**15. Осигуряване на качеството на софтуера.**

* **Проблемът за управление на качеството**

**Осигуряването на качеството на софтуера (OKC)**не може да не се свърже с планирането и прилагането на подходящи мерки през целия производствен цикъл. Това е изразено много точно по следния начин:

**"Качеството е едновременно философия и съвкупност от ръководни принципи, които са основата на една *непрекъснато подобряваща се организация,* интегрираща:**

* **фундаментални техники за управление,**
* **постоянни усилия за усъвършенстване,**
* **технически средства,**

 **всичко това в рамките на един дисциплиниран и целенасочен метод."**

Качеството не може да бъде единствено задача, специално възложена на дадено лице или дори група, а трябва да се разглежда и като приоритетна отговорност на всеки, участващ в разработването на софтуерния продукт. Следователно методите и конкретните мерки на ОКС трябва да са такива, че да мотивират, ангажират и задължават всеки да работи за качеството на продукта.

**ОКС** е тясно свързано с жизнения цикъл на програмния продукт, по-точно с всяка негова фаза. Планирането му следва да се извърши още в началните фази и да обхваща всички следващи фази. Всеки междинен продукт, резултат от дадена фаза, заедно със своите характеристики следва да бъде точно дефиниран. На тази основа следва да се предвидят начини за обективно установяванедоколко удовлетворително е реализиран този междинен продукт (за крайния продукт казаното се подразбира). За да се осъществи тази дейност по един сис-
тематичен и ефективен начин, теорията препоръчва да се изработи **програма за
осигуряване на качеството на софтуера (ПОКС)***.*

* **Компоненти на програмата за осигуряване на качеството**

**Фактори**

Няколко фактора оказват важно влияние върху ПОКС. Без да се впускаме в подробности, ще ги изброим:

* изисквания към *графика;*
* разполагаемия *бюджет;*
* технологичната *сложност* на продукта;
* предполагаемия *размер* на продукта;
* относителния *опит* на разработчиците;
* хардуерните и софтуерните *ресурси,* предвидени за процеса на разработване;
* изискванията на *договора* за възлагане.

В момента на създаването на ПОКС трябва да се анализира и установи доколко всеки от горните фактори (а евентуално и някои други) ще й окаже влияние.

**Прегледи**

Под **преглед** (**review**) се разбира дисциплинирана групова дейност, насочена към изследване на продукт или процес.Ефективното изпълнение на прегледа изисква умело съставяне на групата с оглед комбиниране различните необходими квалификации. Отличават се **3 вида прегледи**:

**A. *Пробег (walkthrough)****.* Това е неформален преглед на софтуерния продукт. Обикновено не се подчинява на строги правила. Прилага се най-често върху първичния код на междинните продукти.

**Б. *Инспекция (Inspection)****.* Това е дисциплиниран формален преглед на всякакъв тип продукти.

**B**. ***Проверка на конфигурацията (Configuration Audit)****.* Много често крайното приемане на софтуерния продукт се основава на редица проверки на конфигурацията. Целта им е да установят доколко крайният продукт удовлетворява изискванията, формулирани първоначално. Тези проверки се делят на 2 категории:

**B.1. *Функционални.***Основната цел на функционалните проверки е да

преодолеят ефекта от многобройните тествания и съответни корекции. Напълно е възможно при установяване на дадена грешка в процеса на разработване и последващото нейно отстраняване да е била внесена друга грешка. Колкото подълго се разработва даден продукт, толкова повече опасността от такива "вторични", неотстранени грешки нараства. Крайната функционална проверка цели да покаже пълната липса на такива остатъчни грешки.

**Оценяване**

Оценка се прави обикновено от един специалист. Целта й е да се установи съответствие на всички характеристики на всеки продукт с формулираните изисквания. По същество това е функция по контрол на качеството на продукта. Тя обаче осигурява информация и за процеса на разработване. Поради тези причини всички дейности по оценяване трябва да се планират подробно и резултатите им да се използват възможно най-пълноценно. За отделните фази следва да се предвидят следните примерни дейности по оценяване.

**1. Изисквания към продукта***.* Отразяват на едно първо, но достатъчно точно приближение възгледите на потребителя за продукта. При планирането следва да се предвиди преглед и оценка на:

* плана за разработване на продукта;
* софтуерните стандарти;
* плана за управление на софтуерната конфигурация;
* плана за осигуряване на качеството;
* спецификацията на изискванията към продукта;
* спецификацията на изискванията към интерфейса.

**2. Общо проектиране.**Изискванията се конкретизират и уточняват, а така също потребителският възглед започва да се третира от гледна точка на реализацията. Съответни на това са и оценките на:

* всички ревизирани производствени планове;
* плана за тестването на софтуера;
* ръководството за оператора;
* ръководството за потребителя;
* ръководството за диагностика,

като последните 3 визират всъщност плановете за тези документи, които очевидно в този ранен стадий не могат да бъдат направени по същество.

