UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTA' DI INGEGNERIA

DIPARTIMENTO DI TECNICA E GESTIONE DEI SISTEMI INDUSTRIALI

Tesi di Laurea Triennale

Software open source per la simulazione discreta: OMNeT++

Relatore: Ch.mo Prof. Giorgio Romanin Jacur

Laureando: Gianluca Zanin

Matricola: 5890893-IG

Anno Accademico 2010-2011

A mamma, papà e Chiara P.

INDICE

INTRODUZIONE

L'ingegnere di fronte a fenomeni naturali, ad esempio della fisica, o artefatti, generati cioè da meccanismi artificiali creati dall'uomo (ad esempio il comportamento di un calcolatore oppure di un processo di produzione manifatturiero), si trova continuamente nella necessità di descriverli, di impiegare queste descrizioni per le realizzazioni che gli sono richieste e, una volta che i sistemi sono funzionanti, di controllarli nella loro fase operativa (Carlucci, Menga 1998).

Negli ultimi trent'anni la teoria dei controlli automatici e con lei la teoria dei sistemi dinamici hanno avuto uno sviluppo eccezionale nell'ambito ingegneristico. Queste teorie, sviluppatesi nel settore dell'elettronica, hanno immediatamente trovato applicazioni con una diffusione a macchia d'olio; si può infatti affermare che oggi non esiste comparto tecnologico, che si deve confrontare con fenomeni dinamici, in cui queste siano sconosciute. Si è assegnato, all'idea di dinamica e al concetto di stato, il nome di sistemi dinamici ad eventi discreti, in quanto hanno un comportamento caratterizzato da uno stato che assume valori discreti nel tempo ed ha transizioni da uno dall'altro di questi valori in corrispondenza di particolari eventi temporali.

La simulazione è una tecnica molto importante di ricerca operativa, ed è forse la più usata in senso assoluto a supporto delle decisioni aziendali (Martinoli,1993).

Tramite la simulazione è possibile determinare la capacità di produzione di impianti e linee di produzione o lo stoccaggio di magazzini, gestire i colli di bottiglia, stabilire adeguati livelli delle scorte, migliorare i sistemi di preparazione degli ordini e ottimizzare i tassi di produzione. Inoltre è possibile testare scenari diversi per migliorare il processo prima di far funzionare il sistema nel mondo reale.

Applicando un modello appropriato di simulazione di un sistema è possibile valutare con precisione l'impatto di modifiche strutturali o di cambiamenti logico-organizzativi.

Un modello di simulazione di un sistema (es. produzione, movimentazione materiali, logistico) o anche solo di un processo può essere realizzato utilizzando **OMNeT++**, un software di simulazione ad eventi discreti.

Nei primi due capitoli di questa tesi si intende approfondire i concetti e le definizioni di modello e di simulazione di un sistema, mentre si dedicherà il capitolo successivo nel mostrare il funzionamento del software OMNeT++.

CAPITOLO 1
Classificazione dei sistemi e dei modelli

Negli ultimi vent'anni, con intensità sempre crescente alla stessa velocità di sviluppo delle tecnologie informatiche ed elettroniche, si è evidenziata la necessità di studiare i sempre più numerosi sistemi realizzati dall'uomo, tendenzialmente molto complessi, considerati non tradizionali rispetto alle trattazioni classiche proprie della Teoria dei Sistemi e del Controllo. Storicamente, infatti, detta disciplina ha trattato e tratta di sistemi a variabili continue modellati da equazioni differenziali o alle differenze e come tale sembra inadeguata a soddisfare la richiesta di un'appropriata rappresentazione dei sistemi suddetti.

Questi sono sistemi dinamici i cui stati assumono diversi valori logici o simbolici, piuttosto che numerici, in corrispondenza dell'occorrenza di eventi che non sempre possono essere descritti in termini numerici (processi produttivi, le reti di elaboratori, di trasporto, di comunicazioni e sistemi formati per integrazione delle suddette tipologie di sistemi).

Esempi di eventi sono: l'arrivo di un cliente nel sistema o la sua partenza da esso, il completamento di una lavorazione o il guastarsi di una macchina in un sistema di produzione, la trasmissione o la ricezione di un pacchetto di dati in una rete di telecomunicazioni, il verificarsi di un disturbo o il cambiamento del segnale di riferimento in un complesso sistema di controllo.

L'evoluzione nel tempo di un sistema con le caratteristiche suddette sembra essere naturalmente descritta da sequenze di occorrenze di cambiamenti discreti e qualitativi del sistema, ignorando i micro-cambiamenti che avvengono continuamente.

Sistemi con tali caratteristiche, posti in contrapposizione ai sistemi di avanzamento temporale, possono essere rappresentati come sistemi ad eventi discreti.

1. PRINCIPI DI BASE DELLA TEORIA DEI SISTEMI E DEL CONTROLLO

1.1. I CONCETTI DI SISTEMA E DI MODELLO

La definizione più generale del termine *sistema* fa riferimento ad un ente fisico che risponda alla sollecitazione esercitata da una certa azione producendo una certa reazione. In specifico, ad esempio, il dizionario IEEE definisce il sistema come "*combinazione di elementi che cooperano per svolgere una funzione altrimenti impossibile per ciascuno dei singoli componenti*", mentre il dizionario Webster come "*unità complessa formata da molte componenti, spesso diverse tra loro, soggette ad un piano comune o orientate verso un piano comune*".

Si possono riconoscere in queste definizioni, come in molte altre, aspetti comuni che individuano le principali caratteristiche che riconosciamo in un ente che chiamiamo sistema: esso, infatti, è tipicamente formato da diverse componenti interagenti che evolvono verso l'espletamento di una o più funzioni assegnate al sistema stesso.
Le definizioni date sono qualitative, e per sviluppare tecniche di progetto, di controllo e valutazione delle prestazioni di un sistema sulla base di specifiche predefinite, è fondamentale ottenere una rappresentazione quantitativa e possibilmente matematica di un sistema.
Si introduce, quindi, il *modello* che consente di riprodurre il comportamento del sistema per mezzo di qualche metodologia matematica.
Qualsiasi sistema fisico è caratterizzato da variabili fisiche che evolvono nel tempo, che si possono dividere in:

- *Cause esterne al sistema*: grandezze il cui andamento nel tempo può essere indipendente dal tipo di sistema;
- *Effetti*: grandezze il cui andamento nel tempo dipende almeno in parte dal tipo di sistema e dalle cause esterne.

Nella *teoria dei sistemi*, disciplina che tratta delle caratteristiche generali dei modelli matematici, le variabili cause esterne sono chiamate *ingressi del sistema* mentre le variabili effetti costituiscono le *uscite del sistema* .

La rappresentazione grafica è la seguente:

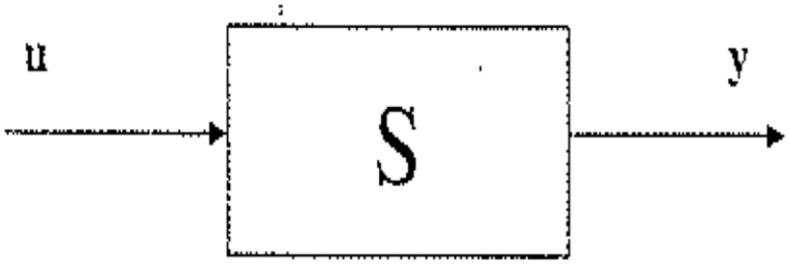


Figura 1

dove $u \in \Re^m$ è il vettore di dimensione m che contiene tutti gli ingressi del sistema e
$y \in$ è $\Re^p$ il vettore delle p uscite, le variabili effetti che interessano.
S può essere considerato come una specie di operatore, che crea una dipendenza degli effetti dalle cause esterne al sistema assegnando ad ogni specifico andamento degli ingressi il corrispondente andamento delle uscite. Nel momento in cui il modello approssima al meglio il sistema, si è giunti alla

progettazione di un modello adeguato. Conseguentemente la rappresentazione della realtà da analizzare potrà essere indicata indifferentemente con modello o sistema.

1.2. IL CONCETTO DI STATO

Legare l'uscita all'ingresso non è poi cosi semplice, in quanto l'uscita ad un dato istante di tempo può dipendere anche dalla storia del sistema.

Si deve introdurre per tale motivo lo *stato* che rappresenta il comportamento del sistema ad un dato istante di tempo, concentrando in sé l'informazione sul passato e sul presente del sistema. Tale grandezza ricopre un ruolo fondamentale e sta alla base del processo di modellazione e di molte tecniche analitiche. In termini formali lo stato di un sistema all'istante τ_0 è la grandezza che contiene l'informazione necessaria in τ_0 per determinare univocamente l'andamento dell'uscita $y(\tau)$, $\forall \tau \geq \tau_0$ sulla base della conoscenza dell'andamento dell'ingresso $u(\tau), \forall \tau \geq \tau_0$, e appunto dello stato in τ_0.

Lo stato si esprime in forma vettoriale, come l'ingresso e l'uscita , nella forma $x \in \Re^n$, e i suoi elementi sono detti *variabili di stato*. L'insieme di tutti i valori che lo stato può assumere viene definito *spazio di stato* di un sistema (X).

Si definiscono equazioni di stato l'insieme di equazioni che determinano lo stato $x(\tau)$ $\forall \tau \geq \tau_0$ sulla base di $x(\tau_0)$ e di $u(\tau)$, $\tau \geq \tau_0$. Attraverso queste equazioni è possibile definire la *dinamica* del sistema, ovvero le relazioni che legano le variabili di ingresso, stato, e uscita.

Tali equazioni possono assumere varie forme; nell'ambito della teoria dei sistemi e del controllo, l'operatore S della figura precedente è costituito tipicamente dal processo di soluzione di un sistema di equazioni differenziali o alle differenze. In particolare, se il sistema è a tempo continuo, S corrisponde alla risoluzione di un sistema di equazioni di struttura:

$x(\tau) = f(x(\tau),u(\tau),\tau)$, $x(\tau 0)=x0$

$y(\tau) = g(x(\tau),u(\tau),\tau)$

che è un modello nello spazio di stato, costituito da un sistema dinamico a tempo continuo. Dato che le variabili di ingresso, di uscita e di stato sono scelte in base agli obiettivi dello studio del sistema che si vuole modellare, tale modello non è mai unico.

Esso può essere rappresentato graficamente nel modo seguente:

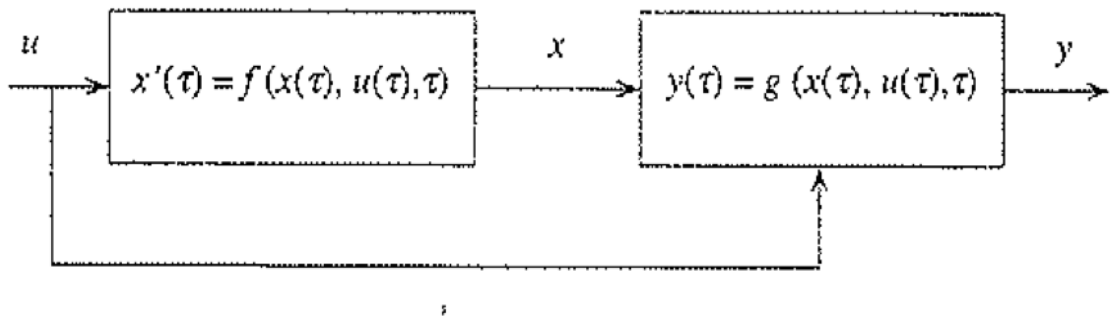


Figura 2

Se, invece, il tempo è discreto si ha un *sistema dinamico a tempo discreto* costituito dalla seguente struttura:

$x(k+1) = f(x(k), u(k), k)$, $x(k_0) = x_0$

$y(k) = g(x(k), u(k), k)$

1.3 IL CONCETTO DI CONTROLLO

In molti casi si vuole giungere alla modificazione di un sistema in modo tale che il suo comportamento migliori secondo criteri definiti. Per fare ciò si devono ricercare delle tecniche che realizzino tale cambiamento. Si parla, quindi, di *controllo* del sistema, che permette di modificare gli ingressi manipolabili del sistema dinamico sotto studio per far si che il suo comportamento si avvicini a quello desiderato. Ad esempio nel caso del robot, dimensionate le inerzie dei bracci, scelte le potenze dei motori adeguate ad imprimere accelerazioni sufficienti a realizzare transitori dinamici nei tempi specificati per le missioni, senza un adeguato controllo con azioni coordinate sui giunti del robot, sarà impossibile fare compiere movimenti anche solo significativi al carico trasportato.

Strutture del controllo:

- *Controllo in anello aperto*: l'azione di controllo non dipende dallo stato attuale del sistema. Il segnale di ingresso è generato da un processo esterno al sistema stesso, attraverso diverse procedure: dall'analisi, da qualche dispositivo fisico, da qualche fenomeno casuale.

  Esempio: Pensiamo alla gestione della produzione a fronte di un picco di domanda. La domanda è vista qui come un disturbo, in quanto le sue variazioni alterano il livello dei magazzini rispetto ai valori normali. Con il modello del sistema, dalla previsione della domanda è possibile determinare direttamente le necessarie correzioni ai tassi di produzione per mantenere i livelli dei magazzini.

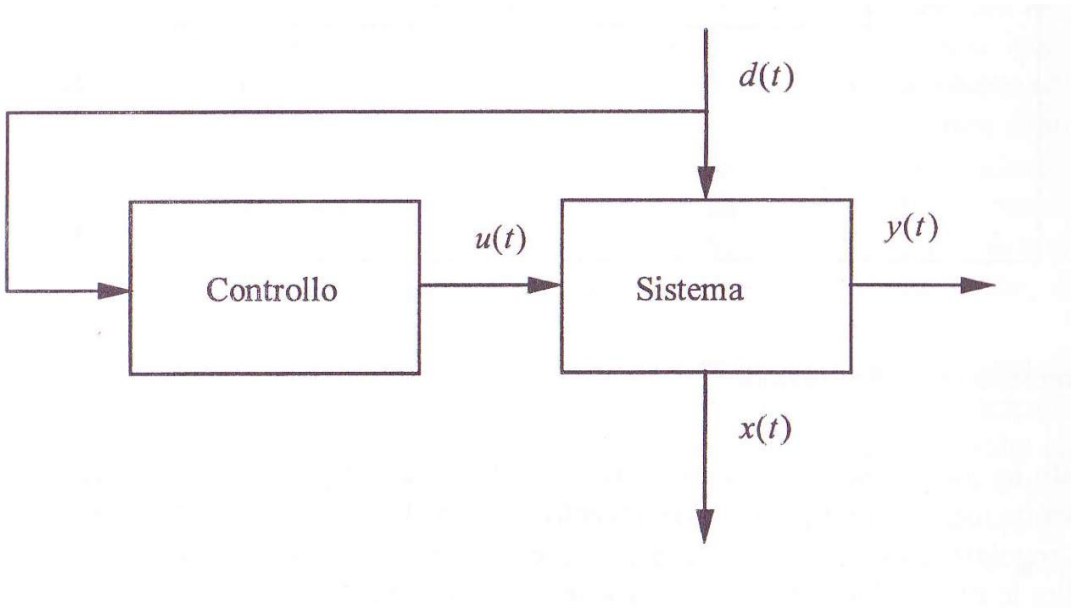


Figura 3

- *Controllo in anello chiuso*: l'azione di controllo dipende dallo stato attuale del sistema. Il segnale di ingresso è determinato continuamente sulla base del comportamento del sistema, attraverso la retroazione. Nel controllo ad anello chiuso le uscite del sistema sono riportate come ingressi. Un esempio di tale struttura è il riscaldamento domestico: l'uscita di questo sistema è la temperatura all'interno della casa, che viene misurata dal termostato e, quando la temperatura scende sotto un determinato livello la caldaia si accende, per spegnersi quando raggiunge il valore desiderato.

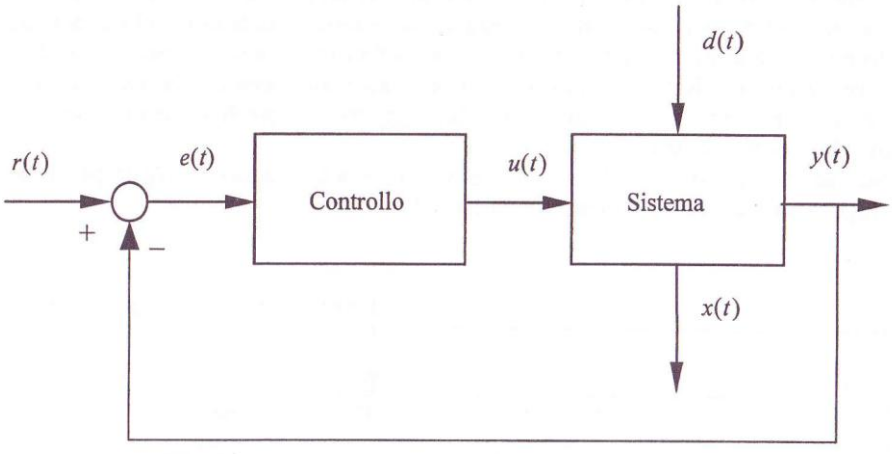


Figura 4

Il controllo in anello chiuso è preferibile a quello aperto in quanto è caratterizzato da minore complessità di implementazione e migliori prestazioni, tanto che la retroazione può produrre una reazione automatica a cambiamenti del sistema inaspettati o a ingressi impredicibili.

2. SISTEMI AD EVENTI DISCRETI

Un sistema ad eventi discreti si può definire come un sistema dinamico il cui comportamento è caratterizzato dall'occorrenza di eventi istantanei con un cadenzamento irregolare non necessariamente noto. Ad esempio si possono trovare sistemi di questo tipo nel settore della produzione, della robotica, del traffico, della logistica e delle reti di elaboratori elettronici e di comunicazione.

Dato che queste applicazioni richiedono controllo, i sistemi ad eventi discreti si propongono come soggetto opportuno per la teoria del controllo. I sistemi ad eventi discreti sono modulari, ovvero composti di elementi che vanno dalla complessità di sistemi ad eventi discreti quasi indipendenti al livello di semplici primitive e possono avere meccanismi di controllo e intercomunicazione, ad esempio per l'abilitazione o la disabilitazione di particolari eventi controllabili e per la segnalazione dell'occorrenza di eventi osservabili da un modulo ad un altro. Attraverso azioni di coordinamento dei meccanismi di controllo e di comunicazione si può garantire che il flusso degli eventi soddisfi le specifiche desiderate.

Fino a pochi decenni fa i sistemi ad eventi discreti erano piuttosto semplici e potevano venire rappresentati e studiati attraverso metodi intuitivi. Con il tempo, però, i sistemi creati dall'uomo sono diventati via via più complessi, anche grazie al veloce sviluppo delle tecnologie legate ai computer. Conseguentemente il livello di complessità dei problemi da affrontare negli ambiti di modellazione, analisi, ottimizzazione e controllo dei sistemi ad eventi discreti è salito cosi tanto che si sono resi necessari metodi più sistematici per le loro analisi e soluzioni.

Per la risoluzione a questi problemi esistono oggi differenti metodologie, nessuna di queste, però, si può considerare indiscutibilmente consolidata, in quanto la teoria dei sistemi ad eventi discreti continua a rimanere oggetto di attività di ricerca scientifica di interesse internazionale.

L'evoluzione della teoria dei sistemi ad eventi discreti sta avvenendo in forte analogia con lo sviluppo della teoria dei sistemi e del controllo. In particolare, raggiungibilità, l'osservabilità, le risposte nel tempo e in frequenza, hanno giocato e continueranno a giocare ruoli molto importanti nello sviluppo dei modelli e degli strumenti di analisi e di controllo per i sistemi ad eventi discreti.

La definizione formale di un sistema ad eventi discreti è la seguente: esso è un sistema il cui comportamento dinamico è caratterizzato dall'accadimento asincrono di eventi che individuano lo svolgimento di attività di durata non necessariamente nota. Formalmente, un sistema ad eventi discreti è caratterizzato da:

- Insieme E degli eventi accadibili;
- Spazio di stato costituito da un insieme discreto X;

- Evoluzione dello stato event-driven, cioè regolata dagli eventi: lo stato evolve nel tempo solo in dipendenza dell'accadimento di eventi asincroni, appartenenti all'insieme E.

L'equazione che descrive l'evoluzione dello stato a partire dallo stato iniziale x0 è

$x_{k+1} = \delta(x_k, ek)$ $\quad k \in N$

dove

- X_{k+1} è lo stato del sistema dopo l'accadimento del k-esimo evento;
- e_k è il k-esimo evento accaduto dall'istante iniziale considerato, che fa transire lo stato da x_k a x_{k+1};
- $\delta: X \times E \rightarrow X$ è la funzione di transizione di stato.

3. MODELLAZIONE DI SISTEMI AD EVENTI DISCRETI

I sistemi ad eventi discreti vengono astratti in Modelli ad Eventi Discreti (MED), i quali sono modelli matematici in grado di rappresentare l'insieme delle tracce generate da un sistema. Le tracce non sono altro che registrazioni di occorrenze di determinati eventi discreti.
Dato che nella maggior parte dei sistemi l'insieme delle tracce può essere infinto, un MED è un modello matematico finito che descrive un insieme finito di tracce di un sistema ad eventi discreti.
Esistono due grandi famiglie di modelli di sistemi ad eventi discreti: i modelli logici e i modelli temporizzati. Nei modelli logici la traccia degli eventi è costituito semplicemente da una sequenza di eventi $\{e_1, e_2, \ldots\}$, in ordine di occorrenza, senza alcuna informazione circa i tempi di occorrenza degli eventi; dato uno stato iniziale x_0, la traiettoria dello stato verrà costruita nel tempo come la sequenza di stati $\{x_0, x_1, x_2, \ldots\}$ risultanti dall'accadimento della sequenza di eventi, ma non è possibile specificare gli istanti di tempo in cui avvengono le transizioni di stato.
Nei modelli temporizzati, invece, la traccia degli eventi è costituita da una sequenza di coppie $\{(e_1, \tau_1),(e_2, \tau_2), \ldots\}$, dove ogni evento ei è accoppiato al suo tempo di accadimento τi, eventualmente stocastico; dato uno stato iniziale x0, la traiettoria dello stato sarà evidentemente ancora la sequenza di stati $\{x_0, x_1, x_2, \ldots\}$ risultanti dall'accadimento della sequenza di eventi; in questo caso però si sa che le transizioni di stato avvengono negli istanti di occorrenza degli eventi.

3.1. MODELLI AD EVENTI DISCRETI LOGICI

I modelli discreti ad eventi discreti logici ignorano i tempi di occorrenza degli eventi, considerando solamente l'ordine con cui essi si verificano. Vengono, quindi, usati questi tipi di modelli quando si vogliono studiare le proprietà dei sistemi ad eventi discreti che non dipendono dalla specifica temporizzazione degli eventi. Una traiettoria del sistema è specificata semplicemente costruendo una stringa di eventi in base all'ordine in cui accadono.

E' fondamentale specificare le sequenze di eventi fisicamente realizzabili nella formulazione del modello. Di solito nei sistemi reali le tracce di eventi ammissibili sono un sottoinsieme ristretto di tutte le sequenze di eventi possibili. Scopo di tutto ciò è determinare quali fra le traiettorie ammissibili hanno certe proprietà specificate.
Per quanto riguarda il controllo, ogni traiettoria ammissibile può assumere le proprietà desiderate, variando attraverso l'uso di azioni di controllo l'insieme delle tracce ammissibili. Alcune proprietà di interesse sono: corretto uso delle risorse, coordinamento dei processi costituenti nel perseguire un determinato obiettivo, stabilità, comportamento dinamico soddisfacente, etc.
I modelli ad eventi discreti logici sono utilizzati come base di calcolo (verifica o sintesi dei sistemi ad eventi discreti), e per lo studio delle proprietà qualitative del sistema.

3.2. MODELLI AD EVENTI DISCRETI TEMPORIZZATI

Il modello ad eventi discreti temporizzati, al contrario di quello logico, associa alla sequenza degli eventi una qualche forma di temporizzazione.
La formulazione del modello procede con la specificazione dell'insieme delle tracce di eventi ammissibili e la temporizzazione ad esse associata.
Adottando un modello deterministico, quest'operazione può essere ancora effettuata tramite una struttura opportuna (es. automi temporizzati, reti di Petri temporizzate).
Adottando, invece, un modello stocastico bisogna fare attenzione, in quanto risulta semplice specificare l'insieme delle traiettorie di stato ammissibili ma complicato definire una misura utile di questo insieme da usare per determinare le distribuzioni delle variabili di interesse. Il limite di questo modello risiede nel fatto che, è impossibile risolvere in forma chiusa parecchi problemi di interesse, in quanto tali modelli sono cosi complicati da impedirne una trattazione analitica. Inoltre, anche nel caso in cui si conoscessero soluzioni in forma chiusa, la forma della soluzione potrebbe risultare cosi complicata da rendere il suo uso quasi impossibile.

4. SISTEMI IBRIDI

I sistemi ibridi sono sistemi complessi formati da componenti il cui comportamento è caratterizzato da funzioni del tempo e da componenti la cui evoluzione temporale avviene sulla base dell'accadimento di eventi istantanei asincroni.
In natura esistono numerosi tipi di processi ibridi, come conseguenza si rende necessario l'uso di modelli ibridi.

Pensiamo ad esempio al comportamento del traffico stradale. Finché esso rimane regolare, questo sistema può venire descritto da equazioni differenziali.

Nel momento in cui ci fossero delle irregolarità che potrebbero influenzare il suo comportamento, il sistema dovrebbe essere rappresentato da un modello ad eventi discreti.

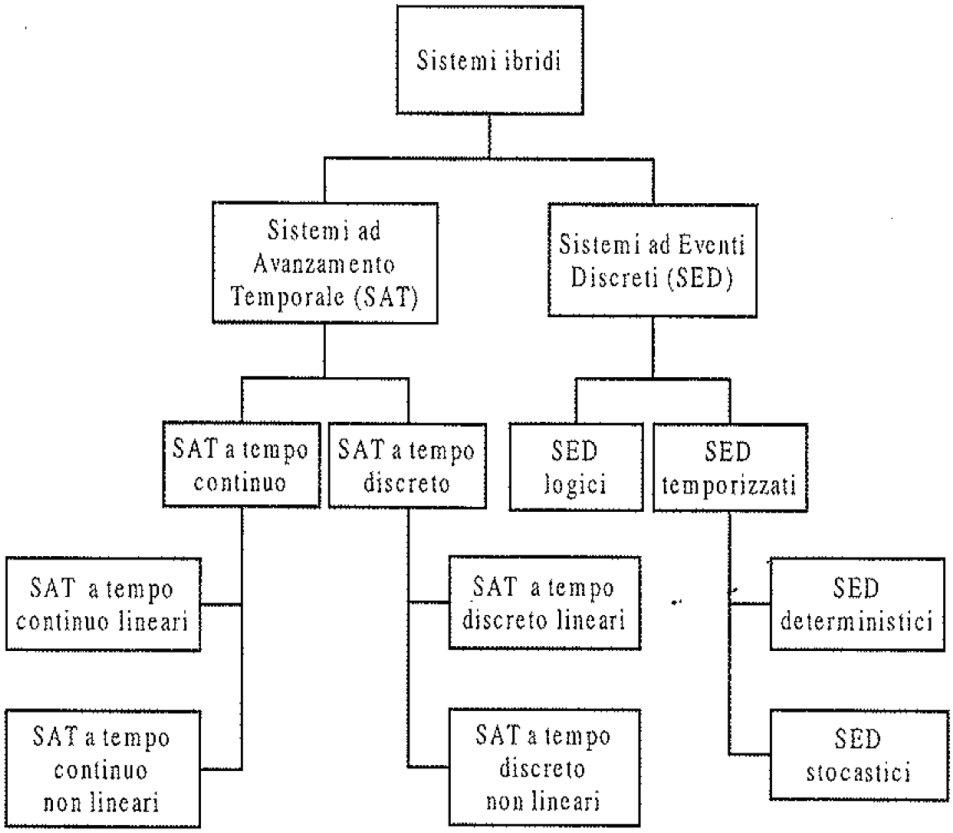


Figura 5

CAPITOLO 2
Simulazione Discreta

Nel capitolo precedente è stato introdotto il concetto di modellazione di sistemi ad eventi discreti; in molti casi tali modelli vengono effettivamente adottati e possono essere trattati per mezzo della teoria della probabilità, di metodi algebrici o di altre tecniche matematiche, al fine di valutare numericamente le misure delle prestazioni del sistema ritenute di volta in volta di interesse. Spesso però accade che la complessità del sistema reale, che si sta analizzando, sia tale da rendere tutte tipologie di modelli praticamente impossibili da risolvere matematicamente.

È in questi casi che tipicamente si fa ricorso alla simulazione, per valutare numericamente il modello del sistema e stimare le grandezze ritenute d'interesse.

Con un programma di simulazione si riproduce, in un ambiente controllato, il funzionamento di un sistema reale al fine di analizzare il suo comportamento nelle diverse condizioni possibili. Per lo studio dell'evoluzione dinamica del sistema va comunque sviluppato un modello, che in questo ambito è ovviamente un modello simulativo. (Di Febbraro *et al.*, 2001; Banks *et al.*, 1996; Fishman,1978)

In generale, la simulazione può essere utilizzata come strumento di ausilio sia per l'analisi sia per la sintesi di sistemi dinamici: permette non solo di effettuare misure delle prestazioni di sistemi al variare di caratteristiche strutturali e/o prestazionali, ma anche di valutare i diversi comportamenti di sistemi in fase di progetto al variare delle condizioni operative.

