DEPENDABILITY
Prima definizione: abilità del sistema di fornire il servizio specificato in presenza di fault
E' la proprietà di un sistema di poter giustificare l'affidabilità di un servizio che fornisce

Ogni sistema è fatto da componenti che sono sistemi a loro volta

Non può essere aggiunta a un design esistente

RUOLI SYSTEM/SW ENGINEER

System engineer: responsabile del design del sistema e usa analisi per modellare la dependability del design-> da qui derivano le specifiche software che portano a cambiare il design se si incontrano dei limiti software

Sw engineer: sw nel sistema che richiede un alto livello di dependability diviene coinvolto in molte funzioni-> difetti software sono diventati cause comuni di fallimenti

DEFINIZIONE SISTEMA

Sistema: un'entità che interagisce con altre (incluso hw,sw, mondo fisico e umano con i suoi fenomeni naturali)

Gli altri sistemi sono l'ambiente per un dato sistema

Il system boundary è la frontiera tra il sistema e il suo ambiente

PROPRIETÀ DEL SISTEMA

Funzione del sistema: specifica funzionale
Comportamento del sistema: sequenza di stati per implementare la funzione
Struttura del sistema: cosa permette al sistema di generare il proprio comportamento, insieme di componenti uniti per interagire, ogni componente è un altro sistema o atomico

Lo stato totale del sistema sono gli stati dei componenti atomici

SISTEMA COME PROVIDER

Servizio fornito dal sistema: comportamento percepito dall'utente
Sistema come provider: il sistema è il fornitore del servizio (l'utente è un altro sistema)

Provider's service interface: Confine del sistema provider
Stato Interno: parte rimanente
Stato Esterno: stato del provider percepito dall'interfaccia
Delivery del servizio: sequenza degli stati esterni

Servizio Utente: interfaccia utente dove viene ricevuto il servizio

DEPENDABILITY TREE

suddivisione in attributi; threats e means

THREATS TO DEPENDABILITY

Fault: causa del problema

Error: il fault causa un errore nello stato del servizio, prima nello stato interno
Failure: l'errore influenza lo stato esterno

Servizio corretto: quando il servizio implementa la funzione del sistema
Service failure: evento che occorre quando il servizio fornito si discosta da quello corretto (dagli stati esterni)

Failure = transizione dallo stato corretto allo stato incorretto
Restoration = transizione allo stato corretto da quello incorretto
Service outage = periodo di fornitura del servizio incorretto

Fault è attivo quando causa un errore altrimenti è dormiente
Una vulnerabilità è un fault interno che permette a un fault esterno di colpire il sistema

Sistema degradato: fallimento di uno o più servizi (fallimento parziale)

ATTRIBUTI DEPENDABILITY

Availability, Reliability, Safety: assenza di conseguenze catastrofiche a utenti e ambiente, Confidentiality, Integrity, Maintainability

MEANS = tecniche per architettare un sistema fault tolerant.

DEFINIZIONE DEPENDABILITY

Dependability: abilità di fornire un servizio che può essere affidabile
La proprietà Dependability è l'abilità di un sistema di evitare service failures che sono più frequenti di quanto accettabile

I requisiti di sistema devono includere la frequenza accettata e la severità dei service failure per ogni classe di fault e un ambiente.

TASSONOMIA DEI FAULT

Ciclo di vita di un sistema: sviluppo -> uso
Fase di sviluppo interazione con ambiente di sviluppo
Fase di uso interazione con ambiente d'uso

Fase d'uso consiste in alternanza di periodi di fornitura del servizio corretto, service outage e spento (shutdown). Service outage avviene con incorrect service, shutdown è interruzione intenzionale. La maintenance può avvenire in tutti e 3 gli stadi

FAULT

tutti i differenti fault che affliggono un sistema durante il suo ciclo di vita non possono essere enumerati, ma classificati

FAULT CLASSIFICAZIONE

Fault di sviluppo: tutti i fault che avvengono durante lo sviluppo
Fault fisico: tutti i fault che affliggono l'hardware
Fault d'interazione: tutti i fault esterni

FAULT NATURALI sono difetti hardware causati da fenomeni naturali senza la partecipazione umana
-dovuti a difetti di produzione durante lo sviluppo
-durante l'operatività
    -interni dovuti a processi naturali come deterioramento
    -esterni dovuti a processi naturali che originano fuori dai confini del sistema e causano interferenza fisica

HUMAN MADE FAULTS
malicious: introdotti con l'obiettivo malevolo di alterare le funzioni di un sistema durante l'uso
    fault logici sia in sviluppo che durante l'operatività
    tentativi d'intrusione che sono considerabili fault esterni durante l'operatività

programmi dipendenti da altri: logic bombs, trojan, trapdoor, virus
programmi indipendenti: worm, zombie
replicanti: virus, worm, zombie

non malicious: sia software che hardware (fault scoperti dopo la produzione di un chip)
    -non deliberati che derivano da errori non voluti
    -deliberati che derivano da decisioni errate, volute. Vengono spesso riconosciuti solo dopo un comportamento inaccettabile del sistema
        -deliberati fault in sviluppo dovuti a tradeoff economici, di prestazione o facilità d'utilizzo
        -deliberati fault d'interazione dovuti all'affronto di una situazione inattesa o violando una procedura operativa senza aver pensato alle conseguenze
oppure -fault accidentali: lo sviluppatore non si era accorto che la conseguenza delle sue azioni avrebbe portato ad un fault
    -fault d'incompetenza: incompetenza per svolgere il lavoro

FAULT d'INTERAZIONE
dovuti a elementi dell'ambiente d'uso che interagiscono con il sistema. occorrono tutti durante la fase d'utilizzo, sono esterni e human made. Hanno bisogno dell'esistenza di una vulnerabilità interna che verrà sfruttata da un fault esterno. L'unica eccezione sono sono fault naturali esterni.
    fault di configurazione del sistema
    reconfiguration fault in caso di cambio dei settaggi precedenti

PERMANENT fault è un fault continuo e stabile. TRANSIENT FAULT è un fault che appare e sparisce in un periodo di tempo breve.

FAILURES
le modalità di fallimento del servizio caratterizzano il servizio incorretto secondo 4 punti di vista:
    -dominio del fallimento
        -content failures: contenuto dell'informazione riportata devia da quella implementata

-timing failures: il tempo di arrivo o la durata dell'informazione fornita devia da quella implementata

-consistenza del fallimento

-fallimenti consistenti: il fallimento viene percepito in modo identico da tutti gli utenti

-fallimenti inconsistenti: alcuni utenti percepiscono differentemente il servizio incorrtto (FALLIMENTI BIZANTINI)

-individuabilità del fallimento riguarda la segnalazione del fallimento del servizio all'utente, e lo stesso meccanismo di segnalazione ha due modalità di fallimento

segnalare la perdita del funzionamento anche se falso (FALSO ALLARME)

non segnalare un malfunzionamento, fallimento non segnalato

-conseguenze del fallimento sull'ambiente, permette di definire le severità del fallimento, due livelli limite:

fallimenti minori quando conseguenze pericolose hanno costi simili al normale funzionamento

fallimenti catastrofici quando il costo delle conseguenze pericolose è addirittura maggiore a quello del servizio funzionante

FALLIMENTI DEL SISTEMA
causati da errori di fault coesistenti (non indipendenti)

FALLIMENTI di SVILUPPO
introdotti dall'ambiente di sviluppo, porta a fermare il processo di sviluppo. Le cause sono specifiche incomplete, cambiate oppure design non adeguato

failure di budget, finiti i soldi

failure di schedule, il sistema è diventato obsoleto

fallimenti di sviluppo parziali non portano alla terminazione e comportano un downgrading delle funzionalità

DEPENDABILITY e SECURITY specification:
gli obiettivi per ogni attributo: Availability, reliability, confidentiality, integrity , mantainability
le classi dei fault attesi
l'environment di utilizzo
le salvaguardie contro condizioni indesiderate
inclusion di tecniche di fault prevention e fault tolerance richieste dall'utente

DEPENDABILITY E SECURITY FAILURES occorrono
quando il sistema soffre failures più frequentemente e severamente di quanto accettabile

SISTEMI FAIL-CONTROLLED: disegnati per fallire solamente in alcune modalità descritte nei requisiti di dependability
SISTEMI FAIL-STOP: gli unici fallimenti sono quelli in cui il sistema si ferma
SISTEMA FAIL-SILENT: come prima ma il sistema non lo comunica
SISTEMI FAIL-SAFE: tutti i fallimenti sono solo minori

ERRORI
un errore è una parte di un sistema che porta al fallimento, viene individuato se viene lanciato un messaggio d'errore, altrimenti è latente. Se un errore porti o no al fallimento del

servizio dipende da: la struttura del sistema e la sua ridondanza, il comportamento del sistema che potrebbe non aver mai bisogno della parte errata o eliminarla. Errori singolo affliggono una sola componente, multiple related errors sono generati da un singolo fault e affliggono più componenti.

CATENA DI THREAT: RELAZIONE TRA FAULT ERRORE E FAILURES
Fault attivo se produce un errore
Attivazione del fault: applicazione di un input a una componente che attiva il fault
Propagazione dell'errore in una componente: causato dal processo di computazione
Propagazione da componente A a B: che riceve servizio da componente A

Il fallimento di una componente causa un permanente o transiente fault nel sistema che la contiene
Il fallimento del servizio di un sistema causa un permanente o transiente fault esterno per altri sistemi che ricevono il servizio da un dato sistema

RIPRODUCIBILITA' dell'ATTIVAZIONE DEL FAULT
identificazione dei pattern che hanno causato uno o più errori
- hard faults ovvero fault che sono riproducibili
- elusive faults ovvero fault che non sono sistematicamente riproducibili (spesso fault di sviluppo residui in software complessi che dipendono da combinazione di condizioni esterne ed interne)

fault fisici transienti e fault di sviluppo elusivi sono considerabili intermittent faults, che sono comunemente chiamati soft errors

DEPENDABILITY MEANS una combinazione dei metodi può essere applicata come mezzo per raggiungere la dependability

MEANS 1: FAULT PREVENTION fa riferimento a tecniche generali di system engineering con lo scopo della prevenzione di fault di sviluppo (metodi formali, sviluppo rigoroso, metodi di controllo qualità) e miglioramento del processo di sviluppo per ridurre il numero di fault introdotti (basato sulle informazioni dei fault nei prodotti e l'eliminazione delle loro cause)

MEANS 2: FAULT TOLERANCE
affrontano i fault a run-time con lo scopo di fornire il servizio corretto in presenza di fault attivati ed errori con:
- error detection e system recovery
- error compensation

ERROR DETECTION and RECOVERY
problema principale:
- identificare tutti i possibili errori con error detection
- assicurare che gli stati d'errore non vengano mai raggiunti o se raggiunti vengano minimizzati gli effetti
- prevenzione della propagazione dell'errore dall affliggere l'operatività delle componenti non fallite

i meccanismi di fault tolerance individano errori (non fault). Fasi della fault tolerance: ERROR DETECTION, ERROR HANDLING, FAULT HANDLING per arrivare all'ERROR RECOVERY

ERROR DETECTION, PROCESSING e FAULT TREATMENT
un errore viene individuato se la sua presenza viene da un messaggio d'errore o segnale, altrimenti è latente. Un esempio classico è la comparazione di due copie che non possono corrompersi insieme

ERROR COMPENSATION (chiamata anche FAULT MASKING)
il sistema contiene abbastanza ridondanza per mascherare gli errori. Ad esempio repliche con majority vote. Per fault hardware sappiamo che i componenti falliscono in maniera indipendente e bastano le repliche, per i software fault le repliche non falliscono in maniera indipendente quindi è utile la design diversity in cui cambiano design e implementazioni

ROLLBACK E ROLLFORWARD error recovery vengono applicati su richiesta dopo la detection.

