



## Manuale Utente

**Nome prodotto:** SGPEM v2

**Data:**

18/07/2006

**Versione:**

1.13

**Stato del documento:**

Formale esterno

**Redazione:**

- |   |                     |       |
|---|---------------------|-------|
| 1 | Marin Pier Giorgio. | firma |
| 2 | Dalle Pezze Piero.  | firma |
| 3 | Bertoli Stefano.    | firma |
| 4 | Perin Michele.      | firma |

**Revisione:**

- |    |                  |                      |       |
|----|------------------|----------------------|-------|
| 1. | Sarto Carlo      | Padova lì 18/07/2006 | firma |
| 2. | Dimartino Orazio | Padova lì 18/07/2006 | firma |

**Approvazione:**

- |    |                   |                      |       |
|----|-------------------|----------------------|-------|
| 1. | Dalle Pezze Piero | Padova lì 18/07/2006 | firma |
|----|-------------------|----------------------|-------|

**Lista di distribuzione:**

- 1 Committente Prof.Tullio Vardanega
- 2 Committente Prof. Renato Conte
- 3 Piero Dalle Pezze
- 4 Stefano Bertolin
- 5 Pier Giorgio Marin
- 6 Michele Perin
- 7 Carlo Sarto
- 8 Orazio Dimartino

**Registro delle modifiche:**

- v. 1.13 18/07/2006  
Aggiunta descrizione procedura per aggiungere processi e risorse ad una nuova configurazione (Perin Michele)
- v. 1.12 20/03/2006  
Aggiunte immagini 3.1, 3.2, 3.3. Aggiornato paragrafo 3.1.1 (Marin Pier Giorgio).



- v. 1.11 17/03/2006  
Aggiunto paragrafo 4.1 (Marin Pier Giorgio).
- v. 1.10 16/03/2006  
Aggiunti paragrafi 2.5, 3.3.4, 3.3.5 (Perin Michele), aggiornamento paragrafo 3.1.3  
divisione paragrafi 3.3.1-3.3.3 (Marin Pier Giorgio).
- v. 1.09 16/03/2006  
Aggiornamento paragrafo 2.3 (Bertolin Stefano).
- v. 1.08 14/03/2006  
Integrazione nel paragrafo 3.1.2. (Piero Dalle Pezze).
- v. 1.07 14/03/2006  
Modificato paragrafo 3.1.3 (Bertolin Stefano).
- v. 1.06 11/03/2006  
Revisione del documento. Integrazione ed aggiunte nel paragrafo 4.2 e 4.3.  
Aggiunti paragrafi Gestione politiche e Scrivere una nuova politica e i  
sottoparagrafi Politica Ordinamento e Politica Assegnazione.  
(Dalle Pezze Piero).
- v. 1.05 09/03/2006  
Correzioni paragrafo 3.1.2, 3.1.4 (Sarto Carlo), correzione paragrafo 1.2  
(Dimartino Orazio).
- v. 1.04 06/03/2006  
Aggiunto paragrafo 3.2 (Marin Pier Giorgio).
- v. 1.03 06/03/2006  
Aggiunti paragrafi 3.1 , 4.2, 4.3, 4.4 (Marin Pier Giorgio).
- v. 1.02 03/03/2006  
Aggiunti paragrafi 2.1, 2.2 (Marin Pier Giorgio).
- v. 1.01 02/03/2006  
Aggiunti paragrafi 1.1-1.6 (Marin Pier Giorgio).
- v. 1.00 27/02/2006  
Stesura bozza iniziale (Marin Pier Giorgio).



## Sommario:

Il documento rappresenta il manuale per apprendere l'utilizzo del prodotto, nelle sue varie parti.

## Indice:

1. Introduzione	pag. 4
1.1 Scopo del documento	pag. 4
1.2 Scopo del prodotto	pag. 4
1.3 Definizione dell'utente del prodotto	pag. 4
1.4 Come leggere il manuale	pag. 4
1.5 Documenti utili	pag. 4
1.6 Come riportare problemi e malfunzionamenti	pag. 4
2. Descrizione generale	pag. 5
2.1 SGPEMv2	pag. 5
2.2 Caratteristiche principali del prodotto	pag. 5
2.3 Requisiti tecnici per il funzionamento del programma	pag. 6
2.3.1 Requisiti minimi per Linux	pag. 6
2.3.2 Requisiti minimi per Windows	pag. 6
2.4 Installazione del programma	pag. 6
2.5 Disinstallazione del programma	pag. 6
3. Istruzioni per l'uso	pag. 7
3.1 Descrizione funzionale	pag. 7
3.1.1 Configurazione iniziale	pag. 7
3.1.2 Simulazione	pag. 9
3.1.3 Gestione Politiche	pag. 12
3.1.4 Help	pag. 12
3.1.5 Barra del menu	pag. 13
3.2 Scrivere una nuova politica	pag. 13
3.2.1 Politica di ordinamento	pag. 13
3.2.2 Politica di assegnazione	pag. 15
3.3 Azioni richieste/permesse	pag. 16
3.3.1 Aggiunta di una politica	pag. 16
3.3.2 Rimozione di una politica	pag. 17
3.3.3 Gestione delle compatibilità	pag. 18
3.3.4 Compilare una nuova politica	pag. 18
3.3.5 Creare la documentazione del codice sorgente	pag. 19
4. Appendice	pag. 19
4.1 Messaggi di errore e loro significato	pag. 19
4.2 Glossario	pag. 21
4.3 Politiche di ordinamento	pag. 23
4.4 Politiche di assegnazione delle risorse	pag. 24



## **1 Introduzione**

### **1.1 Scopo del documento**

Scopo del presente manuale è fornire all'utente il supporto necessario per acquisire un'adeguata conoscenza sia concettuale che pratica delle procedure messe a disposizione dall'applicazione.

### **1.2 Scopo del prodotto**

Lo scopo del prodotto (*SGPEM v2*) è quello di fornire un sistema didattico destinato allo studio dei meccanismi di gestione delle *risorse* in un elaboratore multiprogrammato. Tale software consentirà all'utilizzatore di simulare il comportamento di varie politiche di ordinamento, sulla base di dati da lui immessi, ai fini di illustrare le problematiche inerenti all'ordinamento di *processi*.

### **1.3 Definizione dell'utente del prodotto**

L'utente del prodotto viene identificato principalmente nello studente di un corso universitario di sistemi operativi, che vuole mettere in pratica i concetti appresi a lezione riguardanti le politiche di ordinamento, e l'assegnazione delle *risorse*, oppure semplicemente per esercizio in previsione dell'appello d'esame.

Il prodotto può essere utilizzato anche dai docenti come supporto per la spiegazione degli argomenti, degli esercizi, o per la creazione di questi ultimi.

L'utilizzo del prodotto è anche aperto all'utente che vuole avvicinarsi alla materia, avendo comunque una base teorica sull'argomento.

### **1.4 Come leggere il manuale**

Questo manuale contiene tutte le informazioni necessarie per l'utilizzo di *SGPEMv2* nella preparazione di nuovi esercizi e nell'esecuzione di quelli già disponibili.

Il resto del manuale è organizzato secondo una scelta didattica di partire dalle operazioni di installazione per poi passare all'utilizzo del prodotto.