Si può affermare che l'utilità della simulazione si concretizza nelle possibilità di:

- studiare e analizzare le interazioni tra le singole componenti di sistemi comunque complessi;
- valutare l'impatto che avrebbero potenziali cambiamenti apportabili a un sistema esistente prima di realizzarli;
- valutare le prestazioni che sistemi in fase di progetto avrebbero in differenti condizioni di funzionamento;
- verificare eventuali risultati analitici già ottenuti con altre metodologie di studio.

Per poter realizzare una simulazione corretta ed efficace si deve passare attraverso precise fasi di progettazione.

Formulazione del problema: è la base di ogni studio di simulazione; infatti il problema da affrontare deve essere enunciato nella forma corretta e consona agli obiettivi, per la riuscita di uno studio simulativo.

Creazione del modello: Non ci sono regole fisse e predeterminate per creare un modello adeguato del sistema di cui si intende simulare il comportamento; questa fase si basa sull'abilità di astrarre gli aspetti essenziali di un problema, di individuare e poi variare le ipotesi di funzionamento del sistema da modellare e infine di arricchire ed elaborare il modello fino al raggiungimento di risultati soddisfacenti. In generale, è sempre opportuno partire da un modello semplice per poi articolarlo sempre più, ma solo finché ciò appare necessario per gli scopi per cui il modello viene creato; non è necessario avere una corrispondenza perfetta tra modello e sistema reale, ma basta cogliere gli aspetti fondamentali;

Raccolta dei dati: La fase di raccolta dei dati da utilizzare come ingressi per un programma di simulazione e quella di costruzione del modello sono strettamente legate, dal momento che il tipo e la quantità dei dati di ingresso necessari dipendono dalla complessità assunta dal modello, oltre che, ovviamente, dagli obiettivi dello studio di simulazione. Così, per esempio, se si intende simulare il funzionamento di un sistema a coda, sarà necessario conoscere le distribuzioni degli intervalli di tempo tra i tempi di arrivo dei clienti e i tempi di servizio; se poi l'obiettivo fosse valutare l'andamento della lunghezza della coda al variare del numero di serventi, sarebbe indispensabile anche disporre di dati storici sulle distribuzioni della lunghezza della coda nelle diverse condizioni, da utilizzare nella fase di validazione del modello, descritta più avanti.

Traduzione del modello: una volta realizzato il modello del sistema che si vuole simulare, la fase immediatamente successiva è “tradurlo” in un linguaggio che sia comprensibile ad un elaboratore elettronico. La scelta da compiere a questo punto è tra usare un linguaggio di simulazione di tipo generale (esempi: SIMSCRIPT, MODSIM, GPSS/H, SIMAN V), oppure un programma di simulazione specialistico, diverso a seconda del tipo di sistema sotto studio. I linguaggi di simulazione generali sono usualmente più potenti e ovviamente più flessibili nell'uso, mentre quelli specialistici consentono di sviluppare il modello di simulazione in tempi minori, proprio perché predisposti alla rappresentazione di una specifica tipologia di sistema. Nell'ambito dell'automazione per esempio, per la simulazione ad eventi discreti sono molto utilizzati ARENA, EXTEND, SIMFACTORY.

Verifica: consiste nella verifica del funzionamento del modello software sviluppato; la verifica riguarda soprattutto la struttura logica del programma di simulazione e i moduli di interfaccia di ingresso/uscita.

Convalida: consiste nel determinare se il modello di simulazione rappresenta correttamente il funzionamento del sistema reale; ciò si effettua valutando gli scostamenti tra i risultati forniti dal

modello e quelli del sistema vero, che vengono usati per migliorare iterativamente la qualità della riproduzione del funzionamento del sistema reale, finché questa non viene ritenuta accettabile.

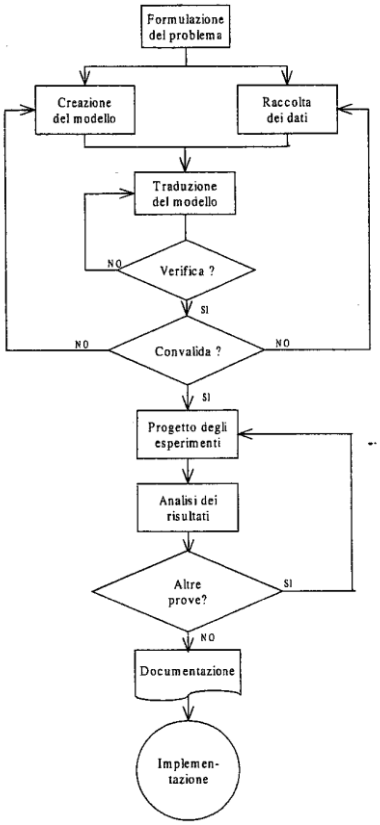


Figura 6

Progetto degli esperimenti: si occupa della determinazione delle alternative da simulare, che spesso dipende dagli esperimenti simulativi già effettuati. In pratica, per ogni modello realizzato, bisogna stabilire il periodo di inizializzazione, la durata del periodo che si vuole simulare e il numero di repliche da effettuare per ogni prova simulativa.

Analisi dei risultati: le prove simulative vengono analizzate per stimare le prescelte misure di prestazioni del sistema; sulla base dell'analisi dei risultati si determina se sono necessarie altre prove simulative.

Documentazione: è opportuno allegare allo studio di simulazione una documentazione adeguata, che consenta di comprendere come le decisioni sono state prese e soprattutto dove intervenire qualora si desideri modificare parametri del modello, misure delle prestazioni, o altre caratteristiche statiche della campagna di simulazione.

Implementazione: se ogni passo precedente è stato compiuto correttamente, si potrà azionare il simulatore senza incorrere in errori o problemi.

1. CONCETTI E PRINCIPI DI FUNZIONAMENTO

In un modello ci sono elementi fondamentali da identificare per lo sviluppo della simulazione di un sistema reale:

- *stato del sistema*: insieme di variabili che contengono tutte le informazioni necessarie a descrivere il sistema in ogni istante;
- *eventi*: avvenimenti istantanei che cambiano lo stato del sistema; un evento si definisce condizionato o dipendente se coincide sempre con un altro evento, primario o indipendente altrimenti;
- *lista (calendario) degli eventi futuri*: lista degli eventi di cui si sono già potute programmare le occorrenze, posti in ordine cronologico;
- *entità*: componente del sistema che richiede una rappresentazione esplicita nel modello;
- *attributi*: proprietà di una data entità;
- *insieme:* gruppo permanente o temporaneo di entità organizzate secondo qualche logica;
- *attività*: periodi di tempo di durata specificata (espressi tramite statistiche);
- *ritardi*: periodi di tempo di durata non specificata;
- *orologio*: variabile che rappresenta il tempo simulato.

Durante la progettazione di un modello per la simulazione ad eventi discreti bisogna individuare il minimo numero di tipologie di eventi, che caratterizzano il funzionamento del sistema; per fare ciò ci si può appoggiare a uno dei due tipi di ragionamento seguenti. Adottare un'*analisi longitudinale*, o *per processi*, degli eventi significa rappresentare la dinamica del sistema, quindi la sequenza degli eventi, in termini di flussi di entità transienti, come ad esempio il movimento di pezzi in un sistema di produzione; in altre parole, si descrivono le azioni che le entità che si muovono nel sistema subiscono.

Il secondo metodo di ragionamento che si può adottare è l'*analisi trasversale* degli eventi, con cui si rappresenta la dinamica del sistema in termini di cicli di entità residenti, come per esempio le risorse di lavorazione in un sistema di produzione; la sequenza degli eventi è individuata sulla base delle sequenze dei cicli che ogni entità residente esegue e dei metodi di interazione con le altre entità del sistema.

In conclusione sarebbe ottimale svolgere entrambe le analisi e rappresentare

anche gli eventi che iniziano un'attività, per poi eliminarli in seconda battuta, in modo da evidenziare le relazioni causa-effetto all'interno del sistema analizzato.

Un concetto importante di cui tener conto è che la simulazione avanza con la scansione del calendario degli eventi futuri riproducendo l'accadimento di una sequenza di eventi. L'accadimento di un evento modifica sicuramente l'orologio della simulazione (*clock*) e la lista degli eventi futuri, in cui viene almeno cancellato l'evento accaduto, mentre altri eventi possono essere inserito cancellati in conseguenza dell'accadimento dell'evento considerato; infine gli eventi futuri nella lista

vengono riordinati in ordine crescente di tempi di accadimento. Solitamente poi l'accadimento di un evento provoca un cambiamento nello stato del sistema e un aggiornamento dei dati che si stanno raccogliendo per le statistiche che formeranno i risultati della simulazione.

Quello che segue è un esempio di creazione di un modello per la simulazione ad eventi discreti che fa riferimento a un semplice sistema costituito da una coda.

Si considera un sistema a coda in cui clienti provenienti da una determinata popolazione si rivolgono, uno alla volta e arrivando a intervalli casuali, a un servente; se un cliente non può essere servito appena arrivato, si accoda e attende il suo turno; appena il suo servizio finisce, il cliente lascia il sistema; per semplicità si suppone che lo stato del servente possa assumere solo i valori "attivo" e "inattivo", ovvero "occupato" e "libero", tralasciando la gestione di possibili interruzioni della sua operatività.

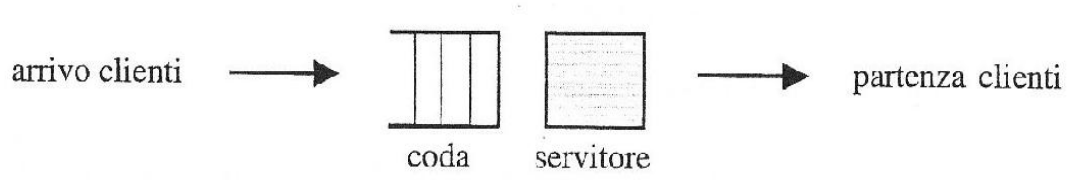


Figura 7

Nella fase di progettazione degli esperimenti simulativi, è necessario:

1. determinare se il sistema viene periodicamente reinizializzato, per esempio a causa

di chiusura notturna, oppure ritorna a uno stato di riferimento, per esempio si svuota,

oppure raggiunge un regime;

2. definire se è richiesto studiare il comportamento del sistema nel transitorio o a regime;

3. stabilire il numero e la durata degli esperimenti in base alle risposte precedenti.

Gli elementi di base di ogni modello simulativo (che sono stati spiegati precedentemente), che dipendono dal sistema che si sta simulando, in questo caso sono particolarizzati come segue:

- stato del sistema: numero di clienti in coda (compreso il cliente che sta ricevendo il servizio), operatività del servente (occupato o libero);
- entità: generatore di clienti, clienti, coda (servente + spazio di accodamento corrispondente);
- insieme: clienti accodati;
- eventi: arrivo di un cliente, inizio di un servizio (evento dipendente), partenza di un cliente, fine della simulazione;
- attività: tempo di interarrivo, tempo di servizio;
- ritardo: tempo di attesa in coda.

Poiché si può escludere l'inizio di un servizio, essendo un evento dipendente, le tipologie di eventi indipendenti ricorrenti rimangono l'arrivo di un cliente nel sistema e la partenza da esso, che coincide con la fine del servizio corrispondente; si è detto che all'accadimento di ogni tipo di evento sono associate determinate azioni; in particolare, le azioni che vengono effettuate in concomitanza con l'arrivo e la partenza di un cliente sono elencate di seguito e schematizzate nei diagrammi di flusso corrispondenti, rappresentati rispettivamente nelle figure seguenti:

Figura 8

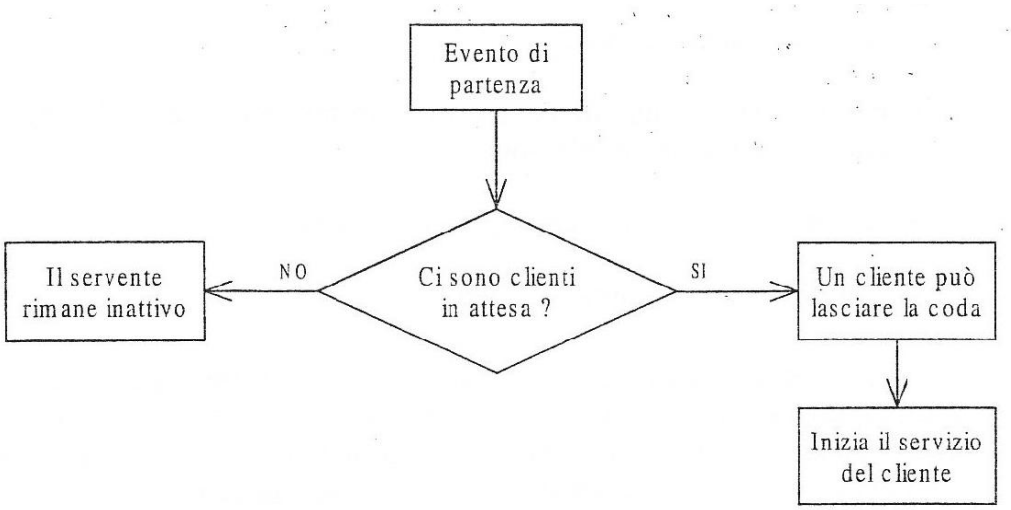


Figura 9

Per la preparazione completa di una simulazione è necessario identificare e calcolare gli indici prestazionali del sistema che si vuole misurare, che solitamente sono:

1. Tempo medio di permanenza nel sistema.

2. Utilizzazione del servente.

3. Lunghezza media della coda.

Nelle ipotesi che la simulazione inizi all'istante τ e termini nell'istante τ_f dopo che *N* clienti, con *N* noto, sono stati serviti dalla coda, le misure prestazionali elencate possono essere definite come segue.

- Tempo medio di permanenza nel sistema

Detto $\overline{\theta}$ il valor medio della variabile aleatoria che rappresenta il tempo medio di permanenza nel sistema, si può ottenere una sua stima $\hat{\theta}$ sulla base dei tempi di permanenza θ_1 ,..., θ_N dei singoli clienti, per mezzo della relazione

che consiste nella semplice $\hat{\vartheta} = \frac{1}{N}\sum_{k=1}^{N} \vartheta_k$ media aritmetica dei tempi di permanenza degli N clienti nel sistema.

- Utilizzazione del servente

Si denoti con υ la probabilità che il server sia occupato durante il periodo di tempo

$T_N = \tau_f - \tau_0$ necessario a completare il servizio degli N clienti, che corrisponde al periodo di tempo in cui la coda è non vuota; definendo inoltre T(i) il tempo in cui la lunghezza della coda è i, i = 0,1…., la stima di υ è data da

$$\hat{\upsilon} = \frac{1}{T_N} \sum_{i=1}^{\infty} T(i) = 1 - \frac{T(0)}{T_N}$$

- Lunghezza media della coda

Detta p_N la probabilità che la lunghezza della coda sia i, i = 0,1, ..., nell'intervallo di tempo necessario a servire gli N clienti, la lunghezza media $\hat{\theta}$ della coda nello stesso

intervallo di tempo si calcola come

$$\bar{x} = \sum_{i=1}^{\infty} i p_N(i)$$

la cui corrispondente stima risulta essere

$$\hat{x} = \sum_{i=1}^{\infty} i \hat{p}_N(i) = \frac{1}{T_N} \sum_{i=0}^{\infty} i T(i)$$

A questo punto è opportuno introdurre qualche concetto sulla **generazione di variabili** in generale e di numeri casuali, quest'ultimi componente indispensabile della simulazione dei sistemi ad eventi discreti; infatti la maggior parte dei linguaggi di simulazione è in grado di generare i numeri casuali necessari per determinare i tempi di accadimento di eventi e i valori di altre variabili aleatorie coinvolte nella simulazione di un sistema ad eventi discreti.

Data una sequenza di numeri casuali, R_1,R_2, . . ., questa deve possedere due importanti proprietà statistiche: *uniformità* tra 0 e 1 e *indipendenza*; in altre parole, ogni numero casuale R_i deve essere un campione indipendente estratto da una distribuzione continua uniforme tra 0 e 1. Uniformità e indipendenza implicano due importanti proprietà:

- dividendo l'intervallo (0,1) in n (sotto)intervalli (classi) di uguale ampiezza, il numero di osservazioni atteso in ogni intervallo è N / *n*, se N e il numero totale di osservazioni;
- la probabilità che in un dato intervallo si estragga un certo campione è indipendente dai campioni estratti in precedenza.

E' palese però, come il solo fatto di generare dei numeri casuali per mezzo di una procedura nota cancella la possibilità di avere vera casualità; infatti, i numeri generati da una procedura non possono che essere *pseudo-casuali*, cioè imitano soltanto le proprietà dei numeri casuali. L'obiettivo della procedura di generazione è dunque produrre una sequenza di numeri tra 0 e 1 che simula, il meglio possibile le proprietà statistiche ideali di distribuzione uniforme ed indipendenza.

In generale, una procedura per la generazione dei numeri casuali deve essere veloce,

avere un ciclo (sequenza di numeri generati che si ripete periodicamente) sufficientemente lungo, non avere ampi intervalli (i cosiddetti *gaps*) tra due numeri generati in successione, generare sequenze replicabili, oltre che, ovviamente, generare numeri con proprietà vicine il più possibile a quelle ideali.

E' un obiettivo molto difficile riuscire a realizzare una tecnica di generazione dei numeri casuali che riesca a comprendere le caratteristiche appena elencate. Solamente a scopo informativo, per non allontanarsi dal tema del capitolo, qui sotto viene descritto brevemente il metodo della congruenza lineare.

Metodo della congruenza lineare. E' la procedura di generazione dei numeri casuali più nota ed applicata, ed è in grado di generare una sequenza di numeri interi X_1, X_2,... tra 0 e m – 1 sulla base della relazione ricorsiva.

$$X_{i+1} = (aX_i + c) \mod m, \qquad i \in \mathbb{N}$$

caratterizzata dai parametri seguenti:

- X_0 : seme;
- a : moltiplicatore;
- c : incremento;
- m : modulo.

E' evidente che la scelta dei valori dei parametri elencati influenza pesantemente le proprietà statistiche della sequenza generata e la lunghezza del ciclo, preferibile più grande possibile.

Per ciò che riguarda la **definizione dei dati di ingresso**, in tutti i casi in cui non sia possibile individuare una distribuzione nota che sia in grado di modellare in maniera adeguata i dati da

utilizzare per la simulazione, è necessario utilizzare la loro distribuzione empirica, continua o discreta che sia.

Nello sviluppo di un modello dei dati di ingresso si distinguono le quattro fasi seguenti:

1. raccolta dei dati dal sistema reale sotto studio;
2. identificazione di una funzione di distribuzione per rappresentare i dati di ingresso;
3. identificazione dei parametri (deterministici e/o stocastici) che caratterizzano una specifica realizzazione della famiglia di distribuzioni selezionata;
4. valutazione della funzione di distribuzione scelta e dei parametri associati.

La prima fase, la raccolta dei dati relativi al sistema reale, è uno dei problemi più importanti e difficili da risolvere nella simulazione; infatti, anche se la struttura del modello è valida, se i dati di ingresso non vengono raccolti e/o analizzati in modo adeguato, o non sono rappresentativi dell'ambiente, i risultati della simulazione risulteranno fuorvianti, il che può ovviamente produrre danni significativi se sulla base di dati tali si mettono in atto strategie decisionali.

Ultimata la raccolta dei dati relativi al sistema che si vuole studiare, è necessario

individuare il metodo più adatto, tra i diversi esistenti, per scegliere la famiglia di distribuzioni statistiche che meglio può rappresentare i dati disponibili; nell'ambito della famiglia individuata viene in seguito selezionata una particolare distribuzione per mezzo di procedure di stima dei parametri. Infine, viene effettuata un'operazione di verifica, per mezzo di test appositi, dell'adeguatezza della funzione scelta per la rappresentazione dei dati di ingresso.

Il processo di **verifica e validazione dei modelli simulativi** riguarda la corretta costruzione del modello, mentre il processo di convalida (o di validazione) riguarda la costruzione del modello corretto.

Nella definizione di un modello simulativo si compiono i seguenti tre passi:

1. osservazione del sistema reale e delle interazioni trai suoi componenti e raccolta di informazioni sul suo comportamento;
2. costruzione di un modello concettuale;
3. trasformazione del modello concettuale in modello "operativo" riconoscibile da computer.

Le relazioni tra le fasi suddette sono rappresentate dal diagramma seguente.

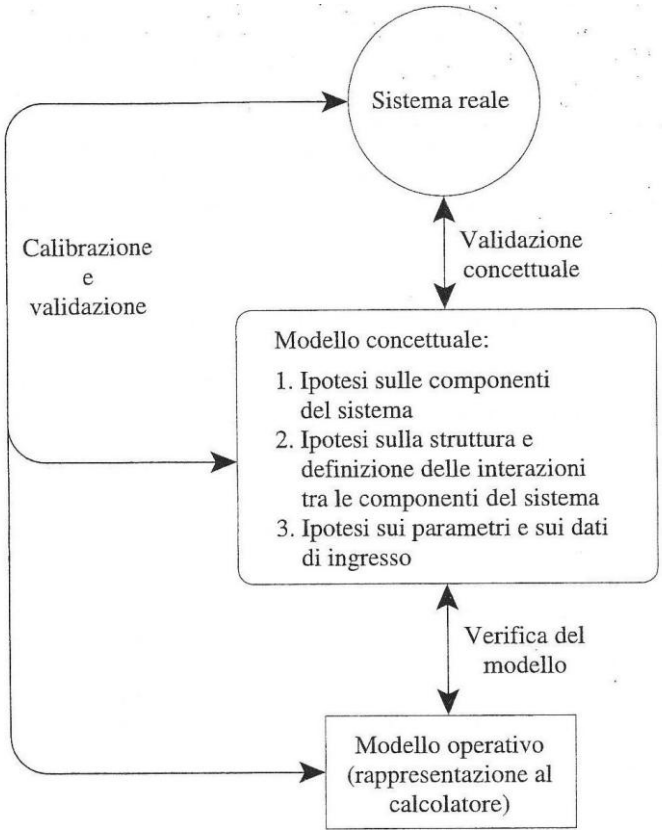


Figura 10

Alcune regole pratiche per la *verifica* sono:

- seguire la sequenza degli eventi;
- sviluppare il diagramma di flusso e verificare il codice in modo da controllare tutte le diramazioni;
- in casi semplici confrontare i risultati ottenuti dal programma di simulazione con quelli ottenuti a mano;
- controllare la funzionalità di ogni singolo oggetto;
- controllare l'assegnazione dei semi dei generatori dei numeri casuali;
- verificare graficamente l'evoluzione temporale di caratteristiche che potrebbero presentare instabilità.

Per esempio si può isolare una risorsa di lavorazione di cui sono noti i tempi di

interarrivo e tempi di servizio e verificare la media e la varianza dei tempi tra due uscite e la stabilità della coda.

La fase di validazione si svolge in genere contemporaneamente alla fase di verifica e prevede regole analoghe, ma in questo caso il confronto avviene con il sistema reale anziché con il modello.

Regole pratiche per la *validazione* sono:

- sviluppare il modello con persone che conoscono il sistema;
- rappresentare il modello graficamente;
- raccogliere dati storici o realistici e controllare che le trasformazioni degli ingressi nelle uscite conducano a risultati statisticamente simili;
- controllare che si possano verificare le condizioni di accadimento di ogni evento e di verifica delle condizioni logiche previste;
- controllare la funzionalità di ogni oggetto, eventualmente realizzando qualche esperimento anche su parti del sistema reale;
- eseguire analisi di sensitività per calibrare correttamente i parametri.

La calibrazione è un processo iterativo di confronto tra realtà e modello al fine di determinare correttamente i valori dei parametri in gioco.

2. ANALISI DEI RISULTATI DELLA SIMULAZIONE

L' analisi dei dati prodotti dalla simulazione consente di valutare le prestazioni del sistema ad eventi discreti che si sta studiando; tale analisi è necessariamente statistica se i dati di uscita della simulazione sono prodotti a partire da variabili di ingresso con valori generati in modo casuale, dal momento che, come si è detto, sequenze differenti di numeri casuali producono tipicamente insiemi di uscite differenti.

Nel caso in cui le prestazioni del sistema siano stimate attraverso un parametro θ,

il risultato di un insieme di esperimenti simulativi è uno stimatore di θ; lo scopo dell'analisi statistica è stimare la varianza dello stimatore, che misura la sua precisione, e/o determinare il numero di osservazioni necessarie per ottenere la precisione desiderata.

L' analisi dei dati di uscita è indubbiamente condizionata dalle sequenze di numeri casuali generati e dalle condizioni iniziali, ma l'elemento più caratterizzante è la durata τ_E della simulazione, che può essere finita o meno. Se la durata è specificata, si parla di simulazione *terminante* o *transitoria*, che inizia all'istante $\tau = 0$ in determinate condizioni iniziali e termina all'istante $\tau = \tau_E$.

Questo tipo di simulazione non è adeguato allo studio di un sistema *non terminante*, che funziona senza interruzioni o comunque ininterrottamente per periodi di tempo molto lunghi. In generale, di

tali sistemi si desidera valutare le prestazioni a regime, quando l'"influenza delle condizioni iniziali si è esaurita o è almeno trascurabile.

Lo studio simulativo del comportamento a regime di un sistema è detto *simulazione non terminante* o *stazionaria*; problemi significativi in questo ambito riguardano la determinazione delle condizioni iniziali e finali.

La scelta del tipo e quindi della durata della simulazione dipende dalla struttura del

sistema da studiare e dagli obiettivi scelti.

Stima degli indici di prestazioni

Si suppone di voler stimare un parametro θ che descrive le prestazioni del sistema a

partire dalla sequenza dei dati di uscita della simulazione $Y_1,\ldots,Y_n$.

Sono necessarie sia la *stima puntuale* del parametro d'interesse sia la *stima dell'intervallo di confidenza*, cioè di un intervallo che includa il valore di θ con una probabilità fissata. Quindi, se date *n* osservazioni $Y_1,\ldots,Y_n$ si desidera una stima del valore "più plausibile" di θ, si determina il valore del suo stimatore "puntuale" definito come

$$\hat{\vartheta} = \frac{1}{n}\sum_{i=1}^{n} Y_i$$

che in generale ha valore atteso E ($\hat{\theta}$) = θ + b, dove b rappresenta un eventuale bias, ossia la deviazione dal valore desiderato, che è una quantità costante e tipicamente non nota; se i dati $Y_1,\ldots,Y_n$ sono indipendenti e identicamente distribuiti (i.i.d.), lo stimatore è non deviato e quindi b = 0.

Se invece, sempre a partite dai dati di uscita $Y_1,\ldots,Y_n$ si vuole stimare un intervallo

che contenga il valore θ con una probabilità desiderata, è necessario stimare la varianza dello stimatore $\hat{\theta}$; ciò si effettua per mezzo dello stimatore della varianza $\sigma^2(\hat{\theta}), \hat{\sigma}^2(\hat{\theta})$, che ha media E $[\hat{\sigma}^2(\hat{\theta})] = B\sigma^2(\hat{\theta})$, dove B è la deviazione della varianza dello stimatore. Se lo stimatore della varianza è non deviato e i dati di uscita seguono una distribuzione normale, allora la variabile aleatoria

$$t = \frac{\hat{\vartheta} - \vartheta}{\sigma^2(\hat{\vartheta})}$$

segue la distribuzione *t* di Student parametrizzata da *n* gradi di libertà, tanti quanti i dati raccolti.

A questo punto è possibile fornire la definizione formale di intervallo di confidenza.

Definizione: L'intervallo di confidenza di percentuale 100(1 - α)% per il parametro θ è dato da

$$\hat{\vartheta} - t_{\frac{\alpha}{2},n}\hat{\sigma}(\hat{\vartheta}) \leq \vartheta \leq \hat{\vartheta} + t_{\frac{\alpha}{2},n}\hat{\sigma}(\hat{\vartheta})$$

dove $t_{\alpha,n}$ è un punto a percentuale 100(1 - α) in una distribuzione *t* con *n* gradi di libertà, cioè $\Pr\{t \geq t_{\alpha,n}\} = \alpha$.

È intuitivo che l'intervallo di confidenza definito risulterà corretto se sia lo stimatore puntuale sia lo stimatore della varianza sono non deviati; per questo motivo un problema rilevante da risolvere nell'analisi dei dati di uscita di uno studio simulativo è ottenere stimatori approssimativamente non deviati per la varianza dello stimatore puntuale.

Ricordando che la varianza dello stimatore $\hat{\theta}$ si può scrivere in funzione della varianza delle *n* osservazioni S^2 come

$$\sigma^2(\hat{\vartheta}) = \frac{S^2}{n}$$

allora uno stimatore della varianza con *n - 1* gradi di libertà è

$$\sigma^2(\hat{\vartheta}) = \frac{S^2}{n} = \frac{1}{n}\sum_{i=1}^{n}\frac{(Y_i - \hat{\vartheta})^2}{n-1}$$

che è non deviata se i dati $Y_1,\ldots,Y_n$ sono statisticamente indipendenti, costituisce una

stima per difetto se i dati $Y_1,\ldots,Y_n$ presentano autocorrelazione positiva, o una stima per eccesso se i dati $Y_1,\ldots,Y_n$ presentano autocorrelazione negativa.

