СОФТУЕРНИ АРХИТЕКТУРИ

спец. Компютърни Науки
ФМИ, СУ „Св. Кл. Охридски“
2024 / 2025 г.

СА – Дефиниция

- Съгласно Software Engineering Institute:

  “Архитектура на дадена софтуерна система е съвкупност от *структури*, показващи различните софтуерни елементи на системата, външно видимите им *свойства* и *връзките* между тях”

Архитектурни структури

- Структура – съвкупност от софтуерни елементи, техните външно видими свойства и връзките между тях;
- Изглед (view) – конкретно документирано представяне на дадена структура;
- Двете понятия в голяма степен са взаимозаменяеми;
- Архитектурните структури се делят най-общо казано на 3 групи:
  - Модулни структури;
  - Структури на процесите;
  - Структури на разположението;

Модулни структури

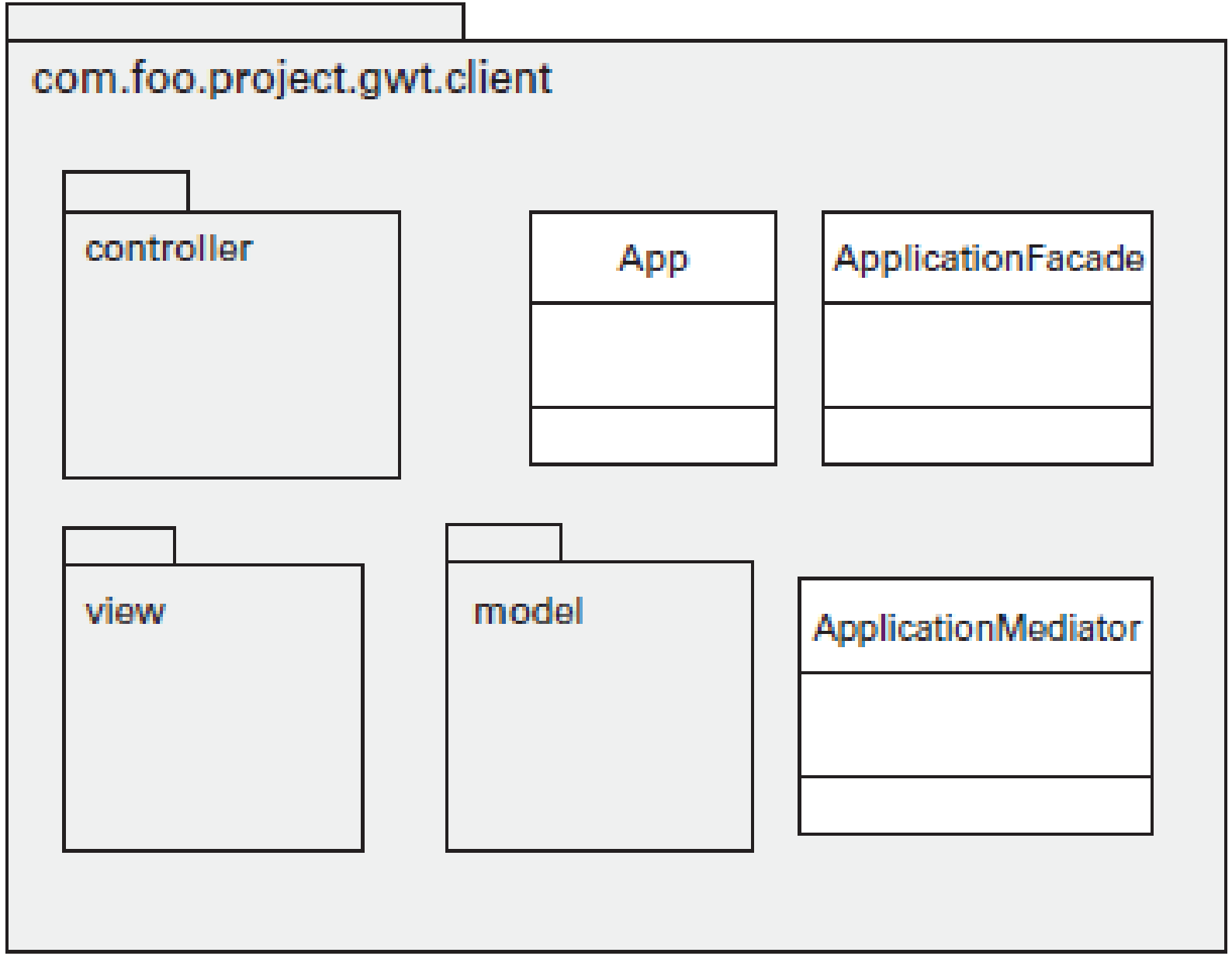
- Елементите в модулните структури са модули – единици работа за изпълнение. Модулите предлагат поглед, ориентиран към реализацията на системата, без значение какво става по време на изпълнението;
- Някои въпроси, на които отговарят тези структури:
  - Коя функционалност в кой модул се реализира?
  - Кои други модули може да използва (и използва) дадения модул?
  - Как са свързани модулите по отношение на специализация и генерализация (наследяване);

