Реализация и тестване на информационни системи

1. Реализация на информационни системи

Реализацията на информационните системи (ИС) е свързана с фазата разработка от жизнения цикъл на системите (виж лекция на тема „Жизнен цикъл на информационните системи"). В повечето модели тази фаза се намира между проектирането и тестването на разработваната система. Основната задача, която трябва да се извърши на фаза разработка, е реализиране на ИС на избран език за програмиране. Като входна информация за тази фаза се използват софтуерната архитектура и спецификацията на системните компоненти. Програмистите (кодировчиците) трябва да реализират проектните модели.

Основни принципи на фаза разработка са:

- Спазване на спецификацията;
- Следване на избран стил на програмиране;
- Постъпково усъвършенстване на ИС;
- Разработване на документация.

Като резултати от тази фаза се получават:

- Изходен код на ИС;
- Документация на разработения код;
- План за тестване;
- Документ за верификация.

Добрият стил на програмиране изисква спазването на методологията на програмиране. Методологията на програмиране е свързана с: избор на език за програмиране; използване на основни концепции; подредба на компонентите за разработка; оценка на грешки при кодиране. Тя се определя и от архитектурата на системата и спецификацията на системните модули. Модулите зависят от избрания подход за проектиране. Например, при структурно проектиране модулите са функционални, при обектноориентирано проектиране – класове, при модулно – функционални модули, модули на обекти от данни и др.

- Аспектите на добрия стил на програмиране са следните:
- Използване на коментари;
- Разработка на пълна документация;

- Подходящ избор на имена и идентификатори;
- Добра структурираност на програмите;
- Осигуряване на преносимост, бързодействие и сигурност (надежност).

Коментарите обикновено са на естествен език. Те не трябва да повтарят написания код. Да не прекалено много. Добре е те да са свързани с програмните единици и параметрите, например с класовете, типовете данни и променливи.

Към документацията на ИС е удачно да се включи и административна информация като: автор (-и) на системата; версия номер; статус (работна, завършена и др.); начин на стартиране.

За да се осигури сигурност (надеждност) на ИС е необходимо кодът на системата да не съдържа грешки и странични ефекти, да е добре тестван.

За да се осигури преносимост на ИС е необходимо кодът да е независим от хардуера и/или операционната система. За тази цел е удачно да се използват стандартни програмни конструкции, както и да се избягват нестандартни решения.

Един ефикасен метод за разработване на разбираеми и надеждни софтуерни системи е постъпковото усъвършенстване (Top down програмиране). При него разработката се реализира в следните стъпки:

1) Разработка на ИС с абстрактни данни (данни само с име) и абстрактни операции. Коментира (най-често на естествен език) се целият софтуер.
2) Усъвършенстване на коментарите – постепенно вместо коментарите се пише код или по-детайлни коментари. Това продължава докато се получи крайният код на системата.

Използването на този подход за реализация на ИС има следните предимства:

Основните нива на усъвършенстване могат да бъдат написани първо на естествен език;

Процесът на разработка е документиран в първичните кодове;

По-лесно и по-бързо се ориентират в същността на програмите нови програмисти;

Програмите са структурирани в две направления – контролни структури и нива на детайлизация.

2. Тестване на информационни системи

Тестването е една от основните и най-важни фази в процеса на създаване на ИС. На фаза се тестват отделните компоненти и тяхната интеграция в единна система. Тестовете са за неоткрити грешки, бизнес логика на софтуера и за сигурност. Резултати от последната фаза са тестовите протоколи и случаи.

Тестването на софтуера е изследване, което се прави с цел да се даде информация на всички заинтересовани страни за качеството на продукта или услугата. С него се гарантира и осигурява качеството на софтуера (Software Quality Assurance). Ролята на Quality assurance (QA) е да се намали риска от проблеми пред крайните потребители.

Тестването се разделя на два основни вида –системно и фунционално.

Основните принципи на тестването като фаза (дейност) в процеса на софтуерна разработка са:

- Тестовете трябва да могат да проследяват изискванията на потребителите.
- Планирането на тестовете трябва да започне преди самото тестване, веднага след спецификацията на изискванията към системата и след като проектът на системата е готов.
- За да бъде тестването ефективно, то трябва да се проведе от независима група за тестване.
- Тестването трябва за започне „в малкото” и да прогресира към тестване „в голямото”. Първите тестове се правят върху компоненти на системата. Последните тестове са за цялата система или за интегрирането и в други системи, ако има такова.

Съществуват два основни подхода за тестване на софтуерни системи: функционален и структурен.

Функционалното тестване (**Black-box)** се нарича още тестване на принципа на черната кутия. То е свързано с външна оценка на качеството на кода от страна на потребителите. Тестовете се осъществяват през графичния интерфейс на системата (Graphical User Interface - GUI). Прави се анализ на изходните резултати в зависимост от набора входни данни. Обект на изследване са функционалните изисквания за системата.

Black-box тестване – има за цел да провери функционалната коректност на системата, правилното приемане на входни данни и

предоставянето на коректни изходни резултати (данни). За провеждането му се използва интерфейсът на приложението.

Black-box тестовете се имат за цел откриването на грешки от следните групи (категории):

- Некоректни или липсващи функционалности;
- Грешки в интерфейсите;
- Грешки в структурите от данни или достъпа до базата от данни;
- Грешки, свързани с производителността и поведението;
- Грешки по време на инициализация и спиране.