La system recovery dipende dalla somma di ERROR HANDLING + FAULT HANDLING

ERROR DETECTION

TIPI di CHECK:
- Check di replicazione, con un COMPARATOR
- REASONABLENESS checks, proprietà semantiche dei dati (es. range array)
- RUNTIME checks, divisione per 0, overflow/underflow
- SPECIFICATION checks, comparando risultati corretti
- REVERSAL checks, usando gli output per ricondurci all'input (f.inversa)
- STRUCTURAL checks, ispezione delle strutture dati
- TIMING checks, timer
- CODES, parity code checksum etc.

APPROCCIO STRUTTURALE all error detection, per prevenire l'error propagation:
- privilegio minimo
- system closure fault tolerance principle (azioni non permesse se non autorizzate)
- modularizzazione, detection errori per ogni modulo
- gerarchia e connessione dei componenti, per analizzare la propagazione errori
- partitioning, moduli funzionalmente indipendenti + moduli di controllo
- azioni atomiche, attività in cui le componenti interagiscono con le altre e non ci sono sovrapposizioni con il resto del sistema

EFFICACIA dell'ERROR DETECTION, misurata da:
copertura: probabilità che errore sia detected
latenza: tempo per la detection
confinamento del danno: path della propagazione dell'errore

ERROR RECOVERY

-Recovery forward da stato scorretto a corretto (nuovo stato), richiede di scovare il danno causato dall'errore

-Backward recovery (tornare ad uno stato corretto) con checkpoint, copia dello stato globale del sistema. Abbiamo perdita del tempo di computazione tra il checkpoint e il rollback e dei dati ricevuti e overhead per salvare lo stato del sistema. Questo tipo di recovery non è utile per hardware fault e fault di progettazione

-EXCEPTION HANDLING con catch() che fa da error recovery
exception d'interfaccia -> errore detected, gestito dal modulo che ha fatto la richiesta
internal local exception -> errore detected, gestito internamente
failure exception -> errore detected, non gestito

-FAULT HANDLING: prevenire la nuova attivazione dei fault
diagnosi -> causa dell'errore in termini di locazione e tipo
isolamento -> fisica o logica del componente faulty
riconfigurazione -> switch a componenti ridondanti funzionanti o riassegnamento task
reinizializzazione -> aggiornamento della configurazione e aggiornamento dei record di sistema

DIAGNOSI

può il meccanismo di error detection individuare il componente faulty con precisione?

diagnosi a livello del sistema tramite un grafo di testing, e comunque non sicuri al 100%

gli stessi componenti di testing potrebbero essere danneggiati

ISOLATION
componenti fallite non possono essere lasciate nel sistema, potrebbero danneggiarlo ulteriormente e addizionarsi nel tempo

RECONFIGURATION
dei componenti faulty deve avvenire fuori dal sistema, che sia fisica o logica

REINITIALIZATION
alla fine la loro esclusione andrebbe ad eliminare la ridondanza ma potremmo andare a reinizializzarli.

MEANS 3: FAULT REMOVAL
TECNICHE DI RIMOZIONE dei FAULT: rimozione per evitare che non vengano più attivati

durante la fase di sviluppo del sistema: consistente in tre fasi: 1. fase di verifica, 2. fase di diagnosi, 3. fase di correzione + 3.a non regression verification per verificare che il fault removal non ha conseguenze indesiderate

1.TECNICHE DI VERIFICA

-senza esecuzione del sistema:

-con verifica statica sul sistema stesso (data flow analisi, abstract interpretation, check compilatore, theorem proving)

-con verifica su di un modello del comportamento del sistema (reti di petri, automa a stati)

-eseguendo il sistema:
con input simbolici al sistema: esecuzione simbolica; con dati reali al sistema: testing reale che si divide in hardware testing e sw testing

Testing deterministico con scelta selettiva del pattern o statistico con scelta dei pattern secondo distribuzione di probabilità

-verifica dei meccanismi di fault tolerance
-verifica che il sistema non faccia più del previsto
-progettazione del sistema per permettere verifica

2. rimozione dei fault durante l'uso del sistema:
MAINTENENCE correttiva rimozione di fault che hanno prodotto errori riportati
MAINTENENCE preventiva rimozione di fault prima che possano causare errori

MEANS 4: FAULT FORECASTING (PREVISIONE)
performando una valutazione del comportamento del sistema rispetto all'occorrenzza dell'attivazione dei fault
VALUTAZIONE QUALITATIVA con identificazione e classificazione delle modalità di faillmento o QUANTITATIVA probabilistica, stimando le misure degli attributi della dependability

GIORNO 1

REDUNDANCY

HARDWARE REDUNDANCY:
passive fault tolerance techniques: uso di FAULT MASKING, grazie a VOTATORI, non richiede azioni e non fornisce detection errori
active fault tolerance techniques: detection location e recovery degli errori. una volta individuato elimina l'errore, riconfigurando il sistema (disconnettendo il componente). utilizzato in applicazione dove risultati temporaneamente errati sono accettati.
hybrid approach: fault masking per fornire alta affidabilità, molto dispendioso

1. TRIPLE MODULAR REDUNDANCY (PASSIVE FAULT MASKING)

tollera un modulo FAULTY, triplicando l'hardware e performando un voto di maggioranza.

2. CASCADING TMR with TRIPLICATED VOTERS

L'effetto del partizionamento dei moduli è che la progettazione può resistere a più fallimenti di quello singolo. Il voter diventa un SINGLE POINT OF FAILURE

3. TMR: il voter

il voto di maggioranza viene calcolato BIT A BIT
VOTO 1 BIT con 3 INPUT BIT = AB+BC+AC
DIFFICOLTA: delay nella propagazione del segnale, tradeoff della raggiunta fault tolerance rispetto al hw richiesto

VOTARE CON SEGNALI ANALOGICI: i segnali convertiti da ADC potrebbero differire per il LSB anche se i segnali sono passati correttamente, e votare nel mondo analogico significa farne la media; o scegliere la media dei due simili; o scegliere la mediana dei 3 segnali

N-MODULAR REDUNDANCY
estensione del concetto TMR a N moduli con un voter, con N numero DISPARI, copertura di m MODULI FAULT con N=2m+1, ex. 5MR ne tollera 2

RIDONDANZA ATTIVA

1. DUPLICAZIONE CON SCHEMA COMPARATIVO (solo ERROR DETECTION)

L'HW viene duplicato, stessa computazione parallela. Una semplice comparazione individua l'errore, l'assunzione è che le due copie non siano faulty allo stesso tempo.Necessità che il comparatore non sia a sua volta faulty.

2. DUPLICAZIONE RICONFIGURABILE (ERROR DETECTION, FAULT IDENTIFICATION, disconnette quello FAULTY e disabilita il comparatore)

di nuovo duplicazione, il comparatore selezione l'output corretto (DUPLEX SYSTEM).
i check per selezionare l'output corretto sono CODING, SELF-CHECKING COMPONENTS, SPECIFICATION CHECKS. Abbiamo gli stessi problemi sul comparatore.

3. STAND-BY SPARING (ERROR DETECTION, FAULT IDENTIFICATION, RECONFIGURATION)

Parte dei moduli sono operativi altri sono solo redundancy, lo switch decide se utilizzare o meno il valore di un modulo, quello faulty viene rimpiazzato. HOTSPARES(ricambio): quelli di ricambio sono sincroni con il resto del sistema, prendono subito il posto dei faulty. WARM: quelli di ricambio sono attivi ma ricevono input solo dopo switch avvenuto. COLD: quelli di ricambio sono spenti fin quando non devono riprendere il posto. Finché l'output è identico, quelli di ricambio non vengono utilizzati.
PAIR AND SPARE APPROACH: un modulo è in DUPLEX, coppie connesse da comparatore e abbiamo lo SPARE module. Se il comparatore capisce di non calcolare il valore corretto avvisa lo switch che cambia sul modulo duplex SPARE.

RIDONDANZA IBRIDA, combina i due approcci
lo svantaggio sta nel fatto che l'abilità di FAULT MASKING si deteriora man mano che le copie falliscono.

RECONFIGURABILE NMR: i moduli sono progettati in una configurazione voting.

N MODULI RIDONDANTI ATTIVI, il VOTER controlla l'output di quelli ACTIVE. LA FAULT DETECTION UNIT(s) compara l'output del voter con quello dei moduli attivi, rimpiazzando quelli che non combaciano con quelli di ricambio. L'affidabilità è garantita fin quando il POOL di RICAMBIO non FINISCE. COPERTURA: un TMR con un ricambio può tollerare due moduli fallimentari

4. INFORMATION REDUNDANCY

CODING: applicazione di ridondanza nell'informazione
se n+c = m bit, avremo 2 alla m combinazioni ma solo 2 alla n valide che sono codeword
PARITY BIT/ODD per singolo bit flip

PARITY CODE è un SEPARABLE CODE, semplice rimozione

EXAMPLES OF CODE
cd- complemented duplication la seconda parte è il complemento della prima
m/n code

HAMMING DISTANCE
numero di posizione per cui due codici differiscono
MIN HAMMING DISTANCE = numero di errori single bit indipendenti che il codice può individuare
Un codice tale che l'HAMMING DISTANCE tra codici è > k può individuare k errori

CODES FOR ERROR CORRECTION
jn codice con al minimum hamming distance di k
corregge fino a d errori dove k=2d+1
individua fino a k-1 errori

CHECKSUMMING
applicato a larghi blocchi di dati in memoria
addizione di tutti i blocchi in uno singolo modulo k, salvato col blocco dati
quando vengono spostati dati, viene ricalcolato checksum per verificare bontà

permette error detection ma non individua error location

CODICI ARITMETICI
se l'input sono codeword e l'operazione viene eseguita correttamente allora anche l'output risultante sarà una codeword
se abbiamo A(b*c) = A(b)* A(c) con * operazione da effettuare

3N CODES, dato ripetuto 3 volte, 2 bit di ridondanza, error checking è performato confermando che la parola ricevuta sia divisibile per 3.