Структури на **процесите**

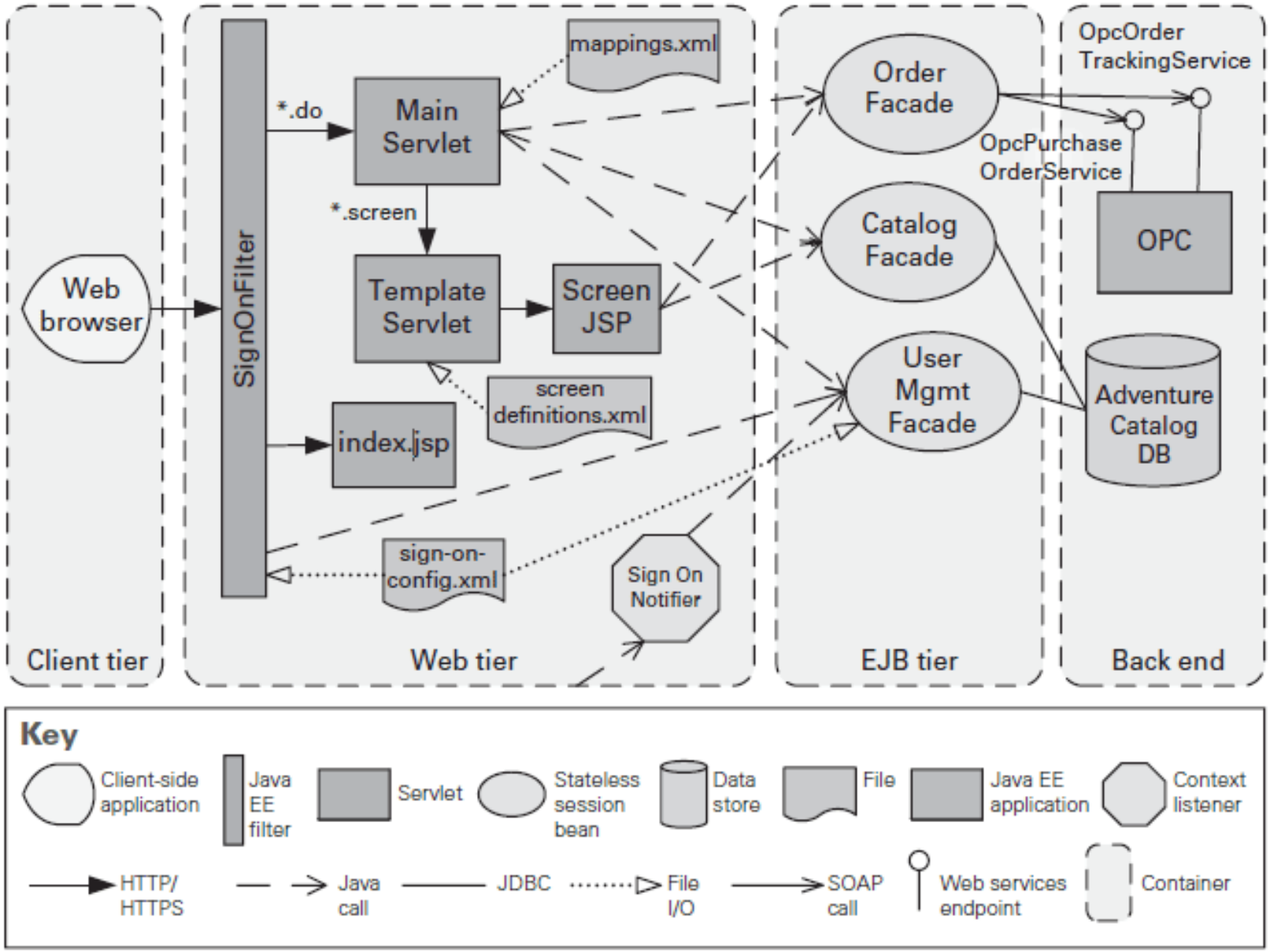
- Елементите са компоненти, които се проявяват по време на изпълнението (т.е. основните изчислителни процеси) и средствата за комуникация между процесите.
- Някои въпроси, на които отговарят тези структури:
  - Кои са основните изчислителни процеси и как те си взаимодействат?
  - Кои са основните споделени ресурси?
  - Как се развиват данните в системата?
  - Кои части от системата могат да работят паралелно?
  - Как се променя структурата на системата докато тя работи?

