



UNIVERSITÀ  
DEGLI STUDI  
FIRENZE

Scuola di Ingegneria

Corso di Laurea in Ingegneria Informatica

Software Engineering for Embedded Systems  
Project work

**Titolo**

*Edoardo Sarri*

7173337

Data

# Indice

<b>1</b>	<b>Introduzione</b>	<b>5</b>
1.1	Capacità . . . . .	5
<b>2</b>	<b>Analisi</b>	<b>6</b>
2.1	Componenti . . . . .	6
2.1.1	Task . . . . .	6
2.1.2	Taskset . . . . .	6
2.1.3	Risorse . . . . .	6
2.1.4	CPU . . . . .	6
2.1.5	Scheduler . . . . .	7
2.1.6	Protocollo di accesso alle risorse . . . . .	7
2.2	Class diagram . . . . .	7
<b>3</b>	<b>Implementazione</b>	<b>8</b>
3.1	Log . . . . .	8
3.2	Rate Monotonic . . . . .	8
<b>4</b>	<b>Dubbi</b>	<b>9</b>
4.1	Domande . . . . .	9

# Elenco delle figure

# Elenco delle tabelle

# 1 Introduzione

L'obiettivo è creare un sistema (in Java) eseguibile da linea di comando che permetta di generare tracce di un'esecuzione. Ogni traccia definita come una sequenza di coppie  $\langle \text{evento}, \text{tempo} \rangle$ .

Un *evento* può essere: rilascio di un job di un task; acquisizione/rilascio di un semaforo da parte di un job di un task; completamento di un chunk; completamento di un job di un task (in realtà questo coincide con il completamento dell'ultimo chunk del task).

## 1.1 Capacità

Vogliamo avere la possibilità di:

- Iniettare fault

Si vuole avere la possibilità di iniettare fallimenti tramite due tecniche: aggiungere un task che fa cycle stealing (implementato tramite un task a priorità massima che non appartiene al task set da schedulare); task programming defect, cioè se abbiamo un'implementazione non funzionante di un task scorretta (es: acquisisce il semaforo ma non si alza la priorità).

- Osservare possibili fallimenti

I possibili fallimenti che si vogliono osservare sono: deadline miss; violazione del tempo di computazione del chunk (sia in eccesso che in difetto).

## 2 Analisi

In questo capitolo analizziamo la struttura del progetto, partendo dai suoi componenti e definendo la loro relazione.

### 2.1 Componenti

#### 2.1.1 Task

Un task è definito da:

- Pattern di rilascio, cioè una distribuzione del tempo tra due arrivi consecutivi; rappresenta il periodo del task. Ci semplifica la vita rappresentando la distribuzione con il suo Sampler (vedi Sirio); aggiungiamo a questi il DeterministicSampler parametrizzato dal tempo di rilascio.
- Numero e tipo di chunk.
- Deadline. Forse conviene considerare la deadline relativa; a lezione abbiamo visto questa soluzione per implementare i vari resource access protocol.
- Priorità nominale e dinamica.

Non ci interessa definire un activation time perché vogliamo considerare il caso pessimo: l'activation time sarà l'istante iniziale per tutti i task.

Un chunk, cioè una computazione atomica del task, è definito da: una distribuzione del tempo di esecuzione; una eventuale richiesta di risorse da usare in mutua esclusione (da acquisire prima dell'esecuzione e rilasciare subito dopo).

#### 2.1.2 Taskset

È un insieme di task. È l'oggetto principale gestito dallo scheduler.

#### 2.1.3 Risorse

Sono le risorse da utilizzare in mutua esclusione. Ogni risorsa è gestita da un semaforo binario.

#### 2.1.4 CPU

È l'unità di elaborazione. Supponiamo essere unica.

### **2.1.5 Scheduler**

È il componente che assegna un task al processore. Per il momento implementiamo solo Rate Monotonic (RM) e Earliest Deadline First (EDF).

### **2.1.6 Protocollo di accesso alle risorse**

È il meccanismo che garantisce la mutua esclusione di una risorsa. Per il momento implementiamo solo Priority Ceiling Protocol (PCP). Forse poi anche Priority Inheritance Protocol (PIP), ma vediamo.

## **2.2 Class diagram**

## 3 Implementazione

### 3.1 Log

Per il logging è stato implementato un semplice logging su un file.

### 3.2 Rate Monotonic

Descrivere come è stato implementato.



# 4 Dubbi

## 4.1 Domande

- Nei fallimenti osservati che vuol dire valutare la violazione del tempo di computazione di un chunk (troppo basso o troppo alto)? Se non viola la deadline allora esegue per il suo execution time altrimenti di più.
- Al momento l'acquisizione delle risorse dei chunks dello stesso task sono indipendenti.
- Manca EDF e la possibilità di iniettare fault.

# Bibliografia

- [1] Laura Carnevali. *Appunti slides Software Engineering for Embedded System*.
- [2] *chatGTP*.
- [3] *documentazione Java, oracle*. <https://docs.oracle.com/javase/8/docs/api/overview-summary.html>.