ECC ERROR CORRECTING CODES

parità a 2 dimensioni:
un errore nel vettore row parity, un errore nel vettore column parity
viene individuato un errore nei due vettori
SEC single error correcting code: individua e corregge errori di 1 bit

HAMMING CODE
bit di parità sparsi lungo tutta la data word -> nell'esempio tutte le posizioni potenza di due: 1,2,4,8 etc, i dati in tutte le altre posizioni
il bit di parità pj copre tutti i bit la cui posizione ha il bit j-esimo meno significativo a 1
ogni bit dato è incluso in un set unico di 2 o più bit parità, determinato dalla forma binaria della sua posizione
ESEMPIO: in pratica il bit di parità 1 copre i bit che tradotti in binario hanno il bit meno significativo settato tipo 1,3,5,7
il bit di parità 2 copre i bit che tradotti in binario hanno il 2secondo bit meno significativo settato 2,3,6,7,10

OVERLAP di un bit di controllo: un bit dato è controllato da più di un bit di parità

RAID: informazione ridondante salvata su dischi multipli per recuperare i fallimenti
MIRRORING, semplice copia RAID 1

CODING, PARITA' BLOCK INTERLEAVED RAID 3
0-1-2-3 PARITA' 0-3
4-5-6-7 PARITA' 4-7

CODING, PARITA' BLOCK INTERLEAVED distribuita RAID 5
PARITA' 0-3 | 0-1-2-3
4 | PARITA' 4-7 | 5-6-7

HAMMING CODE RAID 2
Un sistema RAID 2 divide i dati al livello di bit (invece che di blocco) e usa un codice di Hamming per la correzione d'errore che permette di correggere errori su singoli bit e di rilevare errori doppi

SELF CHECKING CIRCUITRY
necessità di affidabilità sulla correttezza delle operazioni dei comparatori e CODE CHECKERS che vengono utilizzati come funzioni HARDCORE per fault tolerant systems

SELF CHECKING CIRCUIT: dato un set di fault, un circuito ha l'abilità di automaticamente individuare l'esistenza dello stesso durante il normale corso delle operazioni.

IDEA: fault free + input = correct code word; fault + code input: correct code word o code word incorretta.

SELF TESTING CIRCUIT: se per ogni fault del set, il circuito produce un NONCODE OUTPUT per almeno un input.

FAULT SECURE CIRCUIT: se per ogni fault del set, il circuito non produce MAI un INCORRECT CODE OUTPUT per un input (cioè un OUTPUT CORRECT CODE o NONCODE OUTPUT)

TSC TOTALLY SELF CHECKING: se è SELF-TESTING e FAULT-SECURE

TWO INPUT TSC COMPARATOR (0 se bit uguali, 1 altrimenti):
assunzione per il fault-> deve essere singolo
segnali dual rail: segnali codificati per cui i bit vengono invertiti con 0 abbiamo 01 con 1 abbiamo 10

CASO A=B abbiamo c2=1; c1=0 che sono la combinazione che da 0 per c ovvero che i due input sono uguali

CASO A!=B come prima, abbiamo c2=0; c1=1 quindi input diversi

CASO FAULTY:

se abbiamo una linea stuck a 0, avremo in output un errore nel caso A!=B, con stuck at 1 però corretto la CODEWORD

DUNQUE come detto abbiamo almeno un input che ci produce una configurazione scorretta (NON CODE WORD) mentre se l'output è una CODEWORD è giusto

TIME REDUNDANCY TECHNIQUES (I)
Ridurre la spesa di hardware al costo di tempo addizionale

-Ripetizione delle computazioni:
Comparazione di output per trovare fault, rieseguendo anche per capire se disaccordo sparisce o rimane (utile per transient fault non per permanenti).

Uso di un minimo di extra hw per individuare i fault permanenti (codifica i dati prima di eseguire la seconda computazione): al tempo t0 + d inviare il complemento dei messaggi per individuare errori di trasmissione

SOFTWARE REDUNDANCY
il software è soggetto a:
ERRORI DI PROGETTAZIONE (difficili da visualizzare, classificare e correggere)
dovuti a mal interpretazione delle specifiche
dovuti a mal implementazione delle specifiche

L'affidabilità apparente di un pezzo software è correlata a quanto frequentemente errori di progettazione vengono eseguiti rispetto a quelli presenti in totale

FAULT OPERATIVI:
dovuti a usi inattesi

SOFTWARE DEPENDABILITY = tecniche di prevenzione dei fault e strategie di testing

APPROCCIO MULTIVERSIONE = replicazione del software
semplice duplicazione porterebbe a non risolvere il problema, generazione indipendente di n>=2 programmi funzionalmente equivalenti chiamate VERSIONE dalle stesse specifiche
svantaggi: costi nello sviluppo ed esecuzioni concorrenti, non tollera errori di specifica
SOFTWARE VOTER fa da SPOF single point of failure, non replicato deve essere semplice e verificabile, deve assicurarsi che gli vettore dati in input sia identico e deve comparare formati output identici oltre che seguire un protocollo di comunicazione intelligente per attendere l'esecuzione di tutte le versioni e riconoscere se una è in stallo

NSELF CHECKING PROGRAMMING: (basati su test d'accettazione invece di comparare versioni equivalenti).
Girano versioni con i loro test d'accettazione, la SELECTION LOGIC sceglie i risultati da uno dei programmi che passa i test d'accettazione, tollera N-1 FAULT.

DESIGN DIVERSITY:
non si può adottare l'analogia dell'hardware e assumere il fallimento indipendente
evidenza empirica che che ci saranno fault comuni

fault correlati potrebbero risultare da dipendenze nella progettazione separata e implementazioni

diversità funzionale: assegna a versioni indipendenti software funzioni diversificate che calcolano lo stesso task (sensori, attuatori e funzioni che misurano lo stesso fenomeno)

funzioni diverse: funzioni che assicurano indipendentemente che la sicurezza di un sito target sia rispettata

RECOVERY-BLOCK TECHNIQUE: (basato su un test d'accettazione e viene eseguito solamente un'alternativa alla volta)

struttura: ENSURE T; BY P; ELSE BY Q; ELSE ERROR

accettabilità di un risultato viene decisa da un test d'accettazione T, alternativa primaria P e secondaria Q.
1 variabili globali al blocco vengono automaticamente checkpointed se vengono alterate dallo stesso
2 l'alternativa primaria viene eseguita e soggetta al test d'accettazione per trovare errori nel risultato, se il test viene passato si esce dal blocco altrimenti il contenuto viene recuperato dalla cache e eseguito con la seconda alternativa
3 il ciclo viene eseguito finché un alternativa non ha successo o non esistono più alternative, in questo caso viene riportato un errore

combina elementi di checkpointing e backup:
CHECKPOINT = copia dello stato corrente per uso possibile in tecniche fault tolerant, andando a togliere il compito al programmatore quali variabili dovrebbero essere salvate e quando. Strutture linguistiche per blocchi di recupero hanno bisogno di meccanismi utili a fornire recupero retroattivo dell'errore automatico.

DEPENDABILITY EVALUATION

VALUTAZIONE DELLA DEPENDABILITY
I fault (guasti) sono la causa di errori e fallimenti. Il tempo di arrivo degli stessi segue una distribuzione di probabilità? Quali sono i suoi paramteri?

PARAMETRI di VALUTAZIONE

RELIABILITY R(t): in funzione del tempo, è la probabilità condizionale che il sistema lavori correttamente nell intervallo t0,t dato che il sistema stava lavorando correttamente al tempo t0 (R(0) = 1; R(INF)=0). UNRELIABILITY Q(t) = 1 - R(t)

AVAILABILITY A(t): in funzione del tempo, è la probabilità che il sistema operi correttamente e sia disponibile per eseguire le sue azioni all'istante t

DEFINITIONS

PROBABILITA' DI GUASTO, funzione di densità f(t) = dQ(t)/dt = - dR(t)/dt

funzione RATE GUASTI LAMBDA(t) (BATH-TUB SHAPED CURVE)= f(t)/R(t) = -dR(t)/dt - 1/R(t), sempre > 0 nel periodo utile

La relazione esponenziale tra reliability e tempo è conosciuta come exponential failure law

FUNZIONE di RELIABILITY R(t) = e ^ alla -LABMDAt

quindi la funzione f(t) diventa LAMBDAe^-LAMBDAt

Il TIME TO FAILURE di una COMPONENTE può essere modellato da una variabile aleatoria X: UNRELIABILITY = P[X<=t] ; RELIABILITY = 1 - P[X <= t]

MTTF TEMPO MEDIO AL FAILURE: 1/LAMBDA (integrale svolto nelle slide)

FAILURE IN TIME FIT failure rate espressi in un miliardo di ore

CHIP FAILURE RATE = LAMBDA (vedi formula sulle slide)

MODEL BASED EVALUATION OF DEPENDABILITY (un modello è un'astrazione del sistema che sottolinea feature importanti per l'obiettivo dello studio)

1. MODELLI COMBINATORI
2. STATE SPACE REPRESENTATION METHODOLOGIES

MODELLI COMBINATORI
assunzioni: componenti indipendenti, ad ogni componente viene associato un failure rate
la costruzione del modello si basa sulla struttura dei sistemi (connessioni in serie/parallelo)
questo modello non è adeguato per valutare dipendenze complesse.
serie: tutti i componenti devono funzionare = produttoria delle R(t) dei singoli, poi se ogni componente soddisfa la legge dell'exponential failure law con constant failure LAMBDA = e alla -(SOMMATORIA dei LAMBDA)
parallelo: almeno un componente deve funzionare = duale di prima
sistemi M of N = almeno M dei N devono funzionare = SOMMATORIA di (N i, coefficente binomiale) * R alla N-i (t)* (1-R(t))alla i con i numero di componenti fallati

per modelli con ridondanze la system reliability è la reliability di modelli combinatori in serie e parallelo (esempio due processori in parallelo e 3 memorie in parallelo collegati in serie)

TMR: reliability function and mission time
TMR è peggio di un simplex system (vedi slide per conti su tempo medio MTTF) ma TMR ha affidabilità più alta per le prime 6000 ore e lavora ad affidabilità 0.8 per più tempo del simplex. (curva a gomito/ginocchio dovuta a fallimento della ridondanza nel grafico)

HYBRID REDUNDANCY WITH TMR = il primo fallimento occorre se tutti i moduli falliscono o se tutti tranne uno falliscono, aumentare i moduli nell'ibrido aumenta l'affidabilità = (1 - ((1-Rm)alla n + n(Rm)(1-Rm)alla n-1)) = Rsdv(1-Qhybrid)