Структури на **разположението**

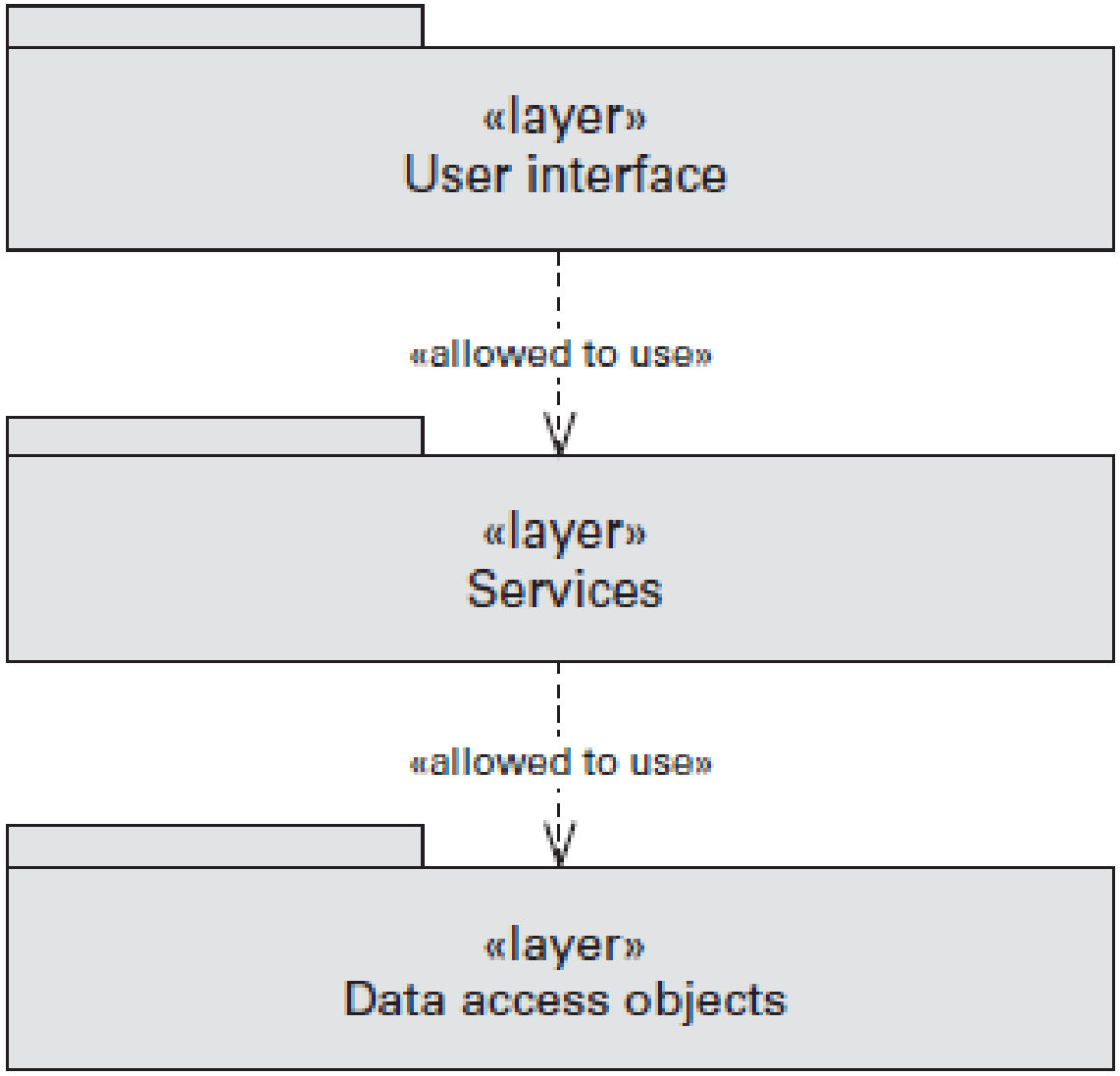
- Структурите на разположението показват връзката между софтуерните елементи и елементите на околната среда, в която се намира системата по време на разработката или по време на изпълнението;
- Някои въпроси, на които отговарят тези структури:
  - На кой процесор се изпълнява всеки от елементите?
  - В кои файлове се записва сорс кода на елементите по време на разработката?
  - Какво е разпределението на софтуерните елементи по екипи, които създават системата?



Paul Clements et all., Documenting Software Architectures: Views and Beyond, Pearson Education, 2011



Paul Clements et all., Documenting Software Architectures: Views and Beyond, Pearson Education, 2011



Paul Clements et all., Documenting Software Architectures: Views and Beyond, Pearson Education, 2011

Понятие за качество на софтуера

- За проектиране на архитектурата е достатъчно да се започне с няколко най-важни изисквания
- Най-важните изисквания ги наричаме архитектурни драйвери
- Съществуват две големи групи изисквания към софтуерните системи