**3. Подробно проектиране.**Резултатите от този етап са подробни спецификации, на основата на които може да бъде извършено програмирането (кодирането) на продукта. Тук се предвижда оценяване на:

* текущо ревизираните планове — отново;
* документа — изход от подробното проектиране;
* документа — проект на интерфейса;
* документа — проект на базата данни;
* тестовите примери за проверка на отделните модули;
* тестовите примери за проверка на интегрираните модули;
* описанието на процедурите по тестване;
* ръководството за програмиста;
* останалите ръководства в новото им състояние.

**4. Програмиране и тестване на отделните модули.** Самите изпълнители тестват своите реализации. През цялото време се извършват оценки на:

* плановете, доколкото търпят промени — както на всеки етап;
* написания първичен код;
* резултатите от тестването на отделните модули;
* всички ръководства.

**5. Интегриране и тестване.**Написаните и проверени отделни модули започват да се интегрират, което изисква оценяване на:

* текущо ревизираните планове;
* резултатите от тестването на интегрирането;
* актуализирания първичен код;
* описанието на приемните тестове;
* всички ръководства.

**6. Тестване на крайния продукт***.* Тук се оценяват:

* текущо ревизираните планове;
* всички ръководства;
* актуализираният първичен код;
* документът за описание и управление на версиите;
* докладът за тестване на крайния продукт (системата).

**Типове оценки**

По-важните от ***типовете оценки***:

* съблюдаване на изисквания формат и стандарти за документацията;
* съответствие с изискванията на договора за разработка;
* вътрешна непротиворечивост;
* добра разбираемост;
* система за лесно намиране на пътя до всеки документ;
* съответствие на продукта с придружаващите го документи;
* коректно извършен анализ на изискванията, проектиране и програмиране (тук следва

да се отбележи, че всъщност се прави оценка не на атрибути на продукта, а на качества на процеса);

* правилно разпределение на ресурсите — по време и памет;
* адекватно проведено тестване за съответствие на продукта с изискванията;
* адекватно съставени тестови примери;
* пълнота на тестването;
* пълнота на регресивното тестване;
* съответствие и непротиворечивост между дефинициите на данните и тяхното

използване.

**Управление на конфигурацията**

Управлението на софтуерната конфигурация обхваща методи и технологии за иницииране, оценяване и управление на измененията в софтуерния продукт след пускането му в експлоатация. Необходимите промени се правят не само в първичния код, но така също в документацията — вътрешна (съпровождаща) и потребителска, в отчетите за тестване, в отчетите за откритите грешки. Следят се и се документират последователно създаваните версии и движението им сред различните потребители.

Ето **основните функции по управления на конфигурацията**, които са решаващи **за запазване качеството на продукта** след влизането му в експлоатация:

* поддържане целостността на продукта;
* напълно контролирано управление на промените;
* управление и контрол на версиите;
* планиране на управлението на конфигурацията.

**Отчитане на грешките**

Ефективна система за регистриране на грешките с оглед последващото им отстраняване трябва да е действена в следните насоки:

1. **Идентификация на грешките***.* Всяка идентифицирана грешка трябва да се

опише ясно и точно. Може да се опише както поведението на продукта в дадената ситуация, така и реалният дефект в софтуера.

1. **Анализ на грешките***.* Трябва да се документира сериозността на грешката и

трудността на нейното отстраняване. Това ще позволи правилно заделяне на не обходимия ресурс, определяне на приоритет и график за корекция. Обикновено грешките се откриват сравнително лесно, често те просто се набиват на очи, но тъй като отстраняването им невинаги е лесно, налага се да се вземат управленски решения, свързани с приоритета им. Впрочем отделянето във времето на анализа от корекцията обуславя разглеждането им като отделни и независими операции.

1. **Корекция на грешките.**Коригирането, освен че се извършва по същество, следва и

да се документира. Описанието на корекцията трябва да съдържа:

* разказвателно описание на корекцията;
* списък на засегнатите модули от продукта;
* пълна идентификация според създадената система на засегнатите документи;
* евентуални промени в тестовите процедури в резултат на корекцията.
1. **Въвеждане на корекцията в експлоатация***.* На практика не е възмож но всяка

направена корекция да отива незабавно при потребителя. Затова обикновено се прави работен екземпляр, в който постепенно се натрупват направени корекции. В определен момент, резултат на управленско решение, се създава нова модификация на продукта, включваща всички направени корекции (а така също подобрения и нови функционалности, ако е имало такива) и тя се предлага официално на потребителите. Задължително се регистрира в коя модификация (или версия) всяка корекция е включена.

