скалируемост и хетерогенност, адаптируемост на архитектурата, надеждност и сигурност.

- Performance - индикация за време на отговор на приложението при зададено действие. Може да бъде измерено в изчисление или цялостно време за отговор на системата при събитие
- Гъвкавост - способността системата да се адаптира към промяна се обкръжение и ситуации, и да се справя с промяни в бизнес потоците и правилата. Лесно конфигурируема или адаптируема в отговор на различни потребителски и системни изисквания.
- Scalability - способността системата да функционира добре при промени в натоварването и потреблението. Принципно с-мата ще може да вдигне изчислителния си ресурс при по-голям период на натоварване.

• Reliability – способността на системата да остане оперативна пред времето. Измерва се като вероятност системата да не фейлне да изпълни функ. си пред определен период от време.

• Security – дефинира начините, по които системата се предпазва от изтичане или загуба на инфо. и вероятността от успешна атака. Гледа да защитава кодовете си и да предотвратява неоторизиран достъп, модификация на инфо. и т.н.

(4) Компоненти и конектори. Типове конектори и техните променливи характеристики. Критерий за избор на подходящи конектори.

Т.к. софт. с-ми са много комплексни, то има три перспективи дефиниращи СА: статична, динамична и allocation перспектива, всяка с нейните брой изгледи:

• Компоненти: елементи активни по време на изпълнение на софт. с-ма (сървлети, илети, ---)

• Конектори: механизми за взаимодействие, които предоставят комуникацията м/у компонентите.

Компонентите използват техните портове, за да комуникират с околната среда, а конекторите играят ролите им.

• Връзки: закачане на порта на компонента с ролята на конектора.

Връзките осигуряват контрол на данните и увеличават на полза на данните.

Конекторите предлагат независими от приложението механизми за взаимодействие, докато компонентите предлагат специфични за приложението функционалности. Конекторите може да бъдат доста сложни и описват подробни и комплексни спецификации. Конекторите са потенциално абстрактни и могат да бъдат параметризирани. Конекторите благоприятстват

от независимостта м/у компонентите и улесняват
гъвкавостта на взаимодействието м/у тях. Конекторите
трябва да ~~бъдат~~ могат да се преизползват.
Конекторите са първокласни архитектурни елементи,
които моделират взаимодействията м/у компонентите,
и правилата, засягащи тези взаимодействия.
Има 4 основни типа услуги за конектори и всяка
роля се разделя на подтипове.

- комуникиращи
- координиращи
- преобразуващи
- улесняващи

(5) Архитектурни стилове

Арх. стил е съвкупност от елементи и типове, релации
заедно с ограниченията по използването им. Те са
patterns, шаблони в изпълнение, които са обособени като
отделен елемент, за да могат да бъдат преизползвани
заради свойствата им.
Стиловете помагат на дизайнерите и разработчиците
на системата да се възползват от вече придобитите
знания в тези области. Описват една малка част от
архитектурата. Обикновено арх. е изградена от
комбинация на арх. стилове. На всяка перспектива
разглеждане арх. стилове за нея. Перспективите са
трите обширни начини, според които архитект би анализирал
система. Даже и да е избран перспектива има още неща,
които трябва да се конкретизират като колко елемента
ще са ограничени, какви връзки ще има помежду им,
как ще се използват и конфигурират и д.р. Всички
тези избори водят направени, правят цялостния продукт
може да бъде наречен арх. стил.
За Module View има Декомпозиционен стил,
Генерализиращ стил, Слоест стил и д.р.
За C&C View има Dataflow, Event-based, Call-return,
Data-centered

(6) Разпределени претови > децентрализирани архитектури.

Call-return arch family е пример за децентрализиран, разпределен и претов набор от арх. стилове. Cloud computing и Client-server.

Service-oriented arch

Като основа има service-oriented computing, но също предлага нови възможности като организиране, отприване, сигурност, управляемост и посредничество.

Cloud

Това е Producer-Consumer модел, чието създаване е повлияно от консюмъристките интернет сървиси. Позволява самообслужване, опции за снабдяване и конекции.

Клаудът представлява индустриализацията на снабдяването на IT услуги. Всъщност, иновацията е в това, че крайния потребител се самообслужва и самоуправлява.

Има различни видове клауд: публичен, частен и хибриден; за вътрешни здрави и отделно за специфични проблеми.

В частния cloud дейностите/функциите се предлагат като услуги през интернет в рамките на предприятие.

В публичния cloud се предлагат като услуги през интернет. Основен плюс и в двата случая са скалируемостта, абонаментното и бързо осигуряване на ресурси.

Специфичното за частния е високо ниво на виртуализация и възможността за възстановяване. За публичните са консюмъристкото плащане и възможността да се наемат много услуги.

Client-server

Комбинация от слоеве и хранилища. Организиран е като 3 отделни компонента, които си комуникират UI, Business Logic, Shared Data

(7) Анализ и визуализация на СА

Архитектурния анализ е дейността на откриване на важни системни св-ва използвайки системни архитектурни модели.

Неформалните арх. модели и анализи помагат на архитектите да са ясно с желанието на потребителите. Помагат на мениджърите за обхвата на проекта, но не и на devs.

Формалните арх. модели и анализи помагат на архитекта да определи компонентите и вътрешното им представяне. Помагат на devs с решения на ниво имплементация. Помагат на автоматизираната генерация на код.

Основните теми на всеки анализ включват поне една от следните цели:

- Consistency
- Completeness
- Compatibility
- Correctness

Един от начините за визуализация на СА е описанието й да бъде направено с някой от графичните езици за описание на архитектури или конкретно ADL. Предлагат формален начин за описанието на СА. Имат конкретен синтаксис, tools за парсване, визуализация и анализ на архитектурата.