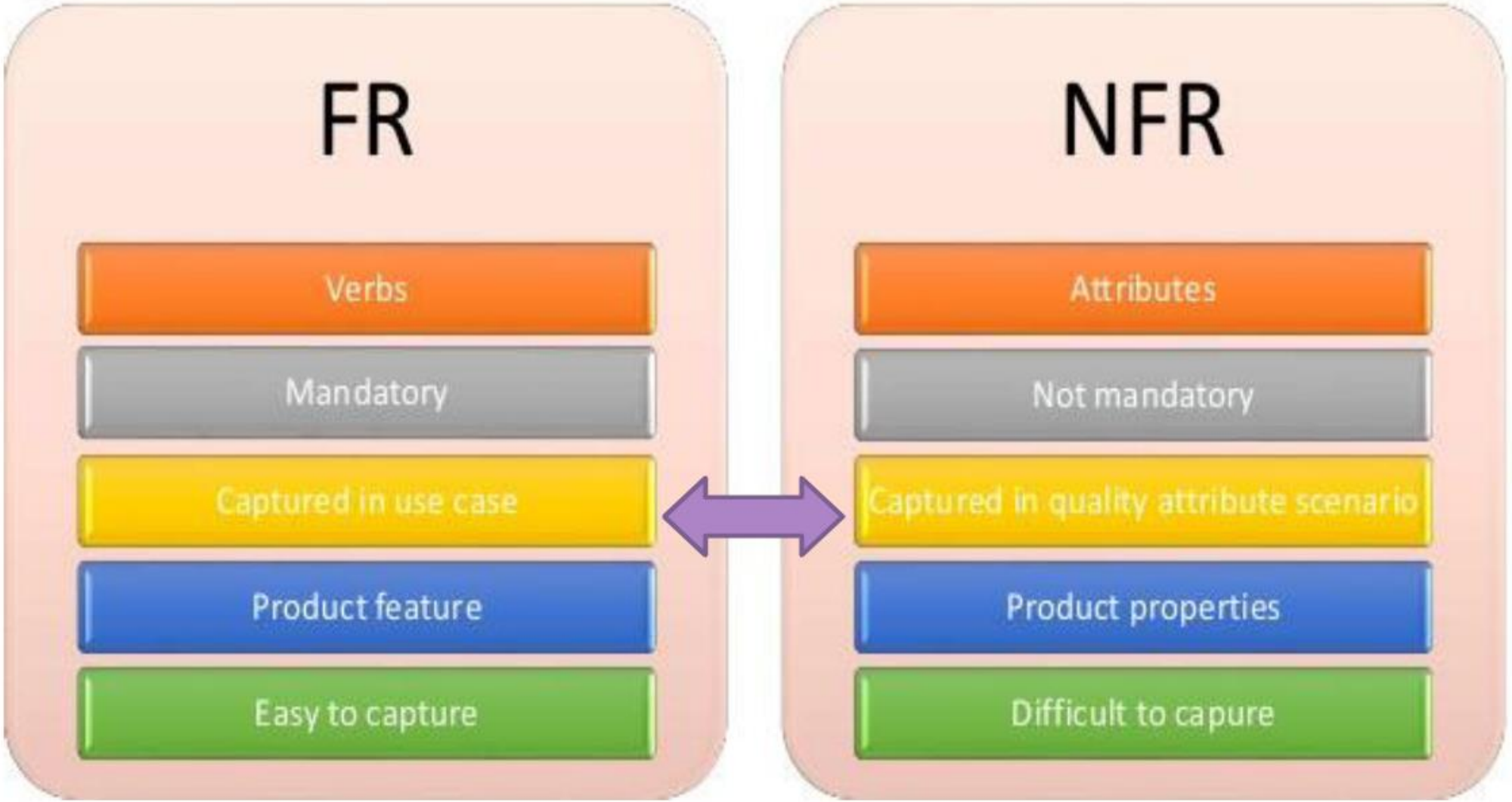
Понятие за качество на софтуера

- Функционални изисквания
- Нефункционални изисквания
  - Технически изисквания към качеството
  - Бизнес ограничения
  - Технологични ограничения

S.No.	Functional Requirements	Non-Functional Requirements
1	They are customer based.	They are based on the developers and technical knowledge of the team.
2	They specify which functionality should to be taken into consider-ation i.e. what needs to be tested.	They focus on how that functionality needs to be tested.
3	Functional testing is performed before the application or software market release.	Non-functional requirements include the maintenance testing, documentation testing and other testing post application market release.
4	It is onlyknown as Functional requirement.	It is also known as Quality requirements.
5	The execution plan for Functional requirement is specified in system design document.	The execution plan for Non-Functional requirement is specified in system design.
6	They include testing of technical functionality of the system.	They include maintenance of qualities like security ,usability etc.
7	In simple terms they are the “What” in testing.	They are the “How” in testing.

Източник: https://www.resourcology.com/blog/functional-vs-non-functional-testing/

Non Functional vs. Functional



Източник: https://kenscourses.com/tc1019fall2016/tag/functional-and-non-functional-requirements/

Функционалност и качества

- Зависят ли функционалността и качествата едно от друго?

- Решенията, които взема архитектът по време на създаването на СА, са определящи за постигането на необходимите нива на съответните качества
  - Например: дадени решения ще доведат до висока производителност, а други до липсата на такава

Функционалност и качества

- Постигането на качествата е въпрос както на архитектурни, така и на не архитектурни решения, например за постигане на **използваемост**:
  - Не архитектурните аспекти включват това, интерфейсът да е ясен и лесен за употреба (широки или тесни менюта) напр. разположение на елементите по екрана, избор на ясен шрифт и подходяща цветова схема, и др.
  - Архитектурните решения включват предоставянето на възможности за cancel, undo, re-use на предишно въведени данни и т.н.

Отношения между качествата

- **Сигурност** vs. **отказоустойчивост** въпреки, че обикновено ги поставят в една и съща графа, те си противоречат сигурността изисква наличието на single point of failure , т.е. системата да може да се компроментира от едно единствено място. Обратно, отказоустойчивостта предполага репликация

- **Преносимост** vs . **производителност** основната техника за постигането на преносимост е изолирането на зависимостта от ОС и хардуер в специализирани модули, което внася допълнителна тежест по време на изпълнението и намалява производителността

Качествени изисквания

- Функционалните изисквания определят **какво** трябва да прави софтуерната система
- Качествените изисквания определят **как** софтуерната система да работи
  - На практика качествата поставят ограничения върху начина по който системата ще се изпълнява

Качествени изисквания

- Примери
  - Performance
  - Reliability
  - Availability
  - Modifiability
  - Usability
  - Testability
  - Survivability
  - Maintainability
  - Accessibility
  - Etc.
- Познати още като нефункционални изисквания / характеристики
  - Други имена: Non-functional Properties/Attributes, System Qualities, “-ilities”, etc.)

Понятие за качество

- Качеството е субективно възприятие – различните ЗЛ могат да не одобрят даден дизайн, тъй като тяхната идея за качество се различава от идеята за качество на архитекта
- Изискванията за качество трябва да се формализират от архитекта посредством т.н. “сценарии за качество”, за да бъдат те поставени на обективна основа
- Сценариите демонстрират какво е качество в рамките на създаваната система, като дават на архитекта и на ЗЛ еднозначна основа за оценка на дизайна

Технологични Качества

- За Технологични Качества се говори от 70-те години на XX-ти век – публикувани са множества дефиниции и класификации
- В класическите дефиниции за качествени характеристики се наблюдават няколко основни проблема:
  - Безсмислено е да се говори за дадена система, че е “изменяема” – всяка система е изменяема по отношение на дадена промяна и неизменяема по отношение на друга промяна. Ситуацията с другите Качества е подобна
  - Често се водят спорове към кое Качество принадлежи даден аспект на системата. Дали отказ в системата е аспект на наличността, на сигурността или на надеждността?
  - За всяко Качество си има собствен речник. Специалистите по Производителност говорят за “събития”, тези по Сигурност – за “атаки”, тези по Надеждност – за откази, и т.н. Всички тези термини всъщност могат да обозначават едно и също събитие
- За избягването на тези проблеми е необходимо да се борави с т.нар. ***сценарии за качество***