E' dunque possibile, in una prima fase, usare il manuale per un corso di auto-apprendimento e, successivamente, proseguire consultando le parti che servono.

### **1.5 Documenti utili**

Altri documenti utili per utilizzare correttamente il prodotto *SGPEMv2* sono i libri Modern Operating System di Andrew S. Tanenbaum, Operating System Concepts di Silberschatz, Galvin, Gagne, e la documentazione javadoc, che verrà fornita in allegato. L'importanza di quest'ultimo documento è dovuta alla possibilità, fornita all'utente, di poter ampliare l'insieme di *politiche di ordinamento e di assegnazione alle risorse*, rispetto a quelle fornite dalla nostra azienda.

L'aggiunta di una nuova politica sarà spiegata nel dettaglio in questo documento, tale funzionalità però richiede una conoscenza dell'architettura, e dei metodi di una *politica di ordinamento* e di *assegnazione*, ecco perché l'importanza dei documenti sopracitati.

### **1.6 Come riportare problemi e malfunzionamenti**

Problemi e malfunzionamenti devono pervenire alla casella di posta elettronica [bluethoth@tiscali.it](mailto:bluethoth@tiscali.it), riportando come oggetto delle e-mail :”Malfunzionamento *SGPEMv2*”.



Le comunicazioni devono contenere le seguenti informazioni:

- Nome e Cognome mittente.
- Data riscontro del problema e/o malfunzionamento.
- Natura del problema e/o malfunzionamento.
- Descrizione dello scenario e delle condizioni sotto le quali si è verificato il problema e/o malfunzionamento.

L'azienda provvederà a gestire e risolvere il problema indicato, secondo le norme interne, per poi fornire un'eventuale versione del prodotto con le opportune correzioni.

## **2 Descrizione generale**

### **2.1 SGPEMv2**

SGPEMv2 è un acronimo per Simulatore della Gestione di Processi in un Elaboratore Multiprogrammato, e dove “v2” sta ad indicare la versione del prodotto.

Il suo scopo è quello di fornire una *simulazione*, che illustri le problematiche inerenti all'ordinamento di *processi* e i meccanismi di gestione delle *risorse* all'interno di un elaboratore multiprogrammato.

### **2.2 Caratteristiche principali del prodotto**

Il prodotto software SGPEMv2 fornisce una *simulazione* grafica e testuale operante su più finestre per illustrare le problematiche inerenti all'ordinamento di *processi* e i meccanismi di gestione delle *risorse* all'interno di un elaboratore multiprogrammato.

Ogni finestra descrive un aspetto significativo riguardante la *simulazione* che si sta eseguendo.

Vengono infatti visualizzati, ad ogni istante della *simulazione*:

- Per ogni *processo*, lo stato attuale.
- Per ogni *risorsa*, quale *processo* la detiene, e quale/i sono bloccati in attesa che si liberi.
- *Il grafo dell'allocazione delle risorse.*
- La rappresentazione grafica dell'avanzamento temporale dell'ordinamento dei *processi*.

Le caratteristiche principali di SGPEMv2 sono:

- E' possibile creare una nuova configurazione dati per la *simulazione*.
- E' possibile modificare una configurazione aperta.
- E' possibile salvare una configurazione.
- E' possibile aprire una configurazione precedentemente salvata.
- La *simulazione* può essere eseguita in più modalità (ad avanzamento continuo, step by step sia avanti che indietro, direttamente alla fine, torna all'inizio).
- La *simulazione* può essere interrotta in qualsiasi momento.
- Ad ogni interruzione è possibile avere delle informazioni statistiche.
- Viene fornito un aiuto sensibile al contesto.

L'utente ha inoltre la possibilità di estendere l'insieme delle politiche fornite, grazie all'architettura del prodotto.



## **2.3 Requisiti tecnici per il funzionamento del programma**

### **2.3.1 Requisiti minimi per Linux:**

- Requisiti software: Java 2 Runtime Environment v 1.4.2 (J2RE).
- Requisiti hardware: Processore Intel i586 o compatibili; 32 MB di memoria RAM (48 MB raccomandati).

### **2.3.2 Requisiti minimi per Windows:**

- Requisiti software: Java 2 Runtime Environment v 1.4.2 (J2RE); Sistemi operativi: Microsoft Windows 98, Windows ME, Windows NT 4.0, Windows XP Home, Windows XP Professional, Windows 2000 Professional, Windows Server 2003.
- Requisiti hardware: Processore Intel Pentium 166MHz o compatibili; 32 MB di memoria RAM.

## **2.4 Installazione del programma**

Il programma viene fornito con un makefile per Linux e un file di installazione per Windows.

Linux:

Dalla shell, posizionarsi nella directory contenente il makefile, e digitare il comando **make install** seguito da invio.

Per l'esecuzione digitare **make esegui** seguito da invio.

Windows:

Fare doppio click sul file WinInstall.bat.

## **2.5 Disinstallazione del programma**

Linux:

Dalla shell, posizionarsi nella directory contenente il makefile, e digitare il comando **make uninstall** seguito da invio.

Windows:

Fare doppio click sul file WinUninstall.bat.

**Attenzione:** Entrambi i comandi non eliminano la directory Politiche contenente gli eventuali sorgenti delle politiche definite dall'utente. La responsabilità di eliminare i file creati dall'utente è lasciata all'utente stesso.



## **3 Istruzioni per l'uso**

### **3.1 Descrizione funzionale**

#### **3.1.1 Configurazione iniziale**

Una volta installato e avviato il programma, l'utente si interfaccia subito con SGPEMv2.

Prima di poter iniziare una *simulazione*, l'utente deve passare per la fase di configurazione del sistema, ovvero deve creare una configurazione dati sulla quale poi avverrà la *simulazione*. Tale configurazione può essere eseguita manualmente, oppure aprendo una configurazione salvata su file.

L'inserimento dei dati per creare una nuova configurazione si articola in tre semplici passi.

Il primo passo consiste nella scelta della *politica di ordinamento* dei *processi* che si intende far simulare, tra quelle fornite nel prodotto.

