10. Софтуерни надеждностни Концепции
10.1. Терминология
Надеждностни системи: системи, които трябва да бъдат надеждни. Надеждните системи са
системи, които отговарят за критични функционални изисквания за надеждност, безопасност или сигурност.
Софтуерна Надеждност: вероятността, че програмата се изпълнява успешно, в съответствие със спецификациите за определен период от време.
$R_{sy} = R_s * R_h * R_o$
Rsy = надеждността на системата
R_s = Софтуерна Надеждност
RH = Хардуерна Надеждност
R_o- операторна Надеждност
Това предполага, хардуер, софтуер и оператора за грешки да бъдат взаимно изключващи се.
Провалът е напускането на поведение на системата при изпълнение на изискванията на потребителите, той е потребителски ориентирана концепция. Липсата на софтуер трябва да се получи по време на изпълнение на програма. Потенциалните повреди намерени от програмисти, в резултат на дизайна проверка, код за четене и други методи не се броят за провал.
Повредата е дефект, който потенциално причиняват повреда, когато се изпълнява.Това е Проектоориентирана концепция. Софтуерна вина е дефект в кода. Тя се причинява от грешка, която е невярна или липсва действие от лице или лица. Грешките са човешки грешки, които да влязат в софтуера. Дефектите са неправилни условия в програма, която обикновено води до грешка.
Неизпълнението на интензитета е един алтернативен начин за изразяване на надеждноста. Надеждността е вероятността, че системата работи без повреда за определен брой физически
единици или на определено време, известни като мисия време. Неизпълнение на интензитета се определя като неуспехи за единица време. Времето е общото времето за изпълнение или могат да бъдат физически единици. Този термин е използвани в софтуерното инженерство за надеждност поради своята простота и интуитивен обжалване.

Клас на неизпълнение на тежестта е съвкупност от пропуски, които имат едни и същи-недостатъци и въздействие върху потребителя. В категорията риск са възложени повреди предимно за използване на неуспешни честоти за да се даде приоритет на неуспехите за резолюция. Общи критерии за класификация включва човешкия живот, разходите и въздействие на способността на системата. Всеки от тези критерии, може да включва много критерии Sub, някои от които могат да бъдат важни за конкретното приложение. Пример, въздействието на разходите може да включва допълнителни оперативни разходи, ремонт и възстановяване на разходите или загуба на настоящи или потенциални бизнес. Способността на системата може да включва под критерии като загуба на критични данни, възстановяване и престой. За системи, за които наличността е важна, повреди, които водят до по-голяма престой често ще бъдат поставени в по-висок клас недостатъчност тежест. Проблемите за потребителите срещат затруднения. Те могат да са резултат от повреда или

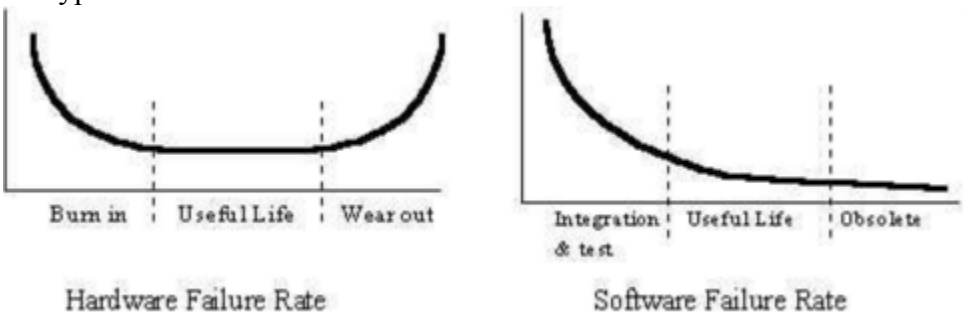
злоупотреби.
Изпълнението на спецификацията е писмено изискване, фигура и фигурата на заслуги, или

параметър, който качествено или количествено определят производителността на системата.
ПОДРАЗБИРАЩИ- спецификацията е едно неписано изискване, което се разбира от голямата част от екипа на проекта. Всички са равностойни по същество на писменото изискване.
Проверката е опит да се намерят грешки от изпълнение на програмата по време на тест или
симулирана среда. Потвърждаването е един опит да се намери грешки от изпълнение на програмата в дадена реална околната среда.
Сертифицирането е официално потвърждение на правилността на дадена програма.
В IEEE определя надеждността като "Способността на една система или компонент, да изпълнява своите необходими функции съгласно гореспоменатите условия за определен период от време. За повечето проекти и мениджъри разработка на софтуер, надеждността е приравнена към коректност, чрез изследване и броят на "бръмбарите" трябва да бъде открит и фиксиран. Въпреки че намирането и оправянето на бъгове, открити в изправност е необходимо да се осигури надеждност, по-добър начин е разработване на мащабен продукт с високо качество през всичките етапи на софтуерния жизнен цикъл. Тоест, достоверността на представения код е свързана с качеството на всички процесите и продуктите на софтуерната разработка; изискваната
документация, кодът, тест плановете и тестовете.
Софтуерът за надеждността се състои от три дейности:
I. Грешка превенция.
II. Откриване на грешки и отстраняване.
III. Измерванията за постигане на максимална надеждност, по-специално мерките, които поддържат първите две дейности.
10.2.1. Грешки, и неуспехи
Условията-грешки, пропуски и неуспехи често се използват взаимозаменяемо, но имат различни значения. В софтуера, грешка обикновено е програмисткото действие или бездействие
която води до повреда. А вината е софтуерен дефект, който причинява повреда,а повреда е