При този тип тестове се изследва взаимодействието между различни функционални модули. Използват се набор от валидни (позитивен тест) и набор от невалидни (негативен тест) входни данни.

При **структурното тестване** (**Write-box)** се изследва софтуерът на системата на базата на познаването на програмния код и архитектурата й. Реализира се чрез:

- Тестване за „покриване" на кода, базирано на структурен анализ (Code coverage measurement) – реализира се чрез специализирани инструменти. Оценките за качество на системата се правят на базата на нейната структура и логика.
- Преглед на кода (Code Review).

  Този тип тестване е познато под името „бяла кутия" (Write-box)

Write-box тестване – структурно тестване, при което се контролира структурирането на кода на системата. То е свързано с процедурните детайли на приложението, с вътрешните логически структури. Чрез това тестване се гарантира, че:

- Всички независими пътища в модула ще бъдат изпълнени поне веднъж;
- Ще се изпълнят всички цикли в определените им граници;
- Ще се изпълнят всички логически решения;
- Ще се осигури валидност на всички вътрешни структури от данни.

Структурното тестване се осъществява по време на тестове на компоненти (Unit testing) и при интеграцията между тях (Integration testing). Като резултат от него може да се подобри бързодействието на системата.

Структурните тестове се делят на следните основни групи:

- ✓ Ориентирани към контролния поток;
- ✓ Ориентирани към потока от данни.

Ориентираният подход към контролния поток се състой в систематично преминаване през програмата, като се разглежда обхвата на изразите. За целта потокът на контрол се представя чрез граф, наречен Control Graph Flow (CGF). Върховете на този граф са линейна последователност от твърдения или изрази, а ребрата – възможните връзки между тези изрази. Един връх в графа може да представлява един или повече изрази, оператори или редове. Ребрата на графа представляват контролния поток на програмния код на системата. Всяко ребро трябва да завършва във връх дори когато съответният връх не е някакъв оператор (като например затваряща скоба). Връх, от който излизат повече от едно ребро се нарича условен. Ако условието се състои от повече от един израз, то всеки от тях може да се представи с отделен връх.

В ориентираните към потока от данни тестове се използват следните характеристики на променливите:

- Дефиниция на стойност (def) – присвояване, входен параметър;
- Използване на стойност (c-use) – използва се за изчисление на стойност на израз;
- Използване на стойност (p-use) – в условни разклонения.

На базата на тези характеристики графът на потока от данни (Data Flow Graph, DFG) може да се получи по следната формула

Data flow graph = CFG + def + c-use + p-use

Основните нива на тестване са:

- ✓ **Компонентно тестване (Component Testing, Unit);**

Компонентното тестване е синоним на Unit testing или на модулното тестване. Това е първия етап на тестването, който се прави, когато имаме готова работеща функционална единица. Като тази единица може да бъде метод, функция, процедура или клас. В над 90% от случаите това тестване се прави от програмистите (software developers). В най-общ вид това са бързи тестове, които показват какво е състоянието на самата система.

- ✓ **Интеграционно тестване (Integration Testing)**

Това е тестване, което се прави когато отделните компоненти са интегрирани, с цел да се провери дали те си взаимодействат по спецификация

след интеграцията им. Интерфейси и комуникация между системи също са обект на това тестване.

- ✓ **Системно тестване (System Testing)**

Системното тестване е основното при което проверява дали системата отговаря на спецификациите зададени от клиентите. По време на системното тестване се изпълняват тестови сценарии (test cases). В тестовите сценарии се описва последователността от действия за проверка на определехо функционално или техническо изискване към системата. Състоят се от идентификационната част и последователност от стъпки за изпълнение. Целта е да се верифицира даден процес и да се изследва поведението на системата при определени въздействия. Тестовите сценарии могат да бъдат позитивни и негативни. С позитивните се проверява дали системата прави това, което трябва, а с негативните – дали системата прави нещо, което не трябва.

Системното тестване може да бъде на функционално и нефункционално. Функционалното тестване е свързано с проверка на функционалността на системата. Нефункционално – с тестване на характеристики като производителност, надежност, удобство за използване и др.

- ✓ **Тестване за приемане (Acceptance Testing)**

То често се прави от самите клиенти. Идеята е да се провери от клиентска гледна точка дали системата работи съгласно предназначението спецификациите. Тук важен фактор е самото изпълнение на тестването да бъде на среда максимално близка до реалната.

Тестването на качеството може да бъде ръчно и автоматизирано.

При **ръчното тестване (QA Manual)** тестовете се изпълняват единствено от хора, без да се използват скриптове. В този случай познанията, свързани с програмни езици и писане на код, не се задължителни, а само препоръчителни. Ръчното тестване позволява да се направи оценка на удобството и интуитивността на даден софтуер. Използват се специализирани платформи за запис на грешки и бъгове или така наречените Bug Tracking Systems (Jira, Bugzilla, Redmine, Mantis и други). Ролята на QA е да опише намерените дефекти максимално ясно, за да може програмиста да ги види и отстрани.

При автоматизираното тестване тестовият процес се извършва посредством софтуер и скриптове. Целта е тестването да бъде изпълнено за възможно най-кратки срокове без намеса на ръчен тестер.