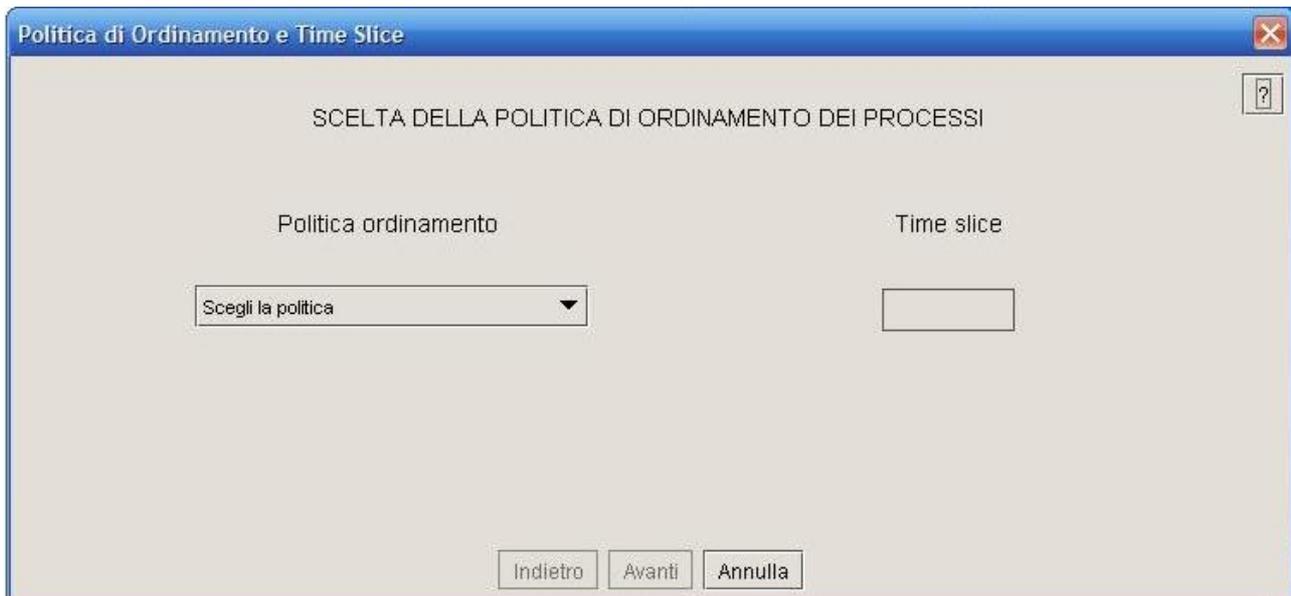


Fig. 3.1 Inserimento dati (primo passo).

Le politiche di ordinamento disponibili sono:

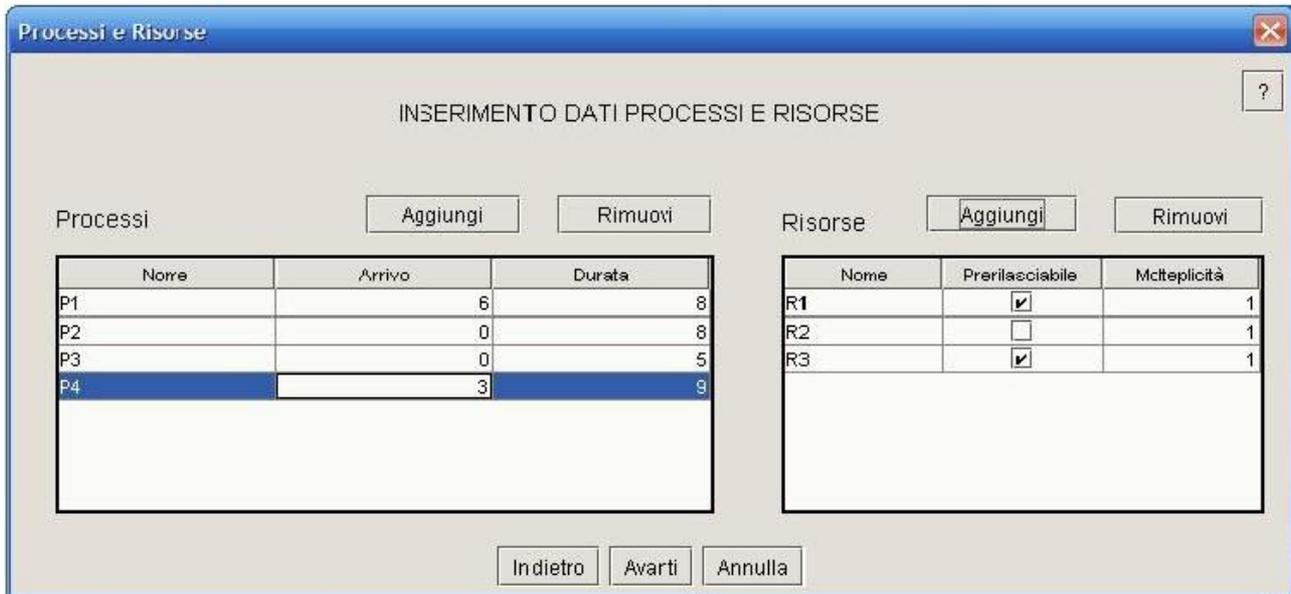
- First Come First Served.
- Shortest Job First.
- Shortest Remaining Time First.
- Round Robin.
- Round Robin con priorità.
- Round Robin con priorità con prerilascio (per priorità).
- Highest Response Ratio Next.
- Priorità.
- Priorità con prerilascio.

Per le politiche che prevedono l'inserimento del valore di time slice, è previsto un'apposita area testuale per poterne inserire il valore. Tale area sarà attiva solo

per le politiche che richiedono questo valore.

Una volta scelta la politica sarà possibile passare al secondo passo della configurazione cliccando sul bottone avanti che si attiverà.

Nel secondo passo l'utente deve inserire i *processi* (max 20) e le *risorse* (max 20), che vanno a costituire i dati principali della *simulazione*.



Nome	Arrivo	Durata
P1	6	8
P2	0	8
P3	0	5
P4	3	8

Nome	Prerilasciabile	Mltiplicità
R1	<input checked="" type="checkbox"/>	1
R2	<input type="checkbox"/>	1
R3	<input checked="" type="checkbox"/>	1

Fig. 3.2 Inserimento dati (secondo passo).

L'inserimento viene fatto tramite una tabella che ha come colonne i campi dati caratterizzanti un *processo* o una *risorsa*.

E' disponibile anche un tasto rimuovi, per rimuovere la riga, corrispondente ad un *processo* o ad una *risorsa*, selezionata.

Alla scelta della politica d'adottare, come detto, si passa allo step del wizard che consentirà l'inserimento di un nuovo processo. Tale operazione sarà possibile cliccando sul tasto aggiungi. A quel punto verrà aggiunta una nuova riga sulla tabella dei processi, riga in cui sarà possibile inserire tutti i parametri che caratterizzeranno il nuovo processo da inserire.

Viene impedito l'inserimento di dati errati (che non rispettano la correttezza logica), e nel caso di mancanza di dati, il sistema provvede ad inserire valori di default.

Una volta inserito almeno un *processo*, viene attivato il pulsante avanti, che, se non sono state inserite *risorse* conclude la fase di configurazione permettendo l'avvio della *simulazione*, altrimenti fa avanzare la fase di configurazione al terzo ed ultimo passo.



In questo passo, l'utente deve inserire le richieste dei *processi* alle *risorse*. Anche qui l'inserimento è reso semplice dalla tabella in cui i dati devono essere inseriti, ma anche sicuro sempre rispetto a errori logici e mancanza di dati. L'inserimento di una nuova risorsa è analogo a quello di un nuovo processo. Unica variante è poi la necessità di associare ogni risorsa ai processi che la usano.

Nome	Arrivo	Durata
P1		8
P2	0	8
P3	0	5
P4	0	9

Nome	Prerilasciabile	Molteplicità
R1	<input checked="" type="checkbox"/>	1
R2	<input type="checkbox"/>	1
R3	<input checked="" type="checkbox"/>	1

Processo	Risorsa	Durata	Istante Richiesta
P2	R2	3	0
P3	R3	4	0

### 3.3 Inserimento dati (terzo passo).

L'utente, per aggiungere una nuova richiesta di un processo ad una risorsa, deve selezionare un *processo* e una *risorsa*, tra quelli inseriti precedentemente, e che vengono visualizzati in apposite tabelle di riepilogo, premere il pulsante "Avanti" e procedere poi con l'inserimento dei dati mancanti.