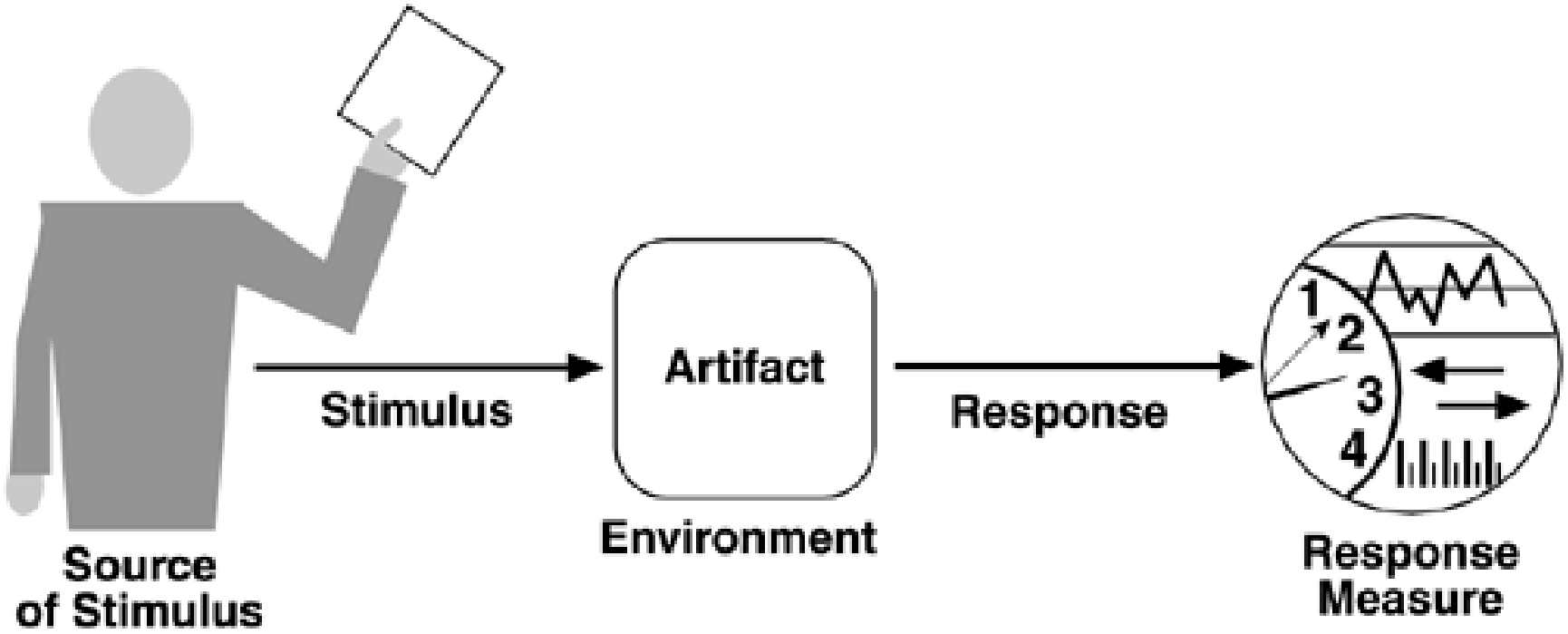
Сценарии за Качество

- Сценарият за качество е специфично изискване към поведението на системата в дадена ситуация, в светлината на дадено качество
- Сценариите за качество играят същата роля за дефиниране на нефункционалните изисквания, каквато роля играят сценариите за употреба (use-cases) за дефиниция на функционалните изисквания

Сценарии за Качество

- Всеки сценарий описва някаква случка и се характеризира с 6 компонента:
  - Въздействие
  - Източник
  - Обект
  - Контекст
  - Резултат
  - Количествени параметри

Сценарий за качество



Source: Bass et all, Software architecture in practice

Компоненти на сценариите за Качество

- **Въздействие** – състояние/събитие, което подлежи на обработка;
- **Източник** – обект (човек, система или нещо друго) който генерира въздействието;
- **Обект** – системата, или конкретна нейна част, върху която се случва въздействието;
- **Контекст** – Условията, при които се намира обекта по време на обработка на въздействието;
- **Резултат** – действията, предприети от обекта при случването на въздействието;
- **Количествени параметри** – резултатът трябва да подлежи на някакви количествени измервания, така че да позволи проверката дали сценарият се изпълнява съгласно изискванията;

Пример за сценарий за надеждност (1)

- Какво означава една система да е надеждна?
- Сценарий за Надеждност
  - По време на експлоатация на системата, външен източник изпраща на процеса X, съобщение за препълване на опашката с потребителски заявки. X трябва да информира оператора за получаването на съобщението и да продължи работа без прекъсване.

Пример за сценарий за надеждност (2)

- Сценарий за Надеждност –
  - “По време на експлоатация на системата,… (Контекст)
  - …външен източник… (Източник)
  - …изпраща на процеса Х…. (Обект)
  - ...съобщение (Въздействие) за препълване на опашката с потребителски заявки …
  - …което се получава от
  - Процесът трябва да информира оператора за получаването на съобщението и да продължи работа… (Резултат)
  - …без прекъсване” (Количествени параметри)

Пример за сценарий за изменяемост (1)

- Сценарий за Изменяемост
  - Преди пускането на системата в експлоатация, клиентът желае промяна на фоновия цвят на екрана. За целта се променя изходния код. Това не трябва да предизвиква никакви странични ефекти в поведението на системата, като времето за извършване на промяната, вкл. тестването й трябва да отнеме по-малко от 3 часа.