MODALITA' NON SERIE/PARALLELA

il sistema è completamente operativo per ogni path da X a Y
L'affidabilità viene calcolata espandendo attorno a un modulo m:
Rsys = Rm * P(sistema funzioni| m funziona) + (1-Rm) * P(sistema funzioni | m fallisca)

P(sistema lavora | B fallisce) = { Rd [1 - (1- RaRe) (1 - RfRc)]}

LIMITE SUPERIORE: Rsys <= 1 - PRODUTTORIA di (1- Rpathi), e viene definito upper bound perchè un componente che fallisce influenza più di un singolo path

Un MINIMAL CUT SET è una lista di sottoinsiemi di componenti tale per cui la rimozione dei componenti del sottinsieme causa il malfunzionamento del sistema

LIMITE INFERIORE: Rsys <= PRODUTTORIA di (1- Qcut) = PRODUTTORIA di Rcuti, dove Qcuti è la probabilità che quel taglio non avvenga+

ANALISI FAULT TREE: è una failure analysis in cui uno stato indesiderato del sistema viene analizzato tramite logica booleana per combinare una serie di eventi a basso livello. Descrive gli scenari di occorrenza d'eventi a livello astratto. Il metodo serve per studiare come il sistema fallisce

TOP EVENT: stato del sistema (G0), basato sullo stato degli eventi base. Gli eventi base sono i componenti del sistema, che a loro volta sono le foglie dell'albero (VERO se faulty, altrimenti FALSO, i nodi sono AND,OR,k-di-N porte), il sistema fallisce se la root è true

CUT SET: per stimare la probabilità dell'evento alla radice, calcola la probabilità d'occorrenza dei suoi tagli e combina queste probabilità
Q_s(t) = probabilità che tutti i componenti del minimal cut siano faulty. La soluzione numerica del fault tree è calcolata con la probabilità d'occorrenza per ogni minimal cut in una sommatoria di Q_si(t)

CONDITIONAL FAULT TREE
uso quando l'indipendenza viene vanificata ovvero la ripetizione di un componente in un fault tree. Q_s(t) = Q_s | c fails(t) Q_c(t) + Q_s | c not fails(t) Q_c(t)

FAILURE MODEL AND EFFECT ANALYSIS
FMEA Failure Mode Effect Analysis è una failure analysis per identificare il rischio di failure di un sistema/componente, considerando una singola feature. Calcolo del RPN, RISK PRIORITY NUMBER nei termini di gravità del failure (S), occorrenza (O) e detectability (D).

Quindi per ogni FAILURE MODE: enumera i potenziali effetti, valuta la S x ogni effetto (da 1 a 10 maggiore è peggio), enumera tutte le possibili cause, valuta l'occorrenza O x ogni causa (da 1 a 10, maggiore è peggio), enumera tutti i processi per controllarla, valuta il detection rating D (da 1 a 10, maggiore è peggio) -> da cui costruzione della tabella FMA con punteggi RPN.

Spesso FMEA viene utilizzato insieme a fault tree quando il fallimento del sistema dipende da più di una singola feature

STATE-BASED MODELS
Reliability dipende dalla frequenza dei fault e la durata dei fault nel sistema. Nei modelli in serie/paralleli la durata non viene considerata, i state-based models numera gli stati del sistema può essere usata per reliability and availability measures: ogni stato rappresenta una combinazione distinta di moduli falliti e funzionanti, il sistema va da stato a stato a causa dei comportamenti dei moduli, le transizioni sono caratterizzate dalla probabilità di fallimento e quella di riparazione, il tempo tra un faut e una riprazione è la durata del fault all'interno del sistema.

Proprietà di MARKOV = lo stato corrente è abbastanza per determinare quello futuro

STEADY-STATE TRANSITION PROBABILITY: la probabilità di transizione sono steady-state se per ogni coppia di stati i e j la probabilità di transizione non dipende dal tempo (in ogni istante la transizione ha la stessa probabilità).

CATENA di MARKOV OMOGENEA = steady-state. Una catena di MARKOV a stati finiti è rappresentabile da una matrice, noi consideriamo quelle OMOGENEE DISCRETE TIME MARKOV CHAINS

TRANSITION PROBABILITY MATRIX
n = numero di stati della catena ; pij = probabilità di muoversi dallo stato i allo stato j
La sommatoria delle pij per ogni i deve essere =1 (MATRICE STOCASTICA)

TEOREMA
per ogni coppia di stati i e j e con n>0, le probabilità rimangono uguali per goni t >= 0

DTMC probabilità sta sempre tra 0 e 1 e la sommatoria delle probabilità delle colonne è data dall'elemento della colonna i esimo

TEOREMA CHAPMAN-KOLMOGOROV
La probabilità ij al tempo n+m è data dalla sommatoria delle probabilità al tempo n e m.

Probabilità che quando il sistema entra nello stato i, vi ci rimanga per n step = $p_{ii}(n-1)(1-p_{ii})$

Un processo di markov si può specificare in termini dello state occupancy vector PHI e la transition Probability matrix

Ogni volta viene ricalcolato lo state occupancy vector con la moltiplicazione delle due matrici dello stato precedente

DTMC: CLASSIFICAZIONE DEGLI STATI
diciamo che uno stato sia accessibile se esiste un t>0 per cui la probabilità di accedervi dallo stato i sia > 0. Una markov chain si dice IRRIDUCIBILE se per tutti gli stati i,j: lo stato i è accessibile da j in un numero finito di step; ogni stato j è accessibile da i in un numero finito di step, ovvero si può andare avanti e indietro per ogni i e j

STATO I RICORRENTE: per ogni j, il fatto di poterci arrivare da i implica anche il contrario.

STATO I TRANSIENTE: esiste un j diverso da i tale che da i si vada a j ma non il contrario

STATO I ASSORBENTE: la probabilità di rimanere in uno stato = 1. questo è anche uno stato ricorrente

dato uno stato ricorrente, sia d il MCD di tutti gli interi m tale che lo la probabilità di Pii alla m > 0 . Uno stato ricorrente è periodico se d > 1. Uno stato ricorrente è aperiodico se d=1.

DTMC: STEADY-STATE BEHAVIOUR

TEOREMA
Per catene di markov IRRIDUCIBILI APERODICHE, per ogni j, per il limite di t -> INF di PHI jesimo, esiste una soluzione ed è indipendente dal PHI INIZIALE.

Lo steady state comportamento della catena di markov è dato dal punto fisso dell'equazione: PHI(t): PHI(t-1)P con PHI = PHI * P PUNTOFISSO

TIME AVERAGE STATE SPACE DISTRIBUTION

Calcolo della time-average state space distribution, chiamata PHI*

Catene di markov IRRIDUCIBILI con stati PERIODICI: lo stato periodico oscilla periodicamente. Il comportamento limitante non esiste:
PRODUTTORIA*(PHI*) = LIM di t->INF (SOMMATORIA dei PHI^i con i che va da 1 a t)/t

DA DISCRETE-TIME a CONTINUOUS-TIME MARKOV CHAIN (CTMC):
X_t con t in T={t1,t2,...} T intervallo di numeri reali. Consideriamo steady-state transition probability (catena omogenea) abbiamo che per ogni t1,t2,... con tutti i t > 0
P{X_t+TAU=j | X_t=i} = P{X_tau=j | X_0=i}, la catena ha una matrice di transizione di probabilità.

CTMC
T_i è il tempo speso in uno stato, per le assunzioni di Markov, T_i è una variabile aleatoria continua con distribuzione esponenziale, e il valore di questa variabile caratterizza il comportamento della catena. -> Per questo il modello di Markov si conforma con le assunzioni che il rate dei failure è costante, portando a una distribuzione esponenziale dell'interrarivvals dei failures.

Per ogni stato i, T_i = e ^ -ai. Se ai = 0, lo stato è assorbente, se ai = INF, lo stato è ISTANTANEO, se 0 < ai < INF lo stato è stabile.
Una CTMC può essere specificata nei termini del suo OCCUPANCY PROBABILITY VECTOR PHI e la sua STATE TRANSITION RATE MATRIX Q

STATE TRANSITION MATRIX Q, gli stati qij = { rate di andare dallo stato i a j se i != j, -SOMMATORIA dei k!=i * qik ALTRIMENTI). La somma di ogni riga deve fare 0.

SIMPLEX SYSTEM WITH REPAIR: DISPONIBILITA', RELIABILITY

L’ AVAILABILITY (DISPONIBILITA’) si misura dalla PHI dello stato 0 (nell’esempio) al tempo t.

LA RELIABILITY R(t) = PHI_0(t) = e ^ -LAMBDAt (NELL’ESEMPIO SENZA REPAIR)
UNRELIABILITY Q(t) = PHI_1(t) = 1 - e ^ LAMBDAt (NELL’ESEMPIO SENZA REPAIR)

SISTEMA TMR CON REPAIR
identificazione degli stati con 0 processori falliti, 1 e 2 su 3 totali

DUAL PROCESSOR SYSTEM con REPAIR: modello AVAILABILITY
A(t) = 1 - PHI_2 (t)
Reliability model con trapping state, R(t) = 1 - PHI_2 (t) = PHI_1 (t) + PHI_0 (t)
modello RELIABILITY (vedi esempi slide)

ESEMPIO CTMC per 2 processori e 3 memorie (strategia di repair = solo uno alla volta, repair rate. Possibile estensione con priorità di repair)

SAFETY- evitare conseguenze catastrofiche
come funzione del tempo, S(t) è la probabilità che il sistema o si comporti correttamente o discontinuerà le sue funzioni in maniera da non causare danni

COPERTURA, COVERAGE. La misura c dell’abilità del sistema di raggiungere un fail-safe state dopo un guasto fault.

ESEMPIO SULLE SLIDE

MTBF - Mean Time Between Failures è il tempo medio tra i failures del sistema, includendo il tempo richiesto per effettuare il repair MTBF = MTTFailure + MTTRepair

STEADY STATE AVAILABILITY (INF in un simplex system) = MU / LAMBDA + MU

RIPETIZIONE GIORNO 1

- SAN (STOCHASTIC ACTIVITY NETWORK)

RETI DI PETRI (PLACES, TRANSITION;ARCS)

ARCHI = da PLACE a TRANSIZIIONE. PLACE contiene un determinato numero di token
PLACES con ARCO a TRANSIZIONE chiamati INPUT PLACE della transizione (PRESET)
PLACES con ARCO dalla TRANSIZIONE chiamati OUTPUT PLACE della transizione (POSTSET)

t è abilitata se PER OGNI PLACE APPARTENENTE ALLA TRANSIZIONE PRESET, M(p) ovvero il MARKING >= PESO DELLA TRANSIZIONE. IL FIRING è ATOMICO (singolo non interrompibile).

