****

**Scuola di Ingegneria**

**Corso di Laurea Triennale in**

**Ingegneria Informatica**

**Dipartimento di**

**Ingegneria dell’Informazione**

**Modellatore e Analizzatore di Stochastic Markovian Fault Tree**

Mistretta Marco

Matricola 7005065

Casciaro Emanuele

Matricola 7012347

Anno accademico 2020-2021

Sommario

**1. INTRODUZIONE** x

1.1 Brevi accenni teorici2

1.2 Motivazione e contenuti3

**2. ANALISI DEI REQUISITI** x

2.1 Use Case Diagram5

The Modeler6

The Analyst7

**3. PROGETTAZIONE** x

3.1 Event9

3.2 Simulatorx

3.3 Plotterx

3.4 DataCenterx

3.5 Masterx

**4. ANALISI DEI REQUISITI** x

Use Case Diagramx

The Modelerx

**5. INTRODUZIONE** x

1.1 Brevi accenni teoricix

1.2 Motivazione e contenutix

**6. ANALISI DEI REQUISITIx**

Use Case Diagramx

The Modelerx

**7. INTRODUZIONEx**

1.1 Brevi accenni teoricix

1.2 Motivazione e contenutix

**8. ANALISI DEI REQUISITIx**

Use Case Diagramx

The Modelerx

1. **INTRODUZIONE**
   1. **Brevi Accenni Teorici**

Uno Stochastic Markovian Fault Tree (SMFT) è un albero che rappresenta le condizioni sotto cui i fallimenti di un insieme di componenti si propagano realizzando un modo di fallimento del sistema, detto Top Event: ciascuna foglia, detta Basic Event, rappresenta lo stato di fallimento di un componente del sistema; i nodi intermedi, detti Gates, esistono in vari tipi (statici e dinamici) che si differenziano per come propagano il fallimento. Un Basic Event (o uno stesso Intermediate Event) può presentarsi in ingresso a più Intermediate Event (Repeated Event) o meno.

**1.2 Motivazioni e Contenuti**

L’obbiettivo di questo progetto era quello di modellizzare uno SMFT generico nel quale: i Basic Event alternano stati di funzionamento e malfunzionamento secondo un semplice modello di Gilber Elliot (i.e. ad intertempo esponenziale), possono essere presenti Repeated Event e i Gate possono essere di natura sia statica che dinamica. Tutto ciò ai fini di: simulare l’esecuzione per un dato stato di inizializzazione, poter ripetere più volte la simulazione in condizioni atte a poter calcolare gli indici di confidenza sul valore medio (e.g. tramite test di Student) e sperimentare la natura ergodica del modello.

1. **ANALISI DEI REQUISITI**

La prima fase di progettazione si è basata sulla stesura dello Use Case Diagram (figura2.1)

**2.1 Use Case Diagram**

Sono stati identificati due Attori fondamentali: “the Modelist” e “the Analyst”. Nell’ottica di come è stato ideato il programma “the Modelist” rappresenta l’utente con esperienza di modellizzazione di SMFT che si occuperà appunto, tramite il programma, della fase di modellizzazione Event per Event, Gate dopo Gate, dell’intera struttura. “The Analyst” rappresenta invece un utente con sufficiente conoscenze di analisi e studio di SMFT, il quale, tramite il programma, sarà in grado di verificare l’ergodicità del sistema e calcolare gli indici di confidenza sul valore medio, istante per istante, della reliability del sistema. Tutto questo tramite simulazioni inizializzate con parametri da lui considerati opportuni.

Analizziamo il diagramma più nel dettaglio.

* **The Modeler**

Come abbiamo detto, The Modeler ha il principale caso d’uso di modelizzare l’intera struttura. Per fare questo può quindì definire nuovi Basic Event e sulla base dei Basic Event già definiti può dunque andare a definire nuovi Intermediate Event. (notare che non esistono Intermediate Event senza figli, un Event senza figli è necessariamente una foglia, dunque per definizione è un Basic Event).

* **The Analyst**

The Analyst ha principalmente bisogno di simulare più e più volte l’esecuzione del SMFT, a partire da inizializzazioni atte a raggiungere i vari scopi rappresentati dai suoi casi d’uso. Per verificare la natura ergodica può infatti simulare più volte il sistema a partire da stati iniziali differenti e verificare, in via d’approssimazione, che dopo un “tempo sufficientemente lungo” (ci soffermeremo successivamente sulla definizione di tempo sufficientemete lungo), tutte le simulazioni si aggirano intorno allo stesso valore percentuale di reliability. Per il calcolo degli indici di confidenza, invece, può simulare più volte un sistema ma a parti

1. **PROGETTAZIONE**

In prima fase ci siamo concentrati sulla progettazione del Class Diagram (UML). Di questo ultimo è possibile visionare uno storica che raccoglie tutte le modifiche apportate dalla prima versione alla versione definitiva (figura 3.1)

**3.1 Event**

* **Composite**

In prima battuta ci siamo concentrati sulle scelte implementative che coinvolgevano la modellazione dell’albero stesso. L’obbiettivo era quello di rappresentare un grafo diretto aciclico (DAG). Il Design Pattern Composite era dunque atto all’utilizzo per raggiungere questo scopo, le sue condizioni di utilizzo erano rispettate

**TBD: Aggiungere Dettagli**

* **Observer**

Avevamo tra le mani la struttura di classi che pensavamo meglio avrebbe potuto rappresentare il SMFT, il nostro successo step logico si è mosso in direzione di affrontare la stesura delle classi che avrebbero modellato questo albero e lo avrebbero gestito. La risposta a queste due questioni era semplice: avevamo bisogno di una Classe EventManager (nata con lo scopo di gestire l’inizializzazione, operazione ripetuta più volte in fase di analisi e stusio del SMFT, e setting dei parametri) e di una classe Modeler (nata con lo scopo di instanziare i Basic Event e gli Intermediate Event, successivamente rinominata EventModeler per conformità con il package).

TBD: EventManager…

TBD: EventModeler…

TBD: Il problema dell’aggiornamento, la scelta di impiegare observer…

**3.2 Simulator**

**3.3 Plotter**

Una volta eseguite le simulazioni, è utile, ai fini analitici, fornire un output dei dati facilmente interpretabile da un utente esterno. È allora necessaria una classe che, una volta raccolti i dati richiesti, li propone all’utente finale, insieme all’albero che rappresenta il sistema, necessario per poter interpretare in modo corretto i dati.

La classe ***HarryPlotter*** si occupa dunque di inoltrare i dati ricevuti ai servizi forniti dalle librerie esterne usate\*, rendendola una possibile istanza di adapter. Inoltre, essendo unico il servizio di stampa, esso deve essere condiviso da tutte le possibili istanze del processo; si è reso allora necessario implementarla come singleton.

**3.4 DataCentre**

Dato che i dati forniti da una singola simulazione non sono portatori di informazione utile, diventa impellente la presenza di un entità che raccolga i dati eseguiti dalle varie simulazioni, li organizzi e, su richiesta, li restituisca secondo in un formato ben definito, che nel nostro caso corrisponde alla quantizzazione dei dati.

**3.5 Master**

1. **STRUTTURA DEL CODICE**

Dalla stesura del Class Diagram è risultato subito evidente la convenienza dell’impiegare il Design Pattern Composite per rappresentare lo SMFT e del Pattern Observer per rendere possibile alla Classe EventManager di rimanere sembre aggiornato sulla struttura del modello.

**4.1 Composite**

**4.2 Observer**

**4.3 Singleton**

1. **TESTING**
2. **MOCKUPS**
3. **CONSIDERAZIONI FINALI**