Consiglio Nazionale delle Ricerche

RsdEditor: un'interfaccia grafica per la specifica delle risorse in un metacomputer Descrizione e guida per l'utilizzo

Ranieri Baraglia, Domenico Laforenza
Mauro Michelotti, Simone Nannetti

Rapporto
CNUCE-B4-1999-013

CNUCE

Pisa

RsdEditor : un'interfaccia grafica per la specifica delle risorse in un metacomputer Descrizione e guida per l'utilizzo

R. Baraglia, D. Laforenza, M. Michelotti, S. Nannetti

CNUCE - Instituto del Consiglio Nazionale delle Ricerche

Via S. Maria, 36 - I-56100 Pisa (Italy)

Tel. +39-050-593111 - Fax +39-050-904052

e-mail: (R.Baraglia, D.Laforenza)@cnuce.cnr.it

Abstract

Questo rapporto descrive gli obiettivi di progetto e le caratteristiche di funzionamento dell'interfaccia grafica RsdEditor per la specifica delle risorse in un metacomputer.

Utilizzando *RsdEditor* è possibile rappresentare le risorse di calcolo, definendo anche una loro strutturazione a livelli, senza la necessità di seguire un particolare ordine nelle operazioni. È possibile salvare il lavoro svolto su file e riutilizzare file contenenti specifiche definite in precedenza; inoltre, l'interfaccia consente di editare tutti gli elementi disegnati per modificarne le caratteristiche.

RsdEditor è stata progettata per facilitare il lavoro sia dell'amministratore del sistema che dell'utente del metacomputer.

Il primo utilizzerà l'interfaccia per specificare le caratteristiche delle risorse che fanno parte del metacomputer, in particolare specificherà le caratteristiche sia delle macchine che delle reti di interconnessione. Il secondo utilizzerà l'interfaccia per specificare una richiesta di risorse da sottomettere al resource manager del metacalcolatore, in modo da farsi assegnare le risorse necessarie all'esecuzione della sua applicazione.

L'idea di realizzare un'interfaccia così fatta nasce da una collaborazione dell'Istituto CNUCE con il $(PC)^2$ (Paderborn Center for Parallel Computing) dell'Università di Paderborn (Germania) nell'ambito del progetto MOL [20].

RsdEditor è stata implementata utilizzando il linguaggio $Java^{TM}$ di *Sun Microsystem* [51], in modo da garantire la sua portabilità sul maggior numero possibile di macchine.

1 Introduzione

L'uso di un insieme eterogeneo di risorse computazionali interconnesse per mezzo di una rete e viste come un singolo calcolatore parallelo a memoria distribuita è oggi una via abbastanza praticata per la soluzione di alcune classi di applicazioni complesse, anche dette *meta-applicazioni.*

In [1] Richard Freund, ricercatore presso il Naval Research Laboratory dell'Università di San Diego (USA), ha osservato che: *"i supercalcolatori vengono utilizzati per la maggior parte del loro tempo nell'esecuzione di codice per cui non sono adatti".*

Da questa osservazione nasce l'idea di partizionare le meta-applicazioni in più moduli in modo da eseguirli su un insieme eterogeneo di calcolatori, facendo si che ciascun modulo sia eseguito sulla macchina che meglio si adatta alla sua esecuzione, riducendo così i tempi di esecuzione. Per definire questo approccio, in letteratura vengono utilizzati termini quali *Distributed Heterogeneous Computing* e *Metacomputing.*

Questo approccio offre notevoli vantaggi in termini di: *potenza di calcolo* (disponibilità di potenza "aggregata" derivante dall'uso congiunto di più risorse computazionali), *espandibilità della configurazione* (il numero ed il tipo delle macchine può variare secondo le specifiche necessità delle applicazioni) e *specializzazione dei moduli* (ciascun modulo di un'applicazione si può avvalere di macchine dedicate).

Un importante obiettivo dei sistemi software usati per lo sfruttamento di questo paradigma di calcolo è quello di offrire all'utente funzionalità elaborative che gli permettano di interagire facilmente con il sistema di metacalcolo "mascherandogli" l'eterogeneità presente a vari livelli. Ogni funzione

dell'ambiente di metacalcolo quale, ad esempio, l'allocazione dei moduli paralleli sui processori, la gestione della sicurezza, dovrebbe nascondere all'utente la complessità della struttura che sta utilizzando. In tal senso, un metacalcolatore deve poter offrire, oltre ad una significativa potenza computazionale, anche la facilità d'uso.

Un *metacomputer* [4] può anche essere visto come un'ambiente dinamico composto da un pool di nodi di elaborazione, distribuiti geograficamente e connessi in rete, che possono essere aggregati dinamicamente per eseguire applicazioni complesse. Tali nodi sono, generalmente, sistemi di elaborazione indipendenti.

La realizzazione di un metacomputer passa attraverso alcune fasi:

- integrazione in rete di risorse hardware e software;
- implementazione di un *middleware* che permetta agli utenti di vedere le risorse che definiscono il metacomputer come un'unica macchina di grande potenza computazionale, rendendo loro trasparente l'uso di funzionalità di basso livello quali, ad esempio, la compilazione e l'allocazione processi-processori.

La naturale eterogeneità di un'ambiente per metacomputing porta a dare una definizione dei vari livelli di astrazione introdotti che sia il più generale possibile. Non volendo limitare a priori la tipologia di macchine e reti utilizzabili in un metacomputer, l'astrazione dall'architettura delle macchine o da particolari configurazioni software diviene un requisito necessario.

Negli ultimi anni le attività di ricerca per la realizzazione di un ambiente di metacalcolo sono state svolte nelle tre aree principali seguenti:

- rete e software di comunicazione;
- gestione della macchina virtuale;
- ambienti di programmazione eterogenei.

Rete e software di comunicazione.

Le connessioni di rete costituiscono un elemento molto importante di un metacomputer perché consentono lo scambio di dati tra le varie macchine che

lo compongono. Non ci sono vincoli sul tipo di rete utilizzato (ATM, Ethernet, ISDN, ecc.); poiché la rete è utilizzata come canale di transito delle informazioni necessarie all'esecuzione di ogni programma, questa deve avere una struttura tale da gestire possibili suoi malfunzionamenti, essere abbastanza veloce da offrire bassi tempi di latenza e permettere lo scambio di grosse quantità di dati.

Conseguentemente, anche il software per lo scambio di messaggi (PVM, MPI, ecc.) ricopre un'importanza notevole, in quanto deve realizzare lo scambio di informazioni fra le macchine sfruttando al massimo la potenzialità della rete fisica sottostante e gestendo nel modo migliore l'eterogeneità esistente tra vari sistemi di elaborazione, in maniera assolutamente trasparente all'utente.

Gestione della macchina virtuale. Dato un insieme di macchine fra loro interconnesse e costituenti un unico sistema di calcolo, occorre che ci sia qualcosa di assimilabile ad un sistema operativo che sia in grado di configurare, gestire e controllare il metacalcolatore.

Lo scopo di un siffatto software è quello di fornire all'utente una visione il più possibile monolitica del metacomputer. Questo implica che il software deve gestire il bilanciamento del carico, l'ottimizzazione dell'uso della rete, la schedulazione dei processi, la gestione di possibili fallimenti hardware/software ed ogni altro aspetto della gestione della macchina parallela che si vuol rendere trasparente all'utente.

Ambienti di programmazione. Un aspetto fondamentale nel settore del metacomputing riguarda la realizzazione di un'ambiente di sviluppo contenente un insieme di funzionalità e di strumenti software che permettano il controllo e la programmazione di un metacomputer.

Un ambiente di sviluppo ideale dovrebbe disporre di:

- strumenti per la gestione delle risorse: rientrano in questa categoria le funzionalità di gestione delle risorse della macchina virtuale, come, ad esempio, l'allocazione dei processi sui nodi di elaborazione.

- strumenti per la configurazione ed il controllo del metacomputer: si tratta di programmi, comandi o altri sistemi, utilizzabili per specificare i nodi di elaborazione costituenti la macchina virtuale, per variarne la configurazione aggiungendo o rimuovendo nodi, per richiedere l'esecuzione

o la terminazione di applicazioni, ecc.;

- strumenti per lo sviluppo di applicazioni: aiutano nella progettazione e nella scritura delle applicazioni, nella distrubuzione dei file sorgenti sui nodi del metacomputer, nella compilazione di questi ultimi, ecc.;

- strumenti per l'analisi delle prestazioni: permettono di analizzare l'esecuzione di un programma. Per raggiungere prestazioni elevate è indispensabile poter esaminare tutti i fattori che contribuiscono ad elevare il tempo di completamento (tempo di calcolo, latenza delle comunicazioni, overhead di varia natura, ecc.) di un'applicazione.

L'interfaccia *RsdEditor* rientra tra gli strumenti per la configurazione ed il controllo di un metacomputer. L'idea di realizzare un'interfaccia così fatta nasce da una collaborazione dell'Istituto CNUCE con il $(PC)^2$ (Paderborn Center for Parallel Computing) dell'Università di Paderborn (Germania) nell'ambito del progetto MOL [20, 21], con l'intento di fornire agli utenti del metacomputer uno strumento visuale con il quale specificare le loro richieste di risorse.

Il progetto MOL utilizza il sistema CCS (Computing Center Software) [37] per la gestione delle risorse. In tale sistema viene utilizzato il linguaggio di specifica RSD (*Resource and Service Description*) [44] per la descrizione delle risorse appartenenti al metacomputer gestito da CCS. Sia CCS che RSD sono stati sviluppati presso il $(PC)^2$.

RsdEditor è stata implementata utilizzando il linguaggio $Java^{TM}$ di *Sun Microsystem* [51], in modo da garantire la sua portabilità sul maggior numero possibile di macchine.

Utilizzando *RsdEditor* è possibile rappresentare le risorse di calcolo, definendo anche una loro strutturazione a livelli, senza la necessità di seguire un particolare ordine nelle operazioni. È possibile salvare il lavoro svolto su file e riutilizzare file contenenti specifiche definite in precedenza; inoltre, l'interfaccia consente di editare tutti gli elementi disegnati per modificarne le caratteristiche.

RsdEditor è stata progettata per facilitare il lavoro sia dell'amministratore del sistema che dell'utente del metacomputer. Il primo utilizzerà l'interfaccia per specificare le caratteristiche delle risorse che fanno parte del metacomputer,

in particolare specificherà le caratteristiche sia delle macchine che delle reti di interconnessione. Il secondo utilizzerà l'interfaccia per specificare una richiesta di risorse da sottomettere al resource manager del metacalcolatore, in modo da farsi assegnare le risorse necessarie all'esecuzione della sua applicazione.

Attualmente, nell'ambito del progetto MOL, l'interfaccia visuale *RsdEditor* ha lo scopo di facilitare la generazione delle specifiche delle risorse che l'utente richiede per l'esecuzione della sua meta-applicazione. Con poche semplici operazioni, l'utente specifica le caratteristiche dei nodi computazionali che intende utilizzare, definendone alcune proprietà peculiari, quali: tipo di processore, quantità di memoria disponibile, ambiente software richiesto, ecc.. La specifica di risorse dell'utente definisce il grafo delle risorse necessarie per l'esecuzione dell'applicazione; ad ogni nodo, che rappresenta una delle risorse richieste, è associata una lista di attributi specificati dall'utente mediante *RsdEditor* . La specifica di risorse effettuata in questo modo non ha lo scopo di individuare esattamente una macchina ben precisa, ma è tale da delineare le caratteristiche generali dei tipi di elaboratori di cui l'utente ha bisogno. Quindi, la specifica identifica una classe di risorse che possono essere usate per eseguire l'applicazione utente. Le classi di risorse specificate dall'utente devono poi essere identificate all'interno del pool di risorse disponibili nel metacomputer, in modo da individuare quelle che meglio soddisfano le richieste dell'utente stesso.

L'amministratore del sistema utilizza *RsdEditor* per specificare le caratteristiche del pool di risorse che fanno parte del metacomputer: egli, in questo modo, definisce il grafo delle risorse disponibili nel metacomputer. Tale specifica deve essere il più possibile completa; le informazioni che è necessario specificare sui nodi del grafo delle risorse disponibili sono, ad esempio, il numero ed il tipo dei processori, la quantità di memoria disponibile per l'esecuzione dei programmi, l'ambiente software offerto per l'esecuzione dei programmi, la classe architetturale (sequenziale, SIMD, MIMD, vettoriale, ecc.), ecc.

Per identificare le risorse richieste dall'utente fra quelle disponibili nel pool è necessario eseguire un confronto fra il grafo delle specifiche fatte dall'utente e quello descrivente le risorse presenti nel metacomputer.

La Figura 1 mostra un possibile utilizzo di *RsdEditor* , in cui l'amministratore

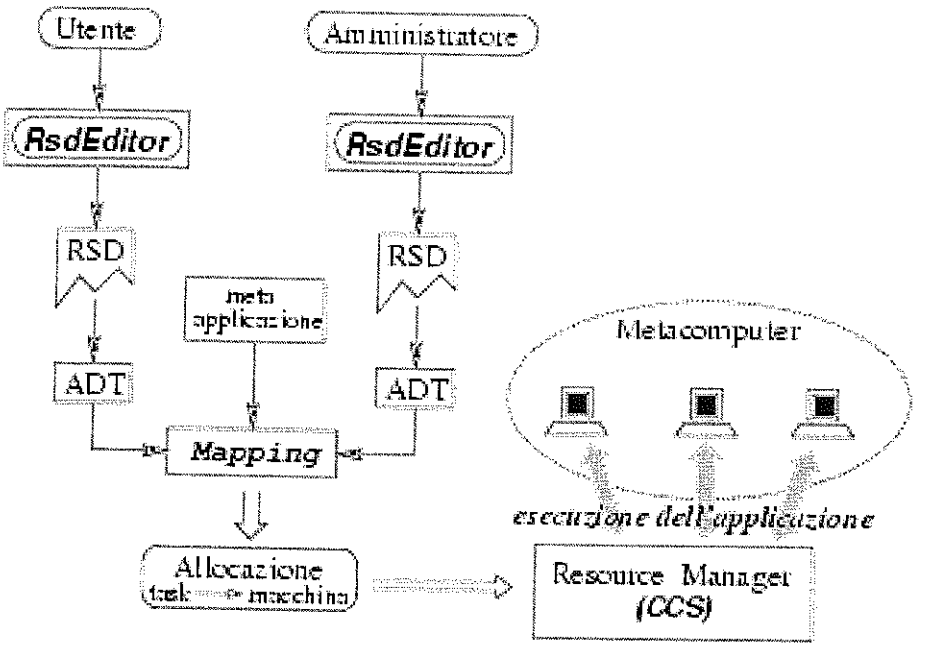


Figure 1: Esempio di utilizzo dell'interfaccia *RsdEditor* .

Figure 1: Esempio di utilizzo dell'interfaccia *RsdEditor* .

del metacomputer utilizza l'interfaccia per definire le caratteristiche del pool di risorse che costituiscono il metacomputer. In figura, **ADT** (*Abstract Data Type*) [44] indica il formato testuale in cui viene convertita la specifica RSD delle risorse. Il formato ADT potrà essere interrogato utilizzando le funzioni fornite dalla API di RSD.

La specifica delle risorse fatta dall'amministratore, permette di mettere a disposizione dell'utente una serie di attributi predefiniti da assegnare alle risorse stesse.

L'utente utilizza *RsdEditor* per specificare le proprietà delle risorse che richiede per eseguire la propria meta-applicazione, definendo nodi di un grafo e specificando degli attributi ad essi associati.

Le due specifiche di risorse definite da amministratore ed utente vengono utilizzate da un algoritmo di mapping per decidere quale insieme di risorse assegnare, fra quelle disponibili nel metacomputer, all'esecuzione della meta-applicazione dell'utente.

Una volta che siano disponibili sia il grafo della richiesta delle risorse che il grafo di specifica delle caratteristiche dei moduli dell'applicazione, diventa possibile utilizzare un algoritmo di mapping costruito per eseguire un'allocazione fra i vari task dell'applicazione e le macchine a disposizione nel metacomputer, svincolando l'utente dal problema di allocare i task sulle varie risorse. Per tale scopo, possono essere utilizzati un ampio insieme di algoritmi di mapping che rappresentano con grafi sia l'applicazione che il metaelaboratore presenti in letteratura, quali ad esempio MARS [23], HMM [57], Suboptimal Heterogeneous Mapping [58].