Пример за сценарий за изменяемост (2)

- Сценарий за Изменяемост
  - “Преди пускането на системата в експлоатация… (Контекст)
  - …клиентът… (Източник)
  - …желае промяна на фоновия цвят на екрана…. (Въздействие)
  - За целта се променя изходния код…. (Обект)
  - Това не трябва да предизвиква никакви странични ефекти в поведението на системата. (Резултат)
  - Времето за извършване на промяната, вкл. тестването й трябва да отнеме по-малко от 3 часа (Количествени параметри)

Групи Качества

- Качествата се разделят на следните три основни групи:
  - **Технологични качества** – напр. Надеждност, Изменяемост, Производителност, Сигурност, Изпитаемост, Използваемост;
  - **Бизнес качества** – напр. време за пускане на продукта на пазара;
  - **Архитектурни качества** – присъщи на самата архитектура като напр. идейна цялост (влияят косвено върху всички останали качества);

Изправност (dependability)

- Състои се от няколко атрибута
  - Надеждност (reliability)
  - Готовност или наличност(availability)
  - Безопасност (safety)
  - Сигурност (security)
  - Отказоустойчивост (fault-tolerance)
  - Възможност за промяна (modifiability)

Сценарии за изправност

- Изправността (dependability) се занимава с отказите (failures), наричани още сривове в системата и свързаните с това последствия.
- Отказ в системата настъпва когато тя престане да предоставя услугите си съгласно спецификацията;
- Отказът е наблюдаем от потребителите (хора или други системи);
- Интерес представляват обстоятелствата, свързани с откриването на отказ, колко често може да се случва отказ, за колко време системата може да преустанови работата си, кога отказите са безопасни, как могат да бъдат избегнати и какви уведомления се генерират когато се получи отказ;
- Трябва да се прави разлика между отказ и дефект. Дефектът (fault) може да стане срив ако не бъде поправен или замаскиран. Сривът е наблюдаемо събитие, докато дефектът не е. Когато дефектът стане наблюдаем, той се превръща в срив.

Сценарии за изправност

- Докато системата е отказала, важна характеристика става времето, за което тя ще бъде поправена;
- Това време може да трае от няколко милисекунди до няколко дни;

Мерни единици за наличност (availability)

- Наличността на системата се дефинира като вероятността тя да бъде в изправност, когато има нужда от нея
- Може да се представи като:

$$\alpha = \frac{\Delta t_f}{\Delta t_f + \Delta t_c},$$

- Където Δt_f е средното време между отказите, а Δt_c е средното време за отстраняване на повредата

Сценарии за изправност

Компонент	**Възможни стойности**
Източник	Вътрешен за системата, външен за системата
Въздействие	Срив: липса на отговор, счупване, ненавременно събитие, неправилни отговори
Обект	Хардуерни устройства, комуникационни канали, постоянна памет, процеси
Контекст	Нормална работа, временна намалена работоспособност, временно решение за отстраняване на срив
Резултат	Системата трябва да регистрира събитието и да: - го запише; - извести когото е необходимо; - да забрани входа от източника; - да остане недостъпна; - да продължи да работи в нормален или специален режим;
Количес-твени параметри	Процент надеждност (α) Интервали (или продължителност), в които системата трябва да е работоспособна Време за отстраняване на срива Интервали (или продължителност), в които системата може да е в състояние на намалена работоспособност

Сценарии за изменяемост (modifiability)

- Изменяемостта на системата е свързана със себестойността на промените. Интерес представляват два въпроса:
  - Какво може да се промени? (Обектът) – Промяна може да се осъществи във всички аспекти на системата, най-често във функциите, платформата (хардуер, ОС, мидълуер и т.н.), обкръжението (другите системи, протоколите за обмен на данни със света и т.н.), качествата на системата (вкл. бъдеща изменяемост), капацитетът (брой потребители, брой операции) и т.н.

Сценарии за изменяемост

- Кога се случва промяната и от кого? (Контекстът) – Промени се правят по време на реализацията (чрез промяна на кода), по време на компилацията (чрез опции на компилатора), по време на билд-а (чрез избор на библиотеките), по време на конфигурацията (посредством параметри) или по време на изпълнение (посредством параметри). Съответно, промените се правят от разработчика, системния администратор или крайния потребител.
- След като промяната е специфицирана, тя трябва да се реализира, тества и внедри, което изисква време и пари, а те могат да се измерят;

Сценарии за изменяемост

Компонент на сценария	**Стойности**
Източник	Разработчици, системни администратори, крайни потребители
Въздействие	Искат да се добави/премахне/промени/ функционалност, качествено свойство, капацитет и др.
Обект	Потребителския интерфейс, платформата, обкръжението, конкретен модул, системата и т.н.
Контекст	По време на разработката, по време на компилацията, по време на билд-а, по време на конфигурацията, по време на изпълнението
Резултат	Намира местата, които подлежат на промяна; прави промените, без това да се отразява на останалата функционалност; проверява промените; внедрява промените
Количествени параметри	Стойност, в смисъл на брой променени елементи, усилие, пари и доколко промяната се отразява на останалата функционалност и свойства

Сценарии за производителност (performance)

- Различни събития (прекъсвания, съобщения, заявки от потребителя, или пък самото преминаване на времето) се случват и системата трябва да реагира на тях;
- Предмет на производителността е времето, за което системата реагира на възникващите събития;
- Едно от нещата, което усложнява ситуацията с производителността е броя различни събития и схемите, съгласно които те възникват;

Сценарии за производителност

- Източниците на събития могат да бъдат потребителски заявки, други системи или пък самата система;
- Примери:
  - Уеб-базирана финансова система получава заявки от потребителите, вероятно десетки или стотици хиляди на ден;
  - Система, за управление на двигател на автомобил получава събития просто с течение на времето и контролира както запалването, така и смесването на гориво и въздух за постигането на максимална мощност;