FIRING RULE vedi dalle SLIDE

TRANSITION SEQUENCE = se M [t1,...,t_n > M ^ n allora t1,...,tn è una transition sequence
ABBIAMO ANCHE LA PROPRIETA' TRANSITIVA

EVOLUZIONE DELLA RETE
l'esecuzione è non deterministica, l'ordine non è determinato del firing.

RISOLUZIONE DEI CONFLITTI
Se il FIRING coinvolge due transizioni con gli stessi token non possono essere abilitate entrambe le transizioni.

NON SEQUENTIAL PROCESS dove il grado di PARALLELISMO è definito dal massimo numero di transizioni abilitate in un taglio del processo.

ESEMPIO PRODUCER/CONSUMER (2 SLOT BUFFER), vedi slide

GRAFO DI RAGGIUNGIBILITA'
rappresenta i reachable marking, può essere infinito se non ci sono limiti sul numero di token sui place

CONDIZIONI su MARKINGS RAGGIUNGIBILI
usate per controllare se avvengono alcuni stati BAD (violazione della mutua esclusione)

TRANSITION INVARIANTS
una sequenza di transizione che porta indietro al marking da cui partiva (POSSIBILE LOOP)

PLACE INVARIANTS
proprietà globali di marking sempre soddisfatte: il numero di token in tutti i marking raggiungibili soddisfano qualche invariante lineare

DEADLOCK
un marking viene chiamato deadlock se nessuna transizione è abilitata nel marking

TIMED TRANSITION
un'attività che ha bisogno di tempo per essere eseguita, assegnando un delay d a ogni transizione, assegnando un tempo globale alla PETRI NET PN. Quando una transizione è abilitata un timer locale è impostato a d, il timer decresce e quando scade la transizione fa il fire. Se la transizione è disabilitata prima del firing, il timer si ferma e si ha la possibilità di CONTINUE e riprendere o RESTART il timer.

RETE STOCASTICA DI PETRI
quando il delay d di una timed transition è una variabile aleatoria

RETI STOCASTICHE DI PETRI
le abbiamo con il delay che è una variabile aleatoria con distribuzione esponenziale e il reachability graph è una MARKOV CHAIN

RATE TRANSIZIONE = LAMBDA_k della transizione T_k

ASSUNZIONE: delay delle transizioni è indipendente

la risoluzione dei conflitti dipende dal delay delle transizioni coinvolti

STOCHASTIC ACTIVITY NETWORK
variante delle reti stocastiche di petri
Un sistema viene descritto via SAN attraverso 4 insiemi disgiunti di nodi:
PLACES
INPUT GATE
OUTPUT GATE
ATTIVITA' (immediate linee sottili, timed spesse)

CASES vengono utilizzati per modellare comportamenti probabilistici (sulle activity), disegnati come cerchietti sulle activity

Ogni input o output gate è connesso ad una sola attività

CASES e la loro distribuzione: attività istantanee o a tempo potrebbero avere risultati mutamente esclusivi chiamati casi, scelti probabilisticamente in base alla distribuzione associata all'attività. OUTPUT GATES permettono la definizione di marking differenti per le casistiche dell'attività

ALTRE ESTENSIONI: MODULI (sistema composto da più moduli); SHARED VARIABLES (oggetti globali usati per scambiare informazioni tra moduli); EXTENDED PLACES (places il cui marking è una struttura dati invece di interi non negativi)

RATE REWARD in mobius in caso di REWARD agli STATI, IMPULSE rewards in caso di rewards delle transizioni.
Tipo se lo stato che rappresenta il fallimento del sistema (A), media dei valori di ritorno della reward funciton -> if (A->Mark()==0) allora 1 altrimenti 0.

NUMERICAL SOLVER
il modello è il reachability graph. Solo per modelli deterministici e con attività esponenzialmente distribuite. Stati finiti, piccola descrizione state-space

SIMULATION
il modello è un discrete event simulator. Per tutti i modelli, con una descrizione arbitrariamente larga dello state space. Fornisce soluzioni statisticamente accurate in un intervallo di confidenza

- FAULT TOLERANT DISTRIBUTED SYSTEM- BUILDING BLOCKS

ATOMIC ACTIONS: un azione eseguita totalmente o non ha alcun effetto
in sistemi distribuiti: viene eseguito su piu di un nodo, i nodi devono collaborare affinchè sia l'esecuzione delle azioni venga completato correttamente o non si abbia alcun effetto.

Il progettista può associare meccanismi di fault tolerance con le sottostanti azioni atomiche:

limitando la conseguenza della propagazione degli errori quando avvengono fault e localizzare l'error recovery

TRANSAZIONI NEI DATABASE: una sequenza di modifiche ai dati che si muovono da uno stato consistente ad un altro consistente (deve essere una transazione atomica)

COMMIT = transazione terminata con successo

ABORT/ROLLBACK = abort della transazione, nessun operazione eseguita

CONDIZIONI: un failure durante l'esecuzione della transazione risulta in un rollback/abort, un failure successivo all'esecuzione della transazione non ha conseguenze.

TWO PHASE COMMIT PROTOCOL
Un transaction manager TM ; + resource manager RM; logfile ; timeout

tollera: perdita di messaggi, crash dei nodi
periodo di incertezza: se crasha il transaction manager, un partecipante con READY nel suo log non può terminare la transazione

THREE PHASE COMMIT
aggiunta di una fase di precommit. Assumi un crash permanente del coordinatore, un partecipante può sostituire il coordinatore per terminare la transazione. Il partecipante assume il ruolo di coordinatore e decide se fare GLOBAL ABORT se l'ultimo record nel log è READY; GLOBAL COMMIT se l'ultimo record nei log è PRECOMMIT.

RECOVERY and ATOMICITA'
blocco fisico: blocco sul disco
blocco buffer: blocco residente temporaneamente in memoria principale.

I movimenti dei blocchi tra disco e memoria principale avviene tramite le seguenti operazioni:
-input(B) trasferisce il blocco fisico alla memoria principale; -output(B) trasferisce il blocco buffer al disco.

Ogni transazione T_i ha la sua area privata di lavoro in cui le copie locali di tutti i dati sono acceduti e aggiornati, esegue read(X) accedendo X per la prima volta; esegue write(X) dopo l'ultimo accesso a X. Il sistema può eseguire l'operazione di output successivamente.

Per assicurare atomicità nonostante i failures, il primo output è l'informazione che descrive le modifiche in un LOGFILE in uno storage stabile senza modificare il database stesso (LOGBASE RECOVERY)

CHECKPOINTING
output di tutti i blocchi nel buffer di transazioni committate sul disco CK(T_i,...,T_k)
Per recuperare da un system failure:

consultare il log; rifare tutte le transazioni nel checkpoint o partite dopo il checkpoint che sono state committate; eliminare le transazioni nel checkpoint o partite dopo il checkpoint che non sono state committate.

Algoritmo: il log a ritroso fino al checkpoint
T commit -> aggiunti T a RIFAI/REDO
T start, se T non è in REDO, aggiungi ad UNDO
Per ogni T attiva al checkpoint, se T non è in REDO, aggiungi ad UNDO

LOGFILE in uno storage stabile, in più prima del commit di una transazione, salva il blocco del logfile contenente i record delle transazioni sul disco; prima di aggiornare il database, salva il blocco del logfile contenente il record aggiornato sul disco.
E' sempre possibile eseguire redo/undo di una transazione se necessario.

PROBLEMA DEL CONSENSO (BYZANTINE FAULT TOLERANCE IN DISTRIBUTED SYSTEMS)
Il problema del consenso informalmente: come rendere un insieme di processori distribuiti raggiungere un consenso su un valore inviato da un processore nonostante un numero di failure.

Assumi:
n numero dei generali
v(i), l'opzione del generale i-esimo (attacco/ritirata)
ogni generale i comunica il valore di v(i) tramite messaggeri agli altri generali
ogni generale fa una decisione ottenuta da: voto di maggioranza tra valori v(1),...,v(n)

Situazione semplificata: 1 COMANDANTE, n-1 TENENTI.
Come si ottiene il consenso se il comandante manda ad alcuni un comando e ad altri uno opposto?

3 GENERALI: un TENENTE traditore -> NON ESISTE SOLUZIONE
ci fa capire che dobbiamo aggiungere la condizione che alla ricezione di messaggi bisogna dare precedenza a quelli ricevuti dal comandante per rispettare INTERACTIVE CONSINSTENCY

3 GENERALI: un COMANDANTE traditore -> NON ESISTE SOLUZIONE
per affrontare un traditore abbiamo bisogno di almeno 4 generali
BYZANTINE GENERAL PROBLEM
generali tenenti mandando messaggi avanti e dietro tra di loro riportando il comando ricevuto dal generale comandante

ORAL MESSAGE algorithm
assunzioni:
-il sistema è sincrono
-ogni due processi hanno comunicazione diretta tra la rete non prono a fallire se stessa e soggetta ad un delay trascurabile
-possiamo identificare chi è il mittente del messaggio
quindi:

ogni messaggio che è mandato da un processo non faulty è spedito correttamente
il ricevitore di un messaggio sa chi l'ha mandato
l'assenza di un messaggio può essere individuata

l'algoritmo risolve il problema per n = (3m+1) generali in presenza di m traditori.
voto di maggioranza deterministico, in caso di mancanza di messaggio viene assunto che sia retreat, e la funzione majority() ritorno retreat nel caso non esista la maggioranza di valori

ALGORITMO no m
1.C manda il suo valori ad ogni L
2.Ogni L usa il valore ricevuto o il valore retreat se non riceve valori

ALGORITMO m>0
1.C manda il suo valori ad ogni L
2.Sia vi il valore ricevuto da L dal C. L funge da C in OM(m-1) per mandare vi ad ognuno degli altri n-2 generali tenenti
3.Per ogni i != j sia vj il valore ricevuto da L dal tenente precedente e Li usa il voto di maggioranza

4 GENERALI: COMANDANTE traditore, condizioni soddisfatte per lo stesso comando inviato; condizioni soddisfatte per retrait inviato solo a un generale tenente
4 GENERALI: un TENENTE traditore, nel caso in cui invii due messaggi differenti agli altri due tenenti, condizioni soddisfatte;

ORAL MESSAGE algorithm
teorema provato formalmente: per ogni m, l'algoritmo OM(m) soddisfa le condizioni IC1, IC2 se ci sono più di 3m generali e al massimo m traditori. Sia n il numero di generali n>=3m+1

BYZANTINE GENERAL problem
il problema originale dei generali bizantini viene risolto assegnando il ruolo di comandante ad ogni tenente e facendo girare l'algoritmo concorrentemente

Le soluzioni al problema del consenso sono costose. OM(m): ogni L aspetta il messaggio originato da C e trasmesso da m altri L. OM(m) richiede n =3m+1 nodi; m+1 turni; messaggi della grandezza di O(n^m+1) [la grandezza dei messaggi aumenta ad ogni round].