L'allocazione task→risorsa generata dall'algoritmo di mapping dell'ambiente di metacomputing viene utilizzata dal gestore delle risorse del metacomputer, che in Figura 1 è identificato dal sistema Computing Center Software (CCS) [37], usato nel progetto MOL, per avviare e controllare l'esecuzione dei moduli della meta-applicazione.

2 Obiettivi di progetto

L'interfaccia visuale *RsdEditor* è stata sviluppata per rispondere a requisiti ben precisi che non sono presenti in interfacce gia' esistenti [59, 60, 61, 62, 63, 64, 65] Pertanto, uno degli obiettivi di progetto dell'interfaccia grafica è stato quello di coniugare la semplicità e rapidità d'uso con il rigore di un linguaggio di specifica delle risorse quale il linguaggio RSD.

Di seguito sono esposti gli obiettivi perseguiti nella progettazione ed implementazione dell'interfaccia.

2.1 Supporto al linguaggio RSD

L'interfaccia *RsdEditor* nasce con l'intento di fornire un supporto visuale ed intuitivo alla generazione di file, scritti in sintassi RSD, per la specifica delle richieste di risorse.

RsdEditor è progettata per restituire come output un file descrivente le risorse specificate graficamente tramite l'interfaccia; tale file è in formato ASCII e rispetta la sintassi RSD.

2.2 Semplificare la definizione di un metacomputer

L'interfaccia *RsdEditor* risulta semplice da utilizzare e riduce gli sforzi degli utilizzatori nel momento della specifica delle risorse necessarie per l'esecuzione della propria applicazione. Con pochi click del mouse, è possibile completare la rappresentazione grafica delle risorse e generare il file di specifica in sintassi RSD.

2.3 Supporto per l'amministratore del metacomputer

RsdEditor semplifica l'attività dell'amministratore del sistema nella definizione delle caratteristiche delle macchine che fanno parte del metacomputer.
L'amministratore del metacomputer ha la necessità di specificare completamente ed esaurientemente tutte le risorse che fanno parte del metacomputer. L'interfaccia gli consente, quindi, di realizzare graficamente una descrizione dettagliata delle risorse e di memorizzare tale descrizione in un file.

2.4 Supporto per l'utente del metacomputer

L'utente generico è colui che ha i maggiori problemi nello specificare una richiesta di risorse, perché non è sempre intuitivo specificare in maniera rigorosa quelle risorse che sono coinvolte nell'esecuzione di un'applicazione. Da questo punto di vista, *RsdEditor* aiuta l'utente nella specifica della propria richiesta e gli fornisce una soluzione semplice al problema della specifica delle risorse in sintassi RSD. Utilizzando uno strumento completamente visuale, l'utente riesce a realizzare richieste di risorse anche molto complesse (si pensi ad una topologia di interconnessione quale un *grid*), specificando soltanto le caratteristiche delle risorse che compongono la sua richiesta.

2.5 Editing degli elementi

Le funzionalità di editing degli elementi sono importanti perché permettono all'utilizzatore di modificare le caratteristiche specificate per una risorsa senza per questo doverle specificare di nuovo per intero. Inoltre, questa funzionalità allinea *RsdEditor* allo standard attuale degli strumenti di disegno di grafi. L'interfaccia consente di salvare su disco tutto il lavoro di specifica, in modo

da poterlo riutilizzare in un secondo momento esattamente come era stato prodotto la prima volta. Nel caso che venga modificato un oggetto letto da un file, è sufficiente effettuare un nuovo salvataggio per ottenere la versione aggiornata della specifica di risorse.

2.6 Output in formato ASCII e sintassi RSD

L'interfaccia produce in output un file in formato ASCII contenente la descrizione della specifica di risorse in sintassi RSD. In questo modo si rende il formato facilmente trasportabile da una piattaforma ad un'altra, dato che si tratta comunque di un file testuale scritto in sintassi RSD. Inotre, questo accorgimento permette all'utilizzatore di modificare il file a mano nel caso in cui l'interfaccia non si fosse rivelata abbastanza flessibile e potente per le sue necessità. Questo aiuta soprattutto l'amministratore del metacomputer, il quale può utilizzare un qualsiasi editor testuale per modificare tale file.

2.7 Specifica *bottom-up* e *top-down* delle risorse

Una caratteristica fondamantale di *RsdEditor* è la possibilità di specificare le richieste di risorse sia seguendo un approccio *top-down* che un approccio *bottom-up*. Questa particolarità garantisce una buona elasticità all'interfaccia e non vincola l'utilizzatore a seguire un approccio particolare. L'utente può specificare le proprie richieste di risorse cambiando in qualsiasi momento il suo approccio, da top-down a bottom-up e viceversa.

2.8 Organizzazione a livelli delle risorse

RsdEditor , grazie alla possibilità di specificare le risorse sia bottom-up che top-down, prevede la possibilità di dare un'organizzazione gerarchica alle risorse. Questo permette all'amministratore di fornire una descrizione delle risorse che rifletta la loro ripartizione fra i vari siti che fanno parte del metacoputer. Per l'utente, questa caratteristica fa si che egli possa organizzare a livelli le proprie specifiche, consentendogli una notevole flessibilità nella definizione delle proprie richieste di risorse.

2.9 Configurazione mediante file

L'architettura dell'interfaccia prevede l'esistenza di una serie di file di configurazione che sono letti al momento dell'attivazione dell'esecuzione dell'interfaccia e che permettono di personalizzarla in base alle esigenze del singolo utente. Mediante tali file vengono rese disponibili alcune predefinizioni di risorse; si tratta di specifiche di risorse già definite che ogni utente può utilizzare nella specifica della propria richiesta di risorse per velocizzare il proprio lavoro. È inoltre prevista una serie di file che permettono di attivare l'esecuzione dell'interfaccia utilizzando messaggi espressi in varie lingue, in modo da favorire l'utilizzo di *RsdEditor*.

2.10 Estendibilità dell'interfaccia

RsdEditor non è uno strumento "chiuso", ma è possibile estenderne le funzionalità grazie alla sua struttura modulare. Sarà possibile, in futuro, aggiungere nuove funzionalità che rendano *RsdEditor* uno strumento sempre più potente e che venga incontro alle nuove esigenze degli utenti. L'aggiunta di una nuova funzionalità richiederà soltanto di scrivere il modulo che la implementa ed integrarlo con il resto dell'applicazione con poche semplici modifiche.

2.11 Implementazione mediante un linguaggio portabile

Un'ulteriore caratteristica di *RsdEditor* è il linguaggio usato per la sua implementazione. Èstato usato un linguaggio che avesse una discreta diffusione e che garantisse una buona portabilità dell'applicazione; il linguaggio $Java^{TM}$ di *Sun Microsystems* ci è sembrato quello più adatto perché consente di compilare l'applicazione una sola volta e di eseguirla ovunque sia installata una macchina virtuale Java (JVM) compatibile. Inoltre, tale linguaggio sta riscuotendo un notevole successo, e questo fa prevedere un ampliamento del numero di JVM disponibili, permettendo di utilizzare *RsdEditor* su un numero sempre maggiore di piattaforme di calcolo.

3 Funzionamento dell'interfaccia

RsdEditor permette di specificare richieste di risorse in modo intuitivo. Questo capitolo descrive il funzionamento dell'interfaccia mostrandone le potenzialità ed evidenziando le semplificazioni che essa introduce per l'utente.

3.1 Avvio dell'interfaccia

Come già scritto, *RsdEditor* è stato implementato utilizzando il linguaggio di programmazione Java di Sun Microsystem. Per avviare l'editor è necessario utilizzare una Java Virtual Machine (JVM) per interpretare le classi (*bytecode*) generate in fase di compilazione dell'applicazione. Indipendentemente dalla piattaforma che si adotta per l'esecuzione dell'interfaccia, il comando di avvio di una Java Virtual Machine è

```
java <nome-classe>
```

mediante il quale viene eseguita la classe indicata con `<nome-classe>` dalla JVM installata sulla piattaforma che stiamo utilizzando.

Per rendere l'uso dell'interfaccia più semplice, sono stati definiti due semplici script di shell che attivano l'esecuzione di *RsdEditor*. Lo script `Run.bat` attiva l'esecuzione dell'applicazione sulle macchine che usano sistemi operativi Microsoft Windows (Windows 95/98 e Windows NT), mentre lo script `run.sh` attiva l'esecuzione dell'applicazione su macchine che usano sistemi operativi di tipo Unix (Linux e Solaris).

In entrambi i casi, il comando utilizzato è:

```
java RsdEditor.RsdEditor
```

Le classi sono state raccolte in un *package*[1] chiamato `RsdEditor` e la classe principale dell'applicazione si chiama `RsdEditor`.

In Figura 2 è mostrata la finestra visualizzata immediatamente dopo l'attivazione dell'esecuzione dell'interfaccia.

Agendo sul combobox[2] Select Language (vedi Figura 2), l'interfaccia offre la possibilità di selezionare la lingua usata per la visualizzazione sia dei suoi

[1] Un *package* è una collezione di classi, interfacce e sottopackage tra loro collegati.

[2] Un *combobox* è un elenco a discesa in cui è possibile selezionare una sola funzione.

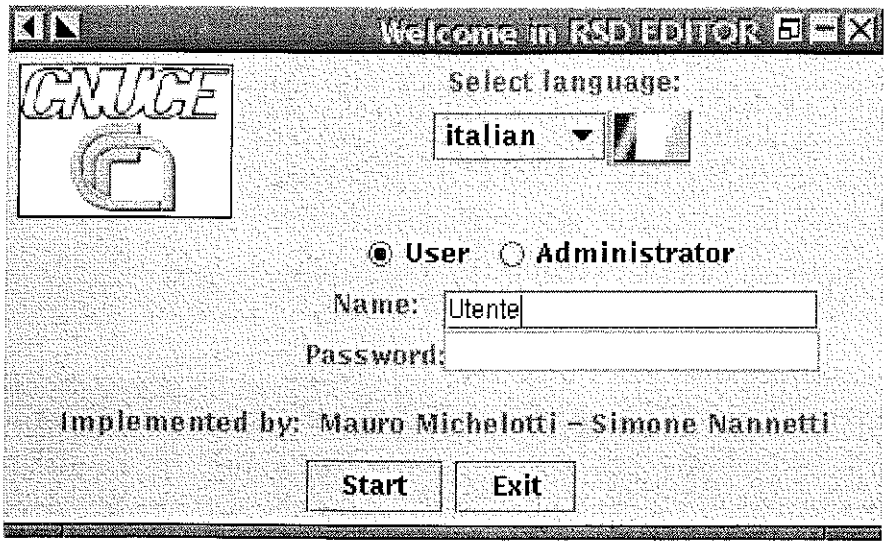


Figure 2: *RsdEditor* : finestra di attivazione dell'esecuzione.

elementi che dei messaggi utente. Al momento sono disponibili due versioni, una in italiano ed una in inglese; in futuro sarà possibile estendere tale disponibilità aggiungendo i file di configurazione dell'interfaccia tradotti nella lingua desiderata.

RsdEditor può funzionare in due diverse modalità:

- Administrator, in cui l'amministratore del metacomputer (riconosciuto grazie all'inserimento di una password nel campo specifico) definisce le caratteristiche delle macchine a disposizione degli utilizzatori;
- User, in cui si consente ad ogni utente di specificare le caratteristiche delle risorse di cui ha bisogno per l'esecuzione della propria applicazione.

Nel caso in cui si selezioni la modalità Administrator viene abiliato il campo Password, nel quale deve essere inserita la password per l'identificazione dell'amministratore del metacomputer, mentre il campo Name viene impostato ad un valore di default. Nel caso in cui si selezioni la modalità User viene abilitato il solo campo Name, in cui l'utente può inserire la propria username, mentre il campo Password viene disabilitato.

Una volta scelta la modalità di avvio ed impostati i parametri richiesti, premendo il pulsante Start viene mostrata la finestra principale di RsdEditor.

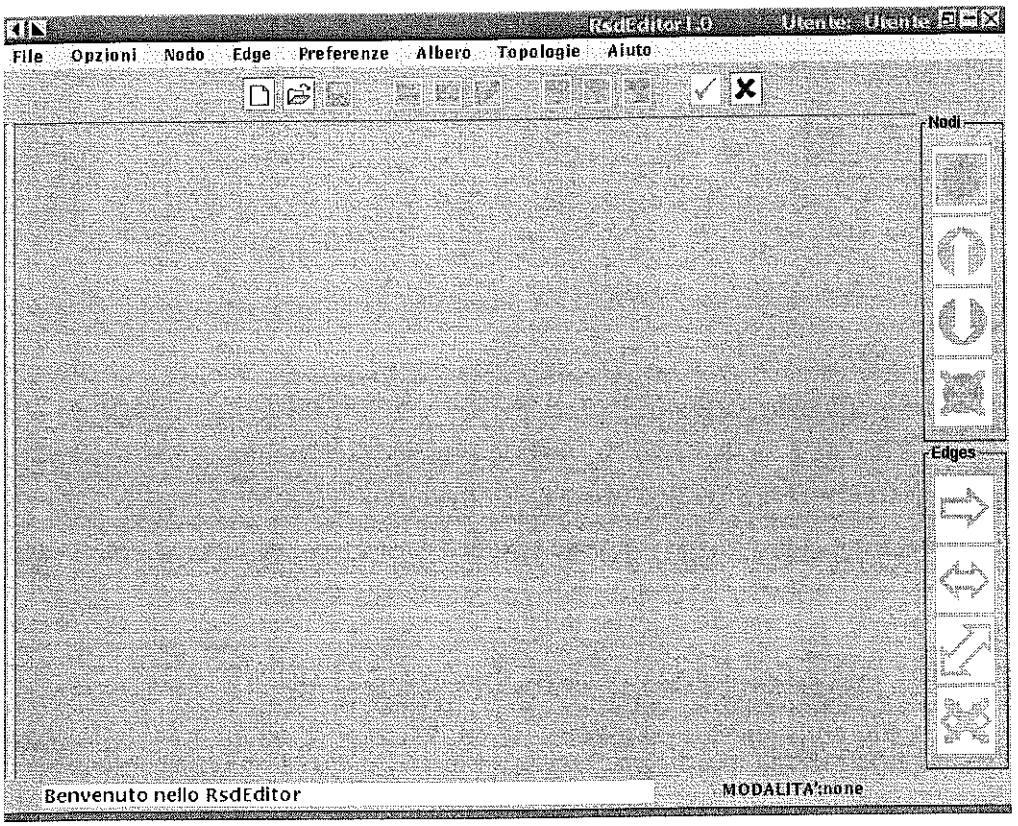


Figure 3: *RsdEditor* : Finestra iniziale.

3.2 Elementi visuali dell'interfaccia

La Figura 3 mostra la finestra visualizzata all'inizio di una sessione di lavoro. Descriviamo brevemente i suoi elementi principali; una descrizione dettagliata del loro funzionamento e del loro utilizzo sarà fornita più avanti in questo Capitolo.

Prima di tutto, notiamo che sulla barra del titolo dell'applicazione è indicato il nome dell'utente specificato usando la finestra di Figura 2 (in tale caso è stata inserita la username "*Utente*").

La barra dei menù contiene i seguenti menù:

- File contiene le funzioni (vedi Figura 4) che consentono di creare un nuovo file di specifica, di aprirne uno esistente, di memorizzare quello corrente, di chiuderlo ed infine di terminare l'esecuzione dell'interfaccia.

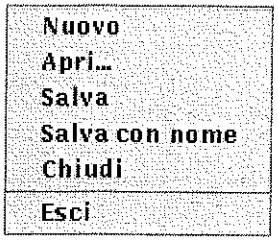


Figure 4: *RsdEditor* : menù File.

Figure 5: *RsdEditor* : menù Opzioni.

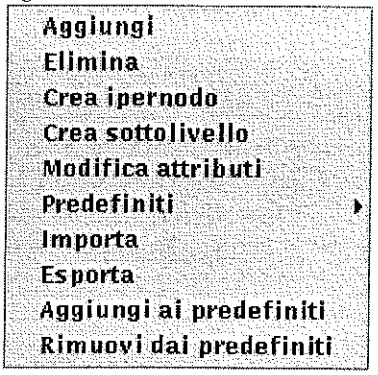


Figure 6: *RsdEditor* : menù Nodo.

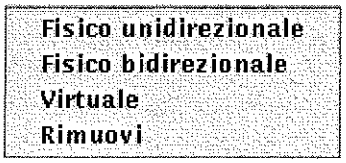


Figure 7: *RsdEditor* : menù Edge.

- Opzioni contiene funzioni (vedi Figura 5) che permettono di annullare la modalità di creazione degli oggetti, di visualizzare il file RSD creato fino a questo momento e di effettuare un refresh degli oggetti disegnati dall'utilizzatore.

- Nodo permette di inserire un nuovo nodo, di eliminarlo, di creare un ipernodo (cioè un nodo che contiene ricorsivamente altri nodi), di creare un sottolivello (per la specifica di risorse in modo top-down) e di modificare gli attributi assegnati ad un nodo (vedi Figura 6). La funzione Predefiniti è un sottomenù che contiene alcune specifiche di risorse già definite ed utilizzabili dall'utente nella costruzione della sua richiesta di risorse. È possibile aggiungere e rimuovere le voci dal sottomenù Predefiniti utilizzando le funzioni Aggiungi ai predefiniti e Rimuovi dai predefiniti. Infine, è possibile importare ed esportare parti di una specifica di risorse usando le funzioni Importa ed Esporta secondo le modalità esposte nel paragrago 3.14.

- Edge permette di inserire un edge all'interno della specifica (vedi Figura

7). Si possono inserire edge fisici, sia unidirezionali che bidirezionali, ed edge virtuali. Infine è possibile rimuovere un edge selezionando il comando Rimuovi.

Figure 8: *RsdEditor* : finestra usata per la specifica delle caratteristiche grafiche di un nodo.

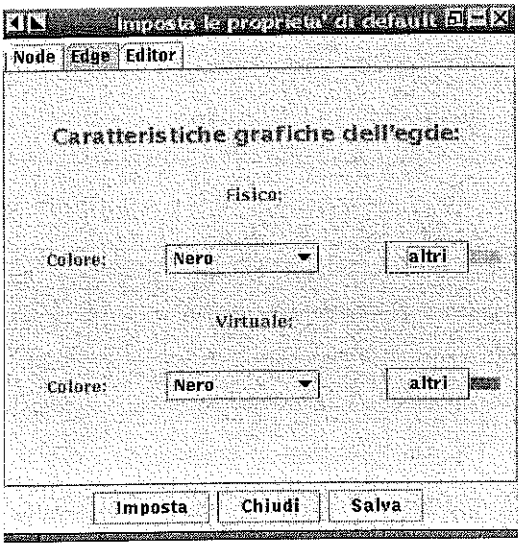


Figure 9: *RsdEditor* : finestra usata per la specifica delle caratteristiche grafiche di un edge.

- Preferenze permette di impostare alcune caratteristiche dell'editor, quali, ad esempio, le caratteristiche grafiche dell'interfaccia e le caratteristiche iniziali dei nodi e degli edge. Una volta selezionata la funzione Imposta viene aperta una finestra composta da tre pannelli[3] sovrapposti, visualizzabili selezionando, con un click del mouse, la funzione corrispondente.

  Il pannello Node permette di impostare le caratteristiche grafiche di default dei nodi che saranno visualizzati dal programma di editing (vedi Figura 8). Utilizzando le funzionalità di tale pannello è possibile definire il tipo di rimpimento dei nodi (Vuoto o Pieno), la loro forma (Ovale o Rettangolare), le loro dimensioni espresse in punti (utilizzando la barra Grandezza), il loro colore (selezionabile da una lista di colori predefiniti o definibile dall'utente utilizzando la finestra che *RsdEditor* apre in seguito

[3]Un *pannello* è un contenitore di componenti grafici Java che rappresenta una sezione di una finestra e che può contenere, a sua volta, altri componenti grafici Java.

alla selezione del pulsante Altri) ed, infine, il tipo di carattere con cui visualizzare il nome del nodo.

Il pannello Edge permette di impostare le caratteristiche grafiche di default per gli edge (vedi Figura 9). È possibile specificare il valore di default del colore con cui saranno disegnati sia gli edge *fisici* che quelli *virtuali*.

Figure 10: *RsdEditor* : finestra usata per la specifica delle proprietà generali dell'editor.

Il pannello Editor permette di specificare i valori di default, per il colore e per il tipo di carattere, utilizzati nella visualizzazione dei componenti grafici di *RsdEditor* (vedi Figura 10).

Infine, i pulsanti posizionati nella parte bassa della finestra permettono, rispettivamente, di impostare i valori specificati per la sessione in esecuzione, di chiudere la finestra e di memorizzare i valori specificati nel file di configurazione dello editor.

- Albero permette la gestione dell'albero di visualizzazione delle risorse specificate (vedi Figura 12). Tale albero viene mostrato quando viene aperto un file di descrizione delle risorse; usando le funzioni del menù è possibile mostrare/nascondere l'albero, espandere ogni elemento e chiudere completamente l'albero. Tali funzionalità saranno viste con maggior

Figure 11: *RsdEditor* : menù Preferenze.

Figure 12: *RsdEditor* : menù Albero.

Figure 13: *RsdEditor* : menù Topologie.

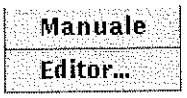


Figure 14: *RsdEditor* : menù Aiuto.

dettaglio nel seguito del Capitolo.

- Topologie permette di inserire, nella richiesta di risorse, nodi complessi che usano al loro interno le topologie di interconnessione Ring, Grid, Star, Torus. Queste quattro topologie sono state ritenute di ampio utilizzo per coloro che intendono utilizzare un cluster di workstation o di personal computer.

- Aiuto fornisce un manuale *on-line* che descrive brevemente come utilizzare le principali funzioni di *RsdEditor* (vedi Figura 14).

Immediatamente al di sotto della barra dei menù ci sono quattro gruppi di pulsanti che hanno lo scopo di velocizzare alcune fra le operazioni ritenute più frequenti. In tale barra degli strumenti, come in quella posta nella parte destra della finestra principale di RsdEditor, alcuni pulsanti sono disattivati; questo perchè le loro funzioni sono abilitate soltanto quando l'intefaccia ha un file di specifica aperto (vedi Figura 21).

Nel primo gruppo troviamo alcune funzioni analoghe a quelle del menù File, cioè apertura di un file esistente, creazione di un nuovo file e salvataggio del file stesso (vedi Figura 15).

Nel secondo gruppo troviamo tre pulsanti; il primo permette di annullare la modalità di disegno, il secondo permette visualizzare, in una finestra a parte, il codice RSD generato dalla specifica di risorse disegnata fino a questo momento, mentre il terzo esegue un redraw della finestra (vedi Figura 16). Tutti questi pulsanti sono scorciatoie per funzioni presenti nel menù Opzioni

Figure 15: *RsdEditor* : primo gruppo di funzioni della barra degli strumenti.

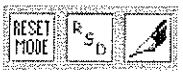


Figure 16: *RsdEditor* : secondo gruppo di funzioni della barra degli strumenti.

Figure 17: *RsdEditor* : terzo gruppodi funzioni della barra degli strumenti.

Figure 18: *RsdEditor* : quarto gruppo di funzioni della barra degli strumenti.

e la loro funzione risulterà più chiara grazie a quanto esposto nel seguito del capitolo.

Nel terzo gruppo abbiamo tre pulsanti di gestione dell'albero descrivente le risorse specificate (vedi Figura 17). Le funzioni dei tre pulsanti sono analoghe a quelle del menù Albero.

Infine, il quarto gruppo prevede due pulsanti che servono, rispettivamente, per chiudere il file di specifica corrente e per chiudere l'interfaccia (vedi Figura 18).

Nella parte destra della finestra principale di *RsdEditor* troviamo due gruppi di pulsanti funzionali che permettono di velocizzare le principali funzioni di disegno dell'interfaccia. Anche in questo caso i pulsanti vengono disattivati all'attivazione dell'esecuzione di *RsdEditor* , e sono riattivati quando viene aperto un file di specifica (vedi Figura 21).

Il primo gruppo di pulsanti, chiamato Nodi, permette di:

- creare un nuovo nodo;
- creare un'ipernodo, cioè creare un livello gerarchico superiore contenente tutti gli oggetti appartenenti al livello gerarchico corrente, in modo da realizzare una definizione *bottom-up* della specifica di risorse;
- creare un sottolivello, cioè trasformare un nodo in un ipernodo, in modo da aggiungere risorse ad un livello gerarchico inferiore per consentire una definizione *top-down* della specifica di risorse;

- cancellare un nodo.

Il secondo gruppo di pulsanti, chiamato Edges, permette di:

- creare un edge fisico unidirezionale;
- creare un edge fisico bidirezionale;
- creare un edge virtuale;
- cancellare un edge, sia fisico che virtuale.

Questi pulsanti svolgono funzioni analoghe a quelle presenti nei menù Nodo ed Edge appartenenti alla barra dei menù.

Da notare che tutti i pulsanti (sia quelli della barra superiore che di quella laterale) sono dotati di **tooltip**, cioè di una piccola finestra che *RsdEditor* visualizza quando, con il mouse, ci si posiziona su di un pulsante. Tale finestra contiene una breve descrizione della funzione associata al pulsante. Questo accorgimento consente all'utente un'interazione più rapida con *RsdEditor* e semplifica la fase di apprendimento delle funzionalità dell'interfaccia stessa.

Nella parte inferiore della finestra principale di *RsdEditor* è stata prevista una *barra di stato*. Essa contiene un campo in cui vengono riportati i messaggi, sia di errore che informativi, per l'utente. Il campo MODALITÀ indica la modalità di disegno correntemente usata; ad esempio, viene indicato se il prossimo click del mouse provoca la creazione di un nuovo nodo o la cancellazione di un oggetto.

La parte centrale della finestra principale di *RsdEditor* , che sarà chiamata **workspace**, è quella che contiene l'area di disegno ed è attiva quando si sta definendo una specifica di risorse.

3.3 Creazione di una nuova specifica

La creazione di un nuovo file di specifica delle risorse può avvenire in due modi: selezionando la funzione Nuovo dal menù File oppure premendo il pulsante

rappresentato dall'icona ed indicato dal tooltip con la scritta "*Crea un nuovo file*". Ovviamente, il risultato che si ottiene è lo stesso, cioè l'apertura della finestra di Figura 19, in cui si chiede di specificare il nome da assegnare al file.

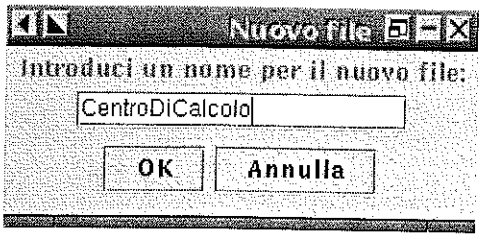


Figure 19: *RsdEditor* : finestra per l'assegnazione del nome ad un file.

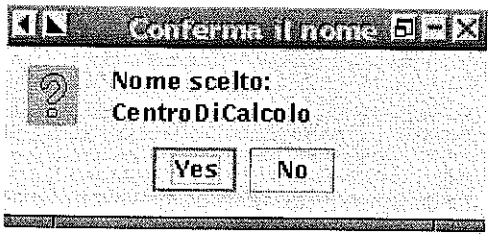


Figure 20: *RsdEditor*: finestra di conferma del nome assegnato ad un file.

Eseguita tale operazione, premendo il pulsante OK viene visualizzata la finestra di Figura 20 che permette di confermare le informazioni prima specificate.

Definito il nome del file, viene visualizzata la finestra di Figura 21, in cui, come si può vedere, tutti i pulsanti sono adesso attivi e, quindi, è possibile iniziare la definizione della nostra specifica delle risorse.

3.4 Aggiunta di un nodo

In sintassi RSD, un nodo rappresenta una risorsa. Per inserire un nodo nella nostra richiesta di risorse si può utilizzare la funzione Aggiungi del menù Nodo oppure si può usare il pulsante a cui è associata l'icona e che è indicato dal corrispondente tooltip con la scritta "*Aggiungi un nuovo nodo*".

Eseguita la scelta, la barra di stato segnala di aver rilevato l'evento mostrando il messaggio

"*Seleziona un punto nel workspace per aggiungere un nodo*"

e, contemporaneamente, indicando che l'interfaccia si trova nella modalità di disegno *"Aggiungi nodo"*.

In questo caso, la barra di stato indica l'azione che l'utente deve compiere per completare l'operazione richiesta; in questo caso l'utente deve selezionare con il mouse un punto nel workspace per indicare la posizione in cui disegnare

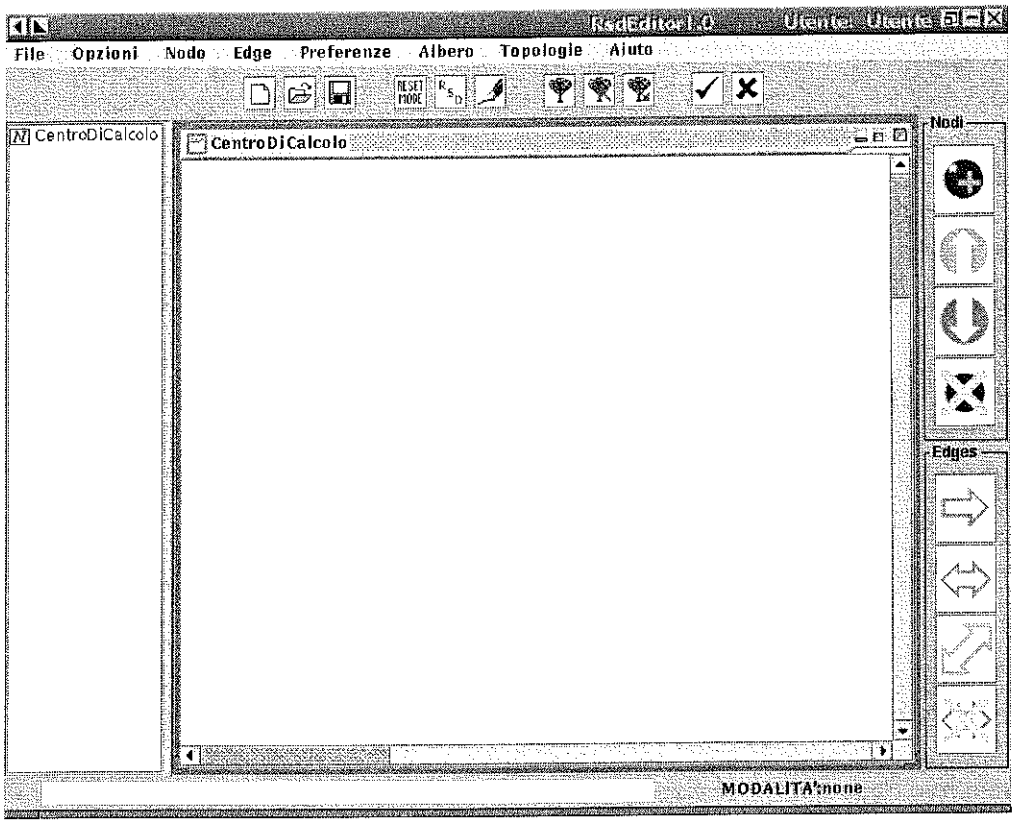


Figure 21: *RsdEditor* : finestra iniziale con un file di specifica vuoto.

il nodo.

Dopo aver selezionato un punto nel workspace, si apre automaticamente la finestra di Figura 22; essa è composta da tre pannelli sovrapposti che si visualizzano selezionando le rispettive voci.

- Proprietà grafiche: usando la finestra di Figura 22 è possibile impostare le proprietà grafiche del nodo. Si può scegliere il tipo di riempimento del nodo (vuoto o pieno), la forma (ovale o rettangolare), la dimensione espressa in punti (variabile fra 30 e 100), il colore del nodo (da scegliere in una lista di colori predefiniti o da creare mediante il pulsante Altri) ed il font da utilizzare per visualizzare il nome del nodo.

- Attributi RSD: con la finestra di Figura 23 è possibile assegnare degli

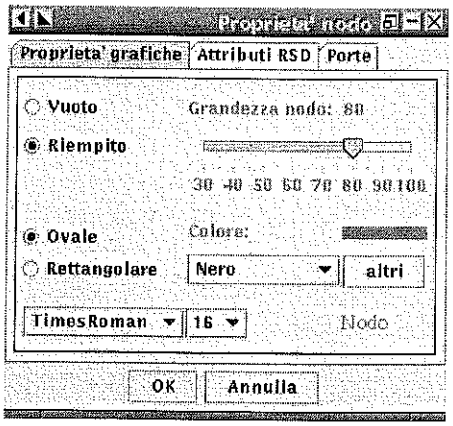


Figure 22: *RsdEditor*: finestra per l'impostazione delle proprietà grafiche di un nodo.

attributi al nodo che si sta disegnando. Si può assegnare un nome, che sarà visualizzato nel workspace al di sotto del nodo, per identificare l'oggetto in modo univoco; *RsdEditor* propone automaticamente un nome che può essere modificato dall'utente. Inoltre, è possibile specificare degli attributi scegliendo fra quelli predefiniti o definendone di propri. Il combobox di sinistra contiene le categorie di attributi disponibili; selezionando un elemento da talc lista, nel combobox di destra vengono inseriti gli attributi della categoria selezionata. Quindi si può aggiungere la coppia prescelta premendo il pulsante Aggiungi; l'attributo selezionato comparirà nella tabella sottostante.

Premendo il pulsante Nuovo è possibile specificare un attributo personalizzato, come mostrato in Figura 24. Utilizzando il combobox "editabile" Attributo, è possibile selezionare una delle categorie di attributi predefinite oppure specificarne una nuova; quindi si può inserire il valore desiderato per l'attributo. La coppia <*attributo,valore*> specificata sarà inserita nella tabella degli attributi dell'oggetto; se l'utilizzatore ha avviato l'interfaccia in modalità Amministratore di sistema, allora *RsdEditor* aggiungerà in via definitiva le categorie ed i valori specificati effettuando, in modo automatico, una scrittura sui corrispondenti file di configurazione.

- Porte: utilizzando la finestra di Figura 25 è possibile specificare una

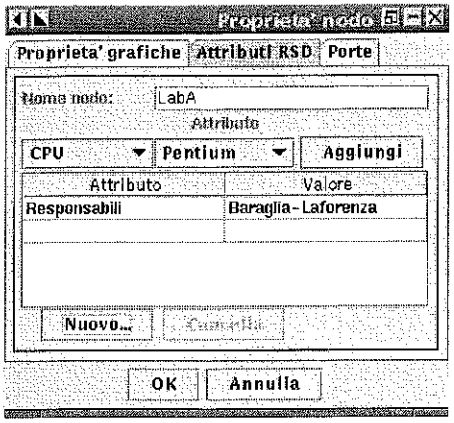


Figure 23: *RsdEditor* : finestra per l'assegnazione degli attributi RSD di un nodo.

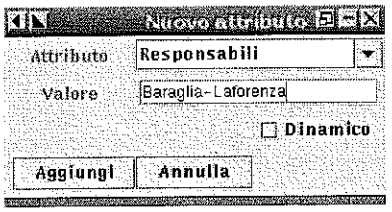


Figure 24: *RsdEditor* : finestra per la specifica di un nuovo attributo.

porta, cioè un'interfaccia del nodo verso l'esterno. La sintassi RSD richiede che in un nodo sia definta almeno una porta affinchè sia possibile collegarlo, tramite un edge, ad un altro nodo. *RsdEditor* permette di specificare una porta premendo il pulsante Aggiungi, con il quale viene visualizzata la finestra di Figura 26. Una volta definita una porta è possibile assegnarle degli attributi, seguendo un meccanismo analogo a quanto visto per il nodo.

Infine, si possono confermare le caratteristiche specificate per il nodo premendo il pulsante OK, in modo che quanto specificato venga visualizzato nel workspace così come è stato definito. In Figura 27, è mostrato un esempio di specifica di tre nodi.

Si noti che ogni nodo viene dotato, al momento della sua creazione e delle successive modifiche, di un tooltip che ne elenca gli attributi; per attivare il tooltip di un nodo è sufficiente spostare il puntatore del mouse sull'oggetto ed attendere per qualche attimo che la finestra sia visualizzata. Questo permette di controllare rapidamente le caratteristiche associate a ciascun nodo, senza la necessità di riaprire la finestra di Figura 22, utilizzabile per la modifica della caratteristiche grafiche del nodo. La libreria *Swing* fornisce una classe standard chiamata `com.sun.java.swing.JToolTip` che permette di inserire tooltip di una sola riga sugli oggetti grafici; per ottenerne una versione multilinea abbiamo utilizzato una classe esterna a tale libreria chiamata

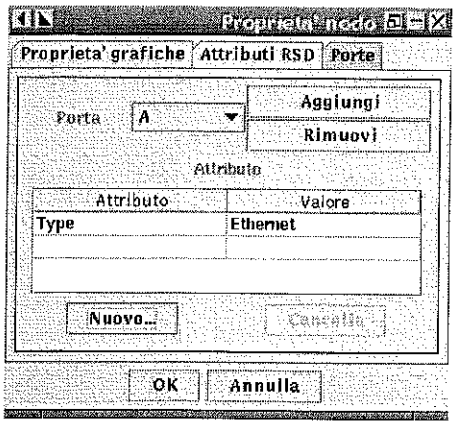


Figure 25: *RsdEditor* : finestra per la definizione di una porta in un nodo secondo la sintassi RSD.

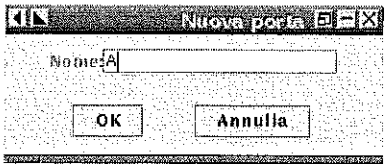


Figure 26: *RsdEditor*: finestra per la specifica di una nuova porta.

`MultiLineToolTipUI`.

3.5 Aggiunta di un edge fisico

Dopo l'inserimento di due o più nodi (Figura 27) è possibile inserire anche uno o più edge, che indicano un collegamento fisico tra due oggetti di tipo `NODE`. La sintassi RSD ammette sia edge fisici *unidirezionali* (cioè che hanno un nodo sorgente ed uno destinazione) che edge fisici *bidirezionali* (in cui entrambi i nodi possono svolgere il ruolo di sorgente/destinazione). Per inserire un'edge fisico sono stati implementati due modi: si possono utilizzare le funzioni Fisico unidirezionale e Fisico bidirezionale del menù Edge (vedi Figura 7) oppure si possono utilizzare i pulsanti:

-  per inserire un edge unidirezionale;
-  per inserire un edge bidirezionale.

Anche in questo caso, la barra di stato indica l'azione che l'utente deve compiere per completare l'operazione di inserimento di un'edge, cioè cliccare, con il tasto sinistro del mouse, sui due nodi che si vogliono collegare tramite

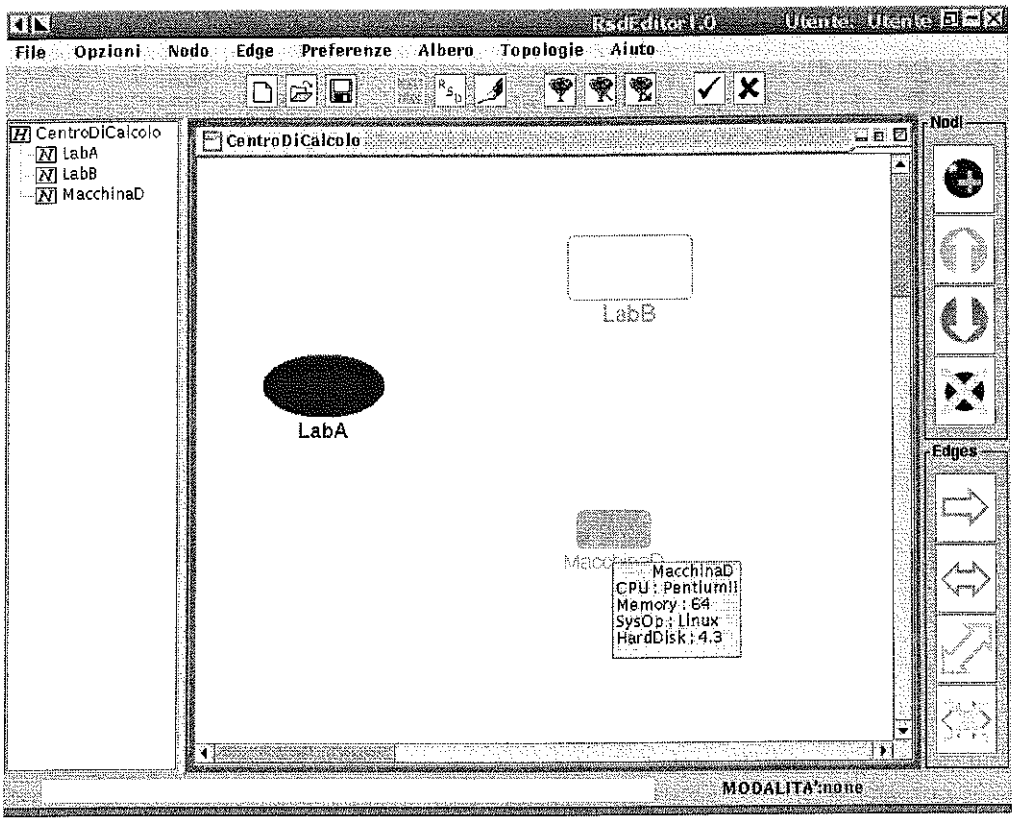


Figure 27: *RsdEditor* : esempio di specifica di alcuni nodi.

lo edge. Se lo edge è unidirezionale, allora il primo nodo selezionato risulterà essere il nodo sorgente. La modalità di disegno, in questo caso, viene impostata ad *Aggiungi edge fisico.*

Una volta selezionati i due nodi estremi dell'arco si apre automaticamente la finestra di Figura 28, utilizzando la quale si possono specificare le proprietà dell'edge.

Analogamente a quanto visto per i nodi, anche per gli edge è possibile specificare un nome personalizzato; *RsdEditor* , comunque, propone un nome di default. La finestra denominata Proprietà RSD evidenzia anche come lo edge è stato costruito: indica il nodo sorgente, quello destinatario ed il tipo di edge: unidirezionale o bidirezionale. Inoltre, è possibile specificare degli attributi scegliendo fra quelli predefiniti o definendone di propri. Come nel caso della specifica degli attributi di un nodo, il combobox di sinistra contiene

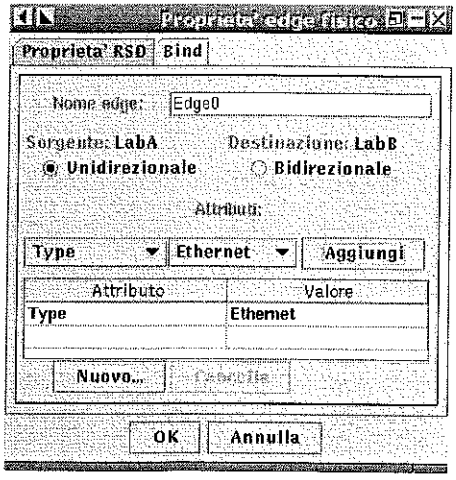


Figure 28: *RsdEditor*: finestra per la specifica delle caratteristiche grafiche e degli attributi RSD di un edge fisico.

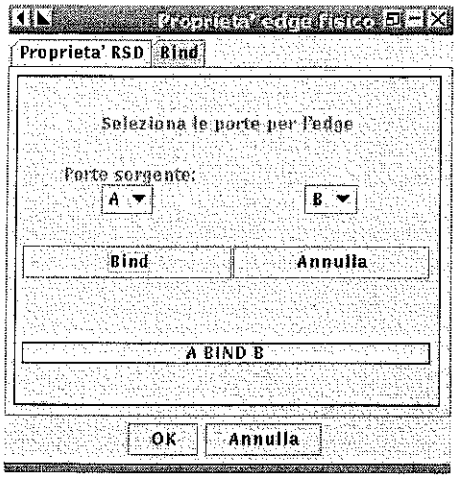


Figure 29: *RsdEditor*: finestra per la specifica del binding dello edge con le porte definite nei nodi estremi.

le categorie di attributi disponibili; selezionando un elemento da tale lista, nel combobox di destra vengono inseriti gli attributi della categoria selezionata. Quindi si può aggiungere la coppia prescelta premendo il pulsante Aggiungi; l'attributo selezionato comparirà nella tabella sottostante. Infine, è possibile inserire nuovi attributi definiti dall'utilizzatore mediante la finestra in Figura 24, che può essere aperta utilizzando il pulsante Nuovo.

La finestra di Figura 29, denominata Bind, permette di definire il cosiddetto *binding* dello edge, cioè di specificare quale delle porte definite su di un nodo viene utilizzata dallo edge. Tramite i due menù a tendina *Porta sorgente* e *Porta destinatario* è possibile impostare quale delle porte utilizzare fra quelle definite ai due estremi del collegamento; sulla parte bassa della finestra, una label indica il binding attualmente impostato.

Da notare che il binding dello edge è un vincolo della sintassi RSD. Quando si definisce uno edge, la sintassi che si deve utilizzare è la seguente:

```
EDGE LabA_LabB {
    NODE LabA PORT A <=> NODE LabB PORT C;
}
```

Come si vede, la porta è un elemento fondamentale della sinassi RSD e, quindi, l'interfaccia richiede prima di tutto che siano definite le porte all'interno dei

nodi e poi che sia definito un binding dello edge con le porte coinvolte. In questo modo si riesce a generare un file di specifica in sintassi RSD completo e corretto.

In Figura 30 è mostrato un'esempio di inserimento di due edge fisici. Si

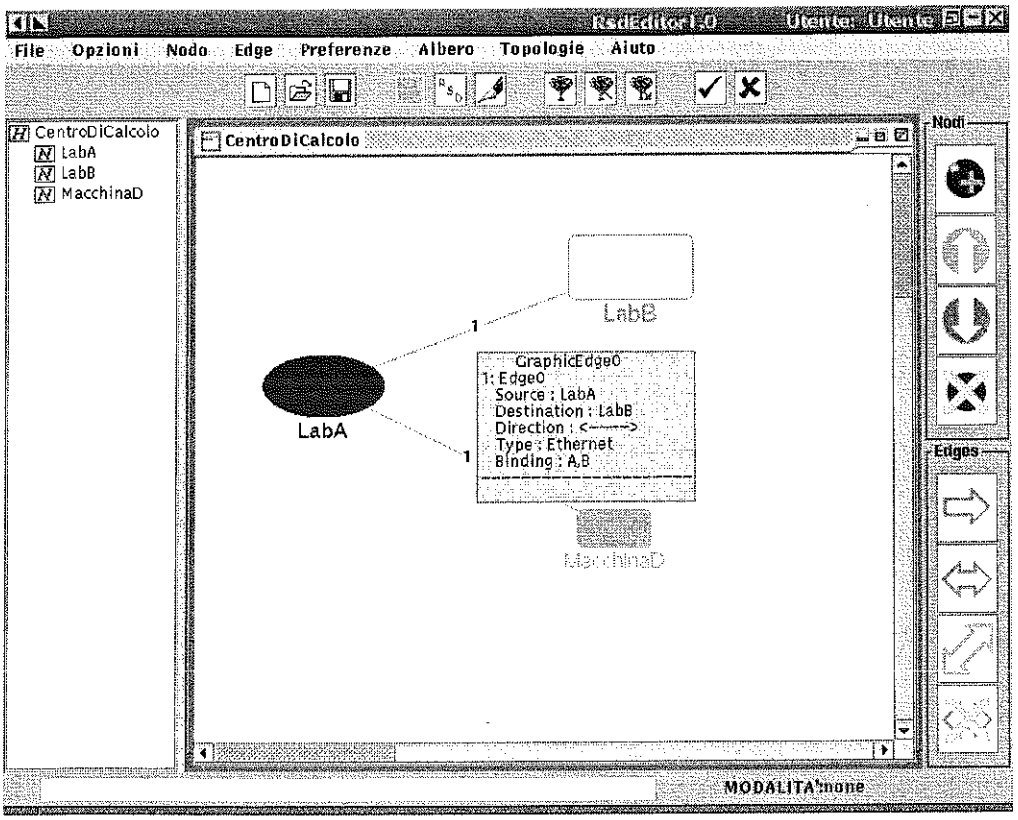


Figure 30: *RsdEditor* : esempio di specifica di alcuni edge fisici.

noti che ogni edge fisico viene dotato, al momento della sua creazione e delle successive modifiche, di un tooltip che ne elenca gli attributi. Le modalità di attivazione di tali tooltip sono analoghe a quelle dei nodi. Inoltre, è possibile definire più edge fisici fra due stessi nodi: in questo caso, il panello di controllo dello edge (cioè il riquadro colorato al centro della linea che lo rappresenta) indica il numero di edge fisici che collegano i due estremi.

3.6 Aggiunta di un edge virtuale

In sintassi RSD, uno edge virtuale, o verticale, è usato per indicare il trasferimento di un oggetto di tipo PORT all'oggetto di tipo NODE appartenente al livello immediatamente superiore della gerarchia delle risorse che si sta descrivendo. Ovviamente, per specificare un edge virtuale è necessario che siano definiti alcuni nodi su livelli diversi e che almeno uno di essi abbia una porta.

Per specificare uno edge virtuale si può utilizzare la funzione Virtuale del menù Edge oppure il pulsante  indicato dal tooltip con la scritta *''Crea un edge verticale"*.

In entrambi i casi, la barra di stato indica l'azione da compiere per completare il comando: *"Seleziona il nodo sorgente"*. Una volta selezionato il nodo desiderato, *RsdEditor* apre la finestra mostrata in Figura 31 per specificare le caratteristiche dello edge.

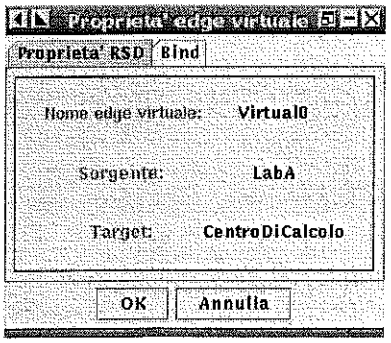


Figure 31: *RsdEditor* : esempio di specifica di edge virtuale.

Figure 32: *RsdEditor* : finestra usata per la specifica del binding dello edge con la porta definita sul nodo sorgente.

La finestra Proprietà RSD mostra il nome dello edge virtuale, il nome del nodo sorgente e l'ipernodo verso cui la porta selezionata viene esportata. Viste le caratteristiche del linguaggio RSD, non sono necessari meccanismi di immissione dati, poichè per gli edge virtuali non sono ammessi attributi. La Figura 32 mostra la finestra Bind, attraverso la quale si specifica quale delle porte definite nel nodo sorgente saranno esportate verso il livello gerarchicamente superiore. Anche in questo caso, data la sintassi del linguaggio RSD, è necessario definire quale porta rendere visibile al livello superiore fissando un

binding. Il linguaggio RSD, nel caso degli edge virtuali, permette due diverse sintassi: con la prima sintassi, si trasferisce la porta con lo stesso nome con cui è stata definita, mentre con la seconda sintassi RSD permette di assegnare alla porta un nome diverso. A questo scopo, la finestra di Figura 32 mette a disposizione il campo "Esporta con il nome:", che permette all'utente di assegnare un nuovo nome alla porta. Tale campo, però, non è sempre attivo: ci sono dei casi in cui è necessario impedire all'utente di rinominare la porta, in modo da generare un file di specifica corretto. La sintassi RSD, nel caso in cui lo edge virtuale rinomini la porta, prevede la dichiarazione di una porta con lo stesso nome nell'ipernodo del livello superiore che la eredita. Di seguito è riportato un esempio.

```
NODE Padre {
   PORT exp;

   NODE Uno {
      PORT portUno;
      };

   ASSIGN NODE Uno PORT portUno <=> PORT exp;
};
```

Il nodo `Uno` definisce una porta di nome `portUno`; quindi, lo statement

```
ASSIGN NODE Uno PORT portUno <=> PORT exp;
```

rende visibile al livello superiore, ed esattamente nel nodo `Padre`, la porta `portUno` rinominandola in `exp`. Per generare una sintasi RSD corretta, è necessario che nel nodo `Padre` venga definita la porta mediante lo statement `PORT exp`. In Figura 33 è mostrato un esempio di inserimento di uno edge virtuale. In tale esempio, poichè *CentroDiCalcolo* non è una risorsa (e quindi non è un nodo) ma è il nome del file che contiene la specifica delle risorse, non è possibile definire in esso una porta. Quindi, se viene specificato uno edge virtuale che ha come estremo un nodo appartenente al livello gerarchico più alto della specifica, il campo Esporta con nome rimane disabilitato, in modo da impedire all'utilizzatore la rinimina della porta utilizzata, dato che tale rinomina

richiederebbe la specifica, al livello superiore, di una porta. Si noti che *RsdEd-*

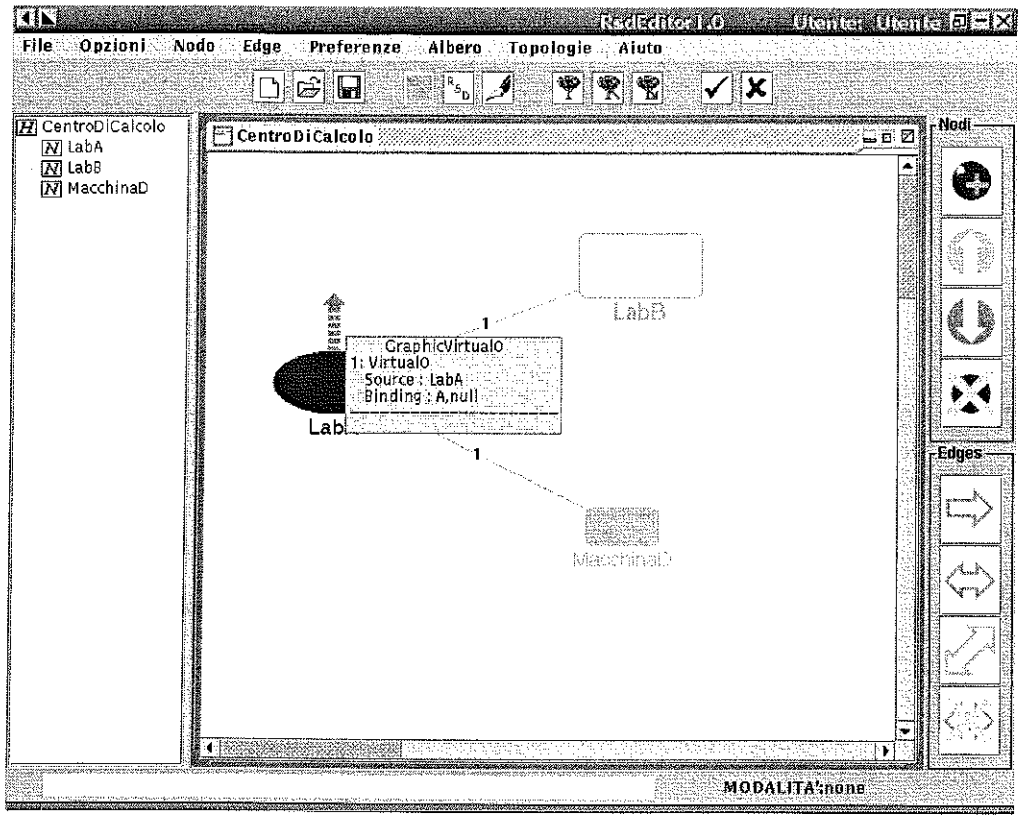


Figure 33: *RsdEditor* : esempio di inserimento di un edge virtuale.

itor costruisce i tooltip anche per gli edge virtuali: tali tooltip mostrano gli attributi specificati per lo edge. Infine, è possibile definire più edge virtuali su uno stesso nodo, in modo analogo agli edge fisici.

3.7 Creazione di un ipernodo (bottom-up)

RsdEditor consente di creare le proprie specifiche di risorse in modalità *bottom-up* utilizzando il meccanismo degli ipernodi. Abbiamo precedentemente scritto che un ipernodo contiene la specifica di altre risorse quali nodi, edge fisici ed edge virtuali, grazie alla possibilità offerta dal linguagio RSD di definire i nodi in modo ricorsivo. Quindi, una volta creati alcuni nodi ed alcuni edge, è possibile raccoglierli tutti all'interno di un ipernodo, rispettando così l'obiettivo

di progetto di consentire la specifica bottom-up delle risorse. Per definire un'ipernodo, *RsdEditor* prevede due possibilità: la funzione Crea ipernodo del menù Nodo e il pulsante

includegraphics../images/hyper.ps

indicato dal tooltip con la scritta *"Crea ipernodo"*.

Dopo aver scelto l'una o l'altra possibilità, *RsdEditor* produce un messaggio per richiedere all'utilizzatore di confermare la creazione di un ipernodo contenente tutti gli oggetti appartenenti al livello gerarchico corrente. In caso di risposta affermativa, la barra di stato indica il messaggio "*Clicca nel workspace per creare l'ipernodo*" e, contemporaneamente, il workspace si svuota per consentire all'utente di selezionare il punto in cui posizionare il nuovo ipernodo.

Una volta selezionato un punto nel workspace, *RsdEditor* apre la finestra mostrata in Figura 22 per consentire all'utente di impostare le caratteristiche grafiche e gli attributi RSD del nuovo nodo. Si noti che un ipernodo possiede tutte le caratteristiche di un nodo "semplice", salvo il fatto che, dal punto di vista della sintassi RSD generata, esso contiene la specifica di alcune risorse al suo interno. Cioòi ha permesso di utilizzare, per la sua specifica, la stessa finestra usata per la specifica di nodi "semplici". La Figura 34 mostra un esempio di ipernodo.

3.8 Creazione di un sottonodo (top-down)

RsdEditor permette di creare specifiche di risorse in modalità *top-down* grazie alla possibilità di definire sottonodi. Anche in questo caso sono disponibili due modi per definire un sottonodo: la funzione Crea sottonodo del menù Nodo ed il pulsante

indicato dal tooltip con la scritta *"Crea sottonodo"*. La creazione di un sottonodo consiste nello specificare una risorsa ad un livello gerarchico inferiore rispetto ad un'altra (dal punto di vista di una struttura ad albero, equivale a specificare il figlio di un certo nodo). La definizione di un sottonodo può essere vista come un'operazione complementare rispetto alla definizione di un ipernodo, in quanto, mentre l'ipernodo raccoglie un gruppo di risorse in un

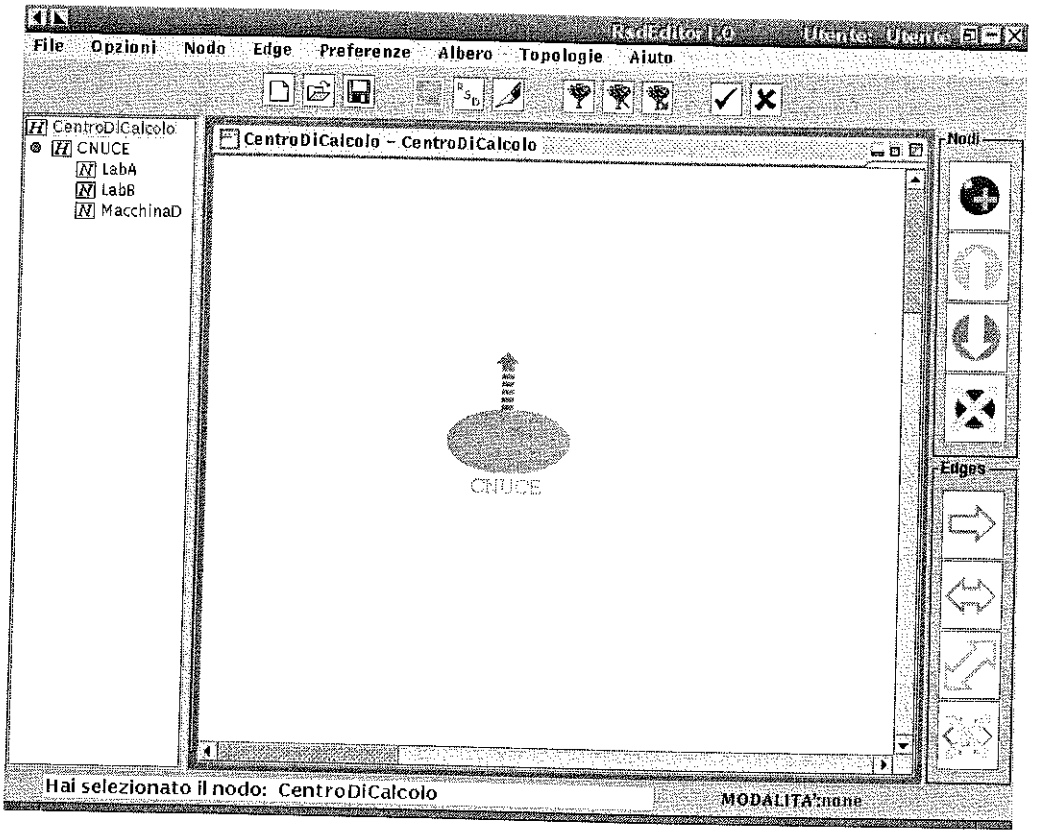


Figure 34: *RsdEditor* : esempio di definizione di un ipernodo.

unico oggetto, la definizione di un sottonodo permette di specificare un oggetto con maggior dettaglio.

Dopo aver avviato la procedura di creazione del sottonodo, la barra di stato mostra il messaggio "*Seleziona un nodo*"; questo significa che l'utente deve selezionare un nodo fra quelli visibili nel workspace, in modo da permettere la specifica di altri oggetti al suo interno. Una volta selezionato il nodo, l'interfaccia può comportarsi in due modi:

- se il nodo selezionato non contiene nessuna specifica di risorsa al suo interno (ovvero se è un nodo foglia nell'albero delle risorse) allora all'utente viene mostrato un nuovo workspace;
- se il nodo selezionato contiene già alcune specifiche di risorse allora all'utente vengono mostrate tali specifiche di risorse.

In ogni caso, a questo punto si può inserire un nodo utilizzando la procedura descritta nel paragrafo 3.4.

La Figura 35 mostra la finestra principale di *RsdEditor* dopo che sono stati inseriti alcuni sottonodi di esempio. Si noti soprattutto l'organizzazione gerar-

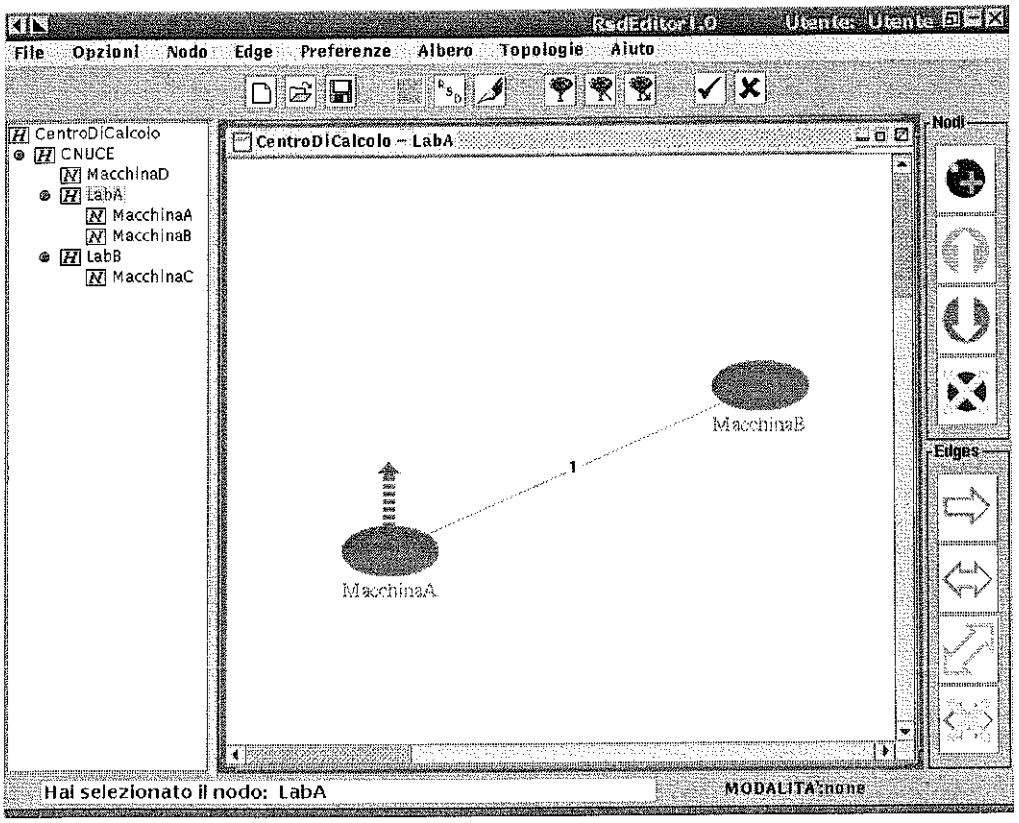


Figure 35: *RsdEditor* : esempio di inserimento di alcuni sottonodi.

chica delle risorse specificate osservanto l'albero delle risorse nella parte sinistra della finestra principale dell'interfaccia.

3.9 Albero delle risorse

Quando *RsdEditor* viene utilizzato per realizzare specifiche complesse di risorse, che coinvolgono numerosi oggetti strutturati su più livelli gerarchici, è facile perdere di di vista la specifica realizzata fino ad un certo momento. Per questo motivo, abbiamo previsto, nella parte sinistra della finestra principale di RsdEditor, di visualizzare una struttura ad albero detta albero delle risorse che

mostra tutti i nodi definiti dall'utente, rispettando l'organizzazione gerarchica ad essi assegnata, e permette, inoltre, di *navigare* all'interno dei vari livelli della specifica stessa.

In Figura 35 è mostrato un esempio di specifica di risorse strutturata su più livelli; l'albero delle risorse permette una visione d'insieme concisa, ed è utile all'utente in quanto gli permette di vedere tutti i livelli su cui si articola la specifica del suo metacomputer e, pertanto, gli facilita il passaggio da un livello all'altro. Infatti, utilizzando il tasto sinistro del mouse, è possibile selezionare un nodo all'interno dell'albero, facendo si che *RsdEditor* lo visualizzi insieme a tutti gli altri oggetti definiti allo stesso livello gerarchico.

L'albero delle risorse mostra i nodi definiti nella specifica utilizzando vari tipi di icone:

- indica un ipernodo, che non è mai una foglia dell'albero delle risorse;
- N indica un nodo "semplice" oppure una topologia di interconnessione ed è sempre una foglia dell'albero delle risorse.

Accanto alle icone viene visualizzato il nome dell'oggetto che esse rappresentano, in modo da rendere intuitiva la navigazione all'interno dell'albero.

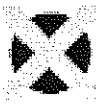
Infine, le voci del menù Albero ed i pulsanti nella barra degli strumenti forniscono alcune altre utili operazioni:

- la funzione Visualizza albero e l'icona  permettono di nascondere/mostrare l'albero delle risorse, in modo da ampliare il workspace;
- la funzione Espandi tutto e l'icona  permettono di aprire ogni livello dell'albero, in modo che siano visibili le risorse in esso specificate;
- la funzione Chiudi tutto e l'icona  permettono di chiudere i livelli dell'albero delle risorse, lasciando comunque visibile l'albero stesso.

3.10 Rimozione di nodi ed edge

Oltre alle funzioni di aggiunta di oggetti, *RsdEditor* fornisce anche funzioni per la rimozione degli oggetti disegnati; è possibile cancellare nodi, edge fisici ed

edge virtuali.

Per la rimozione dei nodi si può utilizzare la funzione Elimina dal menù Nodo oppure il pulsante  indicato dal tooltip con la scritta *"Elimina nodo"*. Una volta selezionata la funzione che permette di rimuovere un nodo la barra di stato invita l'utente a selezionare il nodo da rimuovere con il messaggio *"Seleziona il nodo da rimuovere"*. A questo punto l'utente può selezionare, utilizzando il tasto sinistro del mouse, il nodo da rimuovere. *RsdEditor* , a questo punto, può assumere due diversi comportamenti, a seconda dell'oggeto selezionato. Nel caso in cui il nodo selezionato sia:

- una foglia e non abbia alcun edge ad esso collegato, *RsdEditor* propone un messaggio di richiesta di conferma della cancellazione e, in caso di risposta affermativa, cancella il nodo;
- un ipernodo oppure un nodo agli estremi di un edge, *RsdEditor* produce un messaggio di richiesta di conferma il quale ricorda all'utente che, procedendo con la cancellazione, verranno rimossi tutti i sottonodi e cancellati tutti gli edge che hanno tale nodo come mittente o ricevente.

Per la rimozione degli edge, sia fisici che virtuali, si può utilizzare la funzione Rimuovi del menù Edge oppure il pulsante indicato dal tooltip con la scritta *"Elimina edge"*.

Una volta attivata la procedura di rimozione di un edge, la barra di stato mostra il messaggio *"Seleziona lo edge da rimuovere"*. Quando l'utente seleziona lo edge da cancellare (tasto sinistro del mouse), *RsdEditor* mostra un menù contestuale[4] con un numero di funzioni che varia a seconda del numero di edge che l'oggetto grafico rappresenta. La funzione Elimina tutti permette di cancellare tutti gli edge associati all'oggetto grafico selezionato, mentre le altre funzioni permettono di rimuovere lo edge che ha il nome indicato nella funzione selezionata.

[4]Un menù *contestuale* è un menù associato ad un oggetto visuale e che compare su tale oggetto in corrispondenza di una azione dell'utente.

3.11 Aggiunta di topologie

RsdEditor prevede funzionalità per la specifica di alcune topologie di interconnessione omogenee a livello architetturale. Si tratta di funzionalità che aiutano l'utente nella specifica di richieste di risorse complesse: si pensi, ad esempio, ad un *torus* 4×8, per il quale l'utente dovrebbe specificare **32** nodi e **64** archi. Utilizzando topologie di interconnessione predefinite l'utente può fare la stessa cosa definendo un solo nodo, in modo semplice ed intuitivo. Si noti che *RsdEditor* permette, tramite queste funzionalità, di inserire solo topologie omogenee; nel caso in cui l'utente volesse specificare una richiesta di risorse che coinvolge una topologia di interconnessione eterogenea a livello architetturale può farlo specificando una per una le risorse che ne fanno parte.

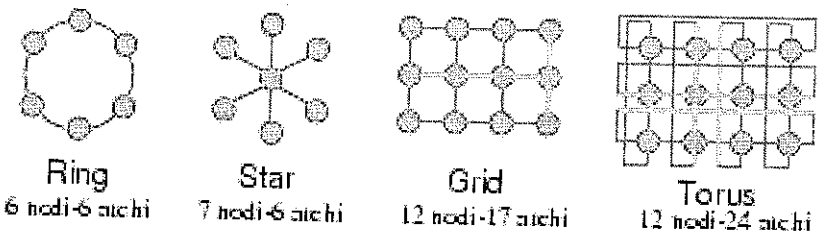


Figure 36: Topologie di interconnessione attualmente supportate da *RsdEditor* .

Figure 36: Topologie di interconnessione attualmente supportate da *RsdEditor* .

Le topologie di interconnessione, mostrate in Figura 36, implementate da *RsdEditor* sono, attualmente, quattro. Il menù Topologie contiene le quattro funzioni che permettono di inserire nel file la loro specifica; una volta selezionata una delle funzioni, la barra di stato visualizza il messaggio

"Seleziona un punto nel workspace per aggiungere la topologia T"

dove T indica Ring, Star, Grid o Torus.

Selezionato un punto nel workspace (tasto sinistro del mouse), viene aperta una finestra che permette di specificare le caratteristiche della topologia scelta. La finestra è composta da quattro pannelli sovrapposti, tre dei quali sono identici per ogni topologia, mentre il primo cambia a seconda della topologia a cui si riferisce.

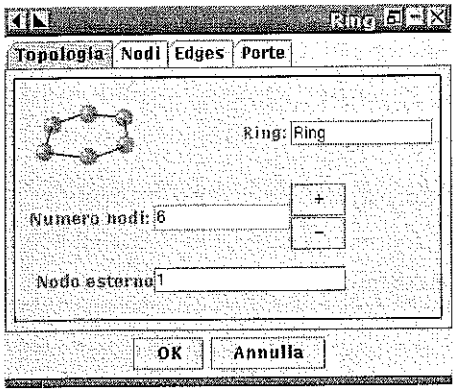


Figure 37: *RsdEditor* : esempio di specifica delle caratteristiche della topologia Ring.

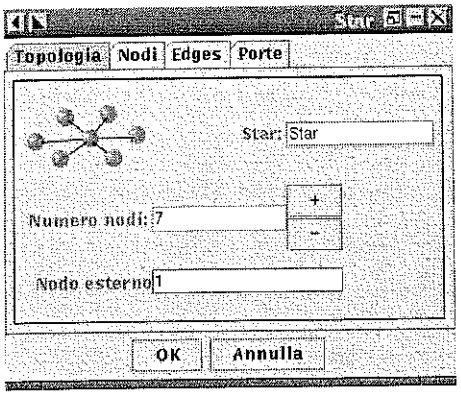


Figure 38: *RsdEditor* : esempio di specifica delle caratteristiche della topologia Star.

In Figura 37 è mostrato un esempio di uso della finestra per la specifica delle caratteristiche della topologia Ring. Utilzzando il pannello Topologia è possibile assegnare un nome alla topologia stessa, definirne il numero di nodi, utilizzando l'apposito campo ad incremento, ed indicare quale di questi possiede il collegamento con l'esterno, cioè a quale nodo assegnare la porta esterna per collegare la topologia con il resto delle risorse.

La Figura 38 mostra un esempio di specifica della topologia Star. In questo caso, le caratteristiche da specificare sono analoghe a quelle specificate per la topologia Ring.

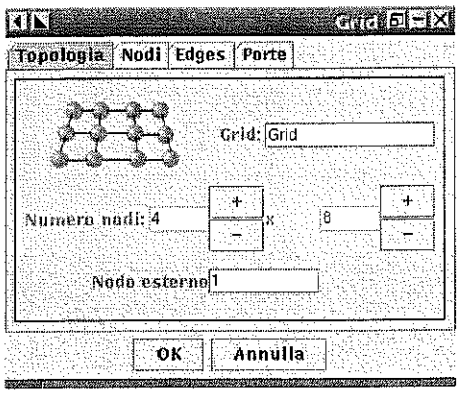


Figure 39: *RsdEditor* : esempio di specifica delle caratteristiche della topologia Grid.

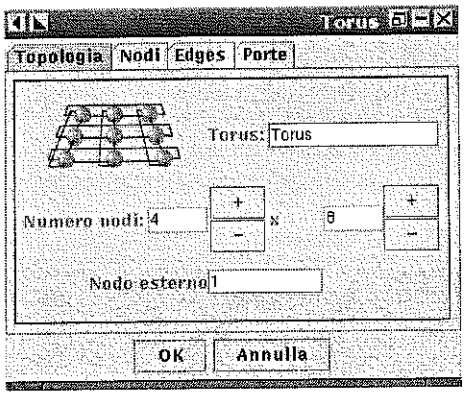


Figure 40: *RsdEditor* : esempio di specifica delle caratteristiche della topologia Torus.

In Figura 39 è mostrato il pannello Topologia usato per la specifica delle caratteristiche della topologia Grid. Anche in questo caso è possibile assegnare

un nome alla topologia ed indicare quale dei suoi nodi possiede il collegamento con un altro nodo esterno alla topologia. Poichè in questo caso la topologia è bidimensionale, per assegnarle una dimensione è necessario specificare due valori distinti; per questo il pannello Topologia propone due campi ad incremento per la definizione del numero di nodi che compongono la topologia.

Infine, la Figura 40 mostra un esempio di inserimento delle caratteristiche della topologia Torus.

I rimanenti pannelli: Nodi, Edges e Porte, sono identici per le quattro topologie in quanto permettono di specificare le caratteristiche generiche dei nodi e degli archi che compongono la topologia. In Figura 41 è mostrata la

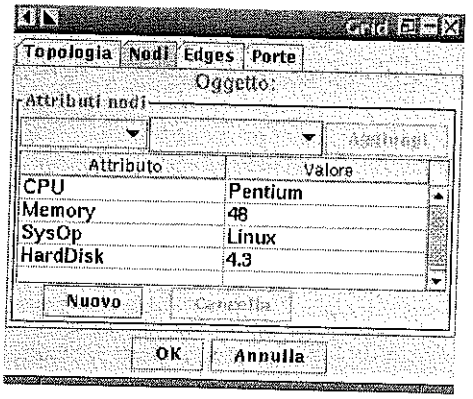


Figure 41: *RsdEditor*: esempio di specifica degli attributi di un nodo nella topologia di interconnessione.

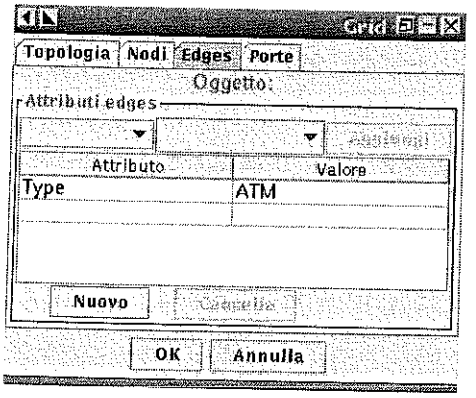


Figure 42: *RsdEditor*: esempio di specifica degli attributi di un edge nella topologia di interconnessione.

finestra che permette di specificare le caratteristiche di un generico nodo nella topologia (ricordiamo che le topologie specificabili usando *RsdEditor* sono omogenee). Gli attributi che si possono specificare per i nodi della topologia sono gli stessi previsti per la specifica di un nodo "semplice" e la loro gestione è identica a quella descritta per i nodi nel paragrafo 3.4.

La finestra mostrata in Figura 42 consente di specificare gli attributi per gli edge presenti nella topologia. Anche in questo caso, tali attributi sono gestiti dall'interfaccia in modo analogo a quanto visto per gli edge fisici nel paragrafo 3.4.

Infine, il pannello Porte, mostrato in Figura 43, permette la specifica delle porte. È prevista la specifica di due porte: quella *esterna* è posseduta dal solo nodo che è collegato con l'esterno, individuato dal valore inserito dall'utente

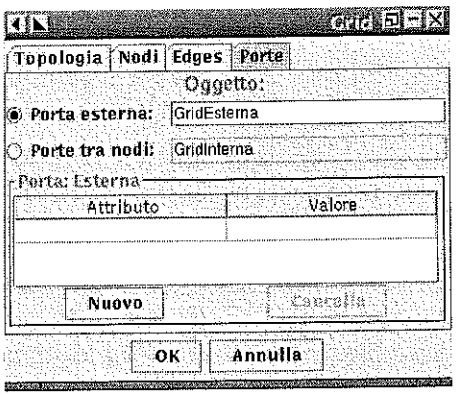


Figure 43: *RsdEditor* : esempio di specifica degli attributi delle porte nella topologia di interconnessione.

nell'apposito campo nel pannello Topologia (vedi Figure 37, 38, 39 e 40), mentre quella *interna* permette di specificare la porta necessaria a collegare tra loro i nodi che costituiscono la topologia. Il pannello Porte, inoltre, permette di modificare il nome delle porte e di impostarne gli attributi.

3.12 Operazioni possibili sui nodi

RsdEditor consente di modificare la posizione di un nodo semplicemente selezionandolo e trascinandolo nella nuova posizione. Conseguentemente, la posizione degli edge connessi al nodo viene modificata in base agli spostamenti che subisce il nodo stesso.

Ad ogni nodo è associato un menù contestuale, apribile usando il tasto destro del mouse. Tale menù contiene tre comandi:

- Modifica: selezionando questa voce viene aperta la finestra mostrata in Figura 22, mediante la quale è possibile modificare le caratteristiche specificate per il nodo. Le modifiche apportate hanno effetto immediato: l'oggetto grafico viene ridisegnato in base alle nuove impostazioni e gli attributi RSD vengono aggiornati.

- Cancella: selezionando questo comando, *RsdEditor* avvia la procedura di cancellazione del nodo.

- Esporta: questo comando permette di esportare il nodo.

Un menù contestuale analogo a quello appena descritto è previsto anche per i nodi topologia; le funzionalità supportate sono uguali a quelle viste prima. Infine, anche i nodi Topologia possono essere riposizionati nel workspace effettuando un'operazione di spostamento analoga a quella descritta per i nodi.

3.13 Operazioni possibili sugli edge

Per ogni oggetto grafico che rappresenta, in sintassi RSD, un'oggetto di tipo edge, *RsdEditor* definisce un menù contestuale associato al tasto sinistro del mouse: il numero di voci in tale menù dipende dal numero di edge che l'oggetto grafico rappresenta. Selezionando una delle funzioni di tale menù viene aperta la finestra mostrata in Figura 28, se si è selezionato un edge fisico, oppure la finestra mostrata in Figura 31. Mediante tali finestre è possibile modificare le caratteristiche dell'edge selezionato.

Per modificare il colore di un'edge si utilizza la finestra di Figura 44, sia per gli edge fisici che per gli edge virtuali. Tale finestra viene aperta utilizzando il

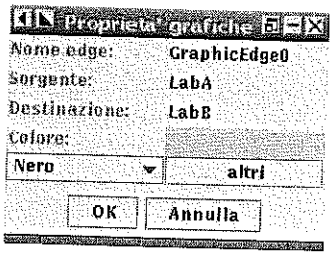


Figure 44: *RsdEditor* : finestra per la specifica delle caratteristiche grafiche degli edge fisici e virtuali.

tasto destro del mouse e permette di modificare il colore dello edge (selezionabile da una lista di colori predefiniti o definibile dall'utente utilizzando la finestra che *RsdEditor* apre in seguito alla selezione del pulsante Altri).

3.14 Importazione ed esportazione di un nodo

RsdEditor offre la possibilità di *importare* ed *esportare* specifiche di risorse o parti di esse, in modo da renderle riusabili in operazioni future.

Per utilizzare queste funzionalità si usano le voci Importa ed Esporta del menù Nodo.

Importa: selezionata questa funzione, un messaggio visualizzato nella barra di stato richiede all'utente di selezionare un punto nel workspace per iniziare la procedura di importazione. Quindi, *RsdEditor* mostra all'utente la finestra di Figura 45, utilizzando la quale l'utente può selezionare il file di specifica da importare nella corrente specifica di risorse. Successi-

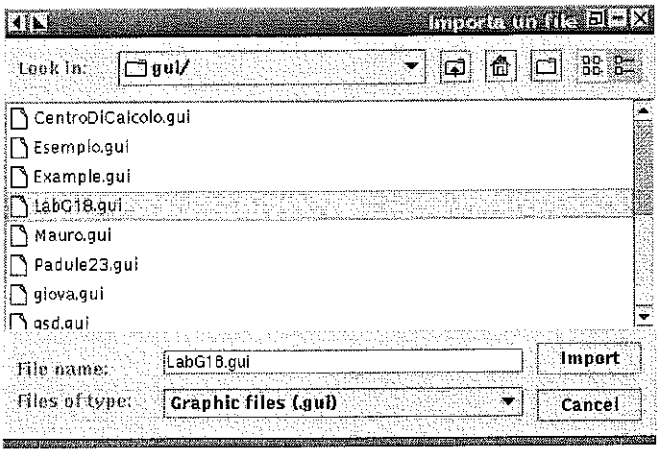


Figure 45: *RsdEditor* : finestra usata per la selezione di un file da importare.

vamente, premendo il tasto Import, può essere letto il file di specifica da importare; i nodi descritti in tale file vengono posizionati nel workspace in corrispondenza del punto selezionato, come mostrato in Figura 46.

Esporta: selezionata questa funzione, *RsdEditor* visualizza un messagio nella barra di stato per richiedere all'utente di selezionare il nodo da esportare e, una volta selezionato, chiede una conferma dell'operazione in corso. A questo punto, *RsdEditor* mostra la finestra di Figura 47, mediante la quale è possibile indicare il nome del file su cui memorizzare il contenuto della parte di specifica precedentemente selezionata. Quest'ultima operazione avviene eseguendo la funzione Export.

3.15 Elementi predefiniti

Oltre a fornire la possibilità di importare ed esportare sottoinsiemi di specifiche di risorse, *RsdEditor* prevede anche alcune predefinizioni di risorse; si tratta di specifiche di risorse già definite che ogni utente può utilizzare nella propria

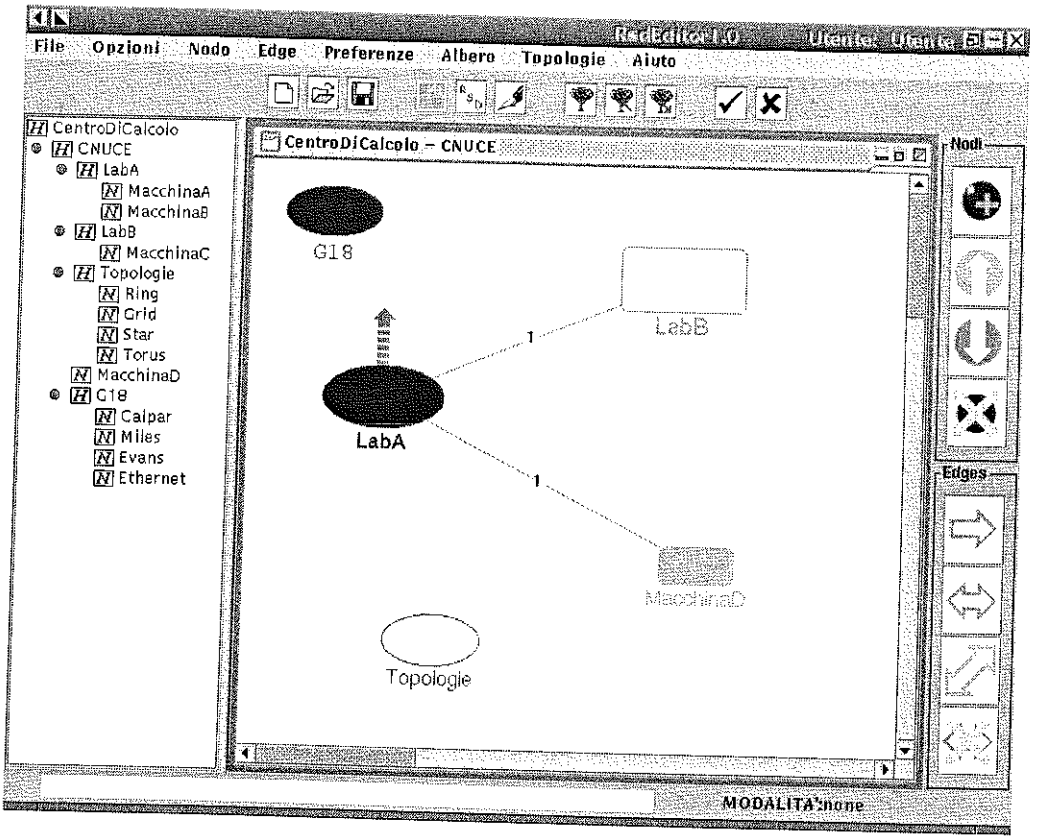


Figure 46: *RsdEditor* : esempio di importazione di una specifica chiamata LabG18.

richiesta per velocizzare il proprio lavoro. Le predefinizioni che l'editor mette a disposizione sono accessibili tramite la funzione Predefiniti del menù Nodo. L'esecuzione di tale funzione produce l'apertura di un sottomenù che contiene le specifiche di risorse predefinite disponibili.

Per inserire una risorsa predefinita in una specifica di risorse è sufficiente selezionarla fra quelle disponibili: *RsdEditor* visualizza un messaggio nella barra di stato per richiedere all'utente di selezionare un punto nel workspace e, quindi, inserisce l'oggetto nel punto fissato.

La lista degli oggetti predefiniti può, comunque, essere aggiornata mediante l'eliminazione/inserimento di vecchie/nuove predefinizioni. Le predefinizioni di risorse sono organizzate in *categorie*, in modo da renderne più semplice la ricerca.

La rimozione di una specifica predefinita è effettuata eseguendo la fun-

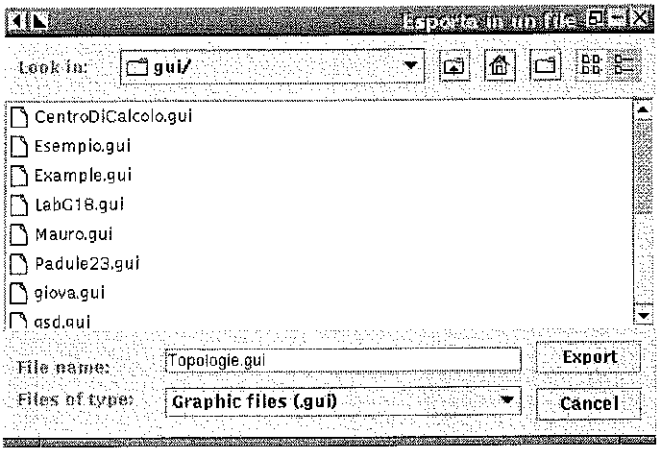


Figure 47: *RsdEditor* : finestra usata per la specifica del nome con cui l'oggetto selezionato viene esportato.

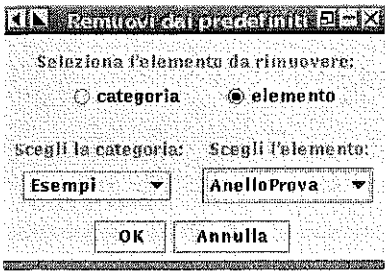


Figure 48: *RsdEditor* : finestra per la rimozione di specifiche di risorse predefinite.

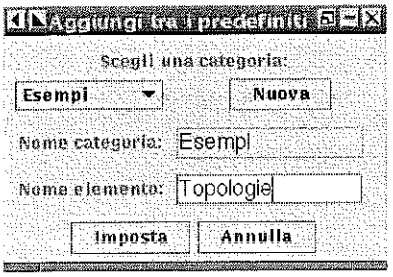


Figure 49: *RsdEditor* : finestra per l'aggiunta di specifiche di risorse fra quelle predefinite.

zione Rimuovi dai predefiniti del menù Nodo che interagisce con il richiedente mediante la finestra mostrata in Figura 48. È possibile rimuovere un'intera categoria oppure una singola predefinizione; il combobox Scegli la categoria permette di selezionare la categoria da rimuovere, mentre il combobox Scegli l'elemento permette di selezionare una predefinizione. Premendo il tasto OK viene mostrato un messaggio di conferma della cancellazione.

Per inserire una nuova specifica di risorse deve essere eseguita la funzione Aggiungi ai predefiniti del menù Nodo; il messaggio visualizzato nella barra di stato richiede all'utente di selezionare nel workspace la specifica da aggiungere tra quelle predefinite. Dopo aver prodotto un messaggio di conferma, *RsdEditor* visualizza la finestra di Figura 49, nella quale è possibile assegnare un

nome alla specifica selezionata ed aggiungerla ad una delle categorie esistenti, oppure creare una nuova categoria. Quindi, premendo il tasto Imposta, dopo aver proposto un messaggio di conferma, *RsdEditor* rende disponibile la specifica aggiungendola al sottomenù Predefiniti, all'interno della categoria indicata in precedenza. Si noti che la nuova predefinizione diviene immediatamente disponibile.

3.16 Visualizzazione del codice prodotto

RsdEditor mette a disposizione dell'utente la possibilità di visualizzare il codice di specifica in sintassi RSD prodotto fino ad un certo momento. La funzione Visualizza file RSD del menù Opzioni oppure il pulsante nella barra degli strumenti permettono di eseguire tale funzionalità.

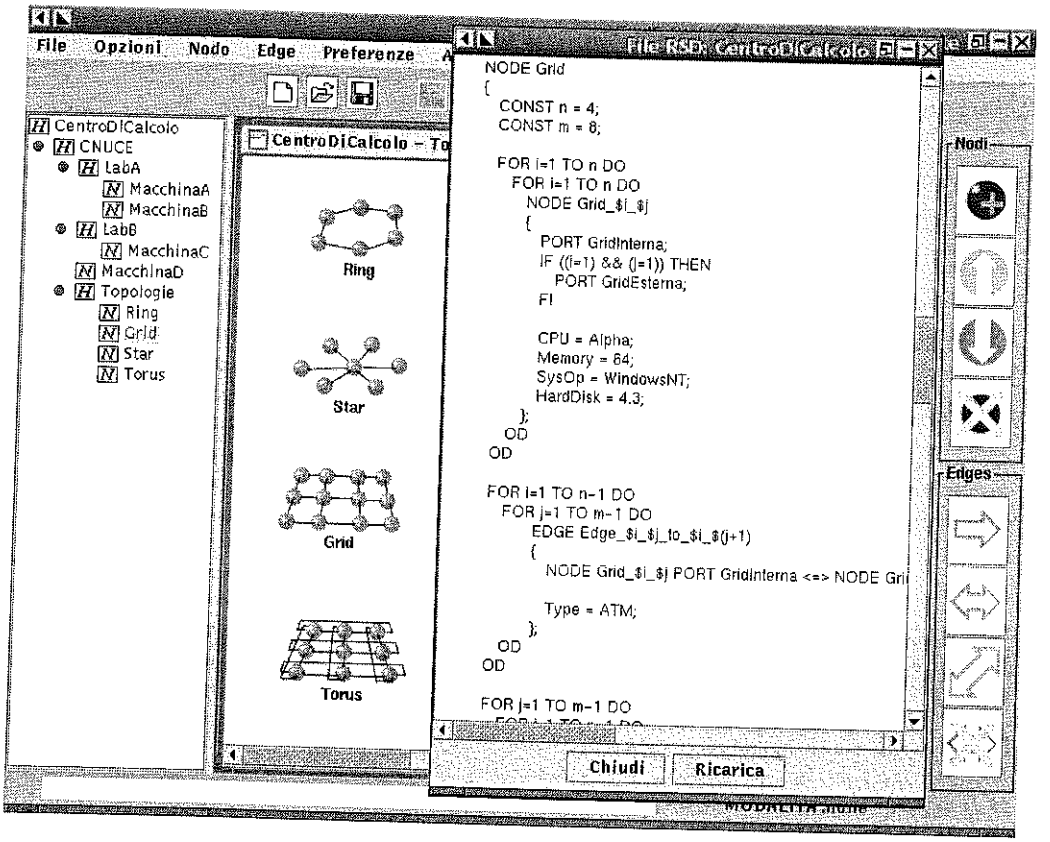


Figure 50: *RsdEditor* : finestra per la visualizzazione di una specifica di risorse in sintassi RSD; la finestra mostra la specifica della topologia di interconnessione Grid.

Attivata l'esecuzione della funzione Visualizza file RSD, *RsdEditor* apre la finestra di Figura 50, che mostra la specifica di risorse realizzata. Tale finestra è l'unica non modale[5] dell'intera applicazione; questo accorgimento permette di mantenere visibile tale finestra durante la fase di specifica delle risorse. Una volta apportate alcune modifiche alla descrizione delle risorse è possibile visualizzarle immediatamente premendo il pulsante Ricarica. Si noti che la finestra permette soltanto la visualizzazione del codice prodotto e non la sua modifica. Infine, per chiudere la finestra si può utilizare il pulsante Chiudi.

3.17 Operazioni di salvataggio

RsdEditor, utilizzando la funzione Salva del menù File oppure il pulsante  nella barra degli strumenti, permette di salvare su file il lavoro svolto fino ad un certo momento. In questo modo, ogni volta vengono generati due nuovi files:

- al primo file viene assegnato il nome

  nomefile`.rsd`

  e contiene la specifica delle risorse i sintassi RSD; *nomefile* è il nome scelto dall'utente al momento della creazione di una nuova specifica di risorse (vedi ***S***3.3).

- al secondo file viene assegnato il nome

  nomefile`.gui`

  e contiene una descrizione formale degli oggetti grafici disegnati dall'utente per la specifica delle risorse. Tale file segue la sintassi del linguaggio **GUI** [45] Tale linguaggio è stato creato con lo scopo di descrivere in modo rigoroso gli elementi visuali che l'utente disegna durante la fase di specifica delle risorse. Inoltre, quando l'utente utilizza una specifica di risorse

[5]Una finestra *modale* ha la seguente caratteristica: quando viene visualizzata, l'input rivolto verso altre finestre dell'applicazione è bloccato fino a quando la finestra modale viene chiusa.

già esistente, tale file gli fornisce la possiblità di visualizzare le risorse esattamente con lo stesso aspetto grafico con cui le aveva specificate.

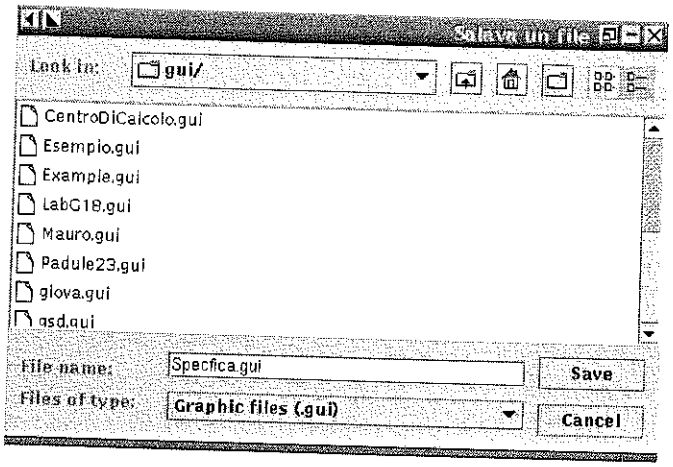


Figure 51: *RsdEditor* : esempio di memorizzazione di una specifica di risorse in un file con nome diverso da quello stabilito inizialmente.

Inoltre, è possibile salvare la specifica di risorse in un file con un nome diverso da quello specificato inizialmente. Un esempio di finestra usata per questa operazione è mostrato in Figura 51.

3.18 Apertura di una specifica esistente

RsdEditor mette a disposizione dell'utente la possibilità di aprire una specifica di risorse realizzata precedentemente e memorizata su file. Questa funzionalità semplifica il lavoro di specifica, in quanto consente di riutilizzare i files memorizzati in precedenza senza dover costruire ogni volta una nuova specifica.

In Figura 52 è mostrata la finestra che permette di selezionare un file di specifica già esistente. Per aprire un file si può utilizzare la funzione Apri... del menù File oppure il pulsante nella barra degli strumenti. È possibile scegliere il file da aprire selzionandolo con il mouse; quindi, premendo il pulsante Open il file viene visualizzato e reso disponibile all'utente per operazioni di editing.

Nel caso in cui *RsdEditor* abbia già un file aperto, viene visualizzato un messaggio per richiedere all'utente di eseguire la memorizzazione del file corrente;

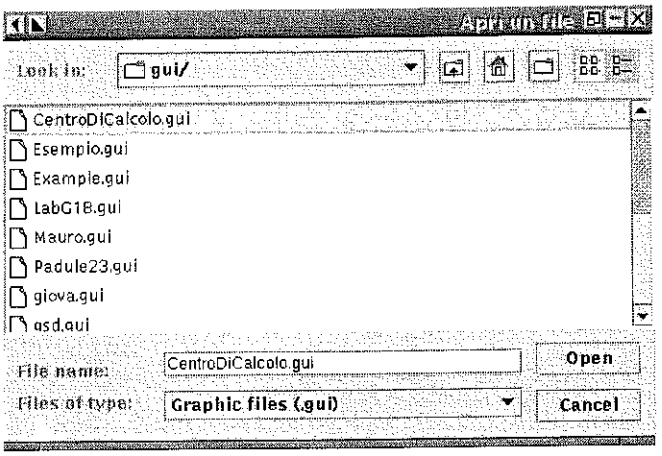


Figure 52: *RsdEditor* : finestra usata per selezionare un file di specifica fra già esistenti.

in ogni caso, poi, viene aperto il file selezionato.

3.19 Chiusura di un file di specifica e di *RsdEditor*

Una volta completato il lavoro di specifica è possibile memorizarlo su file seguendo la procedura descritta il precedenza. Per chiudere il file è possibile utilizzare la funzione Chiudi del menù File oppure il pulsante ✔ nella barra degli strumenti. *RsdEditor* visualizza un messaggio per richiedere all'utente la conferma dell'operazione.

Infine, è possibile terminare l'esecuzione di *RsdEditor* eseguendo la funzione Esci del menù File o medante il pulsante ✘ della barra degli strumenti. In fase di chiusura, *RsdEditor* controlla se un file di specifica è ancora aperto e, in tale caso, lo chiude memorizzandolo su file.

4 Acknowledgment

References

[1] R. R. Freund. *Optimal selection theory for superconcurrency.* In Proceedings of Supercomputing 89, p.699-703, 1989.

[2] A. S. Grimshaw, J. B. Weissman, E. A. West, E. C. Loyot. *Metasystems: An Approach Combining Parallel Processing and Heterogeneous Distributed Computing Systems.* Journal of Parallel and Distributed Computing, 21:257-270, 1994.

[3] L. Smarr, C. E. Catlett. *Metacomputing.* Communications of the ACM, 35(6):45-52, June 1992.

[4] M. Baker, G. Fox. *Metacomputing: Harnessing Informal Supercomputers.* Portsmouth University, preprint, December 1998, **http://www.dcs.port.ac.uk/ mab/Papers/Cluster-Book/**.

[5] F. Ramme. *Building a Virtual Machine-Room - a Focal Point in Metacomputing.* Future Generation Computer Systems (FGCS), 11:477-489, 1995.

[6] D. Laforenza. *HPCN in the New Millennium: Evolution or Revolution.* RCI European Member Management Symposium XI, December 1998. **http://brunello.cnuce.cnr.it/domenico/talks/METACOMP/ index.html**.

[7] R. R. Freund, S. Conwell. *Superconcurrency: A form of distributed heterogeneous supercomputing.* Supercomputing Review, 45-51, October 1990.

[8] A. A. Khokhar, V. K. Prasanna, M. E. Shaaban, C. L. Wang. *Heterogeneous Computing: Challenges and Opportunities.* IEEE Computer, 26(6):18-27, June 1993.

[9] G. C. Fox, R. D. Williams, P. C. Messina. *Parallel Computing: Works!.* Morgan Kaufmann Publisher, Inc., 1994.

[10] Globus Project. **http://www.globus.org**.

[11] I. Foster, C. Kesselman. *Globus: A Metacomputing Infrastructure Toolkit.* International Journal of Supercomputer Applications, 11(2): 115-128, 1997.

[12] I. Foster, D. Kohr, R. Krishnaiyer, J. Mogill. *Remote I/O: Fast Access to Distant Storage.* Workshop on I/O in Parallel and Distribute Systems, 14-25, IOPADS 1997.

[13] I. Foster, C. Kesselman, S. Tuecke. *The Nexus approach to integrating multithreading and communication.* Journal of Parallel and Distributed Computing, 37:70-82, 1996.

[14] Legion Project. **http://www.cs.virginia.edu/~legion/**.

[15] A. Grimshaw, A. Wulf, J. French, A. Weaver, P. Reynolds. *Legion: The next logical step toward the world-wide virtual computer.* UVa CS Technical Report CS-94-21, June 8, 1994.

[16] M. J. Lewis, A. S. Grimshaw. *The Core Legion Object Model.* Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing, Los Alamitos, California, IEEE Computer Society Press, August 1996.

[17] Digital Equipment Corporation, Hewlett-Packard Company, Novell USG, IBM Corporation NCR Corporation, Object Design, SunSoft inc. *The Common Object Request Brocker: Architecture and Specification.* Revision 2.2, February 1998.

[18] A. S. Grimshaw. *Easy to Use Object-Oriented Parallel Programming with Mentat.* IEEE Computer, pp 39-51, Vol. 26, No 5, May 1993.

[19] A. S. Grimshaw, A. Wulf and the Legion team. *The Legion Vision of a Worldwide Virtual Computer.* Communications of the ACM January 1997, Vol. 40, No 1.

[20] A. Reinefeld, R. Baraglia, T. Decker, J. Gehring, D. Laforenza, F. Ramme, T. Römke, J. Simon. *The MOL Project: An Open Extensible Metacomputer.* Proc. Heterogenous Computing Workshop HCW'97, IEEE Computer Society Press, 17-31.

[21] Metacomputer Online (progetto MOL).
http://www.uni-paderborn.de/pc2/projects/mol.

[22] M. Brune, J. Gehring, A. Reinefeld. *A lightweight Communication Interface Between Parallel Programming Environments.* Proc. of the Intern. Conf. on High-Performance Computing and Networking HPCN 97, Vienna, 1997.

[23] J. Gehring, A. Reinefeld. *MARS - A Framework for Minimizing the Job Execution Time in a Metacomputing Environment.* Future Generation Computer Systems (FGCS), Elsevier Science B. V., Vol. 12 No. 1, 1996, 87-99.

[24] M. Danelutto, R. Di Meglio, S. Orlando, S. Pelegatti, M. Vanneschi. *A Methodology for the Development and the Support of Massively Parallel Programs.* J. on Future Generation Computer Systems (FCGS), Vol. 8, 1992.

[25] J. Simon, J. M. Wierum. *Performance Prediction of Benchmark Programs for Massively Parallel Architecture.* 10th Annual Intl. Conf. on High-Performance Computer HPCS'96, Ottawa, Canada, June 5-7 1996.

[26] G. Faieta, M. Formica. *WAMM: Metacomputing: progettazione di una interfaccia per la definizione e la gestione di un metacalcolatore; studio di algoritmi e definizione di una euristica per il mapping in ambiente eterogeneo.* Tesi di Laurea, Università degli Studi di Pisa, Facoltà di Scienze MM. FF. NN., 201-225, 1994.

[27] *WAMM - Wide Area Metacomputer Manager.*
http://miles.cnuce.cnr.it/pp/atti.html oppure
ftp.cnr.it nella directory **/pub/wamm.**

[28] K. Dincer. *World-Wide Virtual Machine: A Metacomputing Environment Integrating World-Wide and High Performance Computing and Communications Tecnologies.* PhD thesis in Computer Science, Department of Electrical Engineering and Computer Science, Syracuse, 1997.

[29] *WebFlow.*
http://osprey7.npac.syr.edu:1998/iwt98/products/webflow.

[30] B. O. Christiansen, P. Cappello, M. F. Ionescu, M. O. Neary, K. E. Schauser, D. Wu. *Javelin: Internet-Based Parallel Computing Using Java.* 1997 ACM Workshop on Java for Science and Engineering Computation, Las Vegas, June 1997.

[31] T. Brecht, H. Sandhu, M. Shan, J. Talbot. *ParaWeb: Towards World-Wide Supercomputing.* In Proceedings of the Seventh ACM SIGOPS European Workshop on System Support for Worldwide Applications, pp 181-188, September 1996.

[32] A. Baratloo, M. Karaul, Z. Kedem, P. Wyckoff. *Charlotte: Metacomputing on the Web.* To appear International Journal on Future Generation Computer Systems.

[33] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, S. Tuecke. *A Directory Service for Configuring High-Performance Distributed Computations.* In Proc. 6th IEEE Symp. on High Performance Distributed Computing, 365-375, IEEE Computer Society Press, 1997.

[34] W. Yeong, T. Howes, S. Kille. *Lightweight Directory Access Protocol.* RFC-1777, 03/28/1995.

[35] W. Yeong, T. Howes, S. Hardcastle-Kille. *X.500 Lightweight Directory Access Protocol.* RFC-1487, 07/29/1993.

[36] K. Czajkowski, I. Foster, C. Kesselman, N. Koronis, S. Martin, W. Smith, S. Tuecke. *A Resource Management Architecture for Metacomputing Systems.* Workshop on Job Scheduling Strategies for Parallel Processing, IPPSISPDP '98, March 1998.

[37] CCS: Computing Center Software.
http://www.uni-paderborn.de/pc2/projects/ccs.

[38] Condor High Throughput Computing.
http://www.cs.wisc.edu/condor/.

[39] GENIAS Software GmbH. Codine 4.1.1 Overview.
http://www.genias.de/products/codine/overview.html.

[40] Network Queuing System (NQS) User's Guide SG-2105 9.0.
http://hpccsrv.gsfc.nasa.gov:8080/library/all/SG-2105.

[41] A. Keller, A. Reinefeld. *CCS Resource Management in Networked HPC Systems.* Proc. Heterogeneous Computing Workshop HCW'98 at IPPS, Orlando, to appear 1998.

[42] M. Brune, J. Gehring, A. Keller, B. Monien, F. Ramme, A. Reinefeld. *Specifyng Resources ans Services in Metacomputing Environments.* Parallel Computing, to appear 1998.

[43] B. Bauer, F. Ramme. *A general purpose Resource Description Language.* TAT 91, Parallelle Datenverarbeitung mit dem Transputer, R. Grebe, M. Baumann, Springer Verlag, Reihe Informatik aktuell, 1991, 68-75.

[44] M. Brune, J. Gehring, A. Keller, A. Reinefeld. *RSD - Resource and Service Description.* Proc. of the Intern. Conf. on High-Performance Computing Systems HPCS 98, Edmonton, Canada, Springer, LNCS, May 1998.

[45] M. Michelotti, S. Nannetti. *Un'interfaccia grafica per la definizione e descrizione delle risorse e servizi di un metacomputer.* Tesi di Laurea, Università degli Studi di Pisa, Facoltà di Scienze MM. FF. NN., Febbraio 1999.

[46] Graph Drawing.
http://www.fmi.uni-passau.de/~himsolt/graphdrawing.html.

[47] CAD, Drawing & Painting Tools.
http://sal.kachinatech.com/E/2/index.shtml.

[48] Auburn University. Drawing Graphs with VGJ.
http://www.eng.auburn.edu/department/cse/research/graph_drawing/graph_drawing.html.

[49] The GML File Format.
http://www.uni-passau.de/Graphlet/GML/.

[50] Michael Himsolt. *GML: A portable Graph File Format.* Technical Report, December 1996.

[51] The $Java^{TM}$ Development Kit (JDK^{TM}).
http://java.sun.com/products/jdk/.

[52] GraVis.
http://www-pr.informatik.uni-tuebingen.de/Forschung/
GraVis/GraVis.html.

[53] Eiffel.
http://www.eiffel.com/ oppure
http://www.elj.com/ oppure
http://www.cm.cf.ac.uk/CLE/.

[54] Ken Arnold, James Gosling
Java - Didattica e programmazione.
Addison Wesley Italia, 1997.

[55] *Java 1.1 Tutto & Oltre.*
Casa Editrice Apogeo Informatica.

[56] A. V. Aho, R. Sethi, J. D. Ullman. *Compilers. Principles, Techniques and Tools.* Addison Wesley, Reading, Mass., 1986.

[57] A. Panciatici, F. Ravaglia *Heterogeneous Computing: progettazione ed implementazione di un'algoritmo per il mapping di applicazioni parallele su metacalcolatori.* Tesi di Laurea, Università degli Studi di Pisa, Facoltà di Scienze MM. FF. NN., 1997.

[58] M. M. Eshaghian, A. C. Parker and Y. C. Wu *A Suboptimal Heterogeneous Mapping.* Tecnical Report, 1996. Departement of Computer and Information Science, Nwe Jersey Institute of Tecnology Newark, New Jersey. URL: `ftp://ftp.njit.edu/pub/cis/mary/Cluster-M/`

[59] *The GML File Format*, URL: `http://www.uni-passau.de/Graphlet/GML/`

[60] Michael Himsolt. *GML: A portable Graph File Format.* Technical Report, December 1996.

[61] GraVis URL: `http://www-pr.informatik.uni-tuebingen.de/Forschung/GraVis/ GraVis.html`"

[62] Eiffel URL: http://www-eiffel.com/ oppure http://www.elj.com/ oppure http://www.cm.cf.ac.uk/CLE

[63] Graph Drawing URL: http://www.fmi.uni-passau.de/ himsolt/graphdrawing.html

[64] CAD, Drawing & Painting Tools URL: http://sal.kachinatech.com/E/2/index.shtml

[65] Auburn University Drawing Graphs with VGJ URL: http://www.eng.auburn.edu/department/cse/research/graph_drawing/graph_drawing.html