Упражнение по Софтуерни архитектури.

Атрибути на качеството.

Име: **Станислав Стоянов**

Фак.номер: **471218066**

# **Въведение**

По-долу са дадени нефункционални изисквания към три вида софтуерни системи:

1. Софтуер за управление на процесите в голяма международна банкова институция (World of BankMan);
2. Софтуер за управление в реално време на системите в автомобил (Car SYS);
3. Real Racing Simulator - Game engine за реалистичен рейсинг симулатор.

# **Основни нефункционални изисквания към трите софтуерни системи**

**World of BankMan** е система за управление на процеси и данни в голяма международна банка. Такъв вид софтуерна система се характеризира с:

* Данните се съхраняват сигурно, включително лични данни, данни за сметки;
* Транзакциите (обменяните данни) между отделните компоненти на системата са защитени (криптирани);
* Гарантира завършеност на транзакциите;
* Резервираност (бекъп) на данните (онлайн и офлалйн);
* Такъв вид системи най-често обменят информация с множестно други системи;
* Често данните се обработват не в реално време, а след края на работния ден;
* Системата има множество компоненти, някои от тях са legacy софтуер (стар, без поддръжка).

**Car SYS** е софтуер, управляващ системите за движение и сигурност в един автомобил. Такъв вид софтуер се характеризира с:

* Обработка на данни в реално време. Това означава, че данните получавани от различни сензори (ABS сензор, сензор за ъгъл на завъртане на волан, сензор за обратно ускорение (удар)) се обработват гарантирано в рамките на милисекунди и при необходимост се подава команда към дадено устройство за изпъление на действие;
* Разработката най-често използва езици от по-ниско ниво като “C” или специализирани езици;
* Хардуерната архитектурата се състои от множество микроконтролери, отговорни за управлението на дадено устройство или компонент;
* Гарантиране на ниски нива на отказоусточивост.

**Real Racing Simulator** е софтуер от типа game engine, използван от разработчиците на игри да създадат собствена игра, ориентирана към жанра на реалистичните рейсинг симулатори. Такъв вид софтуер се характеризира с:

* Оптимално управление на хардуерните ресурси;
* Паралелни изчисления (многонишковост);
* Част от програмния код се използва за различни игрови платформи (PC, PS, XboX);
* Unmanaged код, базиран на езици като C, C++.

# **Задача 1**

В таблицата по-долу попълнете степента на важност на съответния атрибут за качество за съответната софтурена система, използвайки число от 1 до 5, където 1 означава, че съответния атрибут не е толкава важен, а 5 означава, че е от изключителна важност. Добавете още атрибути за качеството, които мислите, че са важни за една или други система

|  |  |  |  |
| --- | --- | --- | --- |
| Quality атрибут | **World of BankMan** | **CarSYS** | **Real Racing Simulator** |
| Responsiveness | 5 | 5 | 4 |
| Usability | 5 | 4 | 4 |
| Distributability | 5 | 3 | 3 |
| Serviceability | 5 | 5 | 3 |
| Extensibility | 5 | 5 | 4 |
| Scalability | 5 | 2 | 5 |
| Maintainability | 5 | 5 | 5 |
| Security | 5 | 4 | 3 |
| Portability | 5 | 5 | 5 |
| Reliability | 5 | 5 | 4 |
| Testability | 4 | 4 | 4 |
| Compatibility | 4 | 3 | 3 |
| Разходи и ползи | 5 | 4 | 4 |
| Интеграция с наследени системи | 5 | 2 | 2 |
| Целеви пазар | 5 | 4 | 5 |

# **Задача 2**

Вие сте ръководител на отдела по поддръжка на системата **World of BankMan** и забелязвате, че напоследък интервала от време MTBF e доста малък и често се случват проблеми със системата. Откривате, че проблемът се състои в това, че един от модулите на системата, често отказва поради голям брой зяавки към него. Как ще решите проблема?

а. Ако имате достъп до сорс кода на този модул.