Сценарии за производителност

- За уеб-базираната финансова система е важно колко транзакции могат да бъдат обработени за 1 мин.
- За системата за управление на двигателя е важно максималното допустимо отклонение от идеалната точка за задействане;
- Във всеки случай, схемите, по които пристигат събитията и схемите, по които те се обработват могат да бъдат описани и това описание служи за създаване на сценариите за производителност;

Сценарии за производителност

- Сценарият за Производителност започва със заявка за извършване на някаква услуга. Обслужването на заявката е свързано с консумацията на някакви ресурси. В същия момент системата може да обслужва паралелно и други заявки.
- Схемата, по която се получават заявките може да бъде характеризирана като периодична или стохастична.
- Напр., периодично събитие може да се появява на всеки 10 ms; Най-често периодичните събития се срещат в системите за реално време;
- Стохастичните заявки пристигат в съответствие с някакво вероятностно разпределение;
- Заявките може да пристигат и спорадично, т.е. не се поддават нито на периодично, нито на стохастично описание;

Сценарии за производителност

- Количествено, резултатите в сценариите за производителност могат да се характеризират чрез различни параметри, напр.:
  - Латентност (latency) – времето между пристигането на заявката и обработката ѝ;
  - Времева граница (Навременност) (deadline) – максимално време за наблюдаема реакция на системата;
  - Отклонение (jitter) – вариация в латентността;
  - Пропускливост (throughput) – броя заявки, които системата може да обработи за определен интервал от време;
  - Брой на необработените заявки (поради претовареност).

Сценарии за производителност

Компонент на сценария	**Стойности**
Източник	Множество източници, потребителски заявки, други системи, вътрешни за системата и т.н.
Въздействие	Случва се периодично, стохастично или спорадично събитие
Обект	Системата или конкретен неин процес
Контекст	Нормален режим; режим на претоварване
Резултат	Обработва събитието; променя качеството на обслужване
Количествени параметри	Латентност; времева граница; пропускливост; отклонение; брой необработени заявки и т.н.

Сценарии за сигурност

- Сигурността е мярка за способността на системата да устоява на опити за неразрешена употреба, без това да пречи на легитимните потребители.
- Опитът да се компроментира сигурността се нарича “атака” (или “заплаха”) и може да приеме множество различни форми.

Сценарии за сигурност

- Атаките, някои от тях отразявани нашироко в пресата, варират от кражба на пари чрез електронен трансфер, промяна на чувствителни данни, кражба на номера на кредитни карти, разрушаване на файлове, DoS атаки и т.н.
- За сигурните системи са характерни невъзможността за отричане (non-repudiation), поверителността (confidentiality), интегритет, осигуреността (assurance), наличността (availability), проверяемостта (auditing)

Сценарии за сигурност

- Невъзможност за отричане –системата не позволява на страните в транзакцията отричането й
- Поверителност – данните и функционалността са защитени от неправомерен достъп
- Интегритет – данните и услугите се предоставят във вида, в който това е предвидено
- Осигуреност – участниците в транзакцията са тези, за които се представят
- Наличност – системата е достъпна за легитимна употреба
- Проверяемост – системата проследява събитията, които се случват в нея.

Сценарии за сигурност

Компонент на сценария	**Стойности**
Източник	Човек или система, който/която е: - идентифициран; неправилно идентифициран; неидентифициран; - външен; вътрешен; оторизиран или не-оторизиран; - с ограничен достъп; с неограничен достъп
Въздействие	Опитва се да види данни; да промени/изтрие данни; да използва системни услуги; да предотврати/намали достъпа до системни услуги
Обект	Системни услуги и данни
Контекст	Online; Offline; свързана; несвързана; с/без защитна стена
Резултат	Допуска потребителя; скрива идентичността му; блокира достъпа до данни/услуги; дава достъп до данни/услуги; дава/отнема права за достъп до данни/услуги; записва опитите за промяна/изтриване; записва данните в криптиран вид; различава необяснимо високи нива на активност; информира заинтересовани лица; ограничава ползваемостта
Количествени параметри	Време/усилия/ресурси, необходими за заобикаляне на сигурността с вероятност за успех; вероятност за разкриване на атаката; вероятност за разкриване на самоличността на извършителите; процент на работоспособност (напр. при DoS); възможности за възстановяване; обхват на пораженията

Сценарии за изпитаемост

- Изпитаемостта е лекотата, с която софтуерът може да бъде накаран да покаже дефектите си посредством извършване на различни тестове.
- Около 40% от себестойността на грамотно изработените системи е свързана с тестването.
- Струва си архитектът да се опита да намали тази стойност.
- По конкретно, изпитаемостта е вероятността по време на следващото пускане на тестовете да бъде открит поне един дефект на системата (ако има такива).
- На практика пресмятането на тази вероятност не е възможно и количествените параметри на сценариите обикновено включват други метрики.

Сценарии за изпитаемост

- За да бъде една система правилно изпитаема, трябва да е възможно да се контролира вътрешното състояние и входовете на всеки един от компоненти й и да е възможно да се наблюдават изходите му;
- Обикновено това се постига посредством специализирани средства за тестване – от обикновен запис/плейбек механизъм за манипулация на интерфейса, до огромни тестови установки, напр. за самолетни двигатели;
- Тестването се прави от различни програмисти, тестери, потребители, по различно време. Тестват се порции от кода, от дизайна, цялата система и т.н.
- Количествените параметри са свързани с това, колко са ефективни тестовете и колко време отнемат за да се достигне предварително ниво покритие.