Una soluzione migliore la possiamo avere con messaggi segnati (autenticati) e il consenso bizantino diventa molto + semplice.
Assunzioni: la firma di un generale leale non può essere falsificata e ogni alterazione del contenuto di un messaggio firmato può essere individuato; tutti possono verificare l'autenticità della firma di un generale

SIGNED MESSAGES algorithm
Sia V un insieme di ordini. La funzione CHOICE(V) ottiene un singolo ordine da un insieme. Limiti: CHOICE(VUOTO) = RETREAT; CHOICE(V)= v se V consiste di un singolo elemento; CHOICE(V)= RETREAT se V consiste di più di un elemento. GENERALE 0 è il comandante

CON C TRADITORE -> manda due messaggi diversi firmati
la scelta viene presa su CHOICE(attack,retreat), e i tenenti sanno che C è un traditore perché la firma appare in ordini differenti dati.

CON TENENTE TRADITORE -> invia ordine opposto. L corretto individua il traditore e obbedisce all'ordine giusto

TEOREMA: per ogni m, l'algoritmo SM(m) risolve il problema dei generali bizantini se ci sono al massimo m traditori e il numero totale di nodi è n >= 2m+1

RISULTATO di IMPOSSIBILITA':
sistemi asincroni distribuiti: non ci sono assunzioni temporali (non ci sono limiti al delay dei messaggi ne all'esecuzione degli step)
alla fine le assunzioni sincrone sono al più probabilistiche: in pratica, carichi inaspettati e variabili sono fonte di asincronia

Il consenso non può essere risolto deterministicamente in un sistema asincrono distribuito che è soggetto anche a un singolo crash, difficoltà nel determinare se un processo sia effettivamente crashato o sia solo molto lento. Fermare un singolo processo ad un momento inopportuno può causare ogni protocollo distribuito a fallire nel raggiungimento del consenso.

ESEMPIO SINCRONIZZAZIONE PROCESSOR:
sincronizzazione loosely: garantisce che processori differenti allocati ad un task stiano eseguendo la stessa iterazione, non ha bisogno di sincronizzazione stretta all'istruzione o al livello del clock
median clock algorithm: ogni clock osserva l'altro e si setta sul valore mediano che vede. Per questo la mediana calcolata da due processori differenti sarà uguale se guardano lo stesso insieme di processori

SINCRONIZZAZIONE CLOCK
se su 3 clock, uno è faulty il median clock algorithm soddisfa le assunzioni ma non il byzantine fault perchè come detto il valore visto di quello faulty potrebbe non essere lo stesso

PROBLEMA DEL CONSENSO SU UN NUMERO MANDATO DA UN PROCESSO R
N = 3+1, con al massimo 1 processo faulty. OM(m) su un numero inviato da un processo e usiamo il median value come majority funciton

CON R FAULTY NODE
tutti gli altri nodi scelgono il mediano dallo stesso set di valori quindi consenso raggiunto

CON ALTRO FAULTY NODE
2 valori di quelli ricevuti dai nodi non faulty sono = a quello di r. Con 3 valori, se almeno 2 sono uguali a v, la mediana è v. Tutti i nodi scelgono quindi lo stesso numero

SECURITY
Tassonomia dei fault: attacchi identificati come fault malevoli

L'accoppiata vulnerabilità e security exploitation rende i failures di sicurezza differenti da quelli classici.
VULENRABILITA': è una debolezza nel sistema/rete che può essere sfruttata per causare danno
EXPLOIT: exploiting è il mezzo attraverso cui una vulnerabilità può essere sfruttata per attività malevoli

SECURITY = resilienza ad attacchi malevoli -> capacità di un sistema di compiere il proprio lavoro in maniera puntuale, in presenza di attacchi

dunque sviluppo di descrizione stocastica di eventi che possono avvenire durante un attacco

modelli per analisi di sicurezza devono descrivere: come e quando un attacco avviene; impatto di un attacco sul sistema quando viene eseguito con successo; meccanismi, effetti e costi del recupero del sistema, la manutenzione del sistema e le difese.

ci sono differenze con la dependability classica

asset: informazioni o risorse che hanno un valore e sono soggette ad attacchi

MODELLAZIONE MINACCE
Processo per identificare capire e comunicare minacce e mitigazioni nel contesto di proteggere qualcosa che ha valore -> porta a decisioni architetturali PROATTIVE che riducono le minacce fin dall'inizio

THREATS VS RISKS and CONTROL
identifica lo scenario della minaccia che sono applicabili allo specifico caso d'uso. In questa fase identifichiamo le motivazioni dietro una minaccia, i mezzi e le tecniche e procedure e i vettori della minaccia
Traduzione di ogni scenario ai rischi correlati che si possono materializzare nell'ambiente.
Quindi il threat modeling è parte del RISK ASSESSMENT
Identifica dunque le contromisure che possono mitigare anche solo parzialmente i rischi identificati

ASSET TANGIBILI vs INTANGIBILI: tangibili tutto ciò che è fisico, intangibili informazioni e dati, software, reputazione etc…

TIPI DI THREAT
Adversarial cyber threats: persone o gruppi cercano di irrompere nel sistema e fare danni
Accidental threats: avviene quando qualcuno fa un errore che lede la sicurezza del sistema
Structural threats: avviene quando controlli software o ambientali falliscono
Environmental threats: avviene quando qualche evento ambientale lede il sistema

INSIDER THREAT
Un insider è interno all'organizzazione ma come minaccia è esterno perché svolge attività fuori dal contesto dell'organizzazione che non può essere controllato. I threats interni non vanno messi in secondo piano rispetto le fonti esterne.0

VULNERABILITA' sono fattori INTERNI e possono essere controllate

CONTROMISURE evita, individua contrattacca o minimizza i rischi di sicurezza a tutti gli asset, individuando vulnerabilità per ridurre la probabilità di attacchi o impatto di minacce. Quindi non andiamo a indirizzare direttamente le minacce, ma i fattori che le definiscono

CLASSIFICAZIONE CONTROMISURE: in base al tempo in cui agiscono (preventive, di detection, correttive); in base alla loro natura (fisiche, procedurali, tecniche, legali e regolatorie)

THREAT MODELING: 4 STEP -> MODELLA IL SISTEMA -> TROVA LE MINACCE -> INDIRIZZA LE MINACCE -> VALIDA

APPROCCI STRUTTURATI E NON: non strutturati = brainstorming; strutturali = asset-centric; attacker-centric; software-centric

SOFTWARE CENTRIC APPROACH (sfrutta le definizioni di TRUST BOUNDARIES e SUPERFICIE D'ATTACCO)
software boundaries mostrano chi controlla cosa e l'attack surface è una trust boundary e una direzione da cui un attaccante può lanciare un attacco

TROVA LE MINACCE -> STRIDE THREAT MODELING, fornisce un metodo per recensire il progetto del sistema ed evidenzia le minacce alla sicurezza. Usa 6 categorie per security threats per recensire il progetto del sistema
-SPOOFING -> autenticità
-TAMPERING -> integrità
-REPUDIATION -> non repudiabilità
-INFORMATION DISCLOSURE -> confidentiality
-DOS -> availability
-ELEVATION OF PRIVILEGE -> authorization

MICROSOFT SECURITY DEVELOPMENT LIFECYCLE (SDL) THREAT MODELING TOOL

THREAT REPORTING TEMPLATE dei threat
ID; NOME; DESCRIZIONE; STRIDE la sua classificazione; MITIGATED; KNOW MITIGATION; INVESTIGATION cosa sappiamo di questa minaccia; ENTRY POINTS mezzi che ha un avversario; ASSETS quali possono essere danneggiati; THREAT TREE come lo possiamo visualizzare

VULNERABILITY DESCRIPTION
ID; NOME; DESCRIZIONE; STRIDE la sua classificazione; DREAD la sua severità; CORRESPONDING THREAT id della sua minaccia corrispondente, BUG il suo id

ADDRESSING THREATS: l'approccio META
M itigating threats -> prendi azioni atte a ridurre la probabilità o l'impatto ad un livello in linea con quello desiderato di tolleranza del rischio
E liminating threats -> elimina la minaccia cambiando i requisiti di sicurezza o altre feature di sistema
T ransferring threats -> dalle ad altre organizzazioni
A accepting threats -> accetta i rischi, controllandoli tramite le risorse disponibili e si preparato a prenderne i costi associati

CWE COMMON WEAKNESS ENUMERATION
una lista sviluppata dalla community di debolezze hardware e software

PLOVER PRELIMINARY LIST OF VULNERABILITY EXAMPLES FOR RESEARCHERS

RISK ASSESSMENT
RISCHIO -> VALUTAZIONE del RISCHIO -> RISCHIO ACCETTABILE? -> METODI di MITIGAZIONE

DREAD risk assessment method
analisi qualitativa del rischio usata da MICROSOFT

CLASSIFICAZIONE DEL RISCHIO SECONDO 5 CATEGORIE:
- D AMAGE POTENTIAL valuta l'estensione del danno se viene sfruttata la vulnerabilità, considerando i tipi di dati e il tipo di accesso
-R EPRODUCIBILITY valuta lo sforzo relativo e la facilità di ripetere l'exploiting del threat
-E XPLOITABILITY si concentra solo sullo sforzo per sfruttare la vulnerabilità
-A FFECTED USERS stima il numero di utenti compromessi rispetto al numero totale di utenti
-D ISCOVERABILITY MEASURES la 'probabilità che una vulnerabilità venga trovata da hackers

RISK = CATEGORIE / diviso 5. RATING per ogni categoria da 1 a 10

OWASP OPEN WEB APPLICATION SECURITY PROJECT risk assessment rating methodologies
stima fattori tecnici e business dell'impatto, si parte da RISK = LIKELIHOOD * IMPATTO

STEP 1: Identifica il rischio
identifica un security risk che deve essere valutato, raccogli informazioni per la valutazione

STEP 2: Fattori per stimare la likelihood, è una misura rudimentale di quanto facilmente la vulnerabilità verrà scoperta e sfruttata da un attaccante

STEP 3: Fattori per stimare l'impatto
ci sono due tipi d'impatto: il primo è tecnico cioè applicativo, il secondo è al livello del business

STEP 4: Determinare la severità del rischio, la stima della likelihood e dell'impatto vengono messe insieme per calcolare la severità complessiva del rischio (TABELLA)

CVSS Common Vulnerability Scoring System, è uno standard pubblicato che cattura le principali caratteristiche di una vulnerabilità e produce uno score numerico per rifletterne la severità

GRUPPI DI METRICHE: metriche BASE riflette la severità di una vulnerabilità sulla base delle sue caratteristiche, metriche TEMPORAL cambia la base severity in base a fattori che cambiano del tempo, metriche ENVIRONMENTAL cambia score base e temporal sulla base di un ambiente specifico.