In questo passo l'utente deve anche scegliere la *politica di assegnazione* delle *risorse* ai *processi* tra quelle disponibili:

- First Come First Served.
- Shortest Job First.
- Casuale.

Al termine di questa fase, che si conclude solo se è stata inserita almeno una assegnazione e scelta la politica, l'utente può, premendo il tasto Fine, chiudere la fase di configurazione e passare alla *simulazione*.

Altra modalità per configurare una *simulazione*, è aprirne una già creata e salvata. Le configurazioni salvate sono riconoscibili dall'estensione .fcs.

### 3.1.2 Simulazione

Una volta terminata la fase di configurazione, l'utente può condurre la *simulazione*, in più modalità (avanzamento automatico, step by step (sia avanti che indietro), vai alla fine, torna all'inizio).

L'avanzamento di una *simulazione* può essere interrotto in qualsiasi momento, per



poi essere ripreso successivamente.

L'interfaccia grafica principale è suddivisa in più finestre per permettere all'utente di avere una visione completa sull'avanzamento della *simulazione*.

Le finestre presenti sono:

- Avanzamento dei *processi*.
- Assegnazione delle *risorse*.
- Coda dei pronti.
- Lista dei terminati.
- *Processi*.
- *Risorse*.
- Statistiche.

Avanzamento dei processi:

In questa finestra è riportato graficamente come i *processi* sono eseguiti dalla CPU. In particolare, sono rappresentate le unità di tempo di esecuzione di ogni *processo*, secondo la configurazione scelta. Se il *processo* scelto per eseguire dovesse richiedere una risorsa che in quell'istante non è disponibile, il *processo* viene bloccato e la CPU rimane inattiva per un'unità di tempo. Quest'evento è rappresentato con uno spazio vuoto di un'unità. Invece, il tempo di cambio di contesto tra *processi* non è considerato. Ad ogni *processo* è assegnato un colore, in modo da rendere più intuitiva la *simulazione*.

Assegnazione delle risorse:

In questa finestra è rappresentato il *grafo dell'allocazione delle risorse ai processi*. I *processi* vengono rappresentati con dei cerchi colorati, con il nome riportato all'interno, mentre le *risorse* vengono rappresentati con dei quadrati, al cui interno sono presenti, oltre al nome della *risorsa*, dei punti neri che ne indicano la molteplicità.

Le *risorse* presentano colori differenti a seconda esse siano *prerilasciabili* (bianche) o *non prerilasciabili* (grigio).

Un *processo* è in relazione con una *risorsa* tramite una freccia.

Se il *processo* detiene la *risorsa* la freccia è diretta dalla *risorsa* al *processo*.

Se il *processo* ha richiesto una *risorsa* o si trova bloccato in attesa che questa si liberi, la freccia è rivolta dal *processo* alla *risorsa*.

Coda dei pronti:

In questa finestra sono riportati, tutti e solo i *processi* pronti ad eseguire, ordinati secondo la politica di ordinamento scelta. Il primo *processo* della coda dei *processi pronti* è quello che compare in alto alla lista.

Lista dei terminati

In questa finestra sono riportati, tutti e solo i *processi* che hanno terminato la loro esecuzione, in ordine temporale di terminazione. Il primo *processo terminato* è quello che compare in alto alla lista. In ordine di terminazione sono inseriti gli altri.

Processi



In questa finestra sono riportati, in forma tabellare, tutti i *processi*, con a fianco il loro stato (in esecuzione, pronto, terminato, in arrivo, bloccato).

#### Risorse

In questa finestra sono riportate, in forma tabellare, tutte le *risorse*, con a fianco riportati, per ogni *risorsa*, il *processo* che eventualmente la detiene e la coda dei *processi bloccati* in attesa che questa si liberi. Quest'ultima coda è ordinata secondo la politica di assegnazione scelta. Se la *risorsa* è prerilasciabile, la coda dei processi bloccati non è abilitata.

#### Statistiche

Questa finestra visualizzerà i dati statistici. Può essere visualizzata solo a *simulazione* interrotta, e non nello stato iniziale. E' possibile avanzare la *simulazione* manualmente ed osservare l'evoluzione delle statistiche dopo aver attivato il bottone "visualizza statistiche". I dati visualizzati comprendono le statistiche della simulazione fino all'istante corrente (*Throughput*, tempo medio di attesa, tempo medio di risposta, tempo medio di *turn around*) ed le statistiche relative ad ogni *processo* attivato (tempo di attesa, tempo di risposta, tempo di *turn around*, occupazione della CPU, occupazione della CPU percentuale). Per il significato dei dati statistici rilevati, si veda il glossario.



### 3.1.3 Gestione politiche

Il software SGPEMv2 è stato accuratamente progettato tenendo in considerazione la possibilità di estendere l'insieme delle politiche di ordinamento e di assegnazione. La presenza di questo requisito, permette all'utente (vedere paragrafo 1.3 di questo documento) di scrivere una nuova politica in linguaggio di programmazione Java senza dover modificare l'architettura ed il codice sorgente del software. Le norme per scrivere una nuova politica sono precisate nel paragrafo 3.2 di questo documento.

La gestione delle politiche, che comprende l'aggiunta e la rimozione di queste ultime e l'impostazione delle compatibilità tra quelle di ordinamento e quelle di assegnazione, viene avviata selezionando la voce "gestione politiche" dal menù "File".

Selezionando tale voce si aprirà una piccola finestra, suddivisa in due pannelli ("Politiche" e "Compatibilità"), che permetterà di eseguire le operazioni sopra citate:

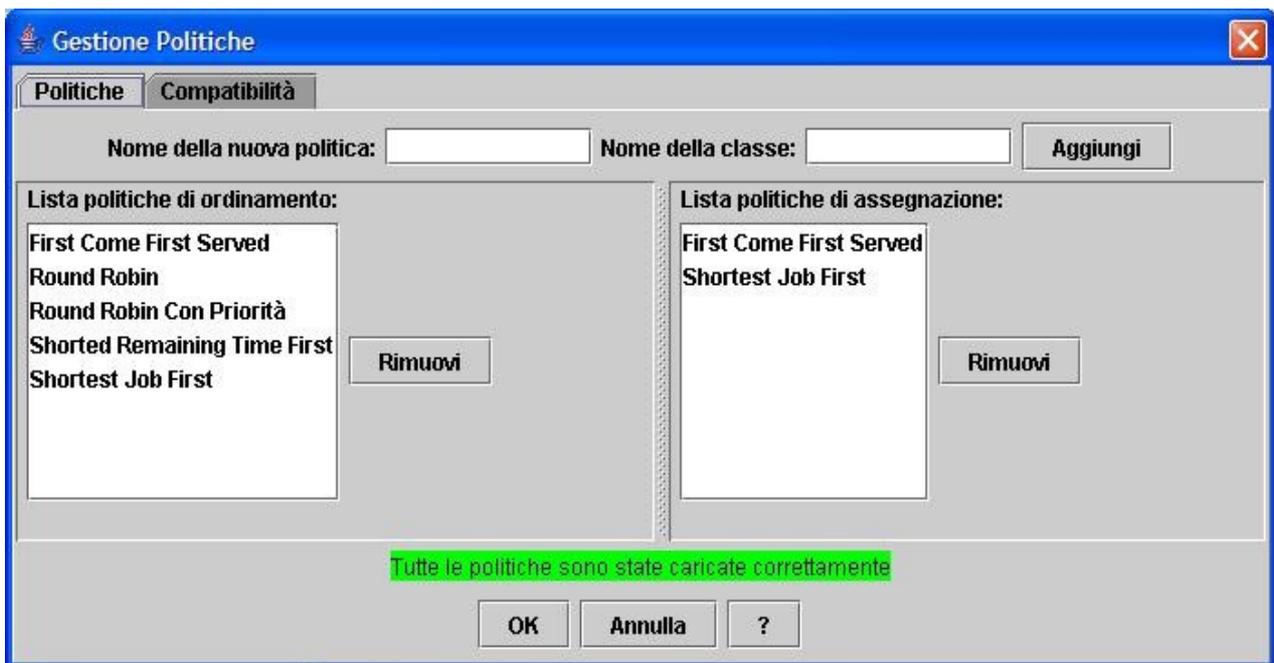


Fig. 3.4 Pannello Gestione Politiche (Politiche).