**-** Ако имам директен достъп до сорс кода на дадения модул от системата, бих използвал различни видове тестване – load/stress тестове, за да видя точно при какво поведение на модула се получава отказ и при какво количество заявки. От друга страна, ако даденият модул вече има готови тестове, бих проверил къде точно се корени проблемът и чрез различни подходи бих написал regression тестове, които ще подсигурят стабилност на системата при други евентуални откази и бъгове. Общо взето основната ми идея е в това чрез различни видове тестове да се подсигури решаване на съответните проблеми и увеличаване стабилността на системата в бъдещо време. От security гледна точка може да се провери дали отказът в системата не се причинява от DDOS атаки към сървъра и изкуствено създаване на голямо количество заявки. При такъв проблем решението е използване на платене платформи, които предварително преди изпращане на заявка от сървъра проверяват надеждността на request заявката (пример за такава платформа е Cloudflare).

b. Ако имате достъп до сорс кода на този модул, но не можете да го променяте поради лицензионни ограничения.

- Чрез разпределеност може да се помисли за изолирано управление на проблемния модул и осигуряване на единно изпълнение на цялата система. Често поради ниските нива на свързаност (loose coupling) определени модули от една система може да предизвикат глобално ниско ниво на производителност на цялата система, докато останалите модули осигуряват напълно очакваните резултати. При начилание на сорс кода може самата архитектура на системата да бъде високо свързана, т.е (tightly coupled) и по този начин да се редуцират евентуалните бъдещи проблеми. Добър подход е и честото актуализиране на логовете за грешките и правилното определяне на приоритета на грешките – info, critical, warning, trace нива на грешки. По този начин без да се променя сорс кодът може лесно да се види къде настъпва критичната грешка в модула и съответно чрез тази информация да се избере подход за отстраняването и.

c. Ако нямате достъп до сорс кода на този модул.

- Може да се помисли за по-добра мащабируемост на системата като се добавят нови физически сървъри или памет към съществуващите, а в краен случай би било възможно и изцяло да се подмени модулът, който често отказва поради липса на физически ресурс.

# **Задача 3**

Вие сте ръководител на екип за разработка на софтуер за управление на кризисни ситуации. Софтуерът ще се използва да известява населението за настъпване на важни събития, а от друга страна всеки потребител ще може да съобщава за настъпване на изключителна ситуация. В основата на система е замислена разработка на мобилно приложение, съвместимо с популярни мобилни операционни системи. Продуктът трябва бързо да бъде изведен на пазара. Кои атрибути на качеството бихте пренебрегнали, за да успеете със спазването на крайните срокове и защо мислите, че точно тези атрибути не са от значение?

- При положение, че софтуерът е за управление на кризисни ситуации, касаещи населението на една държава и необходимостта от бързата му поява на пазара да речем в рамките на 1 седмица, бих пренебрегнал на първо място разпределеността, преносимостта на приложението, тестваемостта и съвместимостта му. Защо? При фиксиран период от време и голямата нужда на пазара за подобен вид приложение, тези атрибути не са съществено важни, защото не допринасят за същевременната производителност на системата и ако трябва да бъдат спазени, то това би значило голямо забавяне до появата на системата на пазара. От друга страна, основното изискване към приложението е то да има високо ниво на надеждност, сигурност на потребителските данни, тъй като потребителят може да предоставя сензитивни лични данни като геолокация, здравословно състояние и др. Мащабируемостта е на второ място по значимост, защото системата трябва да се държи адекватно при рязко увеличаване броя на потребителите и техните заявки. Немислимо е тя да подава грешки или сървърът често да се изключва сам. Тук съществено важно е и поне да се осигури резервен ресурс (сървър), който да бъде използван само при изключително много потребителски заявки. На следващо място стои използваемостта на приложението, то трябва да бъде лесно и запомнящо се за използване от гледна точка на потребителски интерфейс. Като ръководител на екипа, бих използвал всички налични ресурси от гледна точка на разработчици, за да осигуря възможно най-бързата и качествена изработка на системата.