METRICHE BASE
exploitability metrics: vettore d'accesso, complessità dell'accesso, autenticazione
impact metrics: impatto confidentiality, impatto integrità, availability impact
scopus: impatto oltre al componente vulnerabile

METRICHE TEMPORAL
exploitability: la probabilità che una vulnerabilità venga attaccata
remediation level: rimedi alla vulnerabilità non patchata, porteranno a far abbassare il punteggio
report confidence: misura il grado di confidenza nell'esistenza della vulnerabilità e la credibilità dei suoi dettagli tecnici

METRICHE ENVIRONMENTAL
security requirements: confidentiality, integrity, availability requirements, permettono di customizzare il punteggio in base all'importanza degli asset da proteggere
modified base metrics: modifica alla base metrics

CVSS SCORE
RISK = BASE METRICS f (x1,...,xn) da 0 a 10-> rifinite da TEMPORAL METRICS f(y1,...,yn)-> rifinite da ENVIRONMENTAL METRICS f(z1,...,zn) = SCORE + VECTOR STRING

Notare che tutte le metriche devono essere calcolate sotto l'assunzione che l'attaccante abbia già trovato la vulnerabilità per cui non è da considerare il mezzo con cui è stata identificata

ATTACK TREE
root dell'albero è il goal dell'attacco
nodi foglia: metodi base per raggiungere il goal
NODI OR nodi in cui basta anche solo un figlio per avere successo
NODI AND nodo in cui tutti i suoi figli devono avere successo

agli attacchi atomici (nodi foglia) vengono assegnati attributi

il risultato di un'analisi può essere il valore di un attributo nel nodo radice (cheapest attack ad esempio)

minimum cut set -> insieme di attacchi atomici per raggiungere il goal

P_top (t) = P_s1(t) + … + P_sn (t)
con le singole probabilità = probabilità dei componenti del minimal cut set Si

le caratteristiche di un attaccante determinano la parte dell'attack tree su cui bisogna preoccuparsi

solitamente comunque i primi 2 livelli dell'albero sono specifici del sistema in considerazione, mentre quelli più in basso sono comuni a vari attacchi

ADVISE ADversary VIew Security Evaluation
obiettivo: comparare la forza della sicurezza di differenti system architecture
analizzare threat da diversi avversari

executable state-based security model : UN SISTEMA; UNA VISTA DELL'AVVERSARIO; METRICHE DI SICUREZZA
l'attacco viene specificato in una serie di piccoli step -> specifica di ATTACK EXECUTION GRAPH (AEG)
attack decision function -> come l'avversario seleziona il prossimo step più conveniente

ATTACK EXECUTION GRAPH
contiene timing, costi, risultati probabilistici e altre informazioni per ogni step. per questo possiamo analizzare ADVISE models usando simulazioni di eventi discreti

<A,R,K,S,G>
A: insieme degli step dell'attacco
R: insieme dei domini d'accesso del sistema
K: insieme degli elementi conosciuti rilevanti per l'attacco del sistema
S: insieme delle skill dell'attacante
G: insieme dei goal degli attaccanti rilevanti per il sistema

ATTACK STEP
X sono tutti gli stati raggiungibili del modello = s1,...,sn
a_i = <Bi,Ti,Ci,Oi,Pri,Di,Ei>
Bi: X -> {true,false} precondizione da controllare se l'attacco viene abilitato (ha accesso, conoscenza, le skill necessarie all'attacco)

Ti: X * R+ -> [0,1] il tempo richiesto per eseguire l'attack step, con Ti una variabile aleatoria definita su di una funziona di distribuzione di probabilità

Ci : X -> R maggiori di 0. costo per condurre l'attacco a prescindere dal suo risultato

Oi : insieme dei risultati degli attacchi

Pri : X * O -> [0,1] probabilità di avere risultato o in O, dopo l'attack step

Di : X * O -> [0,1] probabilità che l'attacco venga rilevato quando si rivela il risultato o in O

Ei : X * O -> X: prossimo stato quando il risultato o in O si verifica

ATTACK STEP DO-NOTHING
B_dn = precondizione vera
T_dn = tempo tra due occorrenze di do nothing
C_dn = costo zero
D_dn = detectability è zero
E_dn = s è lo stesso dello stato corrente
Pr_dn = 1 c'è solamente un risultato con probabilità 1

Ogni AEG contiene l'a_dn attack step è cio comporta che almeno un attack step ha le precondizioni soddisfatte in AEG

MODELLARE STATO s
uno stato S in X riflette i progressi dell'avversario nell'attacco del sistema

s = <Rs,Ks,Gs>

Rs = insieme di domini a cui l'avversario ha accesso
Ks = insieme di conoscenze dell'avversario
Gs = insieme di obiettivi dell'attacco raggiunti dall'avversario

DEFINIZIONE DEL PROFILO DELL'AVVERSARIO

profilo = <s0,L,V,wc,wp,wd,Uc,Up,Ud,N>

s0 stato iniziale del modello (differente tra attaccante esterno ed interno)
L: funzione di attack skill, mappa ogni attack skill con valore in [0,1]
V: funzione di attack goal value -> va in R maggiori di 0, valore monetario di ogni goal nell'AEG dal punto di vista dell'avversario
VALORE di PAYOFF P(s): di uno stato s in una funzione del valore di tutti i goal V(g) raggiunti nel model state P(s) = f(V(g))
wc,wp,wd: pesi per preferenze (COST,PAYOFF,DETECTION probability)
ATTACK PREFERENCE WEIGHT: attrattività in ognuno dei 3 criteri quando si pianifica un attacco, in [0,1]
Uc,Up,Ud: funzioni d'utilità per costo, payoff, detection probabilità
FUNZIONE DI UTILITA' : mappano i valori nativi di ogni criterio d' attrattività su una scala d'utilità in [0,1]
N: orizzonte di pianificazione

ADVISE MODEL EXECUTION
A_s e l'insieme di attack step disponibili nello stato s, ovvero quelli la cui precondizione è soddisfatta.

L'ATTACK DECISION FUNCTION sceglie il prossimo step dell'attacco e il suo risultato il prossimo stato (in maniera stocastica), processo ripetuto

ATTACK DECISION FUNCTION
attr(ai,s) = wc * C_i(s) + wp * P_i(s) + wd D_i(s)
combinazione lineare delle preferenze dell'attaccante pesate con i dati riguardo l'attack step

P_i (s) = abbiamo detto EXPECTED PAYOFF. SOMMATORIA che cicla su tutti gli outcomes (P(E_i(s,o) * Pri(s,o))) con P(E_i(s,o)) payoff nel prossimo stato
D_i(s) = SOMMATORIA che cicla su tutti gli outcomes (D_i(s,o) * Pri(s,o))), detectability dell'outcome o

B(s) miglior prossimo attack step sarà
{a* in As | attr(a*,s) = max {attr(ai,s) per tutti gli ai in As}}

il risultato di un attack step viene randomicamente generato usando le probabilità della distribuzione e il risultato determina la sequenza della transizione da stati

ALGORITMO
TIME <- 0
STATO <- s0
while il tempo < TAU do:
	Attacki <- B(stato),
	Outcome <- o,	o, Probi(stato)
	Tempo <- Tempo + t,	t, Ti(stato)
	Stato <- Ei (stato, risultato)	Ei, prossimo stato
function
end while

SPECIFICA DELLE METRICHE ADVISE
metriche dello STATO < TAU, LAMBDA, sigma>
TAU è il tempo finale [0,TAU]

LAMBDA è il tipo di metriche di stato:
	EndProb: probabilità di essere nello stato s al tempo TAU con sigma(s) = true
	AvgTime: tempo medio speso in uno stato s tale che sigma(s) = true nell'intervallo [0,TAU]

sigma è la funzione indicatrice di stato s = < R,K,G>
sigma(s) ritorna true per gli stati d'interesse

metriche degli EVENTI < TAU, DELTA, EPSILON>
TAU è il tempo finale [0,TAU]

DELTA è il tipo di metriche di stato, sia epsilon un insieme di eventi:
	Freq: numero di occorrenze di eventi in epsilon nell'intevallo [0,TAU]
	ProbOcc: probabilità che tutti gli eventi in epsilon occorrano almeno una volta nell'intervallo [0,TAU]

EPSILON è l'insieme di eventi nel modello (attack step, attack step outcomes, domini d'accesso, conoscenze e goals)

VEDI SLIDE PER ESEMPI

FUNCTIONAL SAFETY AND CYBERSECURITY ENGINEERING

SAFETY CRITICAL SYSTEM
functional safety = fornisce la sicurezza che il sistema legato alla sicurezza offra la necessaria riduzione del rischio richiesta per raggiungere la safety in presenza di malfunzionamenti
safety critical systems devono essere sviluppati in comply con gli standard certificati

ISO61508 lo standard copre safety related systems quando uno o più incorpora Electrical/Electronic/Programmable Electronic devices, coprendo possibili pericoli=hazards creati in caso di failures delle safety functions svolte dai dispositivi.

La safety lifecycle ha 16 fasi che possono essere divise in:
PHASE 1-5 riguarda l'analisi
PHASE 6-13 riguarda la realizzazione
PHASE 14-16 riguarda l'operatività

Centrali per lo standard sono:
safety lifecycle -> step richiesti per raggiungere la richiesta functional safety
risk and safety functions -> funzione di likelihood dell'evento pericoloso e la sua severità
safety integrity levels -> sono introdotti per specificare il livello target delle safety functions da essere implementati

per lo standard il livello 0 di rischio non può mai essere raggiunto
la safety va considerata fin dall'inizio
rischi non tollerabili devono essere ridotti

ANALISI dell'HAZARD e RISCHIO
analisi degli hazard: framework basato su 6 categorie di occorrenza e 4 di conseguenza combinate in una matrice (classe 1 completamente intollerabile), da classe 3 tollerabile, in totale 4 classi

analisi del rischio:
identificazione di eventi pericolosi e determinazione della necessaria riduzione del rischio per questi eventi

risk = frequenza x conseguenze

DETERMINARE LA RIDUZIONE DEL RISCHIO
E = evento pericoloso EUC = equipment under control EUC risk = rischio che nasce dagli EUC o dall'interazione con il suo sistema di controllo

EUC risk di E > rischio tollerabile di E quindi ci si assicura che il rischio di E generale nel sistema S venga ridotto fino al rischio tollerabile -> il rischio di E nelle operazioni di S' (S con applicate le funzioni) si chiama RISCHIO RESIDUO