Nella parte inferiore del pannello "Politiche" è possibile leggere l'elenco delle politiche che non sono state caricate con successo (e quindi non disponibili all'utente) all'avvio del programma. In figura abbiamo una situazione in cui tutte le politiche sono state caricate correttamente.

I procedimenti per l'inserimento, la rimozione, e l'impostazione delle compatibilità tra le politiche saranno esaminati in dettaglio nei paragrafi 3.3.1, 3.3.2, 3.3.3.



### **3.1.4 Help**

In qualsiasi istante è possibile richiamare l'Help, o Guida in linea.

L'Help è scritto in formato HTML, e può essere richiamato in modalità sensibile al contesto premendo il bottone "punto di domanda" situato in alto a destra alla finestra.

### **3.1.5 Barra del menu**

Di seguito viene fornito un elenco dei menù e delle voci in essi contenute.

#### File

- Nuova configurazione
- Apri configurazione
- Salva
- Modifica configurazione
- Gestione politiche
- Esci

#### Simulazione

Sono riportati i bottoni per "navigare" nella simulazione. Tali bottoni sono quelli presenti nella tool bar. (Interrompi, avanzamento automatico, step by step avanti e indietro, vai alla fine, torna all'inizio).

E' presente anche un bottone (default layout), per reimpostare la dimensione delle viste.

#### Finestre

Sono presenti tutte le finestre dell'interfaccia grafica. Si possono selezionare o deselegionare, andando così ad aggiungerle o toglierle dall'interfaccia grafica.

#### Help

- Help
- About

## **3.2 Scrivere una nuova politica**

Prerequisiti per scrivere una nuova politica sono le conoscenze della documentazione del package scheduler e del package parametri di questo software e del linguaggio di programmazione Java. Si consiglia inoltre di studiare il codice delle politiche già disponibili per comprendere adeguatamente le funzionalità che una politica di ordinamento deve richiamare affinché l'intero software funzioni correttamente.

### **3.2.1 Politica di ordinamento**

Per scrivere una politica di ordinamento, occorre creare una classe che implementi un'interfaccia tra `Batch`, `Interattivo` o `RealTime`, a seconda che la politica da creare sia utilizzata per sistemi batch, interattivi o real time.

Esempio di una politica di ordinamento per sistemi batch:



```
public class SJF implements Batch {  
  
    ...  
  
}
```

Se la nuova politica di ordinamento è con prerilascio, allora occorre anche implementare l'interfaccia `ConPrerilascio`. Questa fornisce un metodo astratto `PCB minore(PCB pronto, PCB inEsecuzione)` che deve essere implementato per ritornare il minore tra pronto ed `inEsecuzione` secondo il criterio per il quale la politica effettua il prerilascio del *processo in esecuzione* (`inEsecuzione`). Esempio della politica di ordinamento con prerilascio `SRTF`:

```
public class SRTF extends SJF implements ConPrerilascio {  
  
    ...  
  
    public PCB minore(PCB pronto, PCB inEsecuzione) {  
        if (pronto.getIstantiDaEseguire() <  
            inEsecuzione.getIstantiDaEseguire()) {  
            return pronto;  
        }  
        return inEsecuzione;  
    }  
  
    ...  
  
}
```

Se la nuova politica di ordinamento è con priorità, allora occorre anche implementare l'interfaccia `ConPriorita`. Esempio della politica di ordinamento con priorità `Priorita`:

```
public class Priorita implements Interattivo, ConPriorita {  
  
    ...  
  
}
```

Se la nuova politica di ordinamento è con quanti di tempo, allora occorre anche estendere la classe astratta `ConQuanti`. Questa classe fornisce i metodi e i campi dati per la gestione del quanto di tempo e del time slice. Esempio della politica di ordinamento `Round Robin`:

```
public class RR extends ConQuanti implements Interattivo {  
  
    ...  
  
}
```



}

Come si può osservare, è possibile inoltre definire politiche con caratteristiche miste, ossia che implementino più interfacce.

Le funzioni fondamentali per la correttezza di una politica di ordinamento, ereditate dalle interfacce `Politica` e `PoliticaOrdinamento` sono:

- **L'inserimento.** (`public abstract void inserisci(PCB pcb)`)  
Tale metodo deve inserire `pcb` nella struttura adottata per implementare la coda dei *processi pronti*, secondo il criterio della politica che si vuole realizzare.  
Per esempio, una politica `FCFS` esegue l'inserimento in coda. Si noti che se la politica è con prerilascio, si deve valutare la necessità di prerilasciare il *processo in esecuzione*, se questo fosse presente. Il *processo in esecuzione* è ottenibile mediante l'invocazione `scheduler.getPCBCorrente()`, dove l'oggetto `scheduler` di tipo `Scheduler` è il riferimento allo scheduler utilizzato.
- **L'estrazione.** (`public abstract PCB estrai()`)  
Tale metodo si occupa di estrarre un `pcb` dalla struttura che implementa la coda dei *processi pronti*. Per esempio, nella politica `FCFS`, l'estrazione avviene in testa, per la politica `HRRN`, l'estrazione avviene in un modo più complesso.
- **L'esecuzione.** (`public abstract Istante esegui(int istantiSicuri)`)  
Tale metodo deve eseguire il *processo in esecuzione* per un tempo pari a `istantiSicuri`. Per fare ciò occorre semplicemente invocare il metodo `scheduler.incrementaTempoScheduler(istantiSicuri)`. Nel caso di politiche di ordinamento più complesse, come per esempio Round Robin, occorre anche la gestione di altre variabili. Si presti attenzione che la politica di ordinamento può anche richiamare il metodo sopra citato con un parametro inferiore di `istantiSicuri`. Sempre in riferimento alla politica di ordinamento Round Robin, infatti, è possibile infatti che il *processo in esecuzione* finisca il suo quanto di tempo prima di `istantiSicuri`.

### **3.2.2 Politica di assegnazione**

Per scrivere una politica di assegnazione, occorre creare una classe che implementi l'interfaccia `PoliticaAssegnazione`.