1. **Регресивно тестване***.* Вече беше казано, че всяка корекция крие потенциална

опасност от създаване на нови грешки. Изследвания показват, че тази опасност е около 20%. Най-разумната възможност за отстраняването им е в момента на откриването им те да бъдат третирани като стандартни грешки и да бъдат отстранявани по вече описаната схема. Във всички случаи обаче описанието на регресивния тест трябва да включва:

* списъка на отново тестваните компоненти;
* върху коя версия/модификация е направено тестването;
* индикация, дали тестването е било успешно или неуспешно.
1. **Категоризация на грешките***.* Анализът на грешките би се улеснил, ако те се

систематизират с оглед бъдещи анализи. Близко до ума е, че категоризирането ще е най-успешно, ако се направи в момента на отстраняването на всяка грешка. Възможни признаци за категоризация са:

* тип на грешката — от изискванията, от проектирането, от програмирането, от

тестването;

* приоритет на грешката — критична, некритична, козметична;
* честота на грешката — повтаряща се, неповтаряща се.

**Анализ на тенденциите**

Това е пасивна и по-второстепенна дейност, но тя може да насочи към полезни корективи. Състои се в анализиране на определени страни от софтуерния процес и изготвяне на съответни отчети.

1. **Количество на грешките.**Възможно е да се събират данни за количеството

грешки както за целия период на разработването, така и за отделни кратки периоди.

1. **Честота на грешките.**Честотата на грешките следва да се свързва с конкретна

единица — тестова процедура, програмен модул, дял от спецификация. Също след

статистически обработки могат да се правят изводи за латентни грешки в тези единици.

1. **Сложност на програмните модули***.* Известни са немалко метрики, които

позволяват обективно измерване на сложността на програмните единици. При установяване на по-голяма от някаква критична стойност, може да се пристъпи към опит за намаляване на сложността на тази единица.

 **4. Честота на компилациите.**Де Марко е показал, че програмни модули, които са били компилирани често при създаването им, след това се компилират често и при интеграцията и системното тестване. Така че би било оправдано за програмна единица, за която е установено, че при кодирането е била компилирана много често, да се определи задължително процедура по оценяване.

**Проследимост**

Принципът за **проследимостта**означава, че за всяка единица, създадена по време на разработването на софтуера, трябва да може да се проследи от каква друга единица е получена. Така например за даден алгоритъм, въплътен в програма, следва да може да се види резултат на какво решение е по време на проектирането или на създаването на изискванията. Благодарение на това се улесняват оценките и преди всичко отстраняването на грешките.

**Планиране на ПОКС**

Както всяка друга дейност при производството на софтуер програмата за осигуряване на качеството е също обект на грижливо предварително планиране. Практическият съвет в това отношение е да се ползва аналогичен план от предходна разработка. Най-малкото, с което може да бъде полезен подобен план, е, че няма да бъдат пропуснати задължителните елементи.

**Социални фактори**

Все по-често се обръща внимание на социалните елементи, начина на общуване на ръководителите с подчинените и с клиентите, мотивацията на разработчиците. Доколкото осигуряването на качеството е особено силно свързано с общуването между членовете на екипа, често при противопоставящи роли е необходимо да се обърне внимание и на този елемент от софтуерния процес.

1. **Точност.**Абсолютната точност на представянето на данните от всяка оценка или

друга дейност по осигуряването на качеството е задължителна. Практиката показва, че най-ефективната защита на контролираните срещу искания на контролиращите, например за извършване на промени, е позоваване на неточност на данните. При това обикновено се цитират прецеденти и доколкото почти винаги времето за подробно разглеждане и доказване на представените данни не е достатъчно, такова позоваване се оказва достатъчно. Единственият начин за противопоставяне е просто да не се създава нито eдин прецедент за неточност.

1. **Авторитет.**Обикновено отговорниците по осигуряване на качеството получават

значителни пълномощия от ръководството на фирмата. Както и в други дейности oбаче, знае се, че истинският авторитет е много no-малко резултат на административни мерки, отколкото на доказвана непрестанно лична компетентност.

1. **Полза.**По принцип ефективната работа по осигуряване на качеството води до

обща полза за всички участници в процеса на разработване. От друга страна обаче, в краткосрочен и индивидуален план много често това не е така, защото на даден програмист например му се налага да прави корекции, които смята за ненужни и които му отнемат допълнително време и усилия.

1. **Комуникации.**Както вече се каза, осигуряването на качеството в голяма степен

като схема е натрупване и разпространяване на информация. Формите за това са обикновено речеви и писмени в различни модификации. Откъдето следва, че отговорните за осигуряване на качеството трябва да са комуникативни личности, умеещи добре да се изразяват говоримо и писмено.

1. **Постоянство.** He e възможно да се запази доверието, ако концепциите

и произлизащите от тях решения се променят често. В такива случаи скоро идва момент, от който нататък нарежданията просто не се изпълняват в очакване на последваща промяна.

1. **Отмьстителност.**Лице, отговарящо за осигуряване на качеството,

нерядко ще се сблъсква с волни или неволни нарушения. Те биха му дали възможност при следващи случаи да се изкуши да се възползва от създалото се предимство пред нарушителя. Ясно е, че такава злонамереност е вредна за общата атмосфера и за ефективността на работата във фирмата.