7.Техники (тактики) за постигане на качеството на софтуера (част 2)-Резюме

1.Тактики за изменяемост (Modifiability)

-Локализиране на промените: Намаляване на броя на модулите, засегнати от промяна.

Семантична свързаност (Semantic Coherence): Обединяване на функционалности, които са семантично свързани в един модул.

Очакване на промените: Списък на най-вероятните промени и оценка на декомпозицията. Помага ли така направената декомпозиция да бъдат локализирани необходимите модификации за постигане на промяната? Случва ли се така, че фундаментално различни промени да засягат един и същ модул?

Ограничаване на възможните опции: Намаляване на вариациите, които могат да засегнат много модули.

-Предотвратяване на ефекта на вълната: Ограничаване на модификациите до директно засегнатите модули. Ефект на вълната има тогава, когато се налагат модификации в модули, които не са директно засегнати от дадена промяна.

Скриване на информация (Information Hiding): – декомпозиция на отговорността на даден елемент (система или конкретен модул) и възлагането ѝ на по-малки елементи, като при това част от информацията остава публична и част от нея се скрива. Публичната функционалност и данни са достъпни посредством специално дефинирани за целта интерфейси. Това е най-старата и изпитана техника за **ограничаване на промените** и е пряко свързана с “**очакване на промените**”, тъй като именно списъка с очакваните промени е водещ при съставянето на декомпозицията, така че промените да бъдат сведени в рамките на отделни модули.

Ограничаване на комуникацията: Намаляване на броя на модулите, с които даден модул обменя информация - ограничават се модулите, които консумират данни, създадени от модул А и се ограничават модулите, които създават информация, която се използва от модул А.

Поддръжка на съществуващите интерфейси: Избягване на промени в интерфейсите чрез добавяне на нови интерфейси, адаптери или stub-ове.

Използване на посредник (Mediator): Премахване на зависимостите между модулите чрез посредници като wrapper, facade, adaptor, proxy.

-**Отлагане на свързването**: Контролиране на времето за внедряване и себестойността на промяната.

- **Plug-and-Play**: Включване/изключване/замяна на компоненти по време на изпълнението.
- **Конфигурационни файлове**: Задаване на стойности на параметри чрез конфигурационни файлове.
- **Протоколи**: Дефиниране на протоколи, които позволяват промяна на компоненти по време на изпълнение.

2. Тактики за сигурност (Security)

- **Устояване на атаки**:
    - **Authentication**: Проверка на идентичността на потребителите.
    - **Authorization**: Проверка на правата за достъп на потребителите.
    - **Confidentiality**: Криптиране на данни и комуникационни канали.
    - **Integrity**: Използване на чек-суми и хеш-алгоритми.

        - **Ограничаване на експозицията**: Намаляване на местата, чрез които системата може да бъде атакувана.
        - **Ограничаване на достъпа**: Използване на firewall и други средства за ограничаване на физическия достъп.
    - **Откриване на атаки**:
        - **Monitoring**: Наблюдение на системата за подозрителна активност.
        - **Intrusion Detection Systems (IDS)**: Системи за откриване на проникванията.
    - **Възстановяване след атака**:
        - **Audit Trail**: Поддръжка на копие на всяка транзакция за идентификация на извършителя и възстановяване от атаката.

3. Тактики за удобство при тестването (Testability)

- **Специализиран интерфейс за тестване**: Предоставяне на интерфейс за тестващ софтуер, различен от нормалния.
- **Вградени модули за мониторинг**: Поддръжка на информация за състоянието, натовареността и производителността на системата.

4. Тактики за удобство при употребата (Usability)

- **По време на изпълнението**:
    - **Обратна връзка**: Предоставяне на достатъчно информация и възможност за изпълнение на команди като cancel, undo, multiple views.
    - **Моделиране на задачата**: Системата знае какво прави потребителят и му помага (напр. AutoCorrection).

    - **Моделиране на потребителя**: Системата се адаптира към поведението на потребителя.
    - **Моделиране на системата**: Системата предоставя адекватна обратна връзка (напр. progress bar).
- **Свързани с UI**:
    - **Разделяне на интерфейса от реализацията**: Например при Model-View-Controller (MVC).

Заключение

- Постигането на качествата на софтуерната система изисква както архитектурни, така и не-архитектурни решения.
- Качествата не могат да се разглеждат в изолация, тъй като постигането на едно качество може да повлияе на друго.