Esempio di una politica di assegnazione `FCFSAss`:

```
public class FCFSAss implements PoliticaAssegnazione {  
  
    ...  
  
}
```

Le funzioni fondamentali per la correttezza di una politica di assegnazione, ereditate



dalle interfacce `Politica` e `PoliticaAssegnazione` sono:

- L'inserimento. (`public abstract void inserisci(PCB pcb)`)  
Tale metodo deve inserire `pcb` nella struttura adottata per implementare la coda dei *processi bloccati* in attesa della disponibilità di una *risorsa*.  
Per esempio, una politica `FCFSAss` esegue l'inserimento in coda.
- L'estrazione. (`public abstract PCB estrai()`)  
Tale metodo si occupa di estrarre un `pcb` dalla struttura che implementa la coda dei *processi bloccati*. Per esempio, nella politica `FCFS`, l'estrazione avviene in testa.

### **3.3 Azioni richieste/permesse**

#### **3.3.1 Aggiunta di una politica**

Prima di aggiungere una politica all'applicazione, la relativa classe deve ovviamente essere già stata scritta e compilata seguendo le indicazioni del paragrafo 3.2.

Nella parte superiore del pannello "Politiche" vanno inseriti il nome della politica che vogliamo aggiungere, e il nome della rispettiva classe, per poi premere il tasto "Aggiungi". A questo punto l'applicazione caricherà la classe indicata, rilevando le interfacce da lei implementate, per determinarne il tipo. Per cui se la classe implementa l'interfaccia "PoliticaOrdinamento", verrà automaticamente aggiunta alla lista delle politiche di ordinamento; viceversa se implementa l'interfaccia "PoliticaAssegnazione" verrà aggiunta alla lista delle politiche di assegnazione.

Se la politica è di ordinamento verrà oltretutto rilevato se è real time, time sharing, con priorità, verificando se implementa rispettivamente le interfacce "RealTime", "ConQuanti", "ConPriorita".

Se i valori inseriti dall'utente non sono corretti, l'applicazione mostrerà uno dei seguenti errori:

- "Classe non trovata": Viene visualizzato quando la classe indicata non esiste.
- "La classe inserita non è una politica": Viene visualizzato se la classe indicata non implementa né l'interfaccia "PoliticaOrdinamento" né l'interfaccia "PoliticaAssegnazione".
- "Costruttore non accessibile": Viene visualizzato se vi sono problemi di accesso alla classe indicata o al suo costruttore di default (es: costruttore privato).
- "Impossibile istanziare la classe": Viene visualizzato se la classe indicata è astratta o è un'interfaccia, oppure se non possiede un costruttore di default.
- "Nome di politica già presente": Viene visualizzato se il nome scelto per la politica è già in uso per un'altra politica dello stesso tipo. Lo stesso nome può essere usato per due politiche soltanto se una è di ordinamento e l'altra è di assegnazione. È invece possibile associare alla stessa classe più politiche: ad esempio si può aggiungere una politica di nome "Coda FIFO" ed associarla alla classe "FCFS", visto che il suo comportamento è identico a quello della politica "First Come First Served" (a cui è associata la classe "FCFS").



### **3.3.2 Rimozione di una politica**

Per rimuovere una politica basta selezionarla dalla lista in cui è contenuta (pannello “Politiche”) e premere il tasto “Rimuovi”.

Così facendo ad ogni riesecuzione del programma il file corrispondente alla politica rimossa non verrà ricaricato automaticamente. Per ricaricarlo seguire la procedura descritta nel paragrafo precedente, “Aggiunta di una politica”.

Se si vuole eliminare definitivamente anche il file associato alla politica le procedure da seguire sono:

Linux:

Andare nella directory in cui è installato il programma e posizionarsi nella sottodirectory politiche.

Eliminare il file *nomefile.java* corrispondente alla politica che si vuole definitivamente eliminare.

Dalla shell, posizionarsi nella directory contenente il makefile, e digitare il comando **make elimina** seguito da invio.

Windows:

Andare nella directory in cui è installato il programma e posizionarsi nella sottodirectory politiche.

Eliminare il file *nomefile.java* corrispondente alla politica che si vuole definitivamente eliminare.

Fare doppio click sul file WinElimina.bat.

**Attenzione:** Qualora si eseguissero i comandi **make elimina** o **WinElimina.bat** senza aver rimosso dal pannello “Gestione Politiche” la politica associata al file *nomefile.java* che si vuole definitivamente eliminare, all'esecuzione del programma il pannello di gestione delle politiche visualizzerà l'errore:

Non sono state caricate con successo le seguenti politiche: <Nome\_della\_politica (tipo\_della\_politica)>

Per rimuovere tale errore, seguire la procedura descritta in “Rimozione di una politica” per cancellare la politica associata al file *nomefile.java* rimosso dal pannello di gestione delle politiche e riavviare il programma.

### 3.3.3 Gestione delle compatibilità



Fig. 3.5 Pannello Gestione Politiche (Compatibilità).

Non tutte le politiche di ordinamento e di assegnazione delle risorse sono compatibili tra loro. Per impostare le compatibilità basta passare al pannello “Compatibilità”, selezionare dalla combo box in alto una politica di ordinamento, e aggiungere o rimuovere politiche di assegnazione tra quelle compatibili con la politica di ordinamento selezionata (usando opportunamente i tasti “Aggiungi” e “Rimuovi”).

Quando viene aggiunta una politica di ordinamento, la sua lista di politiche di assegnazione compatibili è inizialmente vuota. Analogamente quando viene aggiunta una politica di assegnazione, essa inizialmente non sarà compatibile con nessuna politica di ordinamento.

Una volta eseguite le operazioni sulle politiche l'utente può scegliere se confermare (tasto “Ok”) o annullare (tasto “Annulla”) tali operazioni. Nel primo caso le modifiche avranno subito effetto (es: le nuove politiche inserite saranno immediatamente utilizzabili) e resteranno memorizzate anche dopo il riavvio dell'applicazione.

### 3.3.4 Compilare una nuova politica

Dopo aver creato una nuova politica come descritto al paragrafo “Scrivere una nuova politica”, salvare il file *nomepolitica.java* nella sottodirectory politiche presente nella directory principale in cui è installato il programma.

Dopo di che seguire i seguenti passi:

Linux:

Dalla shell, posizionarsi nella directory contenente il makefile, e digitare il comando **make aggiungi** seguito da invio.



Windows:  
Fare doppio click sul file WinAggiungi.bat.

Dopo di che basterà avviare SGPEMV2 e seguire la procedura descritta nel paragrafo “Aggiunta di una politica” affinché il programma rilevi la nuova politica aggiunta. Dopo aver fatto ciò SGPEMV2 rileverà automaticamente la nuova politica creata in tutti i riavvi futuri.

### **3.3.5 Creare la documentazione del codice sorgente**

Linux:  
Dalla shell, posizionarsi nella directory contenente il makefile, e digitare il comando **make doc** seguito da invio.

Windows:  
Fare doppio click sul file WinDoc.bat.

Entrambi comandi genereranno una sottodirectory denominata “documentazione” in cui sarà riportata la documentazione dei file sorgenti generata con javadoc.

## **4 Appendice**

### **4.1 Messaggi di errore e loro significato**

Se i messaggi riportano un'intestazione del tipo [NOTICE ER000], indicano un'eccezione che viene gestita.

Nel caso invece l'intestazione fosse del tipo [ERROR ER000], l'errore non viene gestito, causando l'uscita dal programma.