Сценарии за изпитаемост

Компонент на сценария	**Стойности**
Източник	Разработчик; интегратор; проверител; проверител на клиента; потребител
Въздействие	Завършена е първоначална версия на анализа; архитектурата; проекта; класа; подсистемата; завършена е интеграцията на подсистемите; системата е цялостно завършена
Обект	Части от анализа, проекта, парчета код, цялата система
Контекст	По време на проектирането; разработката; компилацията; внедряването
Резултат	Дава достъп до вътрешните променливи и състоянието, изчислените стойности, създава тестова установка
Количествени параметри	Процент на изпълнените операции, вероятност за регистриране на дефектите, време за изпълнение на тестовете, дължина на най-дългата верига от зависимости в тестовете, време за подготовка на тестовата установка

Сценарии за използваемост

- Използваемостта е свързана с това, колко лесно потребителя успява да свърши дадена задача и каква подкрепа му оказва системата в това му начинание:
  - Обучение – ако потребителя не е запознат с даден аспект на системата, какво може тя да направи за да улесни процеса на обучение?
  - Ефикасност – какво може да направи системата за да работи потребителя по ефикасно?
  - Устойчивост – какво може да направи системата, така че да намали последствията от потребителски грешки?
  - Адаптивност – как може потребителя (или системата) да се адаптира, така че да улесни работата?
  - Увереност – с какво помага системата за това, потребителя да е сигурен, че е извършил правилно дадено действие?

Сценарии за използваемост

Компонент на сценария	**Стойности**
Източник	Крайния потребител
Въздействие	Иска да научи нова функционалност; да използва системата ефикасно; да намали ефекта от грешки; да адаптира системата; да се чувства уверен
Обект	Системата или някаква нейна част
Контекст	По време на изпълнението/конфигурирането
Резултат	- за научаване на нова функционалност: контекстно ориентирана система за помощ; стандартизиран интерфейс, познат на потребителя; - за постигане на ефикасност: агрегация на данни и функционалност, повторна употреба на вече въведени данни и команди, ефикасна навигация в рамките на екраните и между тях, постоянство в интерфейса, разширено търсене, възможност за няколко едновременни действия; - за намаляване на ефекта от грешки: отмени; откажи; възстановяване от грешки; разпознаване и отстраняване на потребителски грешки; възстановяване на забравена парола; проверка на системните ресурси; - за постигане на увереност: показва моментното състояние; прогрес при дълги операции; визуална обратна връзка; работи в ритъма на потребителя
Количествени параметри	Време за извършване на задачите; брой грешки; брой на решените проблеми; потребителска удовлетвореност; научаване от страна на потребителя; отношението на успешните операции към неуспешните; изгубено време/данни

Други технологични качества

- Тук описаните 6 технологични качества не формират окончателен списък – архитектите боравят (и ще боравят) с всякакви други – scalability, portability, interoperability и т.н.
- Дефиницията на сценариите за тях се прави по същия начин – идентифицират се конкретните елементи на сценариите и се разписват формално.

Бизнес Качества

- В допълнение към технологичните Качества, обикновено системата се оформя от набор от нефункционални бизнес изисквания.
- Те обикновено са свързани със себестойността, времето за изработка, пазара и пазарните условия и т.н.
- По същият начин, по който се формализират технологичните Качества, се формализират и бизнес изискванията, за да няма двусмислие при тяхната комуникация;

Бизнес Качества (2)

- **Време за пускане на продукта на пазара** (Time to market, TTM) – Ако има натиск от страна на конкуренцията, или пък има тесен прозорец за пускане на продукта, времето за разработка става важно. Обикновено TTM се намалява чрез използването на COTS продукти или пре-използването на компоненти от предишни разработки (което пък е свързано тясно с декомпозицията на системата и структурата на употреба)

Бизнес Качества (3)

- **Себестойност и печалба** – всеки проект има бюджет, който не бива да се надвишава. Различните архитектури влекат след себе си различна себестойност. Напр., ако архитектурата разчита на непозната технология, то най-вероятно тя ще излезе по-скъпа. Или ако пък е високо изменяема, тя най-вероятно ще струва повече (но пък ще се спести от поддръжката).
- Тези параметри са определящи за съдържанието на архитектурата.

Бизнес Качества (4)

- **Предвидено време за живот на системата** – ако се предвижда системата да се експлоатира за дълъг интервал от време, следва тя да бъде проектирана така, че да е високо изменяема, скалируема и преносима. Това от своя страна би увеличило TTM, така че отново се налага компромис и за да бъде взето правилното решение, е необходимо да се формализират и приоритизират изискванията.

Сценарий 1

- Клиент изисква добавянето на нова функционалност за разпознаване на реч в системата за автопилот (пр. команда за увеличаване на скоростта на автопилота), когато системата все още в процес на разработване. Промените в системата трябва да се извършат с минимален страничен ефект. Това трябва да се случи до две седмици след получаването на пълнята спецификация.

източник / source,
въздействие / stimulus,
обект / artifact,
контекст / environment,
резултат / response и
количествени параметри / response measure

Сценарий 2

- По време на изпълнение, се засича отказ в сървлета. Системата трябва да регистрира отказа в рамките на 15 сек, и да се възстанови от отказа в рамките на 120 сек.

източник / source,
въздействие / stimulus,
обект / artifact,
контекст / environment,
резултат / response и
количествени параметри / response measure

Сценарии за качество

- Дефинирайте сами сценарии за качество за всяка от следните характеристики
  - Performance
  - Testability
  - Modifiability
  - Usability

За целта може да търсите материали по интернет

Въпроси ?

За подготовката на тази презентация са използвани слайдове създадени от:

- доц. Александър Димов
- проф. Боян Бончев