TOOL per VALUTARE il RISCHIO
HAZOP - hazard and operability study
FMEA - failure mode and effect analysis (già vista): identificazione e valutazione gli effetti del fallimento di componenti e determinazione di cosa possa ridurre o eliminare le chance di fallimento
FMEDA - failure mode and effect diagnostic analysis: estende FMEA per includere tecniche di diagnostica online
ETA - event tree analysis
FTA - fault tree analysis

SAFETY INTEGRITY LEVELS
safety integrity: probabilità dei safety related system di eseguire in maniera soddisfacente le richieste safety functions sotto tutte le condizioni dichiarate in uno specificato periodo di tempo

SIL = discrete level per specificare i safety integrity requirements

standard IEC 61508, 4 SIL, con il SIL 4 quello più affidabile

un SIL viene determinato sulla base di un numero quantitativo di fattori in combinazione con fattori qualitativi come il processo di sviluppo e la gestione del ciclo di vita di safety

il SIL viene associato con un insieme di tecniche raccomandate per lo sviluppo

COMPETENZA DEL PERSONALE
tutto le persone coinvolte nel ciclo di attività di E/E/EPS o software devono avere la formazione appropriata, la conoscenza tecnica, esperienza e qualificazione

FUNCTIONAL SAFETY FuSa STANDARDS: standard nel campo automotive ISO 26262
functional safety: assenza di rischi non ragionevoli dovuti a pericoli causati da comportamenti di malfunzionamento di E/E/EPS

-abilita del sistema di fornire le funzionalità attese durante il ciclo di vita in presenza di malfunzionamenti di E/E/EP componenti
-riduce la probabilità di fallimenti ad un dato tasso accettabile in presenza di malfunzionamenti di E/E/EP
-Mantenere il malfunzionamento sotto controllo per preservare la sicurezza delle persone nell'ambiente circostante
-fornisce un safety lifecycle nel campo automotive
-copre aspetti della functional safety nell'intero ciclo di sviluppo

10 parti normative e 2 linee guida
VOCABOLARIO:
hazard = potenziale fonte di danni

hazardous event = combinazione di hazard e situazione d'operatività
safety = assenza di rischi non ragionevoli; safety goal = requisiti di safety a seguito di un'analisi hazard e risk assessment
safety measure = soluzione per evitare o controllare o individuare o mitigare fallimenti e loro effetti
safe state = modalità operativa di un sistema senza un non ragionevole livello di rischio
safety manager = la persona responsabile della gestione functional safety durante lo sviluppo del sistema
item = sistema o array di sistemi che implementano una funzione al livello del veicolo

ASIL AUTOMOTIVE SAFETY INTEGRITY LEVEL, classificazione del rischio
A->D (con d più rischioso)

analisi del rischio sulla base di 3 fattori: SEVERITY, EXPOSURE, CONTROLLABILITY

MANAGEMENT DELLA FUNCTIONAL SAFETY
1.assegnazione dell'ASIL a eventi hazard (SEC score in A-D)
2.associazione del safety goal per l'item, top level safety requirments, gli eventi con ASIL avranno associato un safety goal
3.derivazione dei requisiti di safety dai safety goal

I requisiti di functional safety vengono allocati agli elementi dell'architettura preliminare (verranno rifiniti in requisiti tecnici da FSR a Technical Safety Requirments)

ANALISI DEI FAILURES DIPENDENTI
l'analisi dei failures dipendenti cerca di identificare i singoli eventi o cause o modalità di fallimento che possono bypassare o invalidare una indipendenza richiesta tra elementi e violare un safety requirement o goal

REFERENCE PHASE MODEL per lo sviluppo del prodotto a livello hardware/software
per hw include: valutazione della violazione dei safety goal dovuti a fallimenti randomici hw
due alternative per valutare il rischio residuo: approccio probabilistico globale / analisi cut set

modello a v per software

SAFETY ANALYSIS
la safety analisi include:
-l'identificazione delle condizioni e cause, includendo fault e fallimenti che possono portare alla violazione del safety goal/requirment
-l'identificazione di requisiti aggiuntivi per il riconoscimento di fault e fallimenti
-la determinazione delle risposte richieste ai fault individuati
-identificazione di requisiti aggiuntivi per verificare che i safety goal/requirements sono rispettati

FUNCTIONAL SAFETY STANDARDS (ISO 21488) - ROAD VEHICLES
SOTIF SAFETY OF THE INTENDED FUNCTIONALITY si applica a funzionalità che richiedono attenzione per essere sicuri

CYBERSECURITY ENGINEERING ROAD VEHICLES
cambiamento del paradigma verso veicoli connessi e autonomi -> pericoli cybersecurity aumentano

FROM SENSING AND COMPUTING TO AUTONOMOUS VEHICLES
i sensori forniscono il quadro della situazione del mondo fisico, e l'informazione viene usata per prendere decisione su quale azione performare. 80 processori embedded con alcuni chiave ECUs Electronic Control Unit

DOMAINS FOR SECURITY IN AUTOMOTIVE
-proteggere ecu individuali: devono essere parti sicure dell'architettura
-proteggere rete del veicolo: veicolo come rete di componenti
-proteggere veicoli connessi: veicoli connessi come nodi in una rete a larga scala C2C

man man che l'evoluzione tecnologica e nuove funzionalità vengono introdotte nel campo automotive un sviluppo conscio della sicurezza è diventato un fattore cruciale.

ATTACK SURFACE AND AUTOMOTIVE CYBERSECURITY
proteggere tutti gli elementi importanti del veicoli da pericoli di cybersecurity

REGULATION AND STANDARDS
r155 regulation vehicle cybersecurity -> stabilisce i requisiti di cybersecurity richiesti per omologazione e registrazione di nuovi veicoli
r156 regulation -> gestione dei software updates

ISO 21434 -> come casa produttrice può dimostrare di essere aderente alla r155
ISO/SAE -> sae = society of automotive engineering, l'obiettivo principale dello standard è rendere i produttori coscienti dell'importanza della cybersecurity nello sviluppo del prodotto -> verso il concetto di security by design

CYBERSECURITY MANAGEMENT
lo standard specifica requisiti per la gestione del rischio di cybersecurity che riguardano il concept, sviluppo del prodotto, operazioni produttive, manutenzione e decomissioning degli E/E nei veicoli da strada considerando anche le loro interfacce e componenti

lo standard è organizzato in clausole (sezioni)

cybersecurity policy ENFORCED BY cybersecurity rules ENABLED BY cybersecurity responsibility ENSURED BY cybersecurity resources

attività distribuite: definisce le dipendenze delle interazioni e le responsabilità

INTERFACE AGREEMENT DOCUMENT: cliente e fornitore identificano le responsabilità

lo standard considera la prospettiva di un oggetto, le sue componenti e interfacce

oggetto(item): componente o insieme di componenti che implementa una funzione a livello del veicolo

un oggetto o componente interagisce con il suo ambiente operativo

CONCEPT PHASE considerazione delle funzionalità del veicolo come implementate negli item, 9.3 definizione degli item(oggetti) e del loro ambiente operativo [1.informazioni esistenti come le reti interne ed esterne;2. definizione dei confini dell'item con descrizione delle interfacce;3. funzionalità dell'item, ciò che sta realizzando per il veicolo; 4. architettura preliminare con identificazione dei componenti dell'item e le sue interfacce; 5. informazioni sull'ambiente operativo rilevanti per la cybersecurity]

ESEMPIO: HEADLAMP SYSTEM
funzioni: accendersi/spegnersi secondo le volontà dell'utente; se è in modalità ad alta luminosità se un veicolo sopraggiunge ne abbassa la luminosità e dopo la rialza quando il veicolo non è più nelle vicinanze

SISTEMA HEADLAMP vedi slide per architettura sistema

vedi slide per esempio della descrizione dell'ambiente operativo

CONCEPT PHASE: 9.3 ITEM DEFINITION -> 9.4 Cybersecurity goals, dal punto di vista dell'utente affetto

THREAT ANALYSIS AND RISK ASSESSMENT METHODS - TARA :
15.3 identificazione asset : le loro proprietà (triade CIA) e i loro scenari di danno (Input:item definiton, OUTPUT: quale proprietà di cybersecurity viene violata portnado in scenari malevoli),
15.4 scenario threat: con modellazione threat tipo EVITA o STRIDE, (INPUT: quello visto in 15.3, OUTPUT: identificazione degli scenari di danno). Relazione molti a molti damage scenario threat scenario
15.5 determinare rating dell'impatto nei vari scenari (SFOP safety financial operational privacy categorie per impatti severe, major, moderate, negligible delle precedenti categorie), (INPUT: quello visto in 15.4 e 3, OUTPUT: damage scenario valutato secondo conseguenze per utenti stradali SFOP)
15.6 identificare la via dell'attacco (con approcci topdown [attack trees, attack graphs] o bottomup [identificare la vulnerabilità]), (INPUT: quello visto in 15.5 4 e 3, OUTPUT:attack paths), attack path associati con threat scenario che realizzano
15.7 identificare la facilità con cui una via d'attacco viene utillizzata (high, medium, low, very low probabilità con conseguente sforzo richiesto -> ATTACK POTENTIAL BASED APPROACH sforzo richiesto per attaccare un componente dall'attacante con parametri che verranno sommati ELAPSED TIME, SPECIALIST EXPERTISE, KNOWLEDGE OF THE ITEM, WINDOW OF OPPORTUNITY, EQUIPMENT; CVSS BASED APPROACH già visto), (INPUT: attack path, OUTPUT:rating facilità attacco)
15.8 determinare la valutazione del rischio (aggregazioni dei rating degli attack path ai threat scenario, matrice = impact rating * attack feasibility rating), (INPUT: 15.7 e 6 e 5 e 4, OUTPUT:per ogni threat scenario, il rischio determinato da impatto dei damage scenario e la facilità dell'attacco dell'associato attack path). Formula per risk value = 1 + I x F, sempre con categoria SFOP

15.9 determinare il trattamento del rischio (INPUT: tutto quello fatto, OUTPUT:per ogni threat scenario, considerando il suo risk value viene selezionato se EVITARE IL RISCHIO rimuovendo le fonti del rischio stesso, RIDURRE IL RISCHIO, CONDIVIDERE IL RISCHIO con assicurazione, MANTENERE IL RISCHIO.)

CYBERSECURITY CLAIMS = giustificazione per aver mantenuto il rischio o condiviso
In caso di riduzione del rischio -> viene specificato un CYBERSECURITY GOAL a cui verra assegnato un:
CAL = CYBERSECURITY ASSURANCE LEVEL, livello di cybersecurity richiesta, indipendente dal risk value, determinata all'inizio dello sviluppo (CAL1-4)

CYBERSECURITY CONCEPT = CYBERSECUIRTY GOALS + CLAIMS , requisiti di cybersecurity per l'item e ambiente operativo e le misure che bisogna implementare per modificare il rischio

SCHEMA finale slide 98 della CONCEPT PHASE in ISO 21434