[NOTICE ER001]

Origine: GestioneFile.saveFile(ConfigurazioneIniziale,JFrame)

Messaggio: Errore durante il salvataggio del file

Note: Lanciato dal metodo java.io.ObjectOutputStream.out.writeObject(Object)

[NOTICE ER002]

Origine: GestioneFile.openFile(JFrame)

Messaggio: Il file selezionato non è valido

Note: Lanciato dal metodo java.io.ObjectInputStream.out.readObject()

[NOTICE ER003]

Origine: GuiBlueThoth.apriFileConfigurazione()

Messaggio: La politica di ordinamento di questa configurazione, non è una politica disponibile.

Note: Questo errore viene lanciato nel caso la politica di ordinamento dei processi specificata nella configurazione che si sta tentando di aprire è non disponibile.



[NOTICE ER004]

Origine: GuiBlueThoth.apriFileConfigurazione()

Messaggio: La politica di assegnazione delle risorse di questa configurazione, non è corretta.

Note: Questo errore viene lanciato nel caso la politica di assegnazione risorse specificata nella configurazione che si sta tentando di aprire è incompatibile o non disponibile.

[NOTICE ER005]

Origine: GestorePolitiche.aggiungiPolitica( String nomePol, String nomeClasse )

Messaggio: Nome di politica già presente

Note: Lanciato quando si tenta di aggiungere una politica usando un nome già presente. L'unico caso in cui viene accettato un nome già utilizzato è quando le due politiche sono una di ordinamento e una di assegnazione.

[NOTICE ER006]

Origine: GestorePolitiche.aggiungiPolitica( String nomePol, String nomeClasse )

Messaggio: Impossibile istanziare la classe

Note: Lanciato se la classe in realtà è una classe astratta o una interfaccia, oppure se nella classe manca il costruttore di default.

[NOTICE ER007]

Origine: GestorePolitiche.aggiungiPolitica( String nomePol, String nomeClasse )

Messaggio: Costruttore non accessibile

Note: Lanciato se non si riesce a creare un'istanza della classe per problemi di accesso alla classe stessa o al suo costruttore.

[NOTICE ER008]

Origine: GestorePolitiche.aggiungiPolitica( String nomePol, String nomeClasse )

Messaggio: Classe non trovata

Note: Lanciato se la classe non viene trovata.

[NOTICE ER009]

Origine: GestorePolitiche.aggiungiPolitica( String nomePol, String nomeClasse )

Messaggio: La classe inserita non è una politica

Note: Lanciato se la classe non è una PoliticaOrdinamento o una PoliticaAssegnazione.

[NOTICE ER010]

Origine: GestorePolitiche.esciConConferma()

Messaggio: Errore nel salvataggio

Note: Lanciato se avviene un errore durante il salvataggio delle impostazioni. In questo caso al riavvio dell'applicazione, le impostazioni sulle politiche torneranno allo stato precedente.

[NOTICE ER011]

Origine: GestorePolitiche.esciSenzaConferma()



Messaggio: Errore nel ripristino

Note: Lanciato se avviene un errore durante il ripristino delle configurazioni precedenti (in questo caso le operazioni effettuate non vengono annullate, ma nemmeno confermate).

Quindi le operazioni effettuate sulle politiche vengono "azzerate" soltanto al riavvio dell'applicazione.

[NOTICE ER012]

Origine: GuiBlueThoth.creaSimulazione(ConfigurazioneIniziale,boolean)

Messaggio: La simulazione non può essere creata. La durata è maggiore di 15000 istanti

## **4.2 Glossario**

**Attribuzione:** Indica i *processi* ai quali la *risorsa* è attualmente attribuita. Vedere molteplicità.

**Deadlock o stallo:** Un insieme di *processi* si trova in una situazione di stallo se ogni *processo* dell'insieme aspetta un evento che solo un altro *processo* dell'insieme può provocare. Affinché uno stallo possa verificarsi, devono valere quattro condizioni:

1. Condizione di mutua esclusione: ogni *risorsa* è assegnata ad un solo *processo*, oppure è disponibile.
2. Condizione hold and wait (prendi e aspetta): i *processi* che già hanno richiesto ed ottenuto delle *risorse* ne possono richiedere delle altre.
3. Condizione di mancanza di prerilascio: le *risorse* che sono già state assegnate ad un *processo* non gli possono essere tolte in modo forzato, ma devono essere rilasciate volontariamente dal *processo* che le detiene.
4. Condizione di attesa circolare: deve esistere una catena circolare di *processi*, ognuno dei quali aspetta il rilascio di una *risorsa* da parte del *processo* che lo segue.

**Grafo dell'allocazione delle risorse:** Grafo composto da *processi* e *risorse*. Il grafo fornisce una rappresentazione grafica dell'assegnazione dei *processi* alle *risorse*. Una freccia dal *processo* alla *risorsa* indica che il *processo* ha fatto una richiesta per la *risorsa*, oppure si trova bloccato su di essa.

Una freccia dalla *risorsa* al *processo* invece indica che il *processo* detiene la *risorsa*.

**Molteplicità:** E' il massimo numero di *processi* ai quali una *risorsa* può essere attribuita nello stesso istante. I *processi* accedono in modo mutuamente esclusivo ad una *risorsa* se questa è *non prerilasciabile* ed ha molteplicità unaria.

**Politica di assegnazione:** Una politica di assegnazione stabilisce l'ordinamento della coda di *processi bloccati* in attesa della disponibilità di una specifica *risorsa non prerilasciabile*. Influisce, quindi, nel modo in cui si intende svegliare i *processi bloccati*. Il risveglio di un *processo bloccato* per una *risorsa*, secondo la politica di assegnazione adottata, si verifica quando un *processo*, che deteneva quella *risorsa*, ne ha completato l'utilizzo.

**Politica di ordinamento:** Detto anche algoritmo di schedulazione. E' l'algoritmo che viene



utilizzato dallo scheduler per scegliere il prossimo *processo* che deve eseguire. In particolare, stabilisce l'ordinamento e l'estrazione dei *processi pronti*. Un algoritmo di schedulazione può essere diviso in due categorie: senza prerilascio (nonpreemptive), se si lascia eseguire ogni *processo* finché non rilascia volontariamente la CPU o non si blocca su un'operazione di input/output, con prerilascio (preemptive), se è stabilita una o più condizioni per le quali, il *processo* che sta eseguendo è forzato a rilasciare la CPU prima del termine della sua esecuzione. Le condizioni di prerilascio vertono principalmente nell'arrivo di un nuovo *processo* che ha priorità (si noti che la definizione di priorità caratterizza la politica) maggiore del *processo in esecuzione* oppure nell'interruzione di clock, dovuta allo scadere di una quantità prefissata di tempo. Nel software SGPEMv2, le politiche di ordinamento senza prerilascio, lasciano eseguire il *processo* finché questo non termina.

**Processo:** Un *processo* è un programma in esecuzione. All'interno di un sistema operativo, un *processo* deve mantenere i valori correnti del program counter, dei registri e delle variabili. Ogni *processo* è mantenuto all'interno dello scheduler, in una struttura dati nominata tabella dei *processi* (process table). Mediante questa struttura, lo scheduler possiede tutte le informazioni sui *processi* attivi in ogni istante. In SGPEMv2, i *processi* sono astrazioni semplificate dei *processi* reali.