недопустима за работата на програмата от програмните изисквания.
При измерване на надеждност, ние обикновено измерване само дефекти, открити и фиксирани. Ако целта е да се създаде мярка за надеждност трябва да обърнем внимание на превенцията,
както и на развитието, започващо от изискванията на фазата – това колко програми са разработени. Важно е да се каже, че има разлика между хардуерен срив
и отказ на софтуера. За хардуера, както е показано на фигура 10.1, когато
компонента е произведен първи първоначалния брой грешки е голям, но след това намалява като дефектните компоненти са установени и отстранени или компоненти те се стабилизират. Компонентът влиза във фазата на полезния живот, когато няколко такива

грешки са открити. Тъй като компонента физически се изхабява курса на повредата започва да се увеличава.

Фигура 10.1:



При софтуера има различна грешка или процент на грешки при идентификация. За софтуера, процентът на грешките е на най-голям в нивото на интеграция и тест. Тъй като е тестван,
грешките са установени и отстранени. Това отстраняване продължава по-бавно по време на
оперативното използване, а броят на грешки непрекъснато намалява, ако не са въведени нови грешки . Софтуерът не са движещите се части и не се износва физически както хардуера, но надживява полезността си и се превръща в отживелица. За да се увеличи надеждността чрез предотвратяване на софтуерни грешки, фокусът трябва да бъде върху цялостните изисквания и подробно тестване и осигуряване на всички изисквания. Фокусът също трябва да е и върхупоправката на софтуера тъй като там започва фазата"полезен живот" ,чието поддържането ще бъде от инжеера. Следователно, за да се предотвратят софтуерни грешки, ние трябва да:
I. Започване с изискванията, като се гарантира развитието на продукта по едни определени

изисквания докато ясно и точно се определи окончателен функционален продукт.
II. Гарантиране на кода,че може лесно да подкрепя поддържането без вливане на допълнителните грешки.
III. Всеобхватна програма за изпитване, която да проверява всички функционалности в посоченото в изискванията.
10.2.2. Отказ на софтуерните механизми
Софтуерните неуспехи могат да се дължат на грешки, неясноти, пропуски или неправилно тълкуване на спецификацията, на която софтуерът е трябвало да отговаря, небрежност или

некомпетентност в писмения код, неадекватно тестване, неточно или неочаквано използване на
софтуера или други непредвидени проблеми. Изкушаващо да се направи аналогия между софтуерна надеждност и хардуерна надеждност,но те имат основни разлики, които ги правят различни механизми за неуспех. Откази на хардуера са най-вече физически дефекти, докато софтуерни грешки са проектиране и дефекти, които са по-трудни за

визуализация, класифициране и откриване . Грешките на дизайна са тясно свързани с човешките фактори при процеса на проектиране, когато има неясноти. В хардуерния дизайн грешка може да има, но физическите повреди обикновено доминират. При софтуера едва ли може да се намери строг критерии, съответстващ на производството при хардуера, ако простото действие на качване на софтуер модули на място не се брои. Поради това качеството на софтуера няма да се промени след като се качил в склада и започне да действа. Опитвайки се да постигнем по-висока надеждност, като просто дублираме еднакви софтуерни модули няма да работи, защото грешките на дизайна не могат да бъдат открити на разстояние. Частичен списък на отделните характеристиките на софтуера в сравнение с хардуера са дадени по-долу:
I.Причини на липсата: Софтуер дефектите са предимно дизайн дефекти.
II. Носене и изчакване: Софтуерът не е свързан с енергопотреблението при фазата на износване и изчакване. Грешките могат да възникнат без предупреждение.
III. Концепция поправка на системата: Периодичното рестартиране може да помогне при определяне на софтуерни проблеми.
IV. Времезависимост и жизнен цикъл: Софтуерната надеждност не е функция на оперативен път.
V. Факторите на околната среда: Да не се засяга софтуерната надеждност, освен ако това може да засегне програма за суровини.
VI. Надеждностни прогнози: Софтуерната надеждност не може да бъде предсказана, тъй като това зависи изцяло от човешкия фактор в дизайна.
VII.Редундация: Не може да се подобри надеждността на софтуера ако се използват еднакви софтуерни компоненти
VIII. Интерфейси: Софтуертените интерфейси са чисто концептуални, а не само визуални.

IX. Мотиватори за неизпълнение на задълженията: Обикновено са непредвидими от анализите на отделните отчети.
Параметър Експлоатация Примерна система
POFOD
(Вероятност за неспазване на Търсенето)
Мярка за вероятността, че системата ще се провали ,когато услугата е отправена към искането.
POFOD = 0.001 означава,че 1 от 1000 услуги може да доведе до провал.
Безопасност на критични и нон-стоп системи като хардуер
Системи за управление.
ROCOF (Процент на неизпълнение появата)
Мярка за честотата на възникване, с която неочакваното поведение е възможно да стане. Ако ROCOF = 2 / 100 означава 2 грешки в 100 оперативните звена време. Тази мярка показва липса на интензивност. Операционни системи, Системи за обработка на транзакциите
MTTF / СВДО
(Средно време на липса)
Мярка за времето между наблюдавана система за неуспехи. Например, един от 500 MTTF / СВДО средства 1 недостатъчност може да се очаква всеки 500 път единици. Това е реципрочно на ROCOF.

Системи с дълги сделки, като например CAD, когато MTTF / СВДО> Транзакция време.
Полза (Наличност)
Как може системата да бъде на разположение за употреба 0.998 означава на всеки 1000 единици на тази система вероятно ще бъдат на разположение за 998 от тях.
Непрекъснато движение системи като телефонникомутационни системи.
10.1. Измерване за оценка на надеждността
(А) . Системата за повреда дава номер на система за вход. Това се използва за POFOD мярка.
Б) Времето между системите за повреди. Това се използва за мярка ROCOF и MTTF / СВДО.
(В) изминали ремонти или подновяване на момента на отказа на системата. Като се има предвид, че системата трябва да бъде постоянно на разположение, това се използва за измерване на наличност.
10.2. Безплатни стратегии за постигане на надеждност
(А) повреди при избягване: Проектирането и изпълнението като процеси трябва да бъдат организирани с цел производство на системи.
(Б) Устойчивост на откази: Тази стратегия предполага, че остатъчните грешки остават в системата. Съоръженията са разположени на софтуер, за да се позволи експлоатация им да продължи когато грешки причинят повреди на системата.
(В) за откриване на неизправности: Грешка, които са разкрити преди софтуерът да е въведен в експлоатация.Процесът на утвърждаване на софтуера използва статични и динамични методи, за да останат всички дефекти, които остават в системата, след изпълнението им.
10.2.1. Избягване на грешки
Избягване на повреди за развитието на софтуера разчита на: -
(А) Наличието на точна (за предпочитане официална) системна спецификация, която е недвусмислено описание на това, което трябва да се прилага.
(Б) Организационна философия на качеството, като от програмистите се очаква да пишат без грешки програмите.
(В) Използване на висок вид език за програмиране, така че да е възможно грешката да е открита от езика компилатор.
(Г) Ограничения за използване на програмни конструкции, като указатели, които по своята същност са застрашени от грешки.
10.2.2. Устойчивост на откази
А)отказоустойчива система може да продължи експлоатацията си след ненастъпило изменение на някоя друга система. Устойчивост на откази е необходима в случаите, когато липсата на система ще доведе до някои катастрофални аварии или когато загубата на работата на системата ще доведе до големи икономически загуби. Например, компютърът в един самолет трябва да продължи работата си докато самолетът се приземи, компютрите в една система за въздушен контрол на трафика трябва да бъдат постоянно на разположение. Никога не може да бъде убедително доказано, че системата е напълно безаварийна.Толерантноси на повредени съоръжения са необходими, така че системата да е устойчива на неуспех. Има четири аспекта за устойчивост на откази:
(А) откриване на грешка: Системата трябва да открие, че дадена комбинация води или ще доведе до повреда в системата.

(Б) оценка на пораженията: Частите на състоянието на системата, които са били засегнати от повредата, трябва да бъдат открити.
(В) повреди при възстановяване: Системата трябва да възстанови състоянието си до известно безопасно състояние. Това може да се постигне чрез коригиране на повредено състояние (Препращане на грешка за възстановяване), или чрез възстановяване на системата до известно безопасно състояние (със задна дата на грешката при възстановяване). За напред грешката възстановяване е по-сложна.
(Г) за отстраняване на дефекти: Това включва промяна на системата, така че вината за повредата да не се повтори. В много случаи, софтуерните неуспехите са преходни и са резултат от
специфични комбинации от системата за вход.Ремонт може да не се наложи ако нормалната обработка може да се възобнови веднага след отстраняването на грешка