Di conseguenza, se θ è uno stimatore puntuale deviato, si può ottenere un intervallo

di confidenza intorno a un valore sbagliato. Se si ha una autocorrelazione positiva e si

usa lo stimatore derivato da S^2 si ottiene un intervallo di confidenza più stretto di quello

reale; se invece si ha una autocorrelazione negativa e si usa lo stimatore derivato da S^2 si ottiene un intervallo di confidenza più largo di quello reale.

È evidente che nei primi due casi l'errore eventuale commesso è veramente significativo, mentre nell'ultimo caso si commette un errore minore, che si concretizza però in uno spreco di tempo, in quanto si raccolgono più dati del necessario.

3. SOFTWARE E LINGUAGGI PER LA SIMULAZIONE AD EVENTI DISCRETI

Facendo uso di un linguaggio di programmazione per la simulazione (SPL) *general-purpose* si è in grado di velocizzare lo sviluppo del modello, la rappresentazione dei dati di ingresso e l'analisi dei risultati. Inoltre, gli SPL hanno incrementato l'utilizzo della simulazione come strumento di analisi, favorito dalla diminuzione progressiva dei costi di sviluppo di un modello simulativo.

Nella scelta di un software per la simulazione ad eventi discreti, l'aspetto principale da tenere in considerazione è il rapporto tra il livello di accuratezza e di dettaglio richiesti e ciò che viene offerto da un programma: con i linguaggi si può modellare praticamente qualunque sistema, con i pacchetti specializzati non è detto.

Gli aspetti che caratterizzano i linguaggi di simulazione riguardano la possibilità di interfaccia con linguaggi di tipo generale (*general purpose*), la capacità di analisi dei dati di ingresso (o la compatibilità con pacchetti software che realizzano tale analisi), la portabilità (o almeno lo sviluppo di un eseguibile portabile), la semplicità e strutturabilità del linguaggio, la flessibilità dell'ingresso (dati in files o interattivi), la disponibilità di un *debugger* interattivo.

Le caratteristiche relative alla capacità di elaborazione sono la velocità, la generazione dei numeri casuali, la gestione dei dati, le librerie disponibili, la gestione di simulazioni non terminanti e di replicazioni indipendenti.

Infine, le uscite dei programmi di simulazione sono caratterizzate dalla possibilità

di avere report standard e/o personalizzati, dalle tipologie di analisi statistiche e dalla rappresentazione grafica dei relativi risultati (istogrammi, torte, ...), dalla compatibilità con fogli di lavoro (tipo Excel) o basi di dati e, in ultimo ma non meno importante, dalle potenzialità di animazione offerte.

Nella storia dello sviluppo degli SPL dal 1955 al 1986, Nance (1993) definisce i requisiti fondamentali che deve avere un SPL come:

- generazione dei numeri casuali;
- generazione delle variabili aleatorie;
- gestione del calendario degli eventi;
- procedure per l'analisi statistica;
- generazione dei rapporti finali.

Nella Tabella seguente viene riportata una suddivisione in 5 periodi dei 32 anni 1955-1986 con riferimento ai linguaggi di simulazione più noti generati in ognuno dei periodi (Nance, 1993).

Il *General Purpore System Simulator* (GPSS®) fu originariamente sviluppato su computer IBM nei primi anni '60; negli stessi anni fu realizzato anche SIMULA®, che era basato sul linguaggio *Algol* e presentava caratteristiche veramente innovative.

SIMSCRIPT® fu invece sviluppato allo scopo di ridurre i tempi di realizzazione dei modelli e dei programmi; i modelli SIMSCRIPT® sono descritti in termini di entità, attributi e insiemi; la sintassi e l'organizzazione dei programmi ricordano quelle del *Fortran*.

Periodo	Linguaggi
1955-1960	*GSP®*
1961-1965	*CLP®, CSL®, DYNAMO®, GASP®, GSPP®, SIMULA®, SIMSCRIPT®, OPS®, SOL® MILITRAN®, QUIKSCRIPT®*
1966-1970	*GPL®, SLANG®, AS®, BOSS®, Q-GERT®*
1971-1978	*SIMPL®, DRAFT®, HOCUS®, PBQ®*
1979-1986	*SIMAN®, SLAM®, INS®*

In *Control and Simulation Language* (CSL®) fu introdotto nel progetto di un linguaggio il concetto di scansione per attività, che rende appunto l'attività l'unità descrittiva di base.

Il *General Activity Simulation Program* (GASP®) portò la possibilità di usare i simboli dei diagrammi di flusso, che consentivano l'approccio alla simulazione sia a esperti

in un settore applicativo digiuni di programmazione sia a esperti programmatori non a

conoscenza delle aree applicative.

GASP® è stato un precursore sia del *Simulation Language for Alternative Modeling*

(SLAM ®) sia del *SIMulation ANalysis* (SIMAN®), due dei linguaggi ancor oggi più noti,

nelle loro versioni più recenti (SLAM II/T ESS ® e SIMAN V®). SLAM ® è stato il primo

linguaggio a includere tre ambienti simulativi: simulazione ad eventi discreti, a tempo

continuo e orientata ai processi; SIMAN® invece è stato il primo SPL eseguibile su un PC e consente di simulare sia ad eventi discreti sia con orientamento ai processi.

Il software per la simulazione ha proliferato negli anni '90, quando sono stati sviluppati numerosi prodotti disponibili sia per scopi generali sia per applicazioni specifiche.

Attualmente tali prodotti sono molto diffusi e disponibili solitamente anche in versioni

per PC comuni.

CAPITOLO 3

1. INTRODUZIONE A OMNeT++

OMNeT++ è un simulatore basato sul linguaggio C++, ad eventi discreti, per modelli di reti di comunicazione.
È un prodotto open-source nato nel 2003, che può essere usato sotto licenza GNU, ovvero senza scopi di lucro, principalmente utilizzato in ambito di ricerca accademica.
La definizione di modulare si ha perché ogni entità presente è rappresentata mediante un modulo, e tutti i moduli sono organizzati gerarchicamente e comunicano tramite scambio di messaggi che viaggiano attraverso porte e connessioni.
È dotato di una interfaccia grafica di alto livello ed è classificato fra i simulatori comportamentali.
I simulatori comportamentali consentono la simulazione su ogni nodo della rete di tutti gli strati dello stack protocollare, al fine di studiare il traffico di messaggi sulla rete ed il comportamento complessivo del modello.

Il simulatore OMNeT++ è anche Component-Oriented, ovvero la sua architettura è costituita da componenti che, interagendo tra loro, consentono l'esecuzione del programma di simulazione.
Fra le varie caratteristiche lo rendono un ottimo simulatore il fatto che facilita l'uso di modelli strutturati e riusabili, supporta l'esecuzione parallela e gli ambienti di esecuzione supportano la simulazione interattiva.
L'ambiente grafico utilizzato è molto intuitivo, esso contiene anche una rappresentazione testuale della topologia del modello.
L'interfaccia grafica per l'esecuzione della simulazione è Tkenv, grazie ad essa è possibile visionare i moduli del modello e vedere l'evolvere dello stato della simulazione.
Da linea di comando vi è una interfaccia (Cmdenv) , inoltre vi sono dei tool grafici (PLOVE e SCALARS) per visualizzare il contenuto degli output della simulazione scritti su appositi file dati generati automaticamente, che l'utente eventualmente può modificare.
OMNeT++ è stato progettato fin dall'inizio con la finalità di supportare la simulazione di reti anche grandi. Oltre la simulazione, OMNeT++ ha un buon algoritmo di debugging, utile a rilevare gli errori simulando la rete passo passo.
Il suo livello di astrazione è molto elevato, ha inoltre un'ampia documentazione e le sue prestazioni durante l'esecuzione sono di livello eccellente.
OMNeT++ fornisce strumenti per creare la simulazione, è dotato di un ambiente appositamente studiato per creare e simulare in tutta comodità, grazie anche a specifici modelli.

I suoi moduli sono ben scritti e totalmente riusabili, possono essere combinati in vari modi per essere utilizzati in maniera differente.

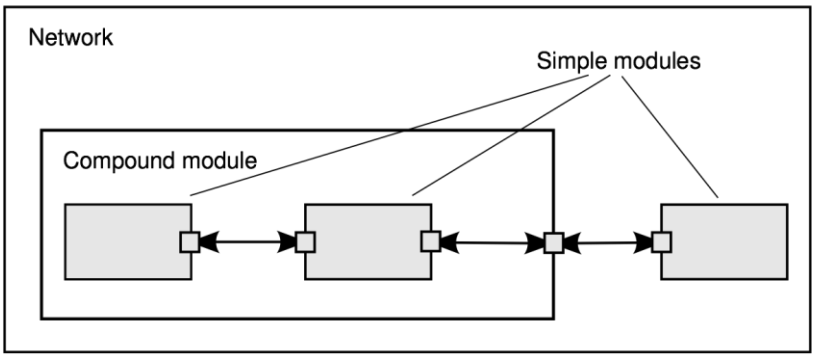
La sua area di applicazione primaria, come detto prima, è la simulazione di network di comunicazione, ma grazie alla sua architettura generica e flessibile è usata con successo in altre aree:

- modellazione di protocolli
- modellazione di code nelle reti
- modellazione di multiprocessori e altri sistemi hardware distribuiti
- convalida di architetture hardware
- valutazione degli aspetti performanti dei sistemi software complessi
- in generale, modellazione e simulazione dei sistemi dove l'approccio a eventi discreti è adatto, e può essere convenientemente mappato in entità in comunicazione attraverso lo scambio di messaggi.

1.1. CARATTERISTICHE DI OMNeT++

1.1.1 *I MODULI*

Un modello di OMNeT++ è costituito da una gerarchia di moduli annidati, che comunicano attraverso lo scambio di messaggi. Il modulo attivo è chiamato semplicemente simple module. I simple modules sono moduli che contengono gli algoritmi del modello e sono implementati dall'utente in C++. Possono essere raggruppati insieme in compound modules (moduli composti) e il numero di livelli di gerarchia è illimitato. L'intero modello, chiamato network in OMNeT++, è esso stesso un compound module.

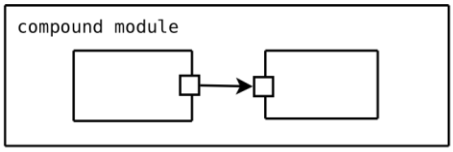


L'utente definisce il modello, che viene descritto attraverso vari tipi di moduli che verranno utilizzati. Prima si definiscono i simple module, per poi definire tipi di moduli sempre più complessi.

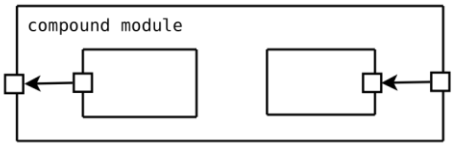
1.1.2. *MESSAGGI, INTERFACCE E CONNESSIONI*

I moduli comunicano tra loro scambiandosi messaggi. I messaggi in una simulazione contengono pacchetti di rete, ma possono contenere anche complessi strutture di dati. I simple modules possono inviare tali messaggi direttamente alla loro destinazione oppure attraverso un percorso predefinito, utilizzando interfacce e connessioni.
Ogni modulo ha sia un' interfaccia di input, che un'interfaccia output. L'interfaccia input serve per ricevere i messaggi, mentre l'interfaccia output per inviarli.

In un network si possono individuare due tipi di connessioni: la connessione tra l'interfaccia di un sub-module e un compound module, e la connessione tra sub-module. Le figure seguenti renderanno più chiara la cosa:



(a) Connessione tra sotto-moduli

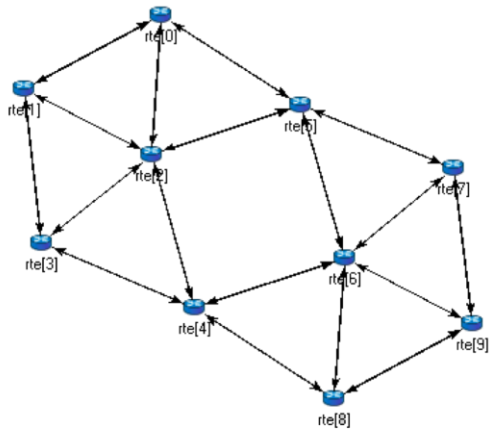


(b) Connessione tra livelli gerarchici

1.1.3. *PARAMETRI DEI MODULI*

Ogni modulo può avere dei parametri. I valori di tali parametri (stringhe, numeri, valori di verità, etc) possono essere assegnati nella definizione del modulo, oppure in un file chiamato **omnetpp.ini**. Dato che questo file ha una propria sintassi, è possibile modificare i valori dei parametri senza dover ricompilare il modello. Tale file viene letto nel momento in cui la simulazione viene eseguita.

1.1.4. *FILE NED*



//

```
// A network
//
network Network
{
submodules :
        node1 : Node;
        node2 : Node;
        node3 : Node;
        …
connections :
                node1. port++ <--> {datarate=100Mbps; } <--> node2.port++;
                node2. port++ <--> {datarate=100Mbps; } <--> node4.port++;
                node4. port++ <--> {datarate=100Mbps; } <--> node6.port++;
                …
}
```

Il codice riportato qui sopra definisce un tipo di network chiamato *Network.* Esso è scritto in linguaggio NED, e conseguente tale file ha un'estensione *.ned*. Tramite il file ned è possibile definire i simple module, i suoi parametri e i compound modules. Come si può vedere dalla figura il network è costituito da un compound module, al cui interno sono presenti dei nodi connessi tra loro a parità di livello gerarchico. Tali nodi possono a sua volta essere dei compound modules.

1.1.5. *PROGRAMMARE GLI ALGORITMI, LIBRERIA DI CLASSI*

I simple module contengono algoritmi, implementati come funzioni in C++. Chi scrive la simulazione può usare tutta la potenza e la piena flessibilità di tale linguaggio di programmazione, supportata dalla libreria di classe di simulazione OMNeT++. Tutti i messaggi, i moduli, le code, ovvero tutti gli oggetti di simulazione sono rappresentati come delle classi in C++. Sono stati progettati per lavorare insieme efficientemente al fine di creare un potente framework di simulazione. Le seguenti classi fanno parte della libreria di classi di simulazione:

- moduli, porte, parametri, canali;
- messaggi, pacchetti;
- classi di container (code, array, etc);
- classi di raccolta dati;
- classi di stima statistica e di distribuzione;

OMNeT++ contiene, quindi, libreria di classi che può essere usate per implementare simple module. Quindi uno dei vantaggi nell'utilizzare il software OMNeT++ è quello di utilizzare degli oggetti già presenti nella libreria ed usufruirne in base alle proprie necessità. Ad esempio i messaggi sono rappresentati dalla classe *cMessage* e dalla sua sottoclasse *cPacket*. *cPacket* viene usato per i pacchetti network (frame, datagrammi, pacchetti di trasporto, etc) in un network di comunicazione, mentre *cMessage* per il resto. Altri esempi di classi sono le classi di generazione di numeri casuali (*normal(), exponential()*), oppure la classe di parametri del modulo *cPar*. Oltre a queste ce ne sono molte altre.

1.1.6. *I COMPONENTI DI UNA SIMULAZIONE*

Riassumendo ciò che abbiamo detto fino ad ora il modello di una simulazione OMNeT++ è costituito dai seguenti componenti:

- un insieme di file NED in cui vengono definiti i moduli (simple e compound) della simulazione, i loro parametri, le interfacce di rete, e le connessioni tra i moduli;
- un file *omnetpp.ini* in cui vengono dati i valori ai parametri dei moduli del modello;
- un insieme di file *.msg* in cui ci sono le definizioni dei messaggi usati nel modello;
- i file sorgente, compilati in C++, dei simple module, ovvero le classi che implementano i simple module, cosi come tutte le ulteriori classi necessarie ai simple module per implementare il comportamento del modello;

Il sistema di simulazione, invece, fornisce le seguenti componenti:

- il kernel di simulazione, che contiene il codice necessario per gestire la simulazione e la libreria delle classi di simulazione;
- le interfacce utente, grafiche e testuali;

La simulazione viene costruita mettendo insieme tutti i componenti precedenti. Quindi prima i file .msg e .ned vengono trasformati in classi C++, e poi tutti i file sorgente vengono compilati e collegati con il kernel di simulazione e l'interfaccia utente scelta.

2. OMNeT++ IDE

La simulazione OMNeT++ è basata sulla piattaforma Eclipse, la quale viene ampliata con nuovi editor, view, wizard e altre funzioni. OMNeT++ aggiunge nuove funzionalità con lo scopo di creare e configurare i modelli (file NED e INI), rendere performanti le esecuzioni batch e analizzare i risultati delle simulazioni.

Dato che l'utilizzo di tale software può risultare piuttosto complesso, affiancherò alla parte teorica di utilizzo del software un esempio pratico. Lo scopo è quello di mostrare come creare, configurare, costruire e analizzare un modello di simulazione in OMNeT++. Verrà perciò creato, passo dopo passo, un network di coda costituito da componenti già definiti nel progetto "queueinglib". Come detto precedentemente questo software di simulazione ha il grande pregio di avere un esteso numero di librerie, con le quali è possibile creare un certo numero di scenari diversi.

2.1. INSTALLAZIONE DI OMNeT++ IDE

OMNeT++ supporta i seguenti sistemi operativi:

- Windows 7, Vista, XP
- Mac OS X 10.5 e 10.6
- Linux

2.1.1. *WINDOWS*

1. Per eseguire OMNeT++ è necessario installare Java runtime (JRE). Scaricare, quindi, Java 5.0 o successivi dal sito http://www.java.com ed installarlo prima di procedere.
2. Scaricare, quindi OMNeT++ dal sito http://omnetpp.org. Essere sicuri di scaricare il file seguente: `omnetpp-4.1-src-windows.zip`.
3. Estrarre il file zip. Per fare ciò, click destro del mouse sul file zip e scegliere *Estrai tutto* dal menù. Si possono utilizzare anche altri programmi per l'estrazione del file (Winzip, 7zip, etc).
4. Controllare che, all'interno della cartella del file estratto, ci siano le cartelle *doc*, *images*, *include*, *msys*, etc e i file *mingwenv.cmd*, *configure*, *Makefile* e altri.
5. Doppio click su *mingwenv.cmd* . Inserire nella finestra che si aprirà il seguente comando:

```
$ ./configure
 $ make
```

   In questo modo il programma verrà installato sul computer.

6. Al termine dell'installazione, digitare sempre su tale finestra *omnetpp.* OMNeT++ verrà quindi avviato.

2.1.2. *MAC OS X*

Sono supportate le seguente release:

- Mac OS X 10.5 (Leopard)
- Mac OS X 10.6 (Snow Leopard)

Testate sull'architettura *Intel 32-bit*.

1. Per eseguire OMNeT++ è necessario installare Xcode. Scaricare Xcode 3.0 dal sito http://developer.apple.com/technology/xcode.html.
2. Scaricare, quindi OMNeT++ dal sito http://omnetpp.org. Essere sicuri di scaricare il file seguente: `omnetpp-4.1-src.tgz`.
3. Aprire un terminale ed estrarre la cartella usando il comando seguente:

   ```
   $ tar zxvf omnetpp-4.1-src.tgz
   ```

   Verrà cosi creata una cartella chiamata `omnetpp-4.1`, contenente i file di simulazione.
4. Sempre sul terminale digitare:

   ```
   $ ./configure
   ```

   ```
   $ make
   ```

   In questo modo il programma verrà installato sul computer.
5. Una volta terminata l'installazione sarà possibile avviare il programma digitando:

   ```
   $ omnetpp
   ```

2.1.3. *LINUX*

1. OMNeT++ richiede che vengano installati alcuni pacchetti sul computer. Questi pacchetti includono il compilatore C++ (gcc), Java runtime, e altre librerie e programmi.
2. Scaricare, quindi OMNeT++ dal sito http://omnetpp.org. Essere sicuri di scaricare il file seguente: `omnetpp-4.1-src.tgz`.
3. Aprire un terminale ed estrarre la cartella usando il comando seguente:

   ```
   $ tar zxvf omnetpp-4.1-src.tgz
   ```

   Verrà cosi creata una cartella chiamata `omnetpp-4.1`, contenente i file di simulazione.
4. Sempre sul terminale digitare:

   ```
   $ ./configure
   ```

```
$ make
```

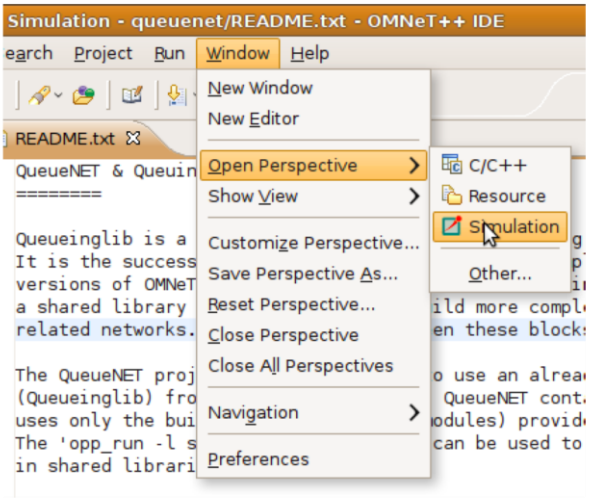
In questo modo il programma verrà installato sul computer.

5. Una volta terminata l'installazione sarà possibile avviare il programma digitando:

```
$ omnetpp
```

2.2. *WORKBENCH (BANCO DI LAVORO)*

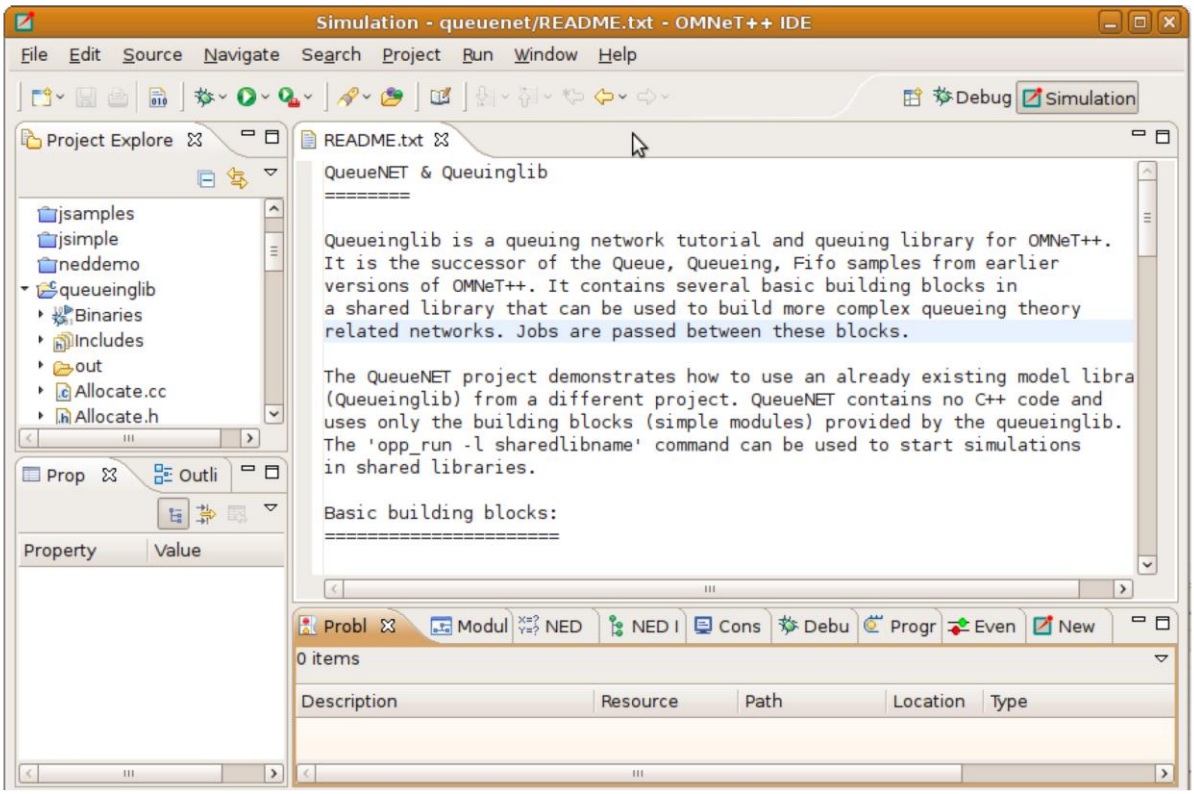
La finestra principale di Eclipse è composta da varie View ed Editor. Sono raggruppati all'interno di Perspectives che definiscono quali View ed Editor sono visibili e come sono dimensionati e posizionati. Eclipse è un sistema molto flessibile e questo permette di personalizzare l'IDE a proprio piacimento, ma lo rende più difficile da descrivere. Prima di tutto si ha la necessità di osservare la stessa cosa. OMNeT++ IDE offre una "Simulation perspective" , che non è altro che una semplice insieme di view selezionate convenientemente, in modo tale da creare i file NED, INI e MSG più facilmente.



2.3. *WORKSPACE (SPAZIO DI LAVORO)*

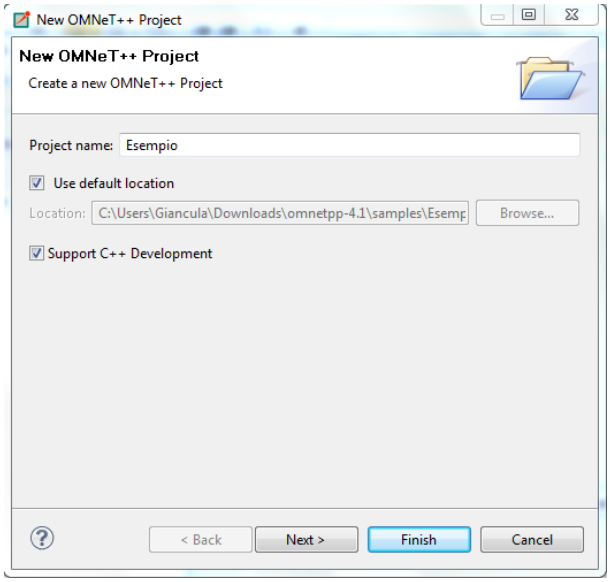
Uno spazio di lavoro è fondamentalmente una directory dove sono collocati tutti i propri progetti. Si possono creare ed usare più spazi di lavoro, passando dall'uno all'altro ogni volta che se ne ha bisogno. Quando si comincia a lavorare sui propri progetti si consiglia di creare il proprio spazio di lavoro selezionando **`File | Switch Workspace | Other.`**

E' possibile esplorare il contenuto dello spazio di lavoro in Project Explorer, Navigator, C/C Projects, etc. Viene raccomandato però l'uso di Project Explorer, il quale si trova nella parte in alto a sinistra della schermata.

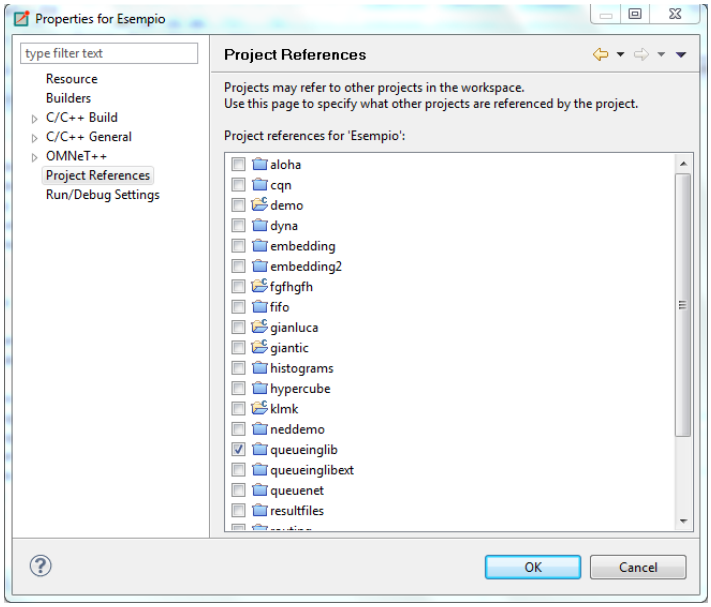


2.4. ESEMPIO

- Creiamo un nuovo progetto di simulazione OMNeT++ , usando OMNeT++ Project wizard. Questo progetto conterrà tutti i file con i quali noi lavoreremo.
- Chiamiamo il nostro nuovo progetto "Esempio".



- Dato che, come detto nell'introduzione, ci si appoggia al progetto queueinglib, è necessario includere tale Progetto. Ciò può essere fatto direttamente dalla finestra Project Properties.
  Click destro del mouse sulla cartella Esempio e successivamente cliccare su properties. Selezionare Project Reference e spuntare la casella del progetto "queueinglib". Cliccare, quindi, su ok.

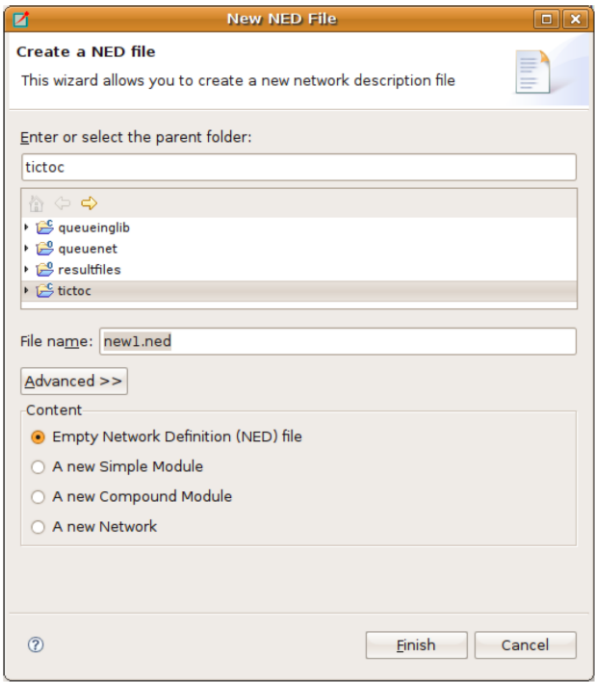


3. MODIFICARE I FILE NED

Quando si clicca su un file .ned nell'IDE, si aprirà nell'editor NED. Il nuovo editor NED è un editor a doppia-modalità. Nella modalità grafica, è possibile modificare il network usando il mouse. Nella modalità testuale è possibile invece lavorare con la sorgente NED. Quando l'IDE rileva un errore nel file IDE, il problema sarà segnato con un indicatore di errore nel Project Explorer e il Problems View verrà aggiornato in modo tale da mostrare la descrizione e la posizione del problema. In aggiunta, i marcatori di errori appariranno nella finestra di testo o sulla rappresentazione grafica del componente affetto dal problema. Il file NED verrà aperto in text mode, nel caso in cui sia affetto da errore. E' possibile passare alla modalità grafica solo nel caso in cui il file NED è sintatticamente corretto.

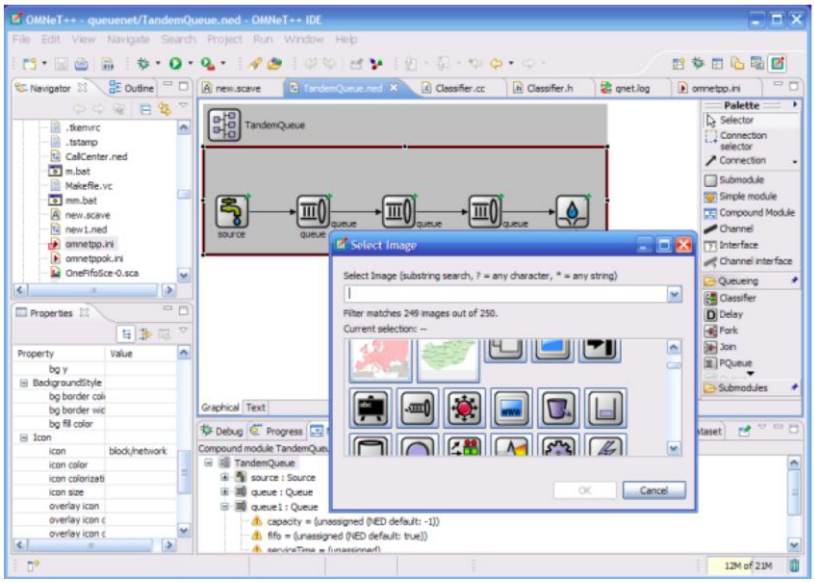
3.1. *CREARE UN FILE NED*

Una volta che si ha un progetto OMNeT++ vuoto, possono essere creati i file NED. Cliccando su **`File|New|Network Description File`** si aprirà un wizard dove sarà possibile specificare la directory di destinazione e il nome dei file/moduli. E' possibile scegliere di creare un file NED vuoto, un simple/compound module, o un network. Dopo avere cliccato su Finish, verrà creato il file NED richiesto.



3.2. *USARE L'EDITOR NED*

Se si vuole aprire un file NED, basta un doppio click del mouse sul file NED nel Project Eplorer. Se il file NED è senza errori, allora sarà aperta una rappresentazione grafica del file, in caso contrario sarà aperta una text view e il testo sarà annotato con marcatori di errore.

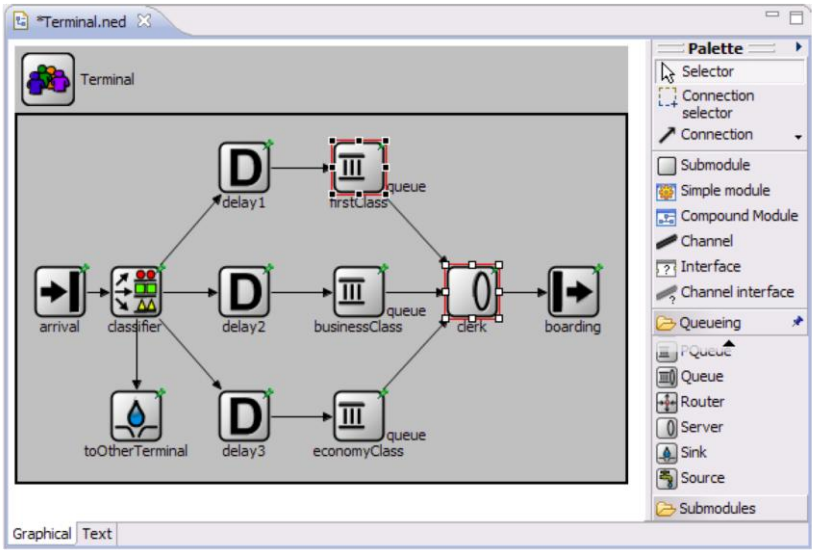


Il NED editor, quindi può modificare i file NED sia graficamente che in text mode, e l'utente può passare dall'uno all'altro in qualsiasi momento, usando le tabs alla base della finestra dell'editor.

3.3. *L'EDITOR NED IN MODALITA' GRAFICA*

L'editor grafico mostra gli elementi visibili del file NED caricato. I simple module, i compound module e i network sono rappresentati da figure o da icone. Ogni file NED può contenere più di un modulo o di un network. Se ciò si verifica, le figure corrispondenti appariranno nello stesso ordine di come sono state trovate nel file NED.

I simple module e i submodule sono delle icone, mentre i compound module e i network dei rettangoli. Le connessioni tra sottomoduli sono rappresentati o da linee o da frecce. Ciò dipende dal fatto che la connessione sia uni o bi-direzionale. I submodule possono essere trascinati o ridimensionati usando il mouse e connessi tramite ***IdealChannel*** nella tavolozza.



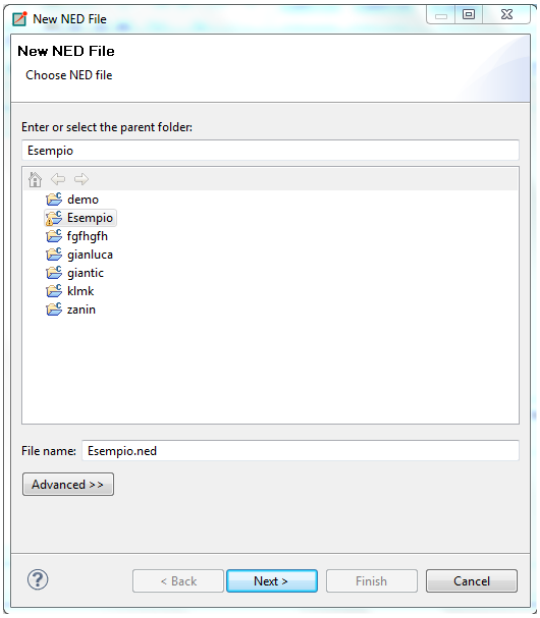
La tavolozza è normalmente alla destra dell'area di editor. La parte superiore della tavolozza contiene gli strumenti di base per le connessioni (Connection, Connection selector, etc). Per usare una voce della tavolozza si clicca semplicemente sopra ad essa. Successivamente, si clicca sul modulo in cui si vuole posizionarla o attivarla. La parte centrale contiene gli elementi di base che possono essere posizionati in un file NED (simple module,compound module, interfacce, canali, etc). Si clicchi su uno di questi e poi sull'area dell'editor per creare un'istanza. La parte inferiore contiene tutti i tipi di modulo che possono essere considerati submodule.

3.4. *L'EDITOR NED IN MODALITA' SORGENTE*

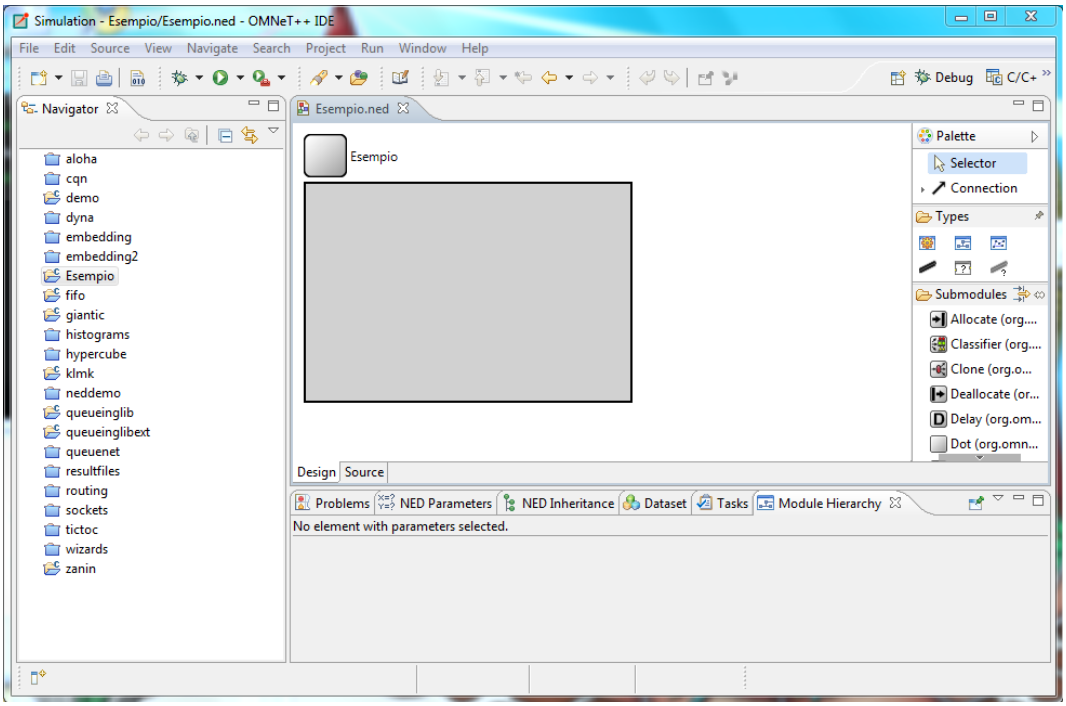
La modalità testo permette che l'utente lavori direttamente con la sorgente NED.

3.5 ESEMPIO

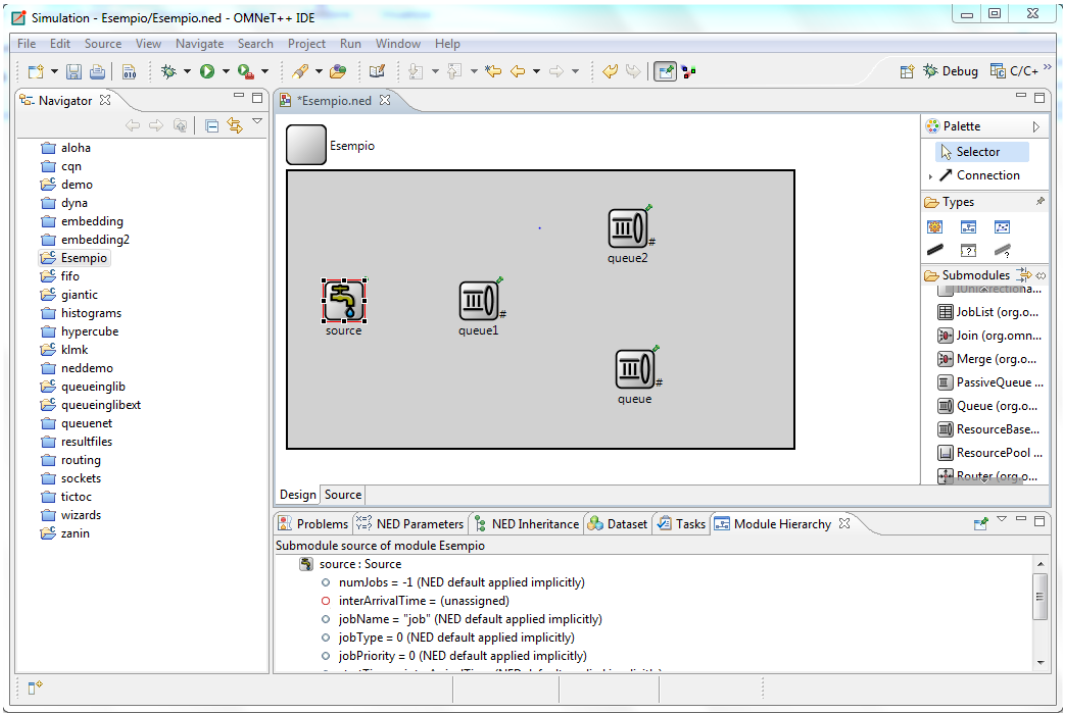
- Creare ora un file .ned.



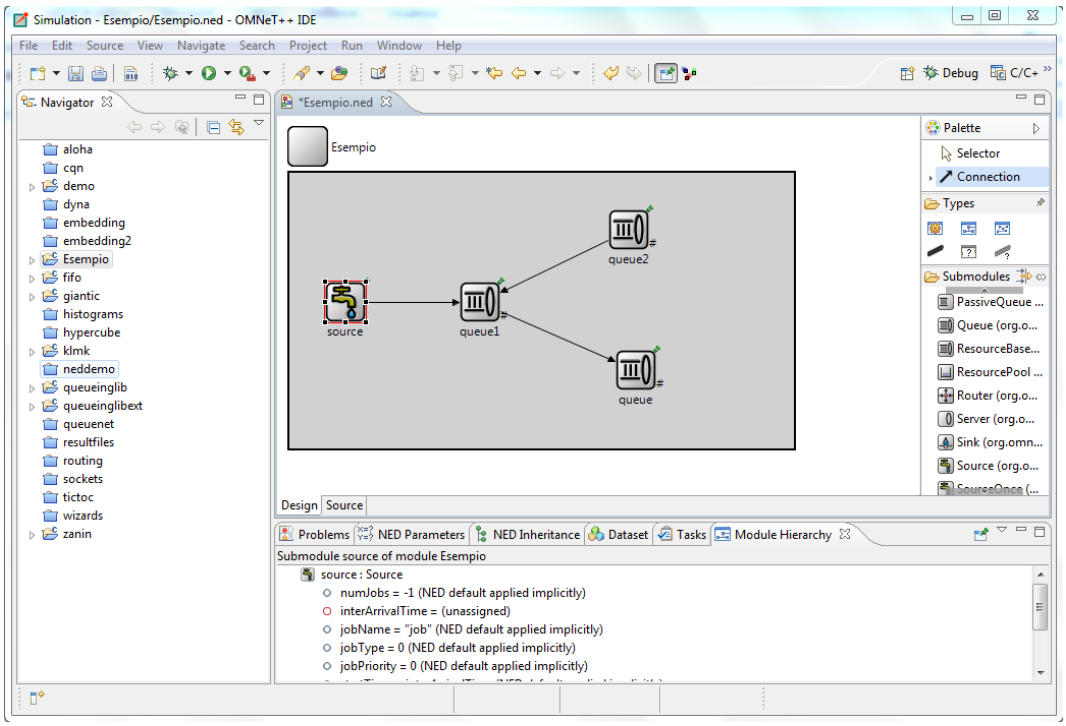
- Creare un network di coda chiuso.



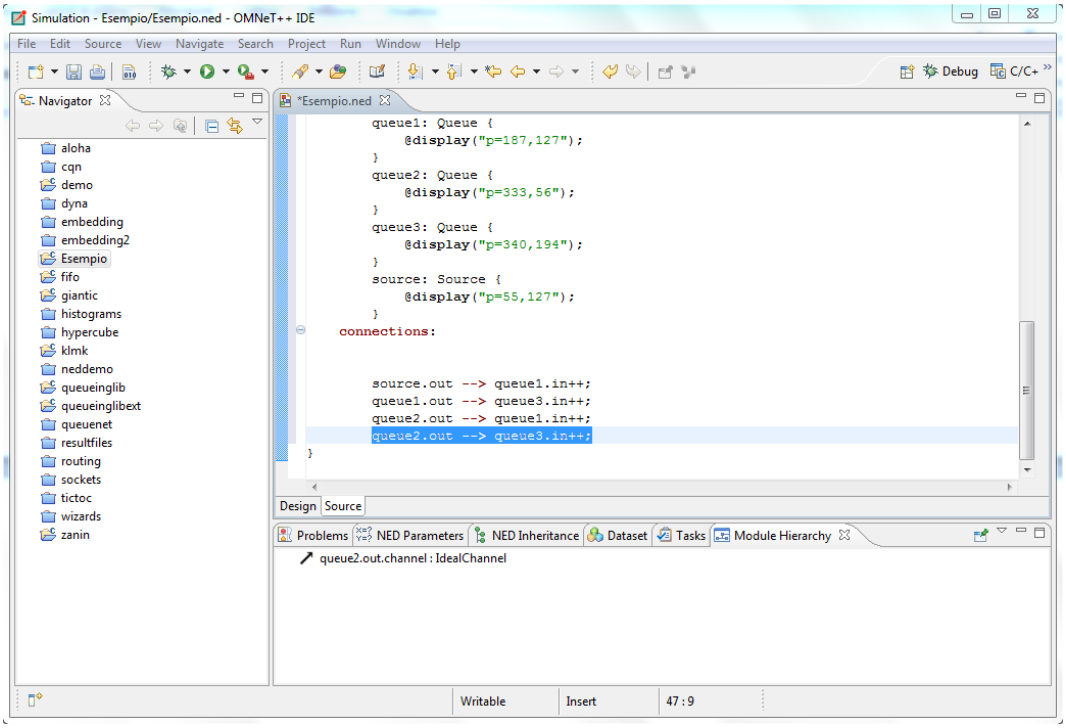
- A tale network aggiungere una singola sorgente e tre code connesse ad anello. I tipi di moduli sono disponibile sulla tavolozza, sul lato destro dell'editor grafico.



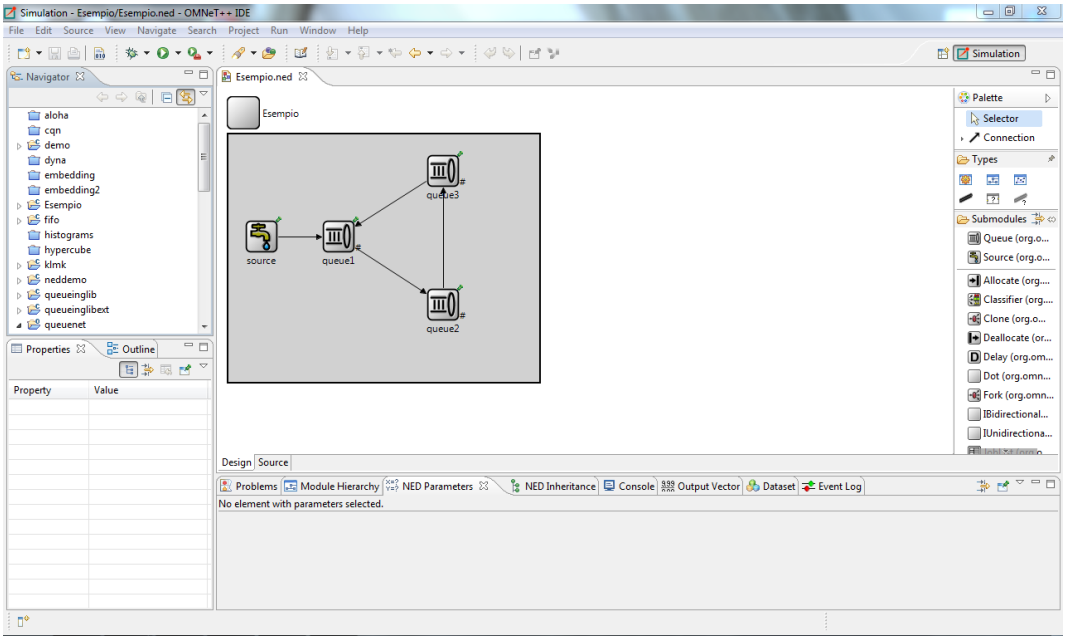
- Ora connettiamo i submodules con la Connection Tool. Quando si crea una connessione si scelgono le porte da connettere dal menù popup.



- E' possibile anche modificare il Network in modalità testo. Aggiungiamo l'ultima connessione scrivendo righe di codice.



Il risultato è il seguente:



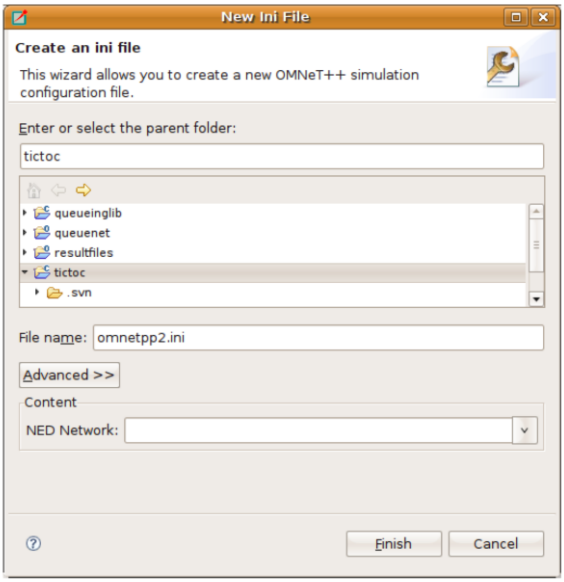
4. MODIFICARE I FILE INI

In OMNeT++, i modelli di simulazione sono parametrizzati e configurati per l'esecuzione della simulazione attraverso la configurazione di un file con l'estensione .ini, chiamati appunto file INI. I file INI sono file di testo, che possono essere modificati usando un qualunque editor di testo. Comunque OMNeT++ 4.1 introduce un programma progettato espressamente per modificare i file INI. L'Editor di File INI fa parte di OMNeT++ IDE ed è molto efficace nell'assistere l'utente nella stesura del file INI. E' a

doppia modalità. La configurazione può essere modificata usando forme e finestre di dialogo, oppure come testo normale.

4.1. *CREARE UN FILE INI*

Per creare un file INI, scegliere **`File | New | Initialization File`** dal menù. Si aprirà un wizard dove si può inserire il nome di un nuovo file e selezionare il nome del network che deve essere configurato.



4.2. *USARE L'EDITOR DI FILE INI*

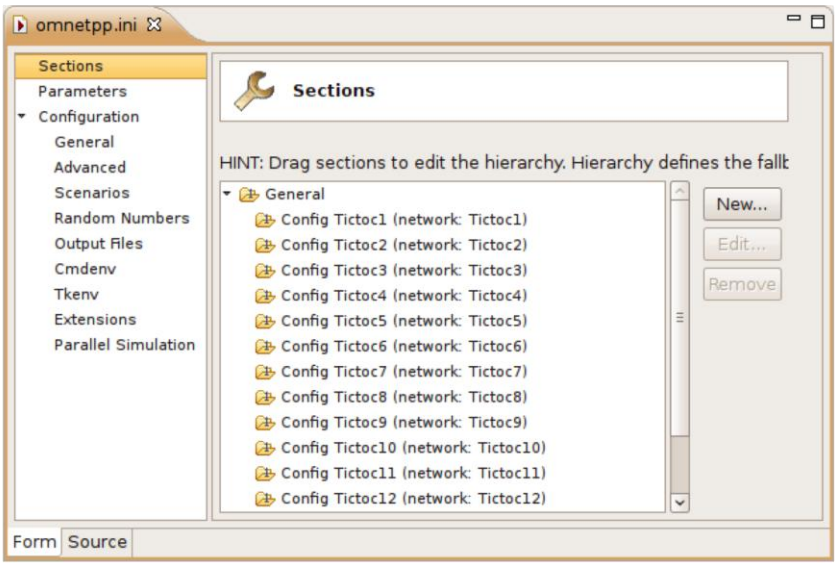
L'editor di file INI ha due modalità. La modalità sorgente (source mode) fornisce un editor di testo mentre nella modalità forma (Form mode) si può modificare la configurazione inserendo i valori nelle righe apposite. Si può passare dall'una all'altra selezionando le tab alla base dell'editor.

4.3. *MODIFICHE IN MODALITA' FORMA*

I file INI contengono la configurazione delle esecuzioni di simulazione. Il contenuto del file INI è diviso in sezioni. Nei casi più semplici, tutti i parametri vengono impostati nella sezione Generale. Se si vogliono creare più configurazioni nello stesso file INI, si può creare altre sezioni di configurazione e fare riferimento a loro con l'opzione ***-c*** quando inizia la simulazione. Le sezioni di configurazione aggiunte ereditano le impostazioni dalla sezione Generale o da altre sezioni di configurazione.

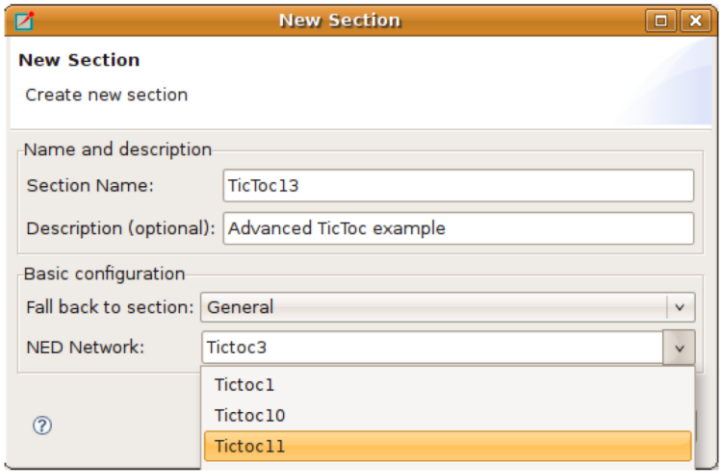
Sulla prima pagina dell'editor di forma, si possono modificare le sezioni. Le sezioni sono disposte come un albero: i nodi ereditano le impostazioni dai genitori. L'icona prima del nome della sezione mostra quante esecuzioni vengono configurate in quella sezione. Si può usare il clicca e trascina (drag and

drop) per riorganizzare le sezioni. Si può eliminare, modificare o aggiungere un nuovo figlio alla sezione selezionata.



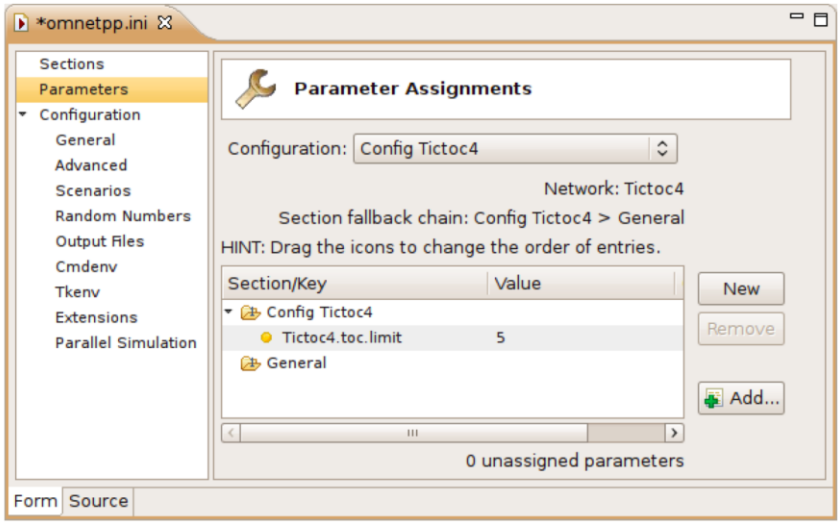
	contains a single run
	contains multiple replications (specified by 'repeat=...')
	contains iteration variables
	contains multiple replications for each iteration

Le sezioni di configurazione hanno un nome e una descrizione opzionale. Si può specificare una sezione di riserva oltre che quella Generale. Se il nome del network non viene ereditato, può essere specificato.



Sulla pagina dei Parametri dell'editor di forma, si possono impostare i parametri del modulo. Prima di tutto, si deve selezionare la sezione dove vengono memorizzati i parametri. Dopo aver selezionato la sezione dalla lista, la forma mostra il nome del network.

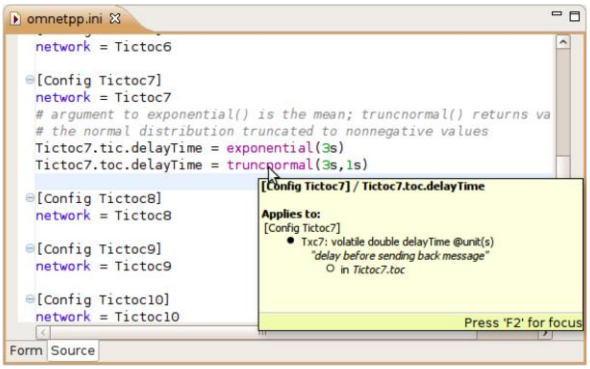
La tabella sottostante mostra le impostazioni correnti della sezione e tutte le sezioni da cui essa ha ereditato le impostazioni. E' possibile spostare i parametri trascinandoli. Se si clicca una cella della tabella è possibile modificare il nome del parametro, il suo valore e il relativo commento. Se si punta il mouse sopra una riga della tabella, il parametro viene descritto sulla finestra che appare.



Possono esserne aggiunti uno dopo l'altro cliccando su **New** e sulla nuova riga della tabella. I parametri selezionati possono essere rimossi con il pulsante **Remove**.

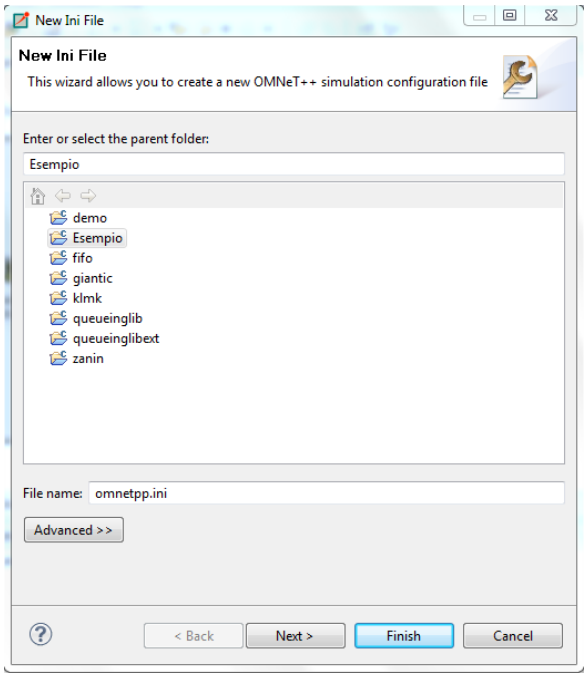
4.4. *MODIFICHE IN MODALITA' TESTO (text mode)*

Se si vuole modificare il file INI come testo normale, si deve passare alla modalità sorgente (Source mode). L'editor offre alcune caratteristiche in aggiunta alle funzioni dell'editor di testo come copia/incolla, annulla/ripeti (undo/redo) e ricerca di testo (text search).

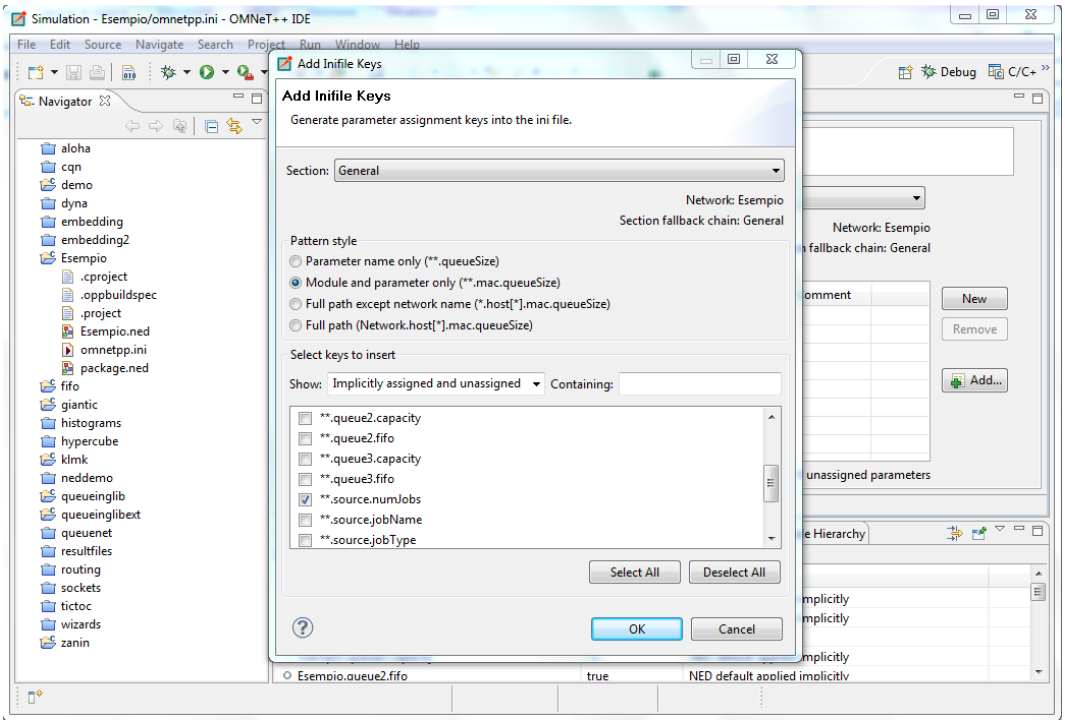


4.5. ESEMPIO

- Il network deve essere configurato prima che possa essere eseguito. Ora creeremo un file INI e li imposteremo i parametri del modello. Useremo l'Ini File Wizard per creare il file.

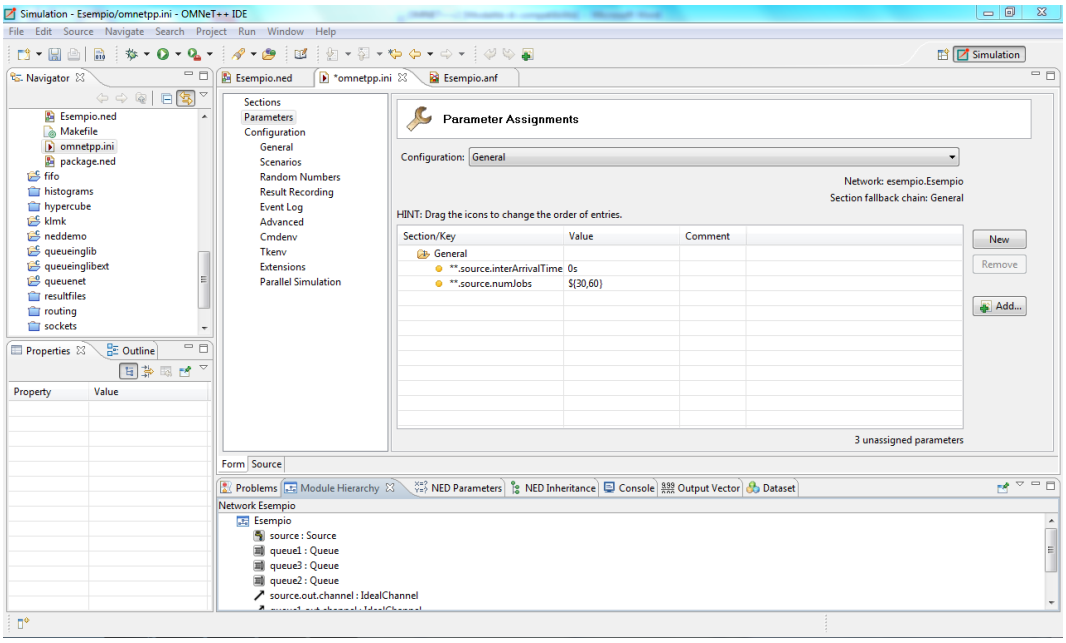


- Il prossimo passo è quello di assegnare i parametri del modello che non hanno valori di default. Cliccare su Add, la finestra che si apre elenca tutti i parametri non assegnati, e li inserisce nel file. Per prima cosa, scegliamo interArrivalTime e il numero di lavori nel (number of jobs) nel submodule Source.

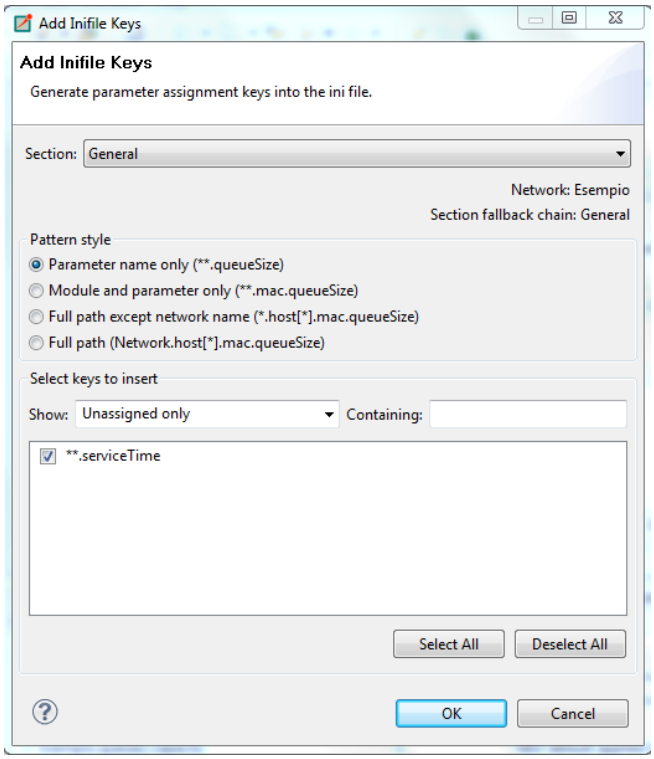


InterArrivalTime sarà impostato a 0s, ciò significa che tutti i lavori saranno iniettati nel network di coda.

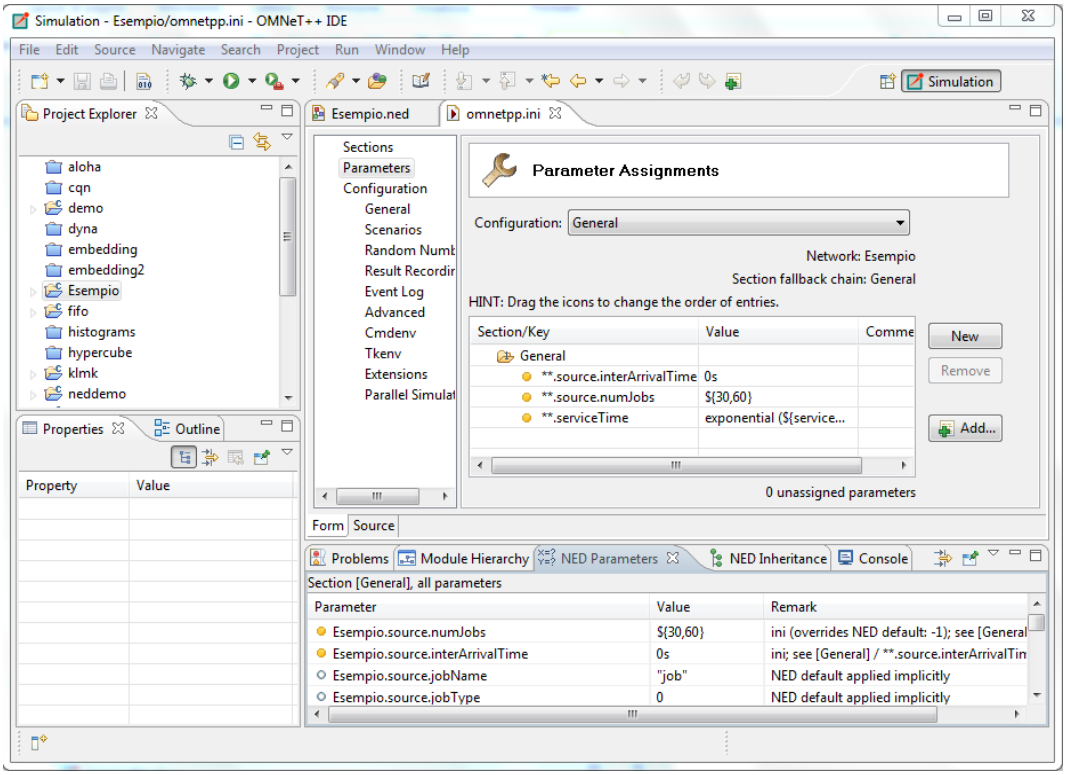
- Noi vogliamo eseguire il modello con due configurazioni, sia con 30 lavori iniziali che con 60. È possibile fare questo utilizzando tale sintassi “${…}”.



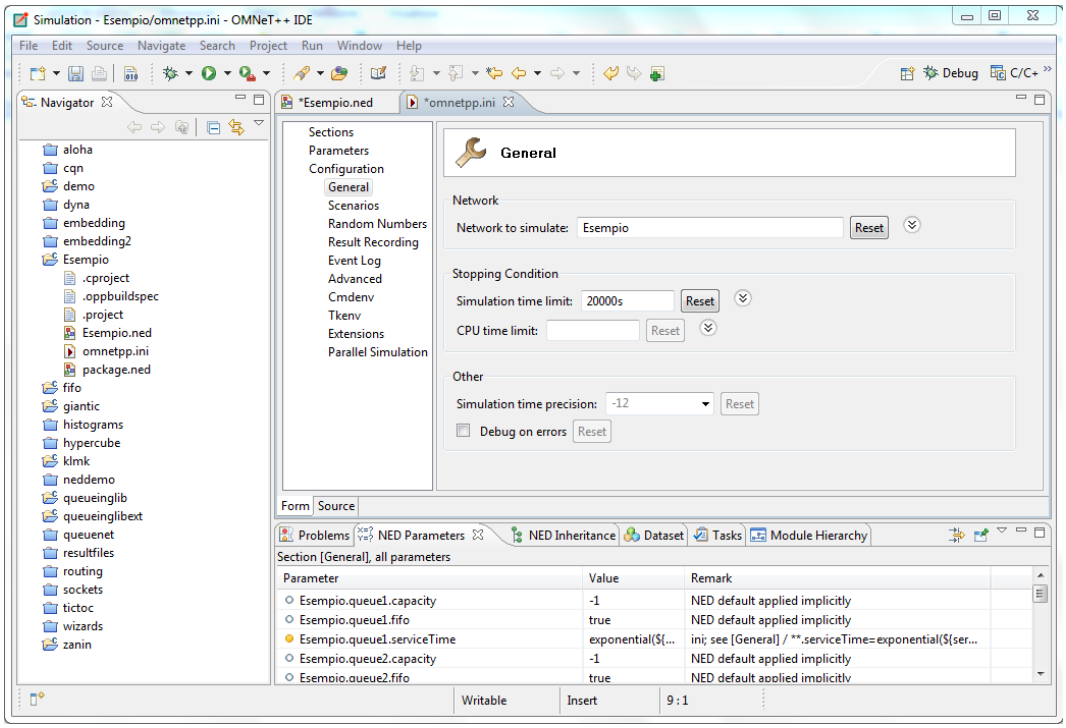
- Ora impostiamo il service time per tutte le code nel sistema. Vogliamo impostare solo i parametri senza valori di default definiti nel file NED.



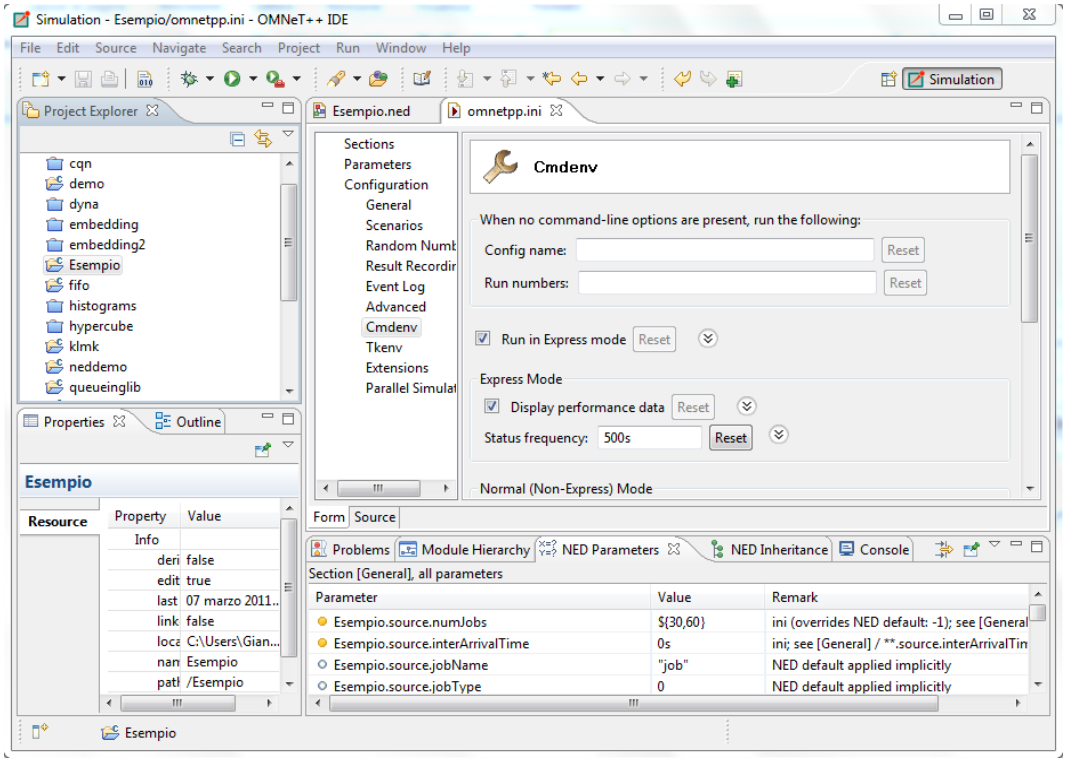
- Vogliamo provare il modello anche con differenti tempi di servizio di coda (tempo necessario ad eseguire le operazioni richieste): scriviamo,quindi, (${serviceMean=1..3 step 1}s). Ciò significa che si simulerà un tempo di servizio medio di 1, 2, 3s.



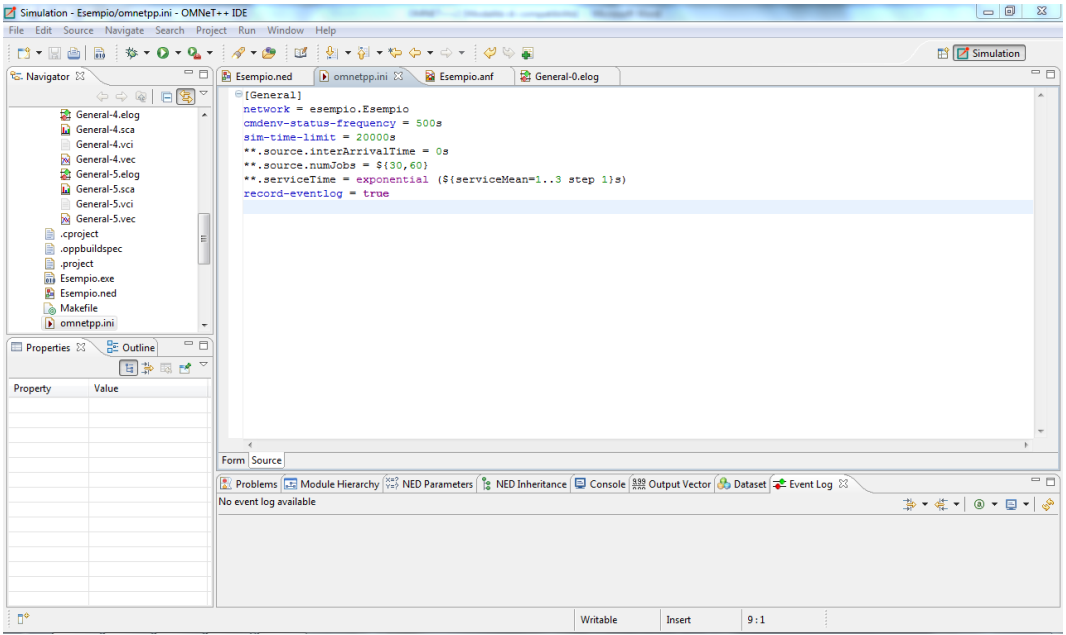
- Specificare per quanto tempo deve essere eseguita la simulazione. (General→Simulation time limit).



- Specificare la frequenza di stato. (Cmdenv→Status frequency).



- Passiamo ora all'interfaccia source e digitiamo la seguente riga di codice: record - eventlog = true

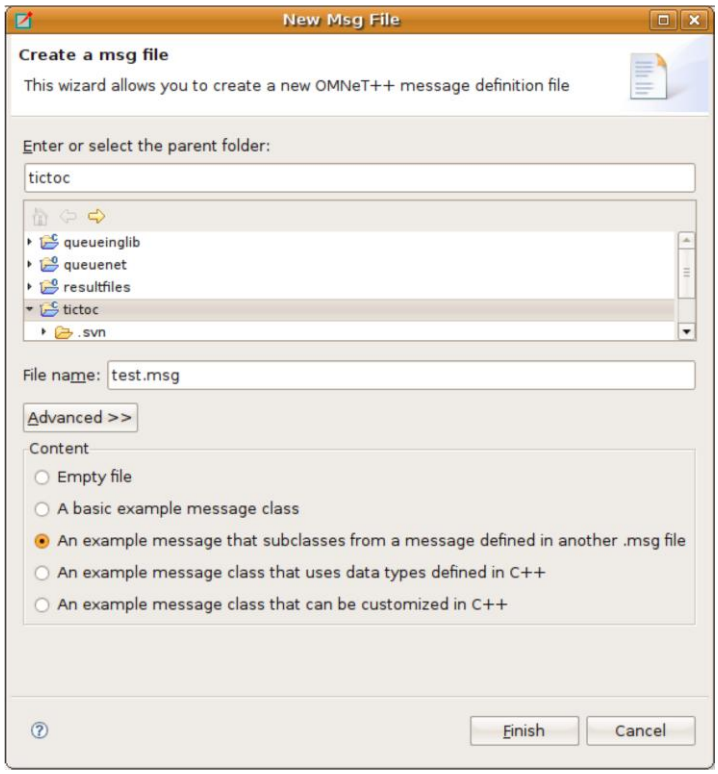


5. MODIFICARE FILE DI MESSAGGIO

5.1. *CREARE FILE MESSAGGIO*

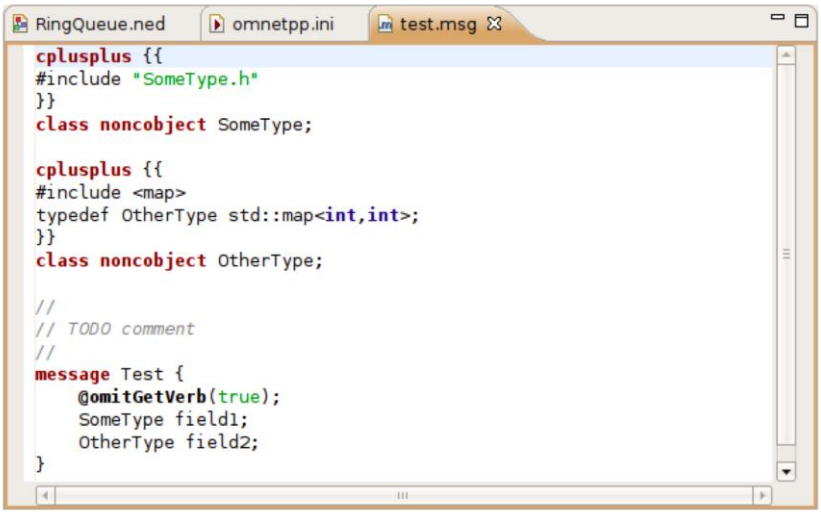
Scegliendo **`File|New|Message Definition (msg)`** si aprirà un wizard in cui si potrà specificare la directory di destinazione e il nome del file per la definizione del messaggio. Si può

scegliere di creare un file MSG vuoto, oppure scegliere tra modelli predefiniti. Una volta premuto Finish, verrà creato un nuovo messaggio con il contenuto richiesto.



5.2. *L'EDITOR DI FILE MESSAGGIO*

L'editor del file messaggio è un editor di testo di base.



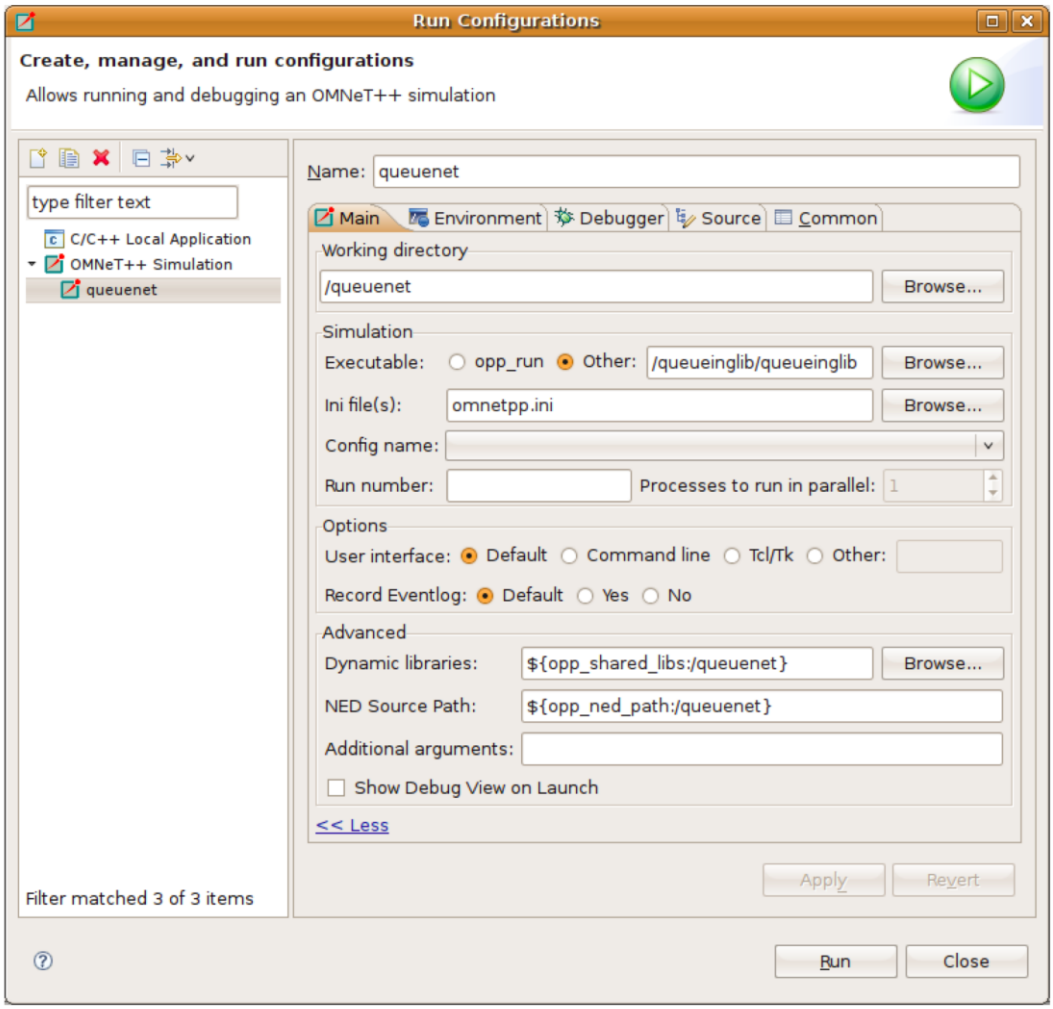
6. CONFIGURAZIONE DI LANCIO E DI DEBUGGING

6.1. *ESEGUIRE UNA SIMULAZIONE*

La nuova versione di OMNeT++ IDE permette di eseguire le simulazioni e i batch di simulazione direttamente dall'IDE.

6.2. *CREARE UNA CONFIGURAZIONE DI LANCIO*

OMNeT++ IDE aggiunge un nuovo tipo di configurazione di lancio a supporto degli eseguibili di simulazione. Questo tipo di lancio è disponibile dalla finestra *Run Configurations*. Per creare una nuova configurazione, si selezioni *Run Configurations* dal menù *Run*. Sulla sinistra apparirà una tendina con i tipi di lancio possibili. Si selezioni OMNeT++ Simulation e si clicchi sull'icona New Launch Configuration (presente sulla parte superiore della tendina). Poi, si immetta un nome di configurazione dove appare *Name*.



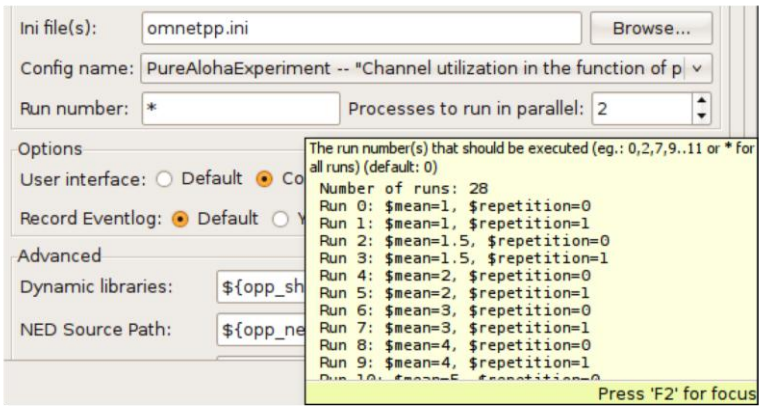
La tab *Main* delle finestra di configurazione è stata progettata in modo da rendere il lancio di simulazione il più semplice possibile. Il solo campo richiesto è la Working directory: tutti gli altri hanno valori di default.

6.3. *ESECUZIONI BATCH*

Nelle versioni precedenti di OMNeT++, per eseguire una simulazione alcune volte con differenti parametri, era necessario creare uno script esterno che creasse un file INI per ogni esecuzione e cominciasse la simulazione con il file specificato. In OMNeT++ 4.1 non è più necessario. La miglioria della sintassi del file INI permette di specificare i cicli per parametri specifici e conseguentemente non è più necessario separare i file INI per ogni esecuzione. In aggiunta, lo stesso IDE supporta la partenza della simulazione alcune volte.

```
[Config PureAlohaExperiment]
description = "Channel utilization in the function of packet generation frequency"
repeat = 2
sim-time-limit = 300min
**.vector-recording = false
Aloha.host[*].iaTime = exponential(${mean=1,1.5,2,3,4,5..21 step 2}s)
```

Se si crea una *Configuration* con una o più variabili di iterazione, si sarà i grado di eseguire più simulazioni. Praticamente l'IDE crea un prodotto Cartesiano e assegna in numero di esecuzione ad ogni prodotto. E' possibile eseguirne uno, alcuni o tutti specificando il *Run number*. Si può specificare un numero, una combinazione di alcuni numeri, o tutti i numeri di esecuzione.



Se si ha un sistema multi-core o multi-processor e un'ampia memoria, è possibile impostare un numero più alto sul campo *Processes to run in parallel* .

6.4. *DEBUG DI UNA SIMULAZIONE*

Se si vuole eseguire un'azione di debug , lo si fa cliccando su **`Run| Debug Configurations`**.

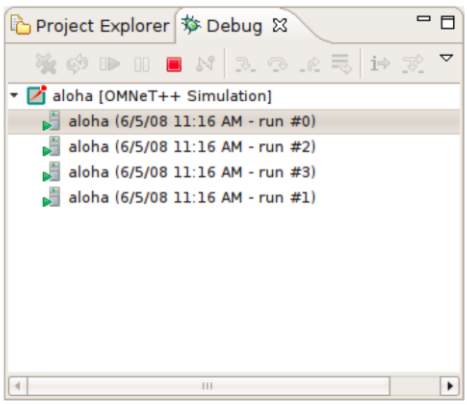
6.5. *CONFIGURAZIONE DI LANCIO RAPIDA*

Nella maggior parte dei casi si vorrà eseguire la simulazione il più presto possibile. Basato sulla selezione corrente, l'IDE offre alcuni metodi veloci per creare una configurazione di lancio. Si può selezionare cliccando su di esso nel Project Explorer oppure aprendolo PER LA modifica. Dopo la selezione, si deve solo cliccare sull'icona *Run* o *Debug* sulla toolbar, o click destro del mouse sul file e scegliere Run As… o Debug As.

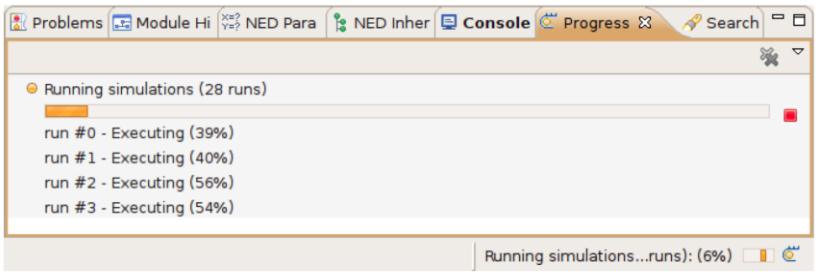
- Se viene selezionata una directory, che contiene un singolo file INI, l'IDE userà tale file per iniziare la simulazione.
- Se viene selezionato un file INI, tale file verrà usato durante il lancio come file INI principale per la simulazione.
- Se viene selezionato un file NED che contiene la definizione di un network, l'IDE farà una scansione dei file INI nei progetti attivi e cercherà di trovare una configurazione che permetta al network di partire.

6.6. *CONTROLLARE L'ESECUZIONE E I REPORT DI PROCESSO*

Dopo aver iniziato un processo di simulazione o una batch di simulazione è possibile tenere traccia dei processi iniziati attraverso la *Debug View*. Per aprire la debug view selezionare **`Window | Show View... | Other... | Debug | Debug.`**

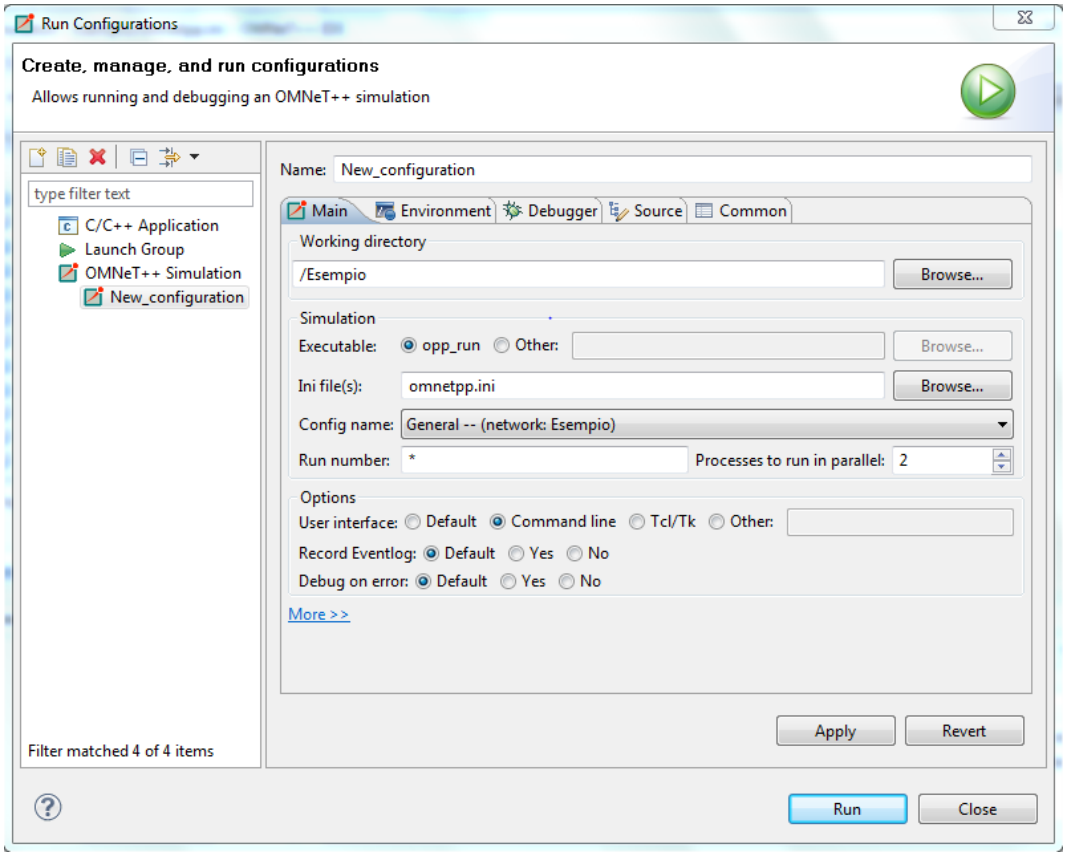


Se la simulazione è stata eseguita nell'ambiente di Command Line, è possibile monitorare il progresso della simulazione nella Progress View.

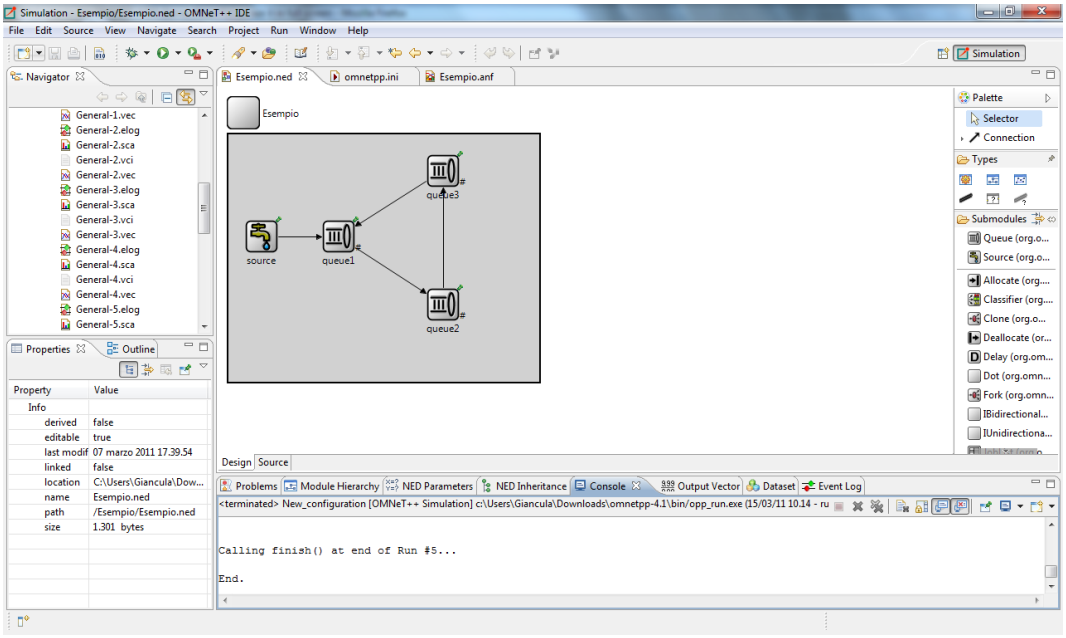


6.7. ESEMPIO

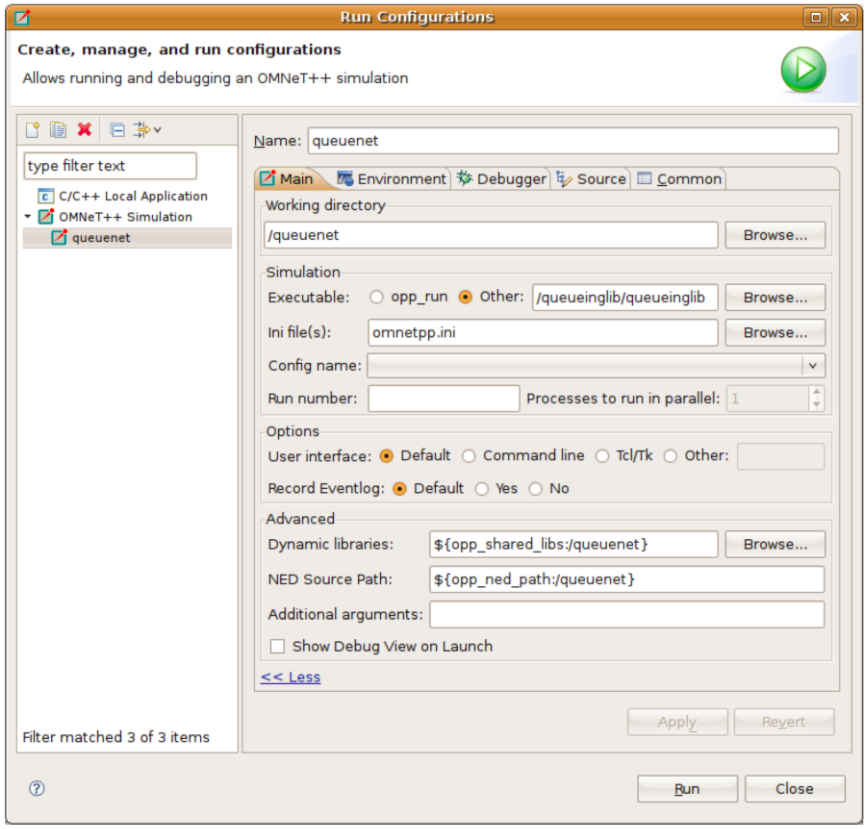
- Ora abbiamo un network definito in un file NED e in un file INI. Come passo successivo vogliamo eseguire la simulazione dall'IDE. Dobbiamo, quindi, creare una Launch Configuration.
  RUN → RUN CONFIGURATIONS → OMNeT++ Simulation → New launch Configuration
  Il programma di simulazione è già stato compilato, e può essere trovato nel progetto "queueinglib". Impostare i seguenti campi, quindi cliccare Run.



- La simulazione è completata. Da notare i file che sono stati creati nella directory del progetto. I file vec e sca contengono le registrazioni delle statistiche della simulazione. I file elog invece contengono la registrazione di ogni messaggio inviato, messaggi di testo di debug, e altro, e possono essere visualizzati sul sequence chart.



7.TKENV



In questa finestra è possibile selezionare l'interfaccia utente che si vuole utilizzare per l'esecuzione della simulazione. OMNeT++ IDE supporta la Command line (Cmdenv) e Tkenv. Della Command line ce ne siamo occupati precedentemente, ora descriveremo brevemente Tkenv.

Tkenv è un'interfaccia grafica di runtime per le simulazioni. Supporta esecuzioni di simulazioni interattive, di animazione, di analisi e di debug. Viene raccomandato per la fase di sviluppo della simulazione, per la presentazione e scopi educativi, in quanto permette all'utente di definire un quadro dettagliato dello stato della simulazione in qualsiasi punto della sua esecuzione, e di sapere cosa accade all'interno del network. Le più importanti caratteristiche sono:

- Messaggio di animazione di flusso;
- Visualizzazione grafica delle statistiche e dei vettori di output durante l'esecuzione della simulazione;
- Finestre separate per ogni output di testo del modulo;
- Esecuzione evento dopo evento, normale e veloce;
- Punti di esecuzione etichettati;
- Finestre di ispezione per esaminare ed alterare gli oggetti e le variabili nel modello;
- La simulazione può essere riavviata;
- Istantanee (report dettagliati relativi ai modelli: oggetti, variabili, etc);

Tkenv rende possibile visualizzare i risultati della simulazione durante l'esecuzione. I risultati possono essere visualizzati come istogrammi e serie storiche dei diagrammi. Questo può accelerare il processo di verifica del corretto funzionamento del programma di simulazione e fornisce un buon ambiente per la sperimentazione del modello durante l'esecuzione.

8. ANALIZZARE I RISULTATI

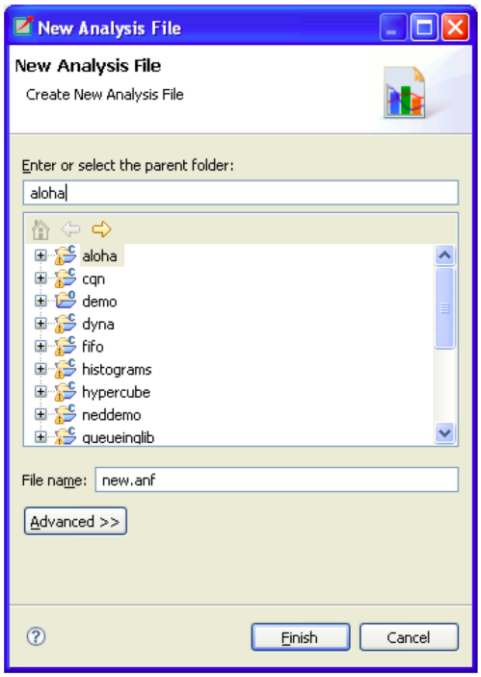
Analizzare i risultati della simulazione è un processo lungo e che richiede tempo. I risultati della simulazione sono registrati come valori scalari, valori vettoriali e istogrammi. L'utente applica poi metodi statistici per estrarre le informazioni rilevanti e trarre le relative conclusioni. Questo processo potrebbe comprendere alcuni passaggi. Di solito si ha il bisogno di filtrare e di trasformare i dati, e tracciare un grafico dei risultati. L'automazione è fondamentale qui. Non si vuole ripetere i passaggi di ricreazione dei grafici ogni volta che si riavvia la simulazione.

In OMNeT++ 4.1 lo strumento di analisi statistiche è integrato con l'ambiente di Eclipse. Le proprie impostazione (ad esempio la ricetta per trovare risultati a partire da dati grezzi) saranno registrate nei file di analisi (.anf) e diventeranno riproducibili istantaneamente. Ciò significa che tutte le elaborazioni e i grafici sono memorizzati come dataset; se, ad esempio una simulazione ha bisogno di essere riavviata a causa di un bug nel modello o di un errore di configurazione, i grafici esistenti non devono essere ricreati di nuovo. I vecchi file saranno rimpiazzati con quelli nuovi e i grafici avranno quindi i nuovi dati.

Quando si crea un'analisi, l'utente prima seleziona l'input dell'analisi specificando il nome del file. I dati di interesse possono essere selezionati nei dataset. L'utente può definire i dataset attraverso l'aggiunta di processi, di filtraggio e la creazione di grafici. Le etichette di sperimentazione, di misurazione, e di replicazione vengono aggiunte per rendere più facile il processo di filtraggio. E' possibile creare regole di filtraggio molto sofisticate. In aggiunta, i dataset possono usare gli altri dataset come input. Questo fa si che i dataset possano costruirsi l'uno con l'altro.

8.1. *CREARE FILE DI ANALISI*

Per creare un nuovo file di analisi, digitare **`File | New | Analysis File`** dal menù. Selezionare la cartella e inserire il nome. Premere *Finish* per creare ed aprire un file di analisi vuoto.



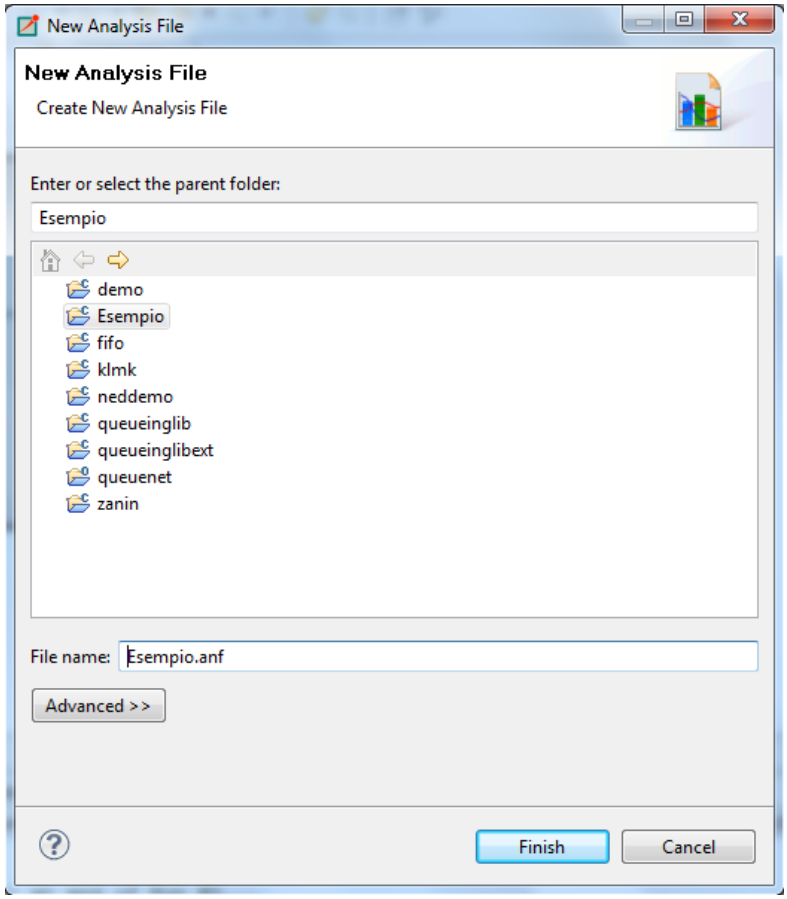
C'è un metodo più veloce per creare un file di analisi per il file di risultato. Basta un doppio click sul file di risultato nel Project Explorer View per aprire la finestra del New Analysis File. I campi sono già riempiti in accordo con la locazione e il nome del file. Premere *Finish* per creare un nuovo file di analisi contenente i file vettori e scalari, i cui nomi corrispondono ai file di risultato. Se il nome del file di risultato contiene un suffisso numerico, allora tutti i file con lo stesso suffisso saranno aggiunti al file di analisi.

8.2. *USARE L'EDITOR DI ANALISI (analysis editor)*

L'Editor di Analisi è implementato come un editor multi-page. Quello che l'editor modifica è la "ricetta": quali file di risultato prendere come input, quali dati selezionare, quali passi di processo applicare, e quali tipi di grafici creare.

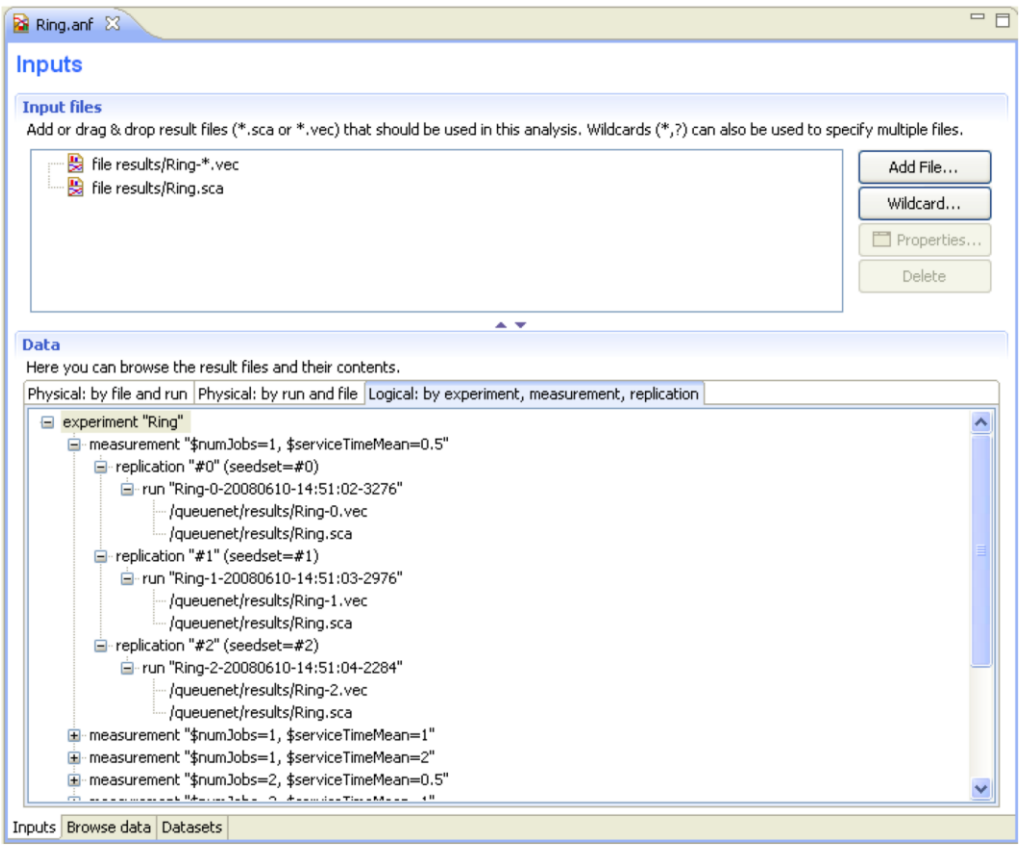
8.3. ESEMPIO

- Analizziamo ora i risultati. Creiamo una nuova Analisi usando uno wizard.



8.4. *FILE INPUT*

La prima pagina visualizza i file di risultato che servono come input per l'analisi. La prima metà specifica quali file selezionare usando filename espliciti e wildcard. I nuovi file input possono essere aggiunti all'analisi trascinando i file vettoriali e scalari da Project Explorer View, o aprendo finestre con i pulsanti Add file…o Wildcard. Se il nome del file inizia con '/,', esso viene interpretato come relativo alla root di lavoro; altrimenti è relativo alla cartella del file di analisi.

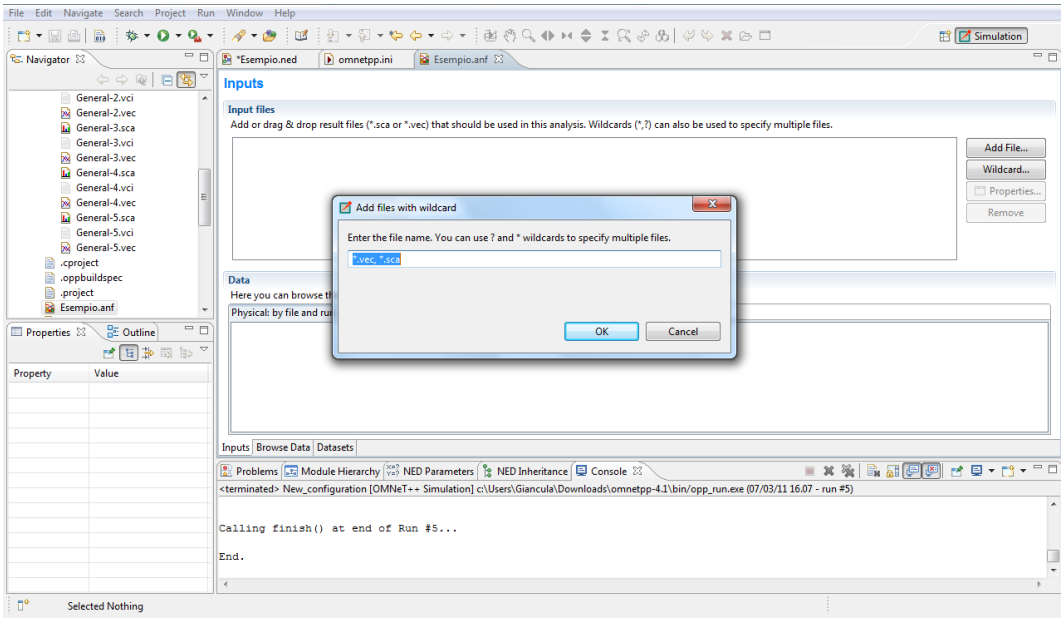


I file input vengono caricati quando il file di analisi viene aperto. Quando il file cambia sul disco, viene automaticamente ricaricato nel momento in cui viene aggiornato lo spazio di lavoro. I file vettori non vengono caricati direttamente: invece, viene creato un file index e gli attributi del vettore vengono caricati dal file index. I file index sono generati durante la simulazione ma possono essere tranquillamente eliminati senza perdita di informazioni. Nel caso in cui il file index venga perso o il file vettore modificato, l'IDE ricostruisce l'indice (index) di background.

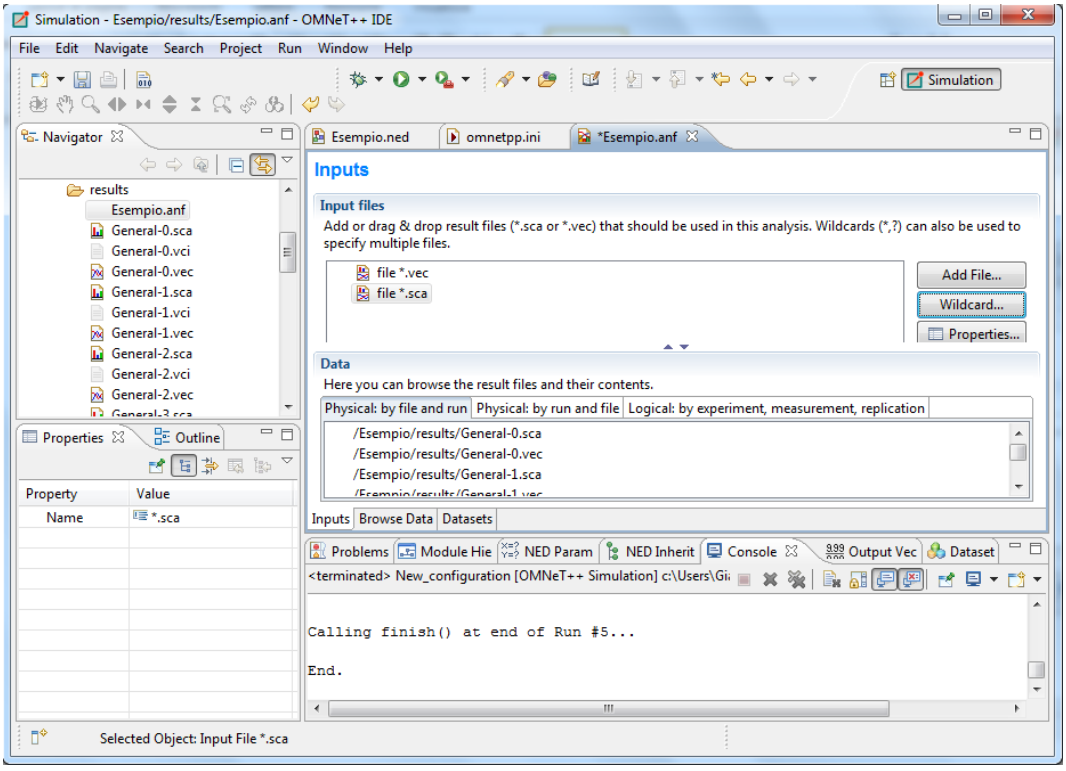
La metà inferiore mostra i file scelti nella specificazione dell'input e quali esecuzioni contengono.

8.4.1.1. ESEMPIO

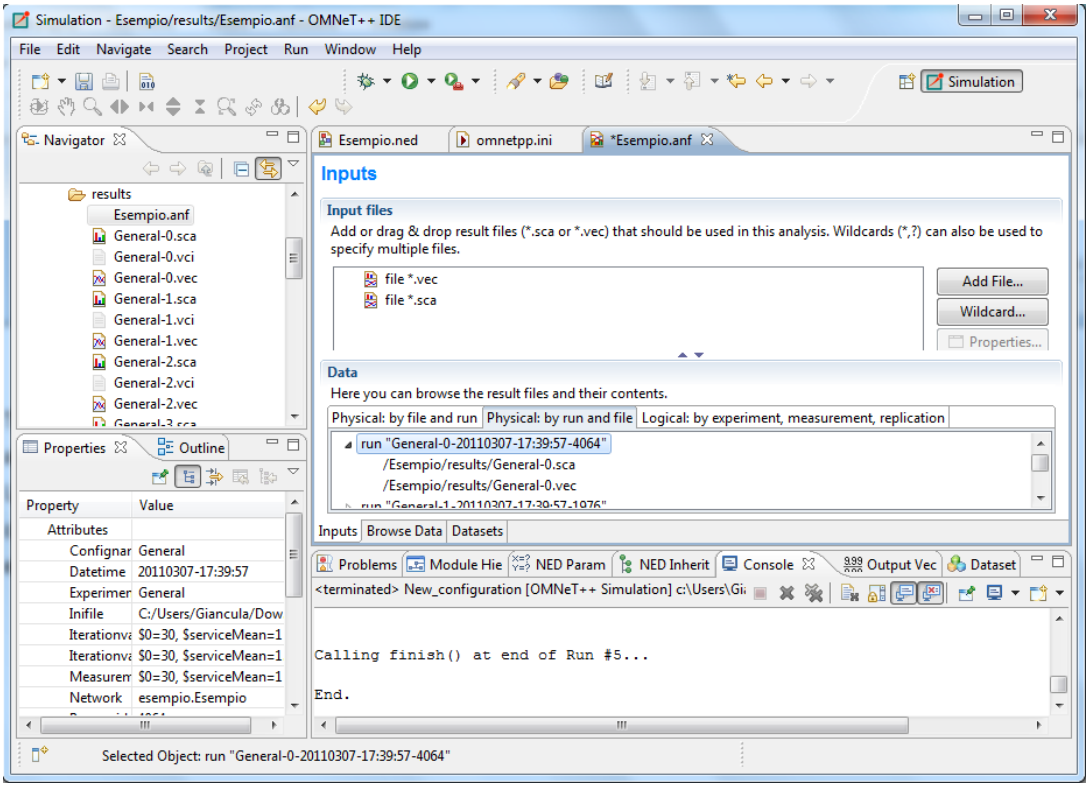
- Aggiungiamo tutti i file dei risultati generati all'Analisi. Potremmo specificare esattamente i nomi dei file oppure trascinarli dal Navigator. Risulta più facile utilizzare le wildcards.



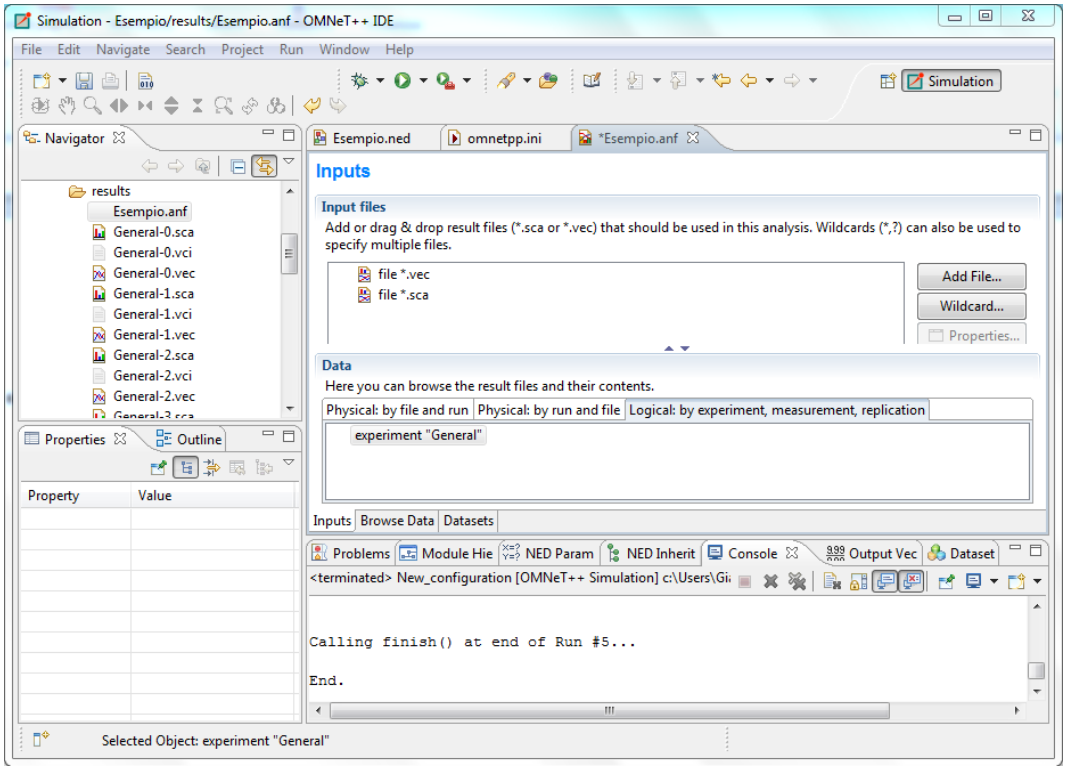
- L'editor elenca I file trovati sullo spazio sottostante, un vettore ed uno scalare per ogni esecuzione.



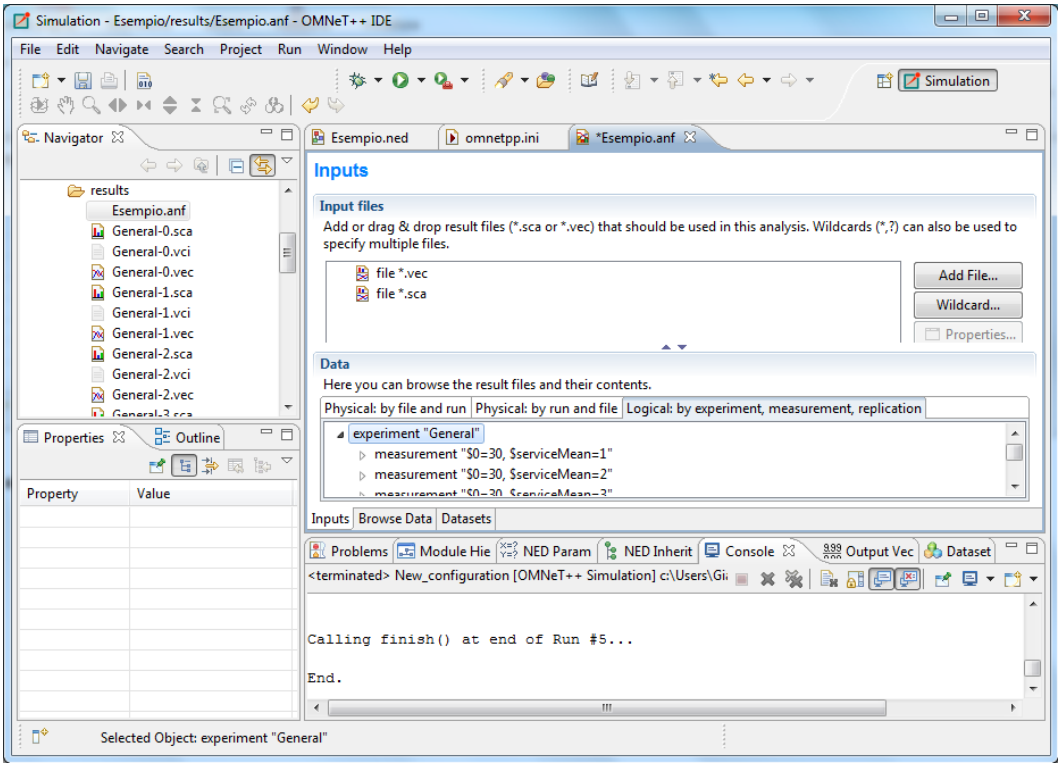
- La seconda tab visualizza quali file ciascuna esecuzione ha generato.



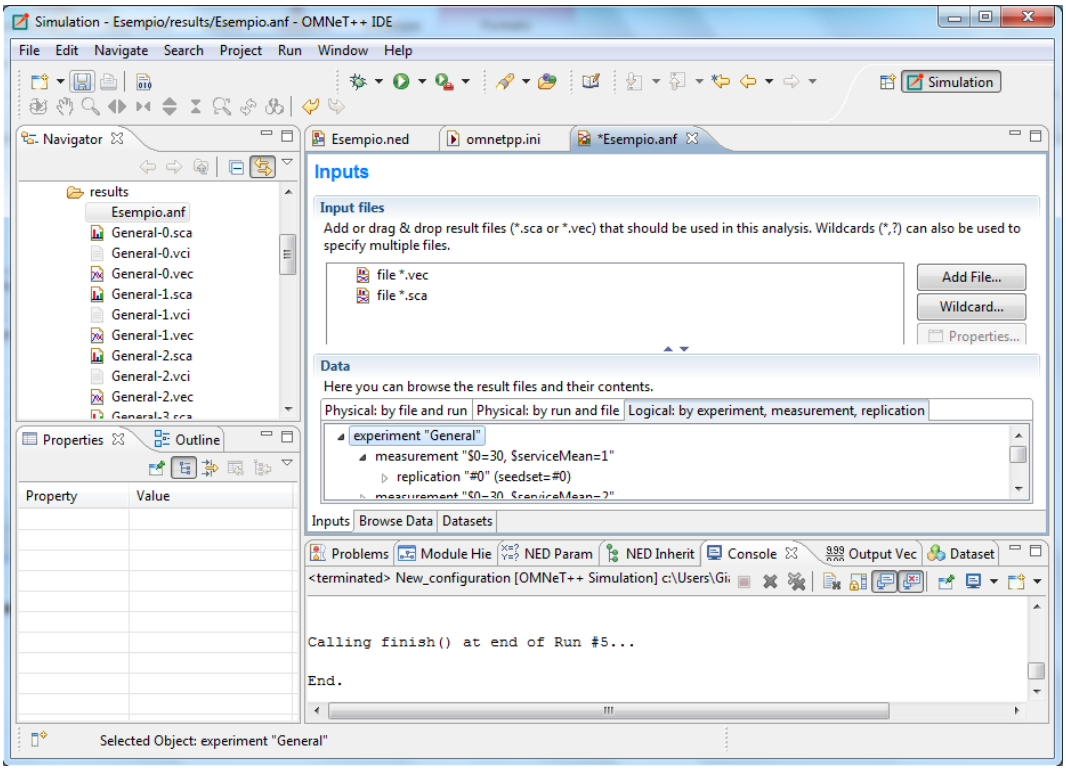
- La terza tab presenta un gruppo logico di esecuzioni. Tutte le esecuzioni che abbiamo appena fatto appartengono ad un ***esperimento*** ,chiamato General (nome del file INI di configurazione).



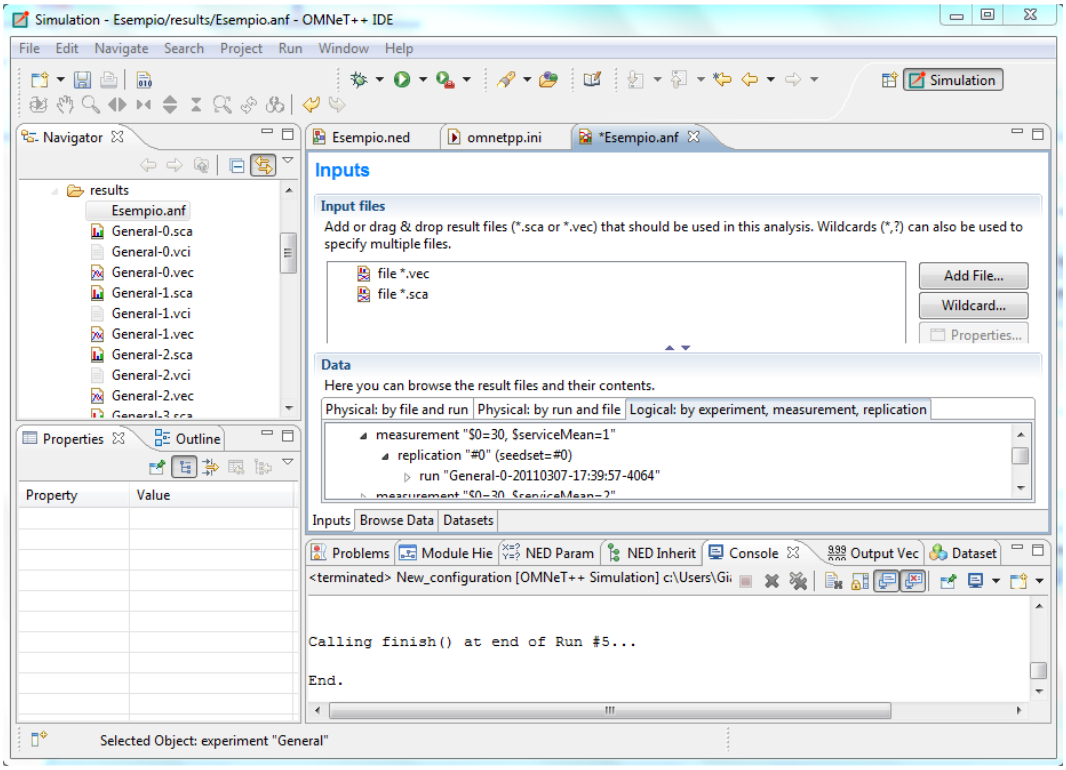
- Ogni esperimento è costituito da diverse misurazioni, ricavate dallo stesso modello di simulazione eseguito con parametri diversi.



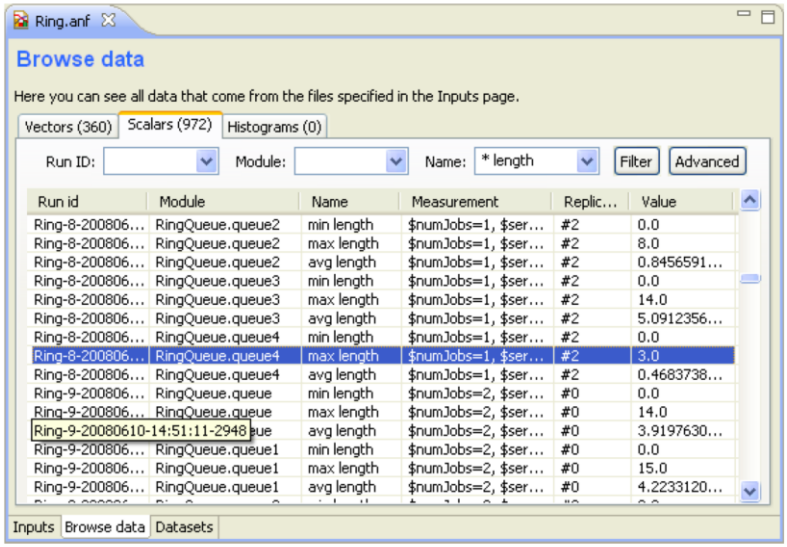
- Ogni misurazione può essere ripetuta con differenti semi affinchè vengano aumentati i risultati statisticamente attendibili, conseguente in alcune replicazioni.



- Ogni istanza di esecuzione riceve un unico Run ID.



8.5. *BROWSING INPUT*



La seconda pagina dell'editor di Analisi visualizza i risultati (vettori, scalari e istogrammi). I risultati possono essere ordinati e filtrati. Il filtraggio semplice è possibile con le caselle combinate (combo box), o quando non è sufficiente, l'utente può scrivere arbitrariamente filtri complessi usando un linguaggio di espressione generico di partner-matching. I dati selezionati o filtrati possono essere tracciati, o salvati nei dataset nominati per un ulteriore processo. Cliccando su *Advanced* si passa alla modalità di filtro avanzata. Nel campo di testo, si può inserire un modello di filtro complesso. Per

nascondere o mostrare le colonne della tabella, aprire *Choose table columns* dal menù e selezionare le colonne che si vogliono visualizzare. Le impostazioni sono persistenti e applicate in ogni editor aperto in successione. Le righe della tabella possono essere ordinate facendo click sul nome della colonna.

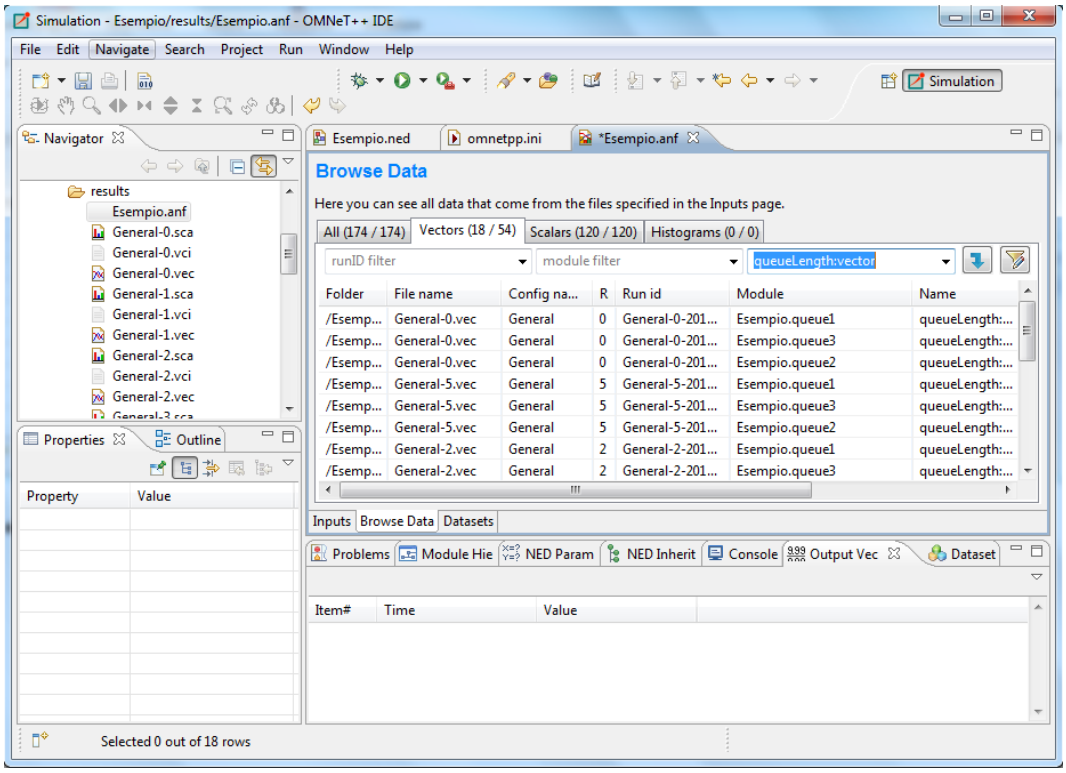
E' possibile visualizzare gli elementi dei dati selezionati su un grafico. Per aprire il grafico, selezionare Plot dal context menù. Il grafico aperto non viene aggiunto automaticamente al file di analisi, cosi si può esplorare i dati aprendo il grafico in questo modo e chiudere la pagina del grafico senza rendere l'editor "sporco".

I dati dei vettori selezionati possono anche essere visualizzati in una tabella. Ci si deve accertare che Output Vector View sia aperto. Se non fosse aperto, si deve aprirlo dal context menù (Show Vector View). Se si seleziona un vettore nell'editor, la view mostrerà il suo contenuto.

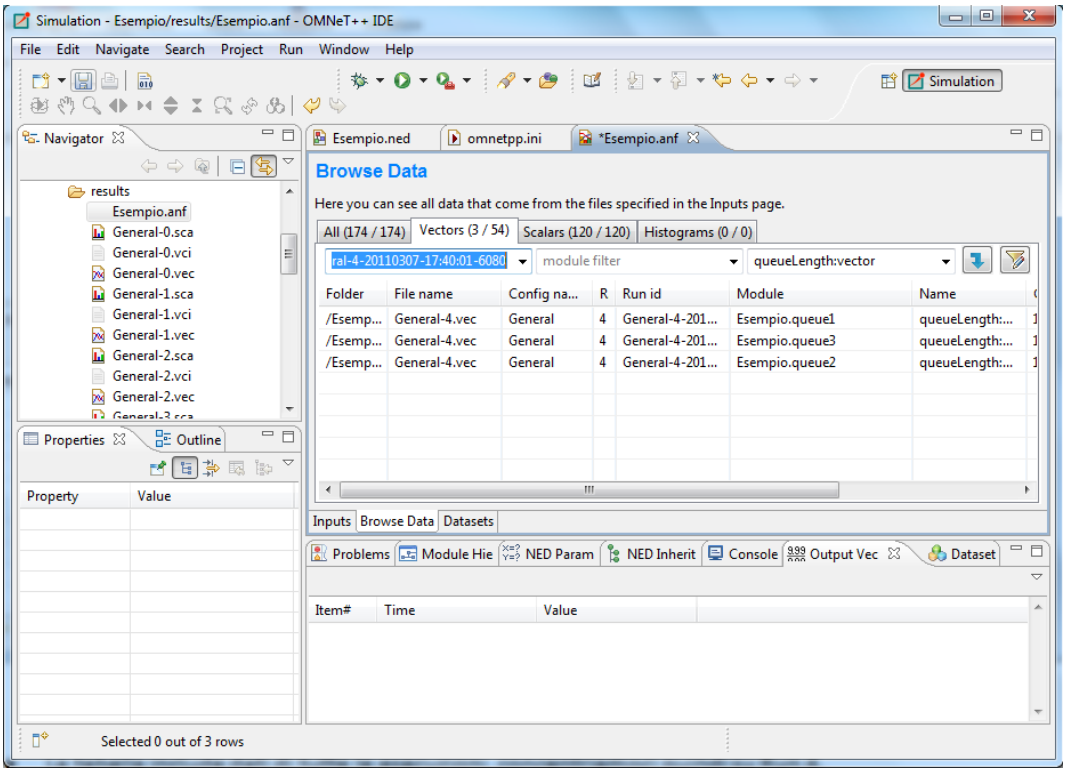
Si può creare un dataset dagli elementi di risultato selezionati. Selezionare Add Filter Expression to Dataset… se si vuole aggiungere tutti gli elementi visualizzati nella tabella. Selezionare Add Filter Selected se si vuole aggiungere solo gli elementi selezionati. Si può aggiungere gli elementi ad un dataset esistente, o si può creare un nuovo dataset.

8.5.1.1. ESEMPIO

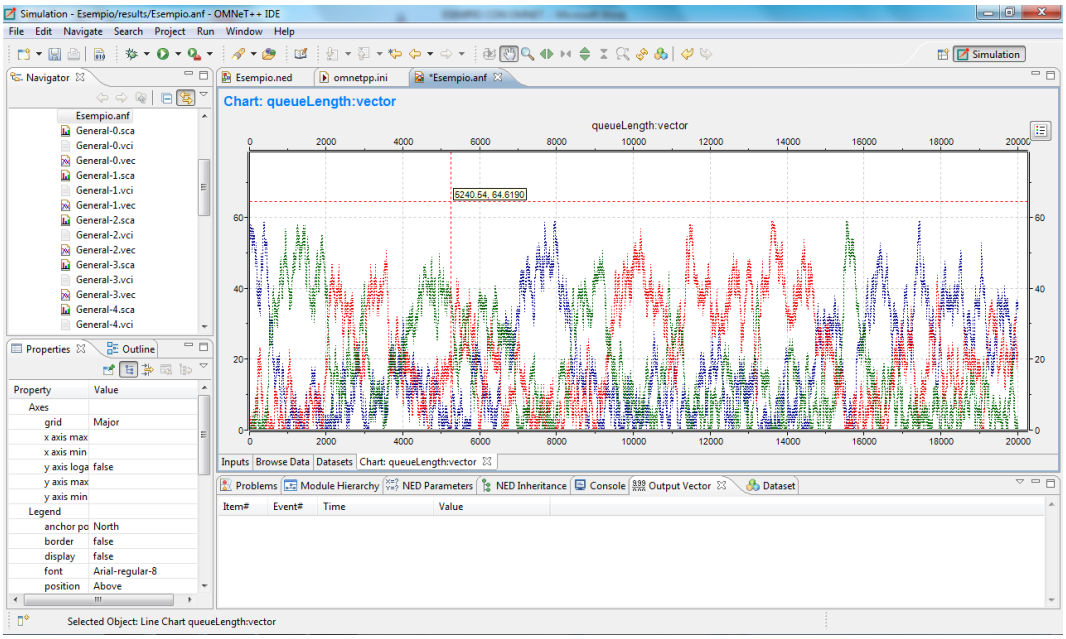
- Passiamo ora alla pagina Browse data. La tabella visualizza i vettori registrati nelle esecuzioni della simulazione. Noi siamo interessati a come le lunghezze della coda cambiano con il tempo, scegliamo perciò "queueLenght:vector" dal filtro.



- La tabella include dati di tutte le esecuzioni, concentriamoci quindi su Run 4.

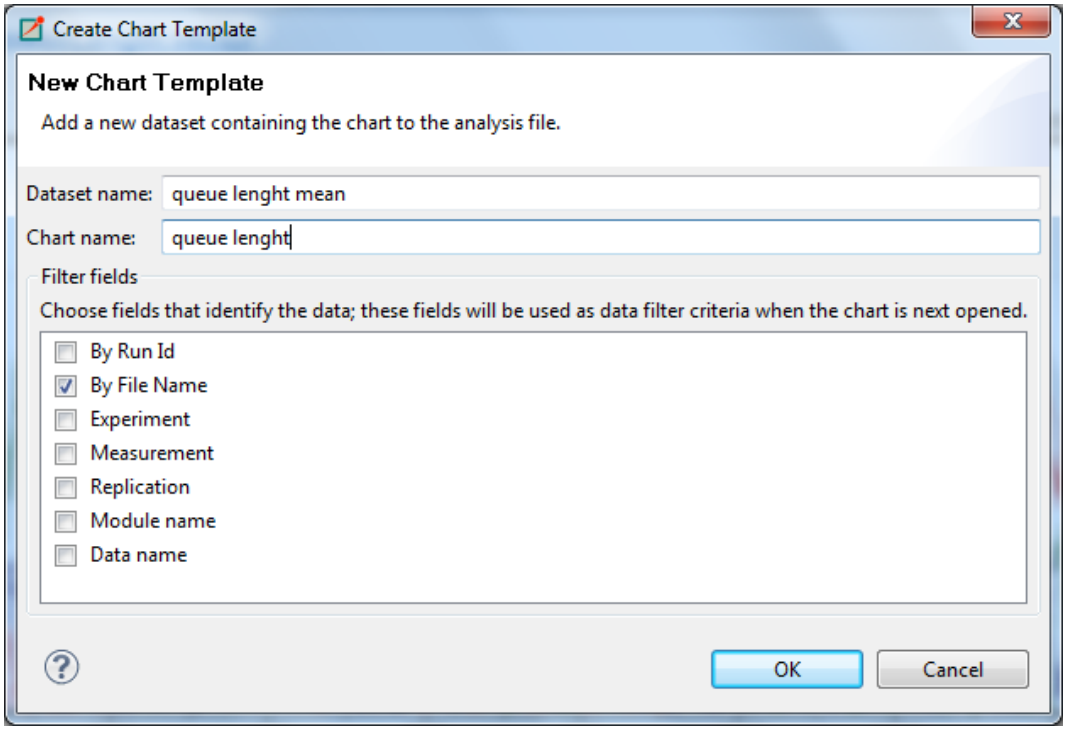


- Ci sono solo tre vettori, uno per ciascuna corsa. Riportiamoli su un singolo grafico. Ctrl+A per selezionarli tutti, poi cliccare su Plot che si trova sulla barra degli strumenti.



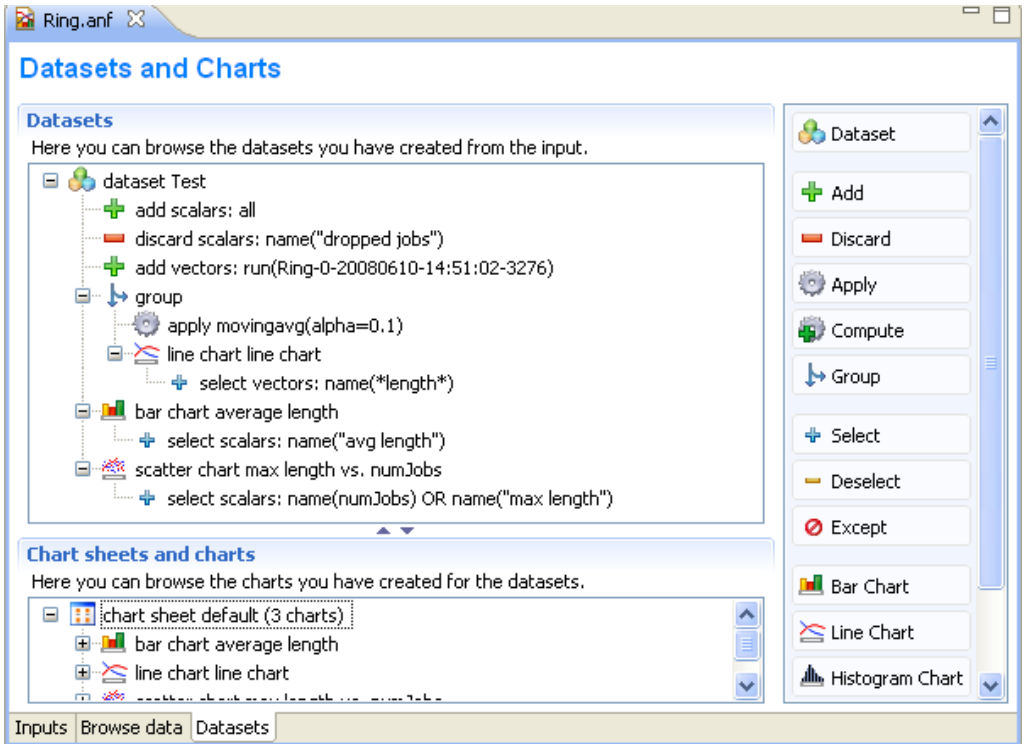
- Possiamo conservare questo grafico come “ricetta”, in modo tale che, quando la simulazione viene ri-eseguita, il grafico può essere creato automaticamente.

  Click destro sull’area del grafico→Convert to Dataset→Inserire quindi nei vari campi i seguenti:



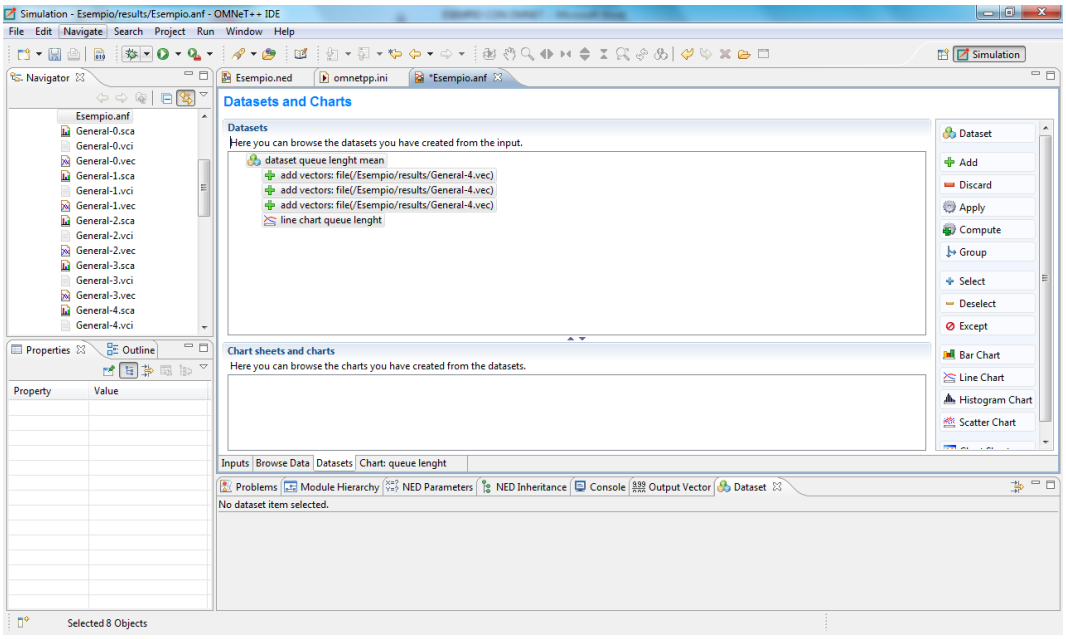
8.6. *DATASET*

La terza pagina visualizza i dataset e i grafici creati durante l'analisi. I dataset visualizzano i set di dati di input, il processo applicato e i grafici. Il dataset viene visualizzato come un albero. Ci sono nodi per aggiungere e scartare i dati, per applicare processi ai vettori, per selezionare gli operandi delle operazioni e i contenuti dei grafici, e per creare grafici.



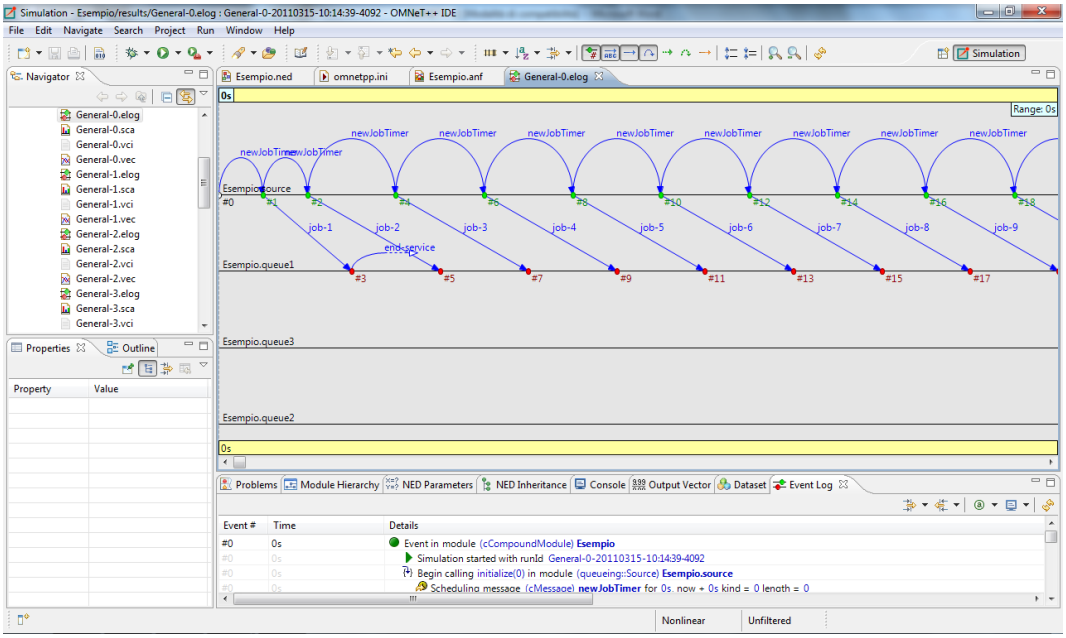
8.6.1.1. ESEMPIO

- La pagina di dataset visualizza le “ricette” usate per creare i grafici e i diagrammi. Esso contiene i passi di processo applicati in ordine dall’alto in basso.

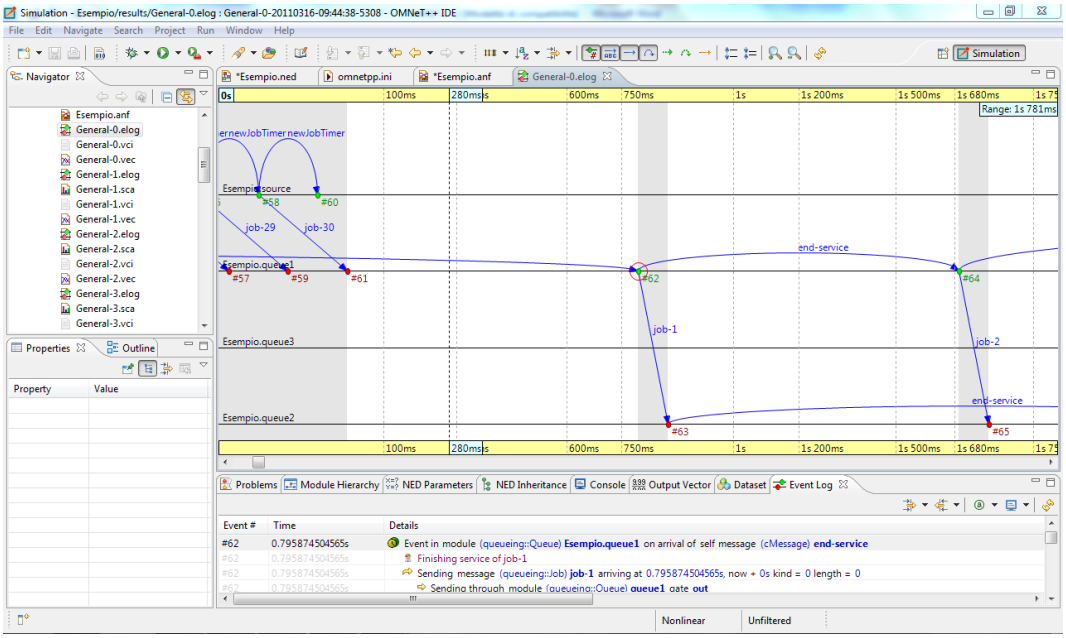


- Salvare l’analisi.

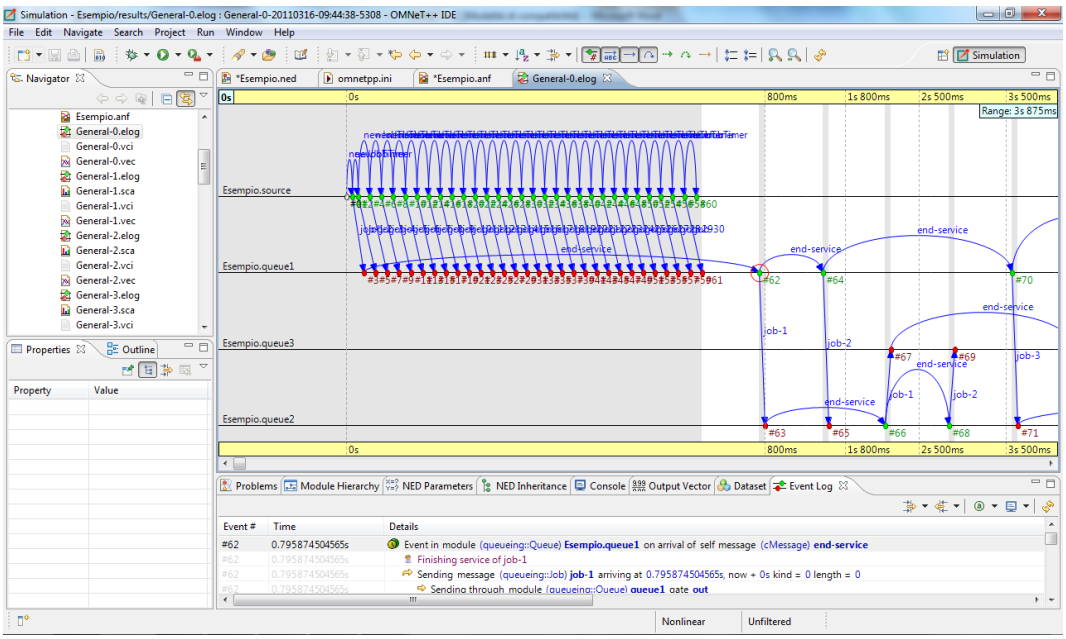
- Ora diamo un’occhiata a uno dei Sequence Chart creati durante la simulazione. È possibile vedere i primi 60 messaggi spinti dentro una delle code.



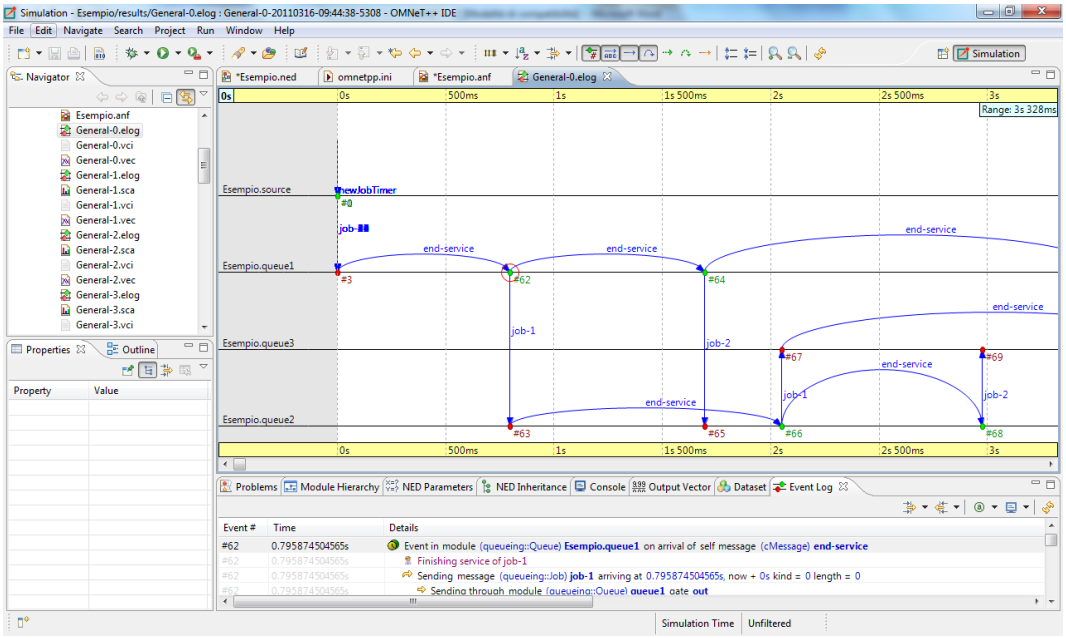
- Andiamo ora dove è stato processato il primo messaggio dalla coda. Quindi click destro del mouse su #3 → Sending (cMessage) end-service (id=1) → Goto Consequence Event.



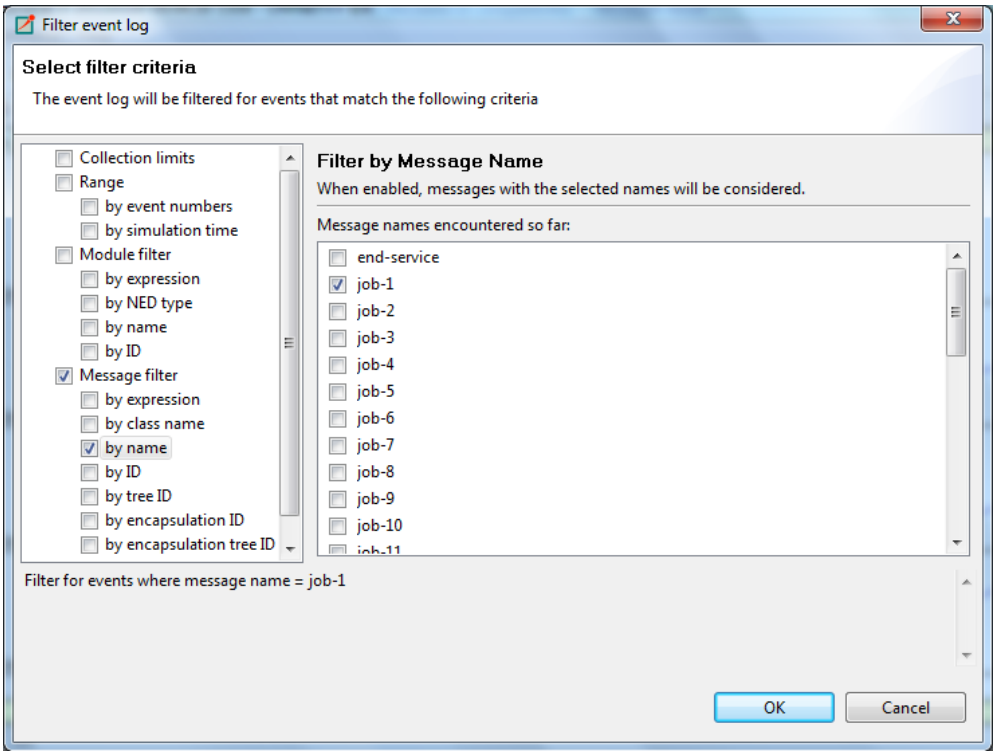
- È possibile zoomare all'indietro per avere una visione complessiva del processo.

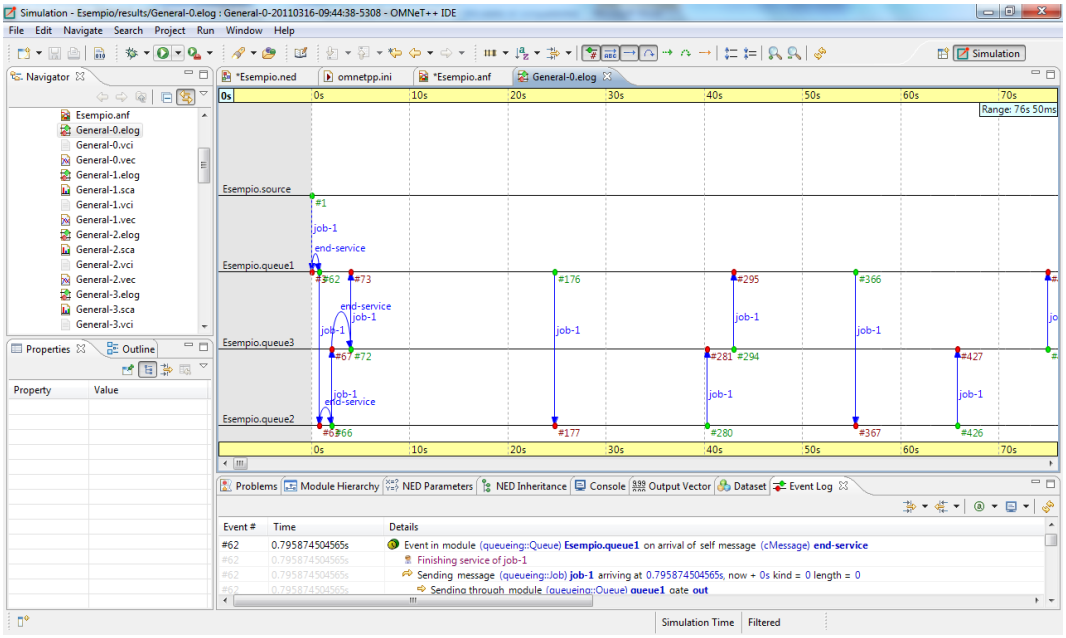


- In modalità di timeline non lineare, i messaggi nella stessa area grigia hanno lo stesso tempo di simulazione. Ora impostiamo la modalità di timeline lineare. Vedremo che i messaggi iniziali saranno inviati al tempo 0 di simulazione (le zone grigie collasseranno in una linea verticale). Cliccare su **Timeline Mode** che si trova sulla barra strumenti in alto e selezionare **Linear**.

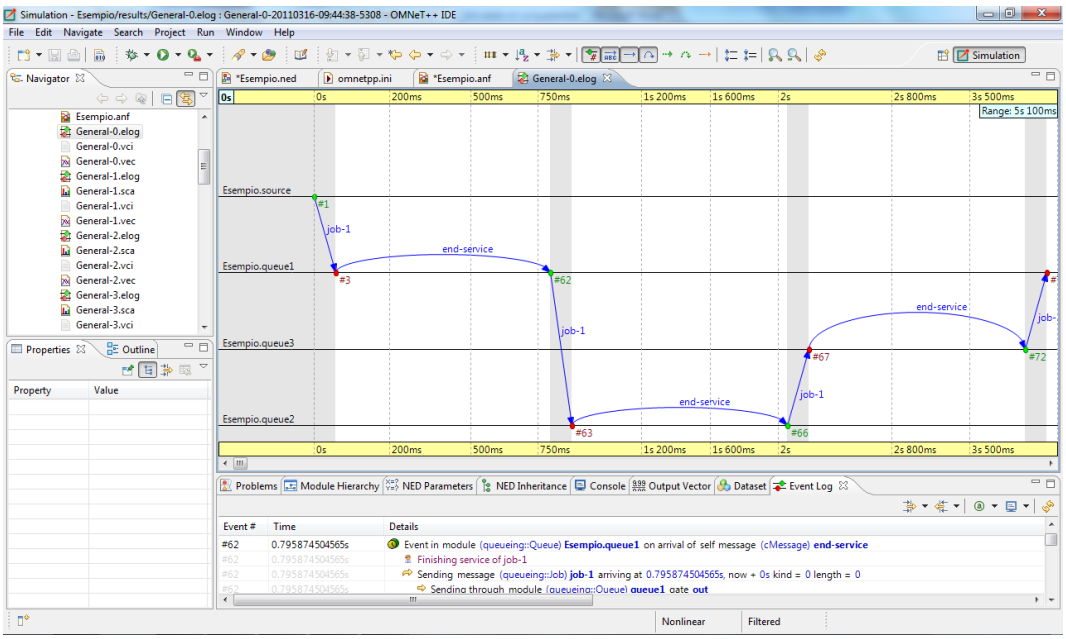


- Filtriamo ora il primo messaggio per vedere come circola nel network. Selezionare **Filter** sempre dalla barra strumenti in alto.

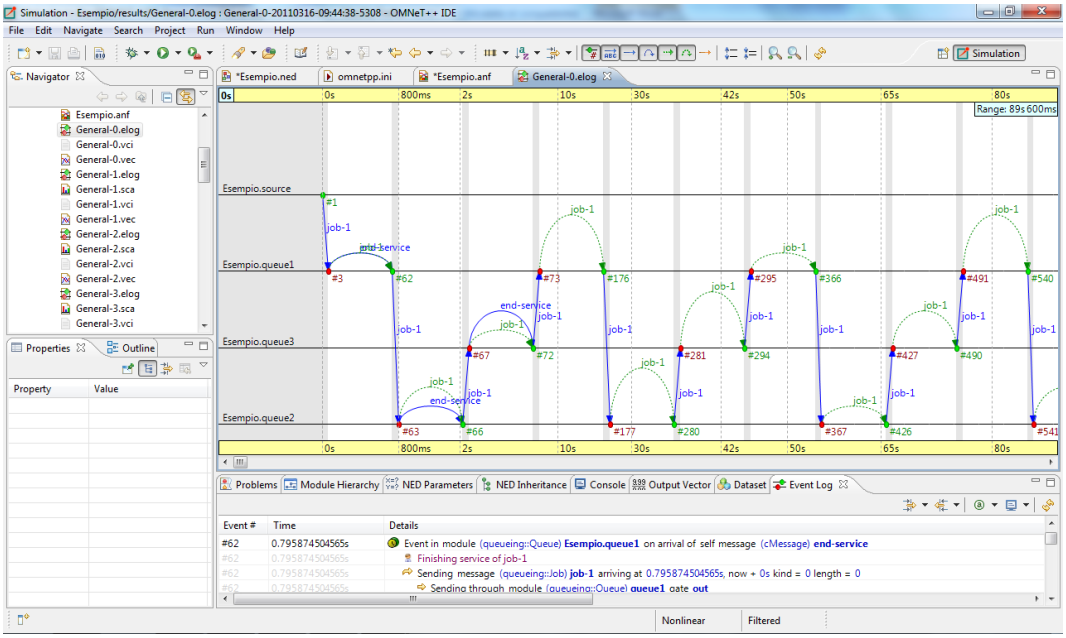




- Impostiamo la modalità non lineare per gira attorno alcune volte.



- Mostriamo dove i messaggi sono riutilizzati attraverso un riinvio degli stessi. Click destro del mouse sull'area del grafico → Show/Hide → Show Self-Message Reuses/Show Other Message Reuses.



9. GRAFICI

L'Editor di Analisi supporta diagrammi a barre, a linee, istogrammi e grafici a dispersione. I grafici sono interattivi: gli utenti possono zoomare, e accedere ai tooltip che danno informazioni riguardo gli elementi dei dati.

I grafici possono essere personalizzati. Alcune delle opzioni personalizzabili includono titoli, caratteri, leggende, linee, colori, stili di linea e simboli.

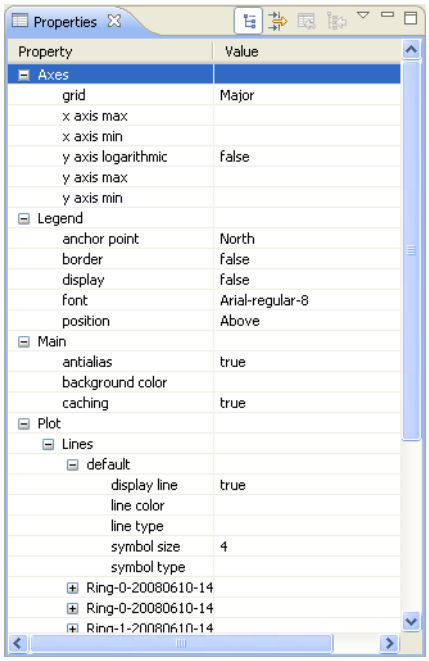
9.1. *CREARE I GRAFICI*

Per creare un grafico, si usa la tavolozza alla destra della pagina del *Dataset.* Si deve trascinare il pulsante del grafico che si vuole creare e rilasciarlo sul dataset in cui si vuole che appaia. Se si clicca sul grafico appena creato, si aprirà una finestra dove si potranno modificare le sue proprietà. In questo caso il nuovo grafico sarà aggiunto alla fine del dataset selezionato o dopo l'elemento del dataset selezionato.

9.2. *MODIFICARE I GRAFICI*

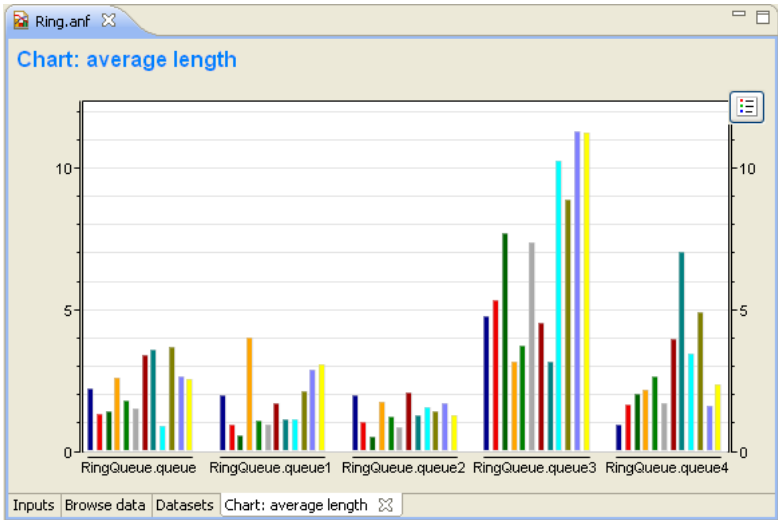
È possibile aprire una finestra per modificare i grafici dal context menù. La finestra è divisa in più pagine. Le pagine possono essere aperte direttamente dal context menù. Quando si seleziona una riga e si sceglie *Lines…* dal menù, si possono modificare le proprietà della riga selezionata.

È possibile anche utilizzare la *Properties View* per modificare il grafico.

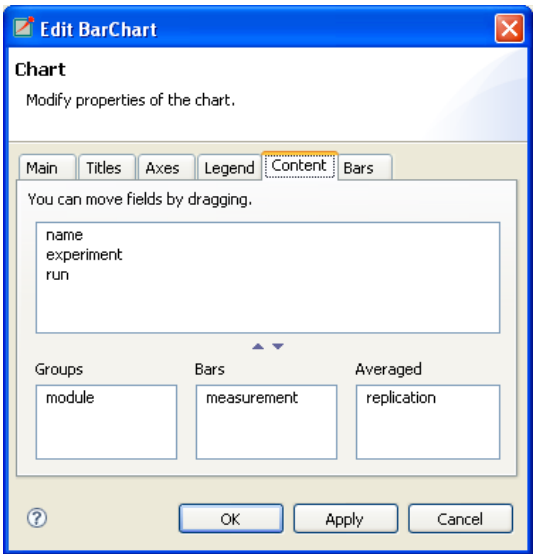


9.3. *DIAGRAMMI A BARRE*

I diagrammi a barre visualizzano gli scalari come gruppi di barre verticali. Le barre possono essere posizionate all'interno di un gruppo, e affiancato ad altri gruppi. La linea di fondo delle barre può essere modificata. Opzionalmente, può essere applicata una trasformazione logaritmica ai valori.

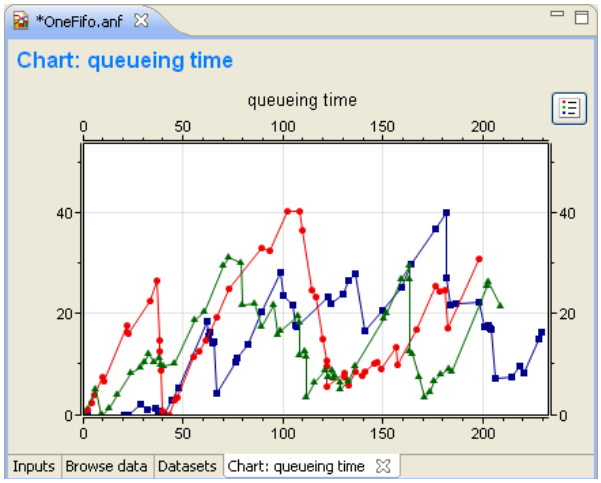


Il contenuto del grafico scalare può essere specificato sulla tab *Content* della finestra *Properties.* È possibile classificare i vari attributi trascinandoli dal box sovrastante ai box sottostanti. Normalmente si vorrà raggruppare gli scalari in moduli ed etichettare le barre con i nomi dello scalare. Questa è l'impostazione di default nel caso in cui si lascia ciascun attributo nel box sovrastante:



9.4. *DIAGRAMMI A LINEE*