**Risorsa:** con questo termine si fa riferimento a qualsiasi cosa un *processo* può utilizzare durante la sua esecuzione. Le *risorse* sono classificate generalmente in prerilasciabili e non prerilasciabili.

**Risorsa non prerilasciabile:** E' una *risorsa* che non può essere ceduta ad un altro *processo* senza provocare il fallimento dell'esecuzione in atto.

**Risorsa prerilasciabile:** E' una *risorsa* che si può togliere al *processo* che la sta usando senza provocare effetti dannosi.

**Simulazione:** Evoluzione temporale della gestione dei *processi*. Può essere vista come una funzione che prende in input la configurazione e restituisce ad ogni istante la situazione corrente (sul *processo in esecuzione*, l'allocazione delle *risorse*, ecc.).

configurazione  situazione corrente

**Sistemi batch:** Sistemi caratterizzati da un ordinamento predeterminato, prerilascio non necessario e da lavori di lunga durata e di limitata urgenza. Questo approccio riduce il cambio di contesto (context switch) tra *processi*, migliorando quindi le prestazioni.

**Sistemi interattivi:** Sistemi con grande varietà di attività e prerilascio essenziale.

**Sistemi real time:** Sistemi caratterizzati normalmente da *processi* di durata ridotta, ma elevata urgenza. Tali *processi* devono eseguire entro una scadenza. I sistemi real time si



classificano in hard real time, nei quali le scadenze sono inflessibili, e in soft real time, nei quali la scadenza è indicativa, ma non necessariamente rigida. L'importanza dell'informazione contenuta e prodotta da un *processo* che non è riuscito a terminare prima della sua scadenza, decade rapidamente o diventa, in certi casi, addirittura nulla. Un sistema real time può dover gestire *processi* periodici (che si verificano ad intervalli regolari di tempo) e *processi* aperiodici o sporadici (che si verificano in modo imprevedibile). Il prerilascio dei *processi*, talvolta, non è necessario a causa della loro breve esecuzione e perché il tempo a disposizione prima di scadere è limitato.

*Stato del processo*: Lo stato del *processo* può essere *in arrivo*, se deve ancora arrivare e per cui non condiziona la simulazione finché non viene attivato, *bloccato*, se è in attesa di una *risorsa* attualmente non disponibile, *pronto*, se è inserito in una può andare in esecuzione, *in esecuzione*, se attualmente sta eseguendo, *terminato* se ha terminato la sua esecuzione.

*Tempo di attesa*: E' il tempo durante il quale il *processo* è in stato di attesa.

*Tempo di risposta*: E' il tempo tra l'invio di un comando e l'arrivo del risultato.

*Tempo di turn around*: E' il tempo medio statistico dal momento che viene richiesto un job batch fino al momento che il job viene completato e misura quanto tempo, in media, l'utente deve aspettare per ottenere i dati in uscita.

Throughput: E' il numero dei job per ora che il sistema completa.

*Time slice*: Nelle politiche di ordinamento con quanti, (per esempio Round Robin), definisce l'intervallo di tempo che ogni *processo* può eseguire. Quando il quanto di tempo del *processo in esecuzione* è terminato (ossia è uguale al valore del time slice), il *processo* viene riposizionato nella coda dei *processi pronti*.

### **4.3 Politiche di ordinamento**

In questo paragrafo verrà fornita una descrizione delle politiche di ordinamento fornite con il software SGPEMv2.

FCFS: (First Come First Served) *Politica di ordinamento* per sistemi *Batch*.

Questo tipo di politica ordina la coda dei *processi pronti* secondo il loro tempo d'arrivo e li fa eseguire uno alla volta fino al completamento.

SJF: (Shortest Job First) *Politica di ordinamento* per sistemi *Batch*.

Tale politica richiede la conoscenza dei tempi di esecuzione infatti viene eseguito sempre il lavoro più breve. Ovviamente, non è una politica equa con i lavori non presenti all'inizio. Può comportare attesa infinita per i processi di lunga durata.

SRTF: (Shortest Remain Time First) *Politica di ordinamento* per sistemi *Batch*.

Questa *politica di ordinamento* ha un comportamento simile alla politica SJF, ma applica il prerilascio quando arriva un nuovo *processo* con tempo di esecuzione minore rispetto a



quello che sta eseguendo. Può comportare attesa infinita per i processi di lunga durata.

RR: (Round Robin) *Politica di ordinamento* tipica dei sistemi *Interattivi*.

La politica Round Robin, è una politica con ordinamento a quanti, ovvero ogni *processo* esegue per al più un quanto di tempo alla volta. Il quanto di tempo di esecuzione è più facilmente conosciuto con il nome di time slice.

RR con priorità: *Politica di ordinamento* tipica dei sistemi *Interattivi*.

Molto simile alla politica Round Robin, con la differenza che ad ogni *processo* è associato una valore di priorità (min 1, max 5). Viene schedulato sempre il *processo* con priorità più alta presente nella coda dei pronti.

RR con priorità con prerilascio (per priorità): *Politica di ordinamento* tipica dei sistemi *Interattivi*.

Applica il prerilascio per priorità alla politica Round Robin con priorità. Ovvero nel momento in cui è pronto ad eseguire un *processo* con priorità più alta di quello che è in esecuzione, quest'ultimo viene rimesso nella coda dei pronti, a favore del primo.

Priorità: *Politica di ordinamento* tipica dei sistemi *Interattivi*.

La coda dei *processi pronti* viene mantenuta rispetto alla priorità del *processo*, dalla priorità più alta alla più bassa. La priorità è attribuita all'inizio ed statica nel tempo. Riflette una misura di importanza del processo. Può comportare attesa infinita dei processi con bassa priorità.

Priorità con prerilascio: *Politica di ordinamento* tipica dei sistemi *Interattivi*.

Simile alla politica Priorità, ma con prerilascio nel caso di arrivo di un *processo* a priorità più alta rispetto a quella del *processo in esecuzione*. Può comportare attesa infinita dei processi con bassa priorità.

HRRN: (Highest Response Ratio Next) *Politica di ordinamento* tipica dei sistemi *Interattivi*.

Il rapporto di risposta di un *processo* e' definito:

rapporto risposta = (tempo atteso di esecuzione + *tempo di attesa* ) / tempo atteso di esecuzione

Il *tempo di attesa* e il rapporto di risposta sono ricalcolati ogni volta che un *processo* deve essere estratto. La scelta quindi è dinamica. Questa Viene sempre scelto il *processo* con rapporto di risposta maggiore. Questa politica, rispetto ad SJF, previene la possibile attesa infinita dei *processi* a lunga durata.

#### **4.4 Politiche di assegnazione delle risorse**

In questo paragrafo verrà fornita una descrizione delle politiche di assegnazione delle risorse ai *processi* fornite con il software SGPEMv2.

FCFS: Questa politica mantiene ordinata la coda dei *processi bloccati* in attesa della *risorsa* associata, rispetto al tempo di richiesta.

SJF: Questa politica mantiene ordinata la coda dei *processi bloccati* in attesa della *risorsa*



associata, rispetto al tempo di esecuzione di questi.

Casuale: In questa politica il *processo* che andrà in esecuzione, una volta che la *risorsa* è disponibile, sarà scelto a caso tra quelli presenti nella coda dei bloccati per quella *risorsa*. Simula il comportamento del risveglio casuale dei thread in Java.