**Universitatea Politehnică Bucureşti**

**Facultatea de Electronică, Telecomunicaţii si Tehnologia Informaţiei**

**Sisteme Critice**

**Student: Stefan Dumitru**

**Grupa: 443A**

**Disciplina: Inginerie Software**

**Profesor: Ştefan Stăncescu**

**Cuprins:**

1. **Prezentare generală, definitii**
2. **Analiza cerintelor de baza**
   1. **Fiabilitatea**
   2. **Siguranta**
3. **Securitate**
4. **Etapele dezvoltarii programelor**
5. **Testarea securitatii si a fiabilitatii**
6. **Exemple si concluzii**
7. **Prezentare generală, definitii**

Siteme critece de siguranta – un sistem a carui defectare induce vatamare, pierderea unei vieti sau pagube serioase ale mediului inconjurator. Un exemplu ar fi sistemul de control al unei rafinarii.

Sisteme critice de misiune – un sistem a carui defectare induce neatigerea scopului unei activitati de tip misiune. Exemplu ar fi sistemul de navigatie al unei navete spatiale.

Sisteme critice de afacere – un sitem a carui defectare induce costuri mari afacerii care se foloseste de acest sistem. Exemplu ar fi sistemul de gestiune al unei banci.

Implicatiile mari ale defectarii unui sistem critic inseamna ca in dezvoltarea sa se vor folosi doar metode de incredere. In consecinta sistemele critice sunt create folosind tehnici consacrate si nu tehnici noi sau experimentale a caror eficienta nu este inca dovedia. Dezvoltatorii unor astfel de sisteme sunt in natura lor conservatori si prefera sa urmeze tehnici mai vechi ale caror calitati si puncte slabe le sunt bine cunoscute. Thehnicile noi care pot parea ca fiind mai bune pot avea problem in folosirea pe lunga durata.

Se poate apela la metode ingineresti mai putin eficiente ca si cost. Spre exemplu metode formale matematice ale dezvoltarii software au fost folosite cu succes pentru sisteme de siguranta si securitate. Unul dintre motive ar fi ca folosirea lor reduce volumul de testare necesar care pentru sistemele critice este mai mult de 50% din de dezvoltare.

1. **Analiza cerintelor de baza**

Analiza cerintelor unui program de monitorizare se face strict in functie de sistemul si parametrii ce trebuiesc monitorizati. Acesti parametrii trebuie definiti foarte bine iar sitemul analizat in detaliu incat sa se poata gasi modalitatea cea mai eficienta de a ii extrage si prelucra.

Daca complexitatea sistemului ce trebuie monitorizat este foarte mare necesitatile nu vor putea fi satisfacute de un singur program de monitorizare deoarece complexitatea acestuia ar fi proportional de mare si ar afecta eficienta sa. In schimb sitemul va fi inpartit in diferite segmente in functie de rolul functional si modalitatea de interconectare cu celelate parti si fiecare dintre aceste segmente va avea propriul program de monitorizare croit pe necesitatile sale specific.

Timpul de reactie este critic in astfel de aplicatii si este dictat de nevoia de valabilitate a sistemului monitorizat.

Atunci cand este prezentat un stimul, aplicatia thebuie sa produca un raspuns intr-un interval de timp specificat. Acesti stimuli sunt de 2 tipuri:

* Periodici – stimulul apare la un interval de timp dat. Exemplu ar fi un senzor de temperature care este interogat la fiacre 5 secunde sau nivelul de incarcare al unui procesor.
* Aperiodici – stimulul apare la timpi inprevizibili deoarece sunt determinati de factori externi aleatori. Exemplu ar fi o pana de curent care declanseaza un senzor de votaj.

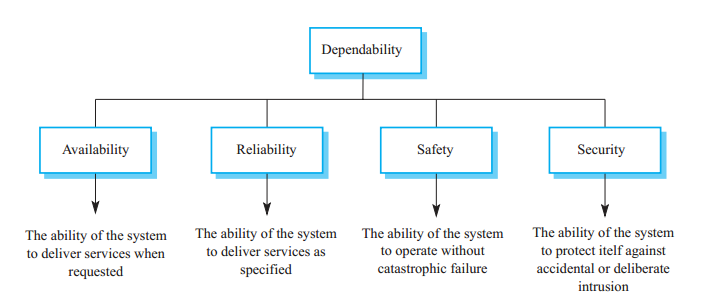
15.1 SensorActModel(13.1).eps                                  0007B854Macintosh HD                   B8AA5F2E:

Sistemul de comutare al stimulilor trebuie sa fie unul care satisface necesitatilede raspuns in timp.

15.2 RTSysModel(13.2).eps                                      0007B854Macintosh HD                   B8AA5F2E:

* 1. **Fiabilitatea**

Cele patru dimensiuni ale calitatii unui sistem critic sunt valabilitatea, fiabilitatea, siguranta si securitatea.



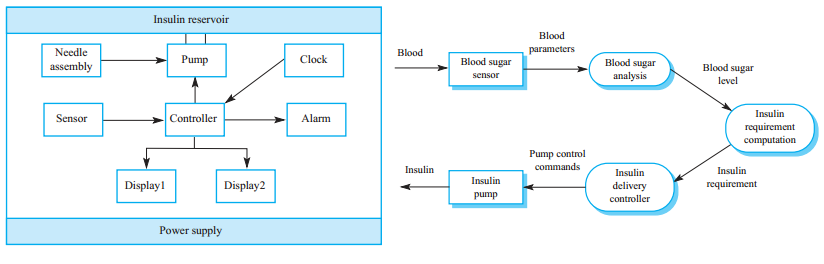
Gradul de calitate reflecta increderea pe cere userul o are in sistem. Sistemele nefiabile sau nesigure sunt respinse de catre utilizator.

Alte caracteristici ale calitatii:

* Reparabilitatea
* Mentenanta
* Capacitatea de supravietuire
* Toleranta la erori

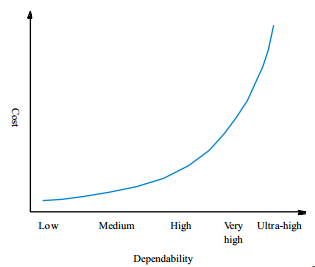
Un sistem poate ceda datorita problemelor hardware, software sau de operare.

*Ex. O pompa de insulina controlata software*



Costurile calitatii cresc exponential cu cresterea in nivel a calitatii.

Acest lucru se intampla deaoarece este necesara folosirea tehnicilor de dezvoltare mai costisitoare cat si hardwareului care sa indeplineasca standardele ridicate de fiabilitate in operare necesare acestor nivele superioare. Deasemenea testarea si validarea unor astfel de sisteme necesare comvingerii clientului ca nivelul cerut a fost atins simt si ele foarte costisitoare.



Prabusirea sistemului – un eveniment ce apare la un moment dat cand sistemul nu mai livreaza un serviciu in modul in care utilizatorul se asteapta si are nevoie.

Eroare de sistem – o stare de eroare a unui sistem care poate duce la comportament neasteptat al sistemului.

Defect de sistem – o caracteristica a sistemului software care poate duce la o eroare de sistem. Spre exemplu incapacitatea de a initia o variabila poate face ca acea variabila sa aiba o valoare gresita atunci cand va fi folosita.

Eroare umana – comportament uman care rezulta in introducerea de defecte in sistem.

Prabusirile sunt de obicei un rezultat al erorilor de sistem care sunt derivate din defecte de sistem. Totusi defectele nu rezulta mereu in erori de sistem.

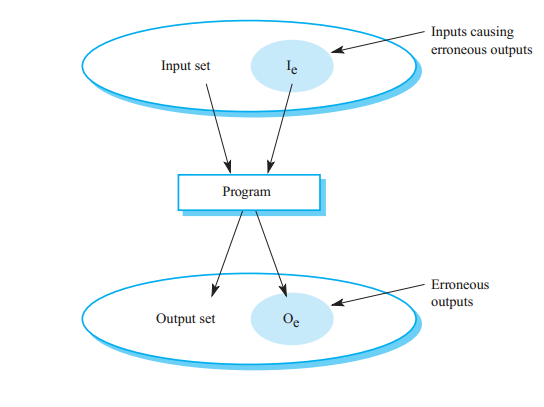
Erorile nu rezulta mereu in prabusirea sistemului. Exista posibilitatea ca ele sa fie detectate si corectate in timp util.

Perceptia utilizatorului nu este mereu aceeasi cu definitia fiabilitatii unui sistem. Presupunerile facute despre mediul unde un sistem va fi folosit pot fi incorecte. Consecintele prabusirii sistemului afecteaza perceptia fiabilitatii. Situatiile in cre consecintele sunt mai serioase, precum nefunctionarea motorului intr-o masina, primesc o pondere mai importanta decat situatiile in care se produce un inconvenient, precum arderea unui bec.

Fiabilitatea se obtine prin abordarea problemelor in diferite stagii ale aparitiei lor:

* Evitarea defectelor – se folosesc metode de dezvoltare care minimizeaza sau detecteaza greselile inainte ca ele sa introduca erori de sistem
* Detectie si indepartare de erori – exista tehnici de validare si testare a sistemului care identifica si corecteaza problemele inainte ca sistemul sa fie dat in folosinta
* Toleranta la defect – se folosesc tehnici care asigura ca defectele nu vor duce la erori sau prabusiri de sistem, tehnici precum redundanta si menajarea axceptiilor.

Modelarea fiabilitatii se poate face prin maparea intrare-iesire. Fiabilitatea sistemului este probabilitatea ca un anume input va introduce o problema in setul de input-uri si va cauza outpu-uri eronate. Persoane diferite vor folosi sistemul in modalitati diferite si deci aceasta caracteristica nu este un sistem static de atribute ci depinde de mediul sistemului.



Indepartarea unui procent din defectele unui sistem nu duce la inbunatatirea fiabilitatii acelui sistem cu acelas procent. Un studiu IBM arata ca indepartarea a 60% din defecte a dus la o imbunatatire a fiabilitatii percepute cu doar 3%.

Unele defecte se afla in parti rar executate ale sistemului si deci care sunt rar intalnite de catre utilizator. Rezolvarea acestora nu inbunatateste cu mult fiabilitatea perceputa.

* 1. **Siguranta**

Siguranta este o proprietate a sistemului care reflecta abilitatea sa de a opera, in parametrii normali sau anormali, fara pericolul de a cauza vatamare sau fara pagube mediului sistemului.

Este din ce in ce mai important sa consideram siguranta software deoarece din ce in ce mai multe sisteme inplementeaza control pe baza softwareului.

Cerintele de siguranta sunt unele de excluziune deoarece ele tind sa excluda situatiile nedorete si nu sa specifice necesitatea unor servicii de sistem.

Sisteme de siguranta critica primara: cedarea lor duce la defectiuni hardware si in consecinta punerea in pericol a personelor.

Sisteme de siguranta critica secundara: defectarea lor poate duce in defectarea altor sisteme de tip primar.

Siguranta si fiabilitatea sunt inrudite dar distincte. In general fiabilitatea este necesara dar nu suficienta pentru a indeplini si conditia de siguranta a sistemului. Fiabilitatea este preocupata cu conformarea la o specificatie data si livrarea serviciului. Siguranta este preocupata cu asigurarea incapacitatii sistemului de a vatama indiferent daca se conformeaza cu serviciul ce trebuie livrat.

Sisteme fiabile si care totusi sunt nesigure sunt cele care se comporta ca in specificatii dar totusi provoaca accidente.

Terminologia in siguranta:

Accident – un eveniment sau secventa de evenimente neplanificate care rezulta in vatamarea sau moartea unei persoane, pagube materiale sau mediului.

Pericol – o conditie care are potentialul de a cauza sau contribui la cauzarea unui accident.

Paguba – o masura a pierderilor rezultate din defectarea sistemului.

Severitatea pericolului – o estimare a celui mai defavorabil caz care ar putea fi provocat de catre un anumit pericol.

Probabilitatea pericolului – probabilitatea de aparitie a unui eveniment care sa produca un pericol

Risc – este estimarea posibilitatii ca sistemul sa produca un accident

1. **Securitate**

Securitatea unui sistem este proprietatea sa de a se proteja impotriva atacurilor externe accidentale sau itentionate. Aceasta este din ce in ce mai importanta deoarece sistemele sunt interconectate incat accesul extern prin internet sa fie posibil.

Securitatea este o cerinta esentiala a disponibilitatii, fiabilitatii si sigurantei. Daca un sistem este interconectat si nesigur atunci fiabilitatea si siguranta sa sunt neadecvate deoarece depind de faptul ca sistemul ce este executat si cel care a fost dezvoltat sa fie aceleasi iar instructiuni externe pot chimba sistemul ce este executat.

Posibile consecinte alea lipsei de securitate:

* Negarea serviciului – blocarea sau icetinirea semnificativa a sistemului
* Coruperea programelor si datelor – prin modificare neautorizata
* Expunerea informatiilor confidentiale

Asigurarea securitatii se paote realiza punctand urmatoarele:

* Evitarea vulnerabilitatilor – prin design
* Detectie si eliminare de atacuri
* Limitarea expunerii

1. **Etapele dezvoltarii programelor**

Procesul de design al unui sistem real time incepe cu identificarea stimulilor si reactia necesara fiecaruia. Apoi pentru fiecare stimul se identifica timpul de raspuns impus. Se agrega stimulii si procesele de raspuns intr-un proces concurent asociat fiecarei clase de stimuli si raspunsuri. Se construiesc algoritmi care sa proceseze fiecare clasa si care sa se incadreze in cerintele de raspuns in timp si apoi totul trebuie integrat folosind un sistem de operare real-time.

Constrangerile temporare pot necesita simulari ample pentru a asigura ca sunt indeplinite de catre sistem. Acest lucru poate insemna ca anumite strategii de proiectare precum cele obiect-orientate nu pot fi abordate datorita complexitatii pe care o implica. Poate insemna chiar necesitatea folosirii unui limbaj de nivel jos pentru a obtine eficienta necesara.

Aici avem un exemplu de sistem de monitorizare in timp real implementat intr-o pompa de benzina.

15.3 PetrolPumpState(13.3).eps                                 0007B854Macintosh HD                   B8AA5F2E:

**Elemente de sistem:**

Procese de control al senzorilor care colecteaza date de la senzori si se pot folosi de buffere pentru a le stoca temporar.

Procesoare de date care proceseaza informatia venita de la senzori

Procesoare de control ale actuatoarelor

**Sisteme de operare real time**

Aceste sisteme de operare sunt special dezvoltate incat sa poata sustie procesele unui sistem real time. Sunt responsabile cu gestionarea proceselor si alocarea resurselor si pot fi bazate pe un kernel standard care a fost sau nu modificat in scopul deservirii unei aplicatii particulare. In mod uzual nu include facilitate precum gestiune de fisier.

Ca si componente ar fi:

* Real-time Clock
* Interrupt handler pentru interogari aperiodice
* Scheduler – pentru a planifica urmatorul process
* Resource manager – aloca resursele de procesare si memorie
* Dispatcher – initiaza executia proceselor
* Configuration manager – responsabil cu configurarea dinamica a sistemului software si hardware. Componentele hardware ar puta fi inlocuite si cele software upgradate fara a opri sistemul.
* Fault manager – responsabil cu detectarea problemelor software sau hardware si luarea de decizii pentru a le remedia

15.4 RTOSComponents(13.4).eps                                  0007B854Macintosh HD                   B8AA5F2E:

Uneori exista procese ce au nevoie de prioritate asupra celorlate. Prioritatea de nivel Interrupt este cea mai inalta posibila si este data proceselor care au nevoie de un raspuns extreme de rapid. Prioritatea de nivel Clock este alocata proceselor periodice. In aceste categorii exista si subnivele.

* Interrupt – controlul este transferat automat la o portiune predefinita de memorie care contine rutina ce trebuie rulata. Aceasta secvemta trebuie sa fie scurta si rapida in rulare. Orice alta comanda de interrupt este dezactivata in acest timp.
* Periodic process – in sitemele real time exista multiple clase periodice de procese fiecare cu perioade, timpi de execitie si termini de livrare diferiti. Managerul de proces selecteaza instant ce este gata de rulare

Managerul de process se preocupa cu setul curent de procese.

15.5 ProcessStartup(13.5).eps                                  0007B854Macintosh HD                   B8AA5F2E:

Sistemele de monitorizare testeaza si interogheaza constant senzorii si raporteaza rezultatele iar actuatoarele hardware sunt controlate de acestea.

15.6 MonControlArch.eps                                        0007B854Macintosh HD                   B8AA5F2E:

Un exemplu concret implementat intr-un sistem de securitate al unei clariri:

15.8 AlarmProcArch(13.7*).eps                                  0007B854Macintosh HD                   B8AA5F2E:

1. **Testarea**

Testarea softwareului sistemelor critice este un process foarte minutios si constituie de cele mai multe ori un procent mare din dezvoltarea produsului. Poate reprezenta chair si 50% din timpul si costul de productie.

Aceasta etapa este foarte importanta deoarece este ultima ce poate scoate la iveala eventuale problem aparute pe parcurs.

1. **Exemple si concluzii**

Sitemele critice exista in majoritatea domeniilor de activitate dar predomina in industrie si transport. Ele pot varia de la o complexitate redusa la una foarte mare.

Complexitate redusa:

* Sistemul de alarma al unei usi. De cele mai multe ori presupune un senzor si o component acustica.
* Termostatul unui frigider industrial. Un simplu senzor de temperature care controleaza functionarea sistemului de refrigeratie.

Complexitate medie:

* Sitemul de franare al unui autoturism. La autovehiculele modern presupune un computer care controleaza sistemul in situatiile extreme, tip ABS, si senzori multipli cum ar fi cel pentru nivelul lichidului de frana sau temperature ambientala.
* Sistemul de control si supervizare al unei cladiri.

Complexitate ridicata:

* Sistemul de control al unei central nucleare. Presupune gestiunea si controlul a mii de senzori.
* Sistemul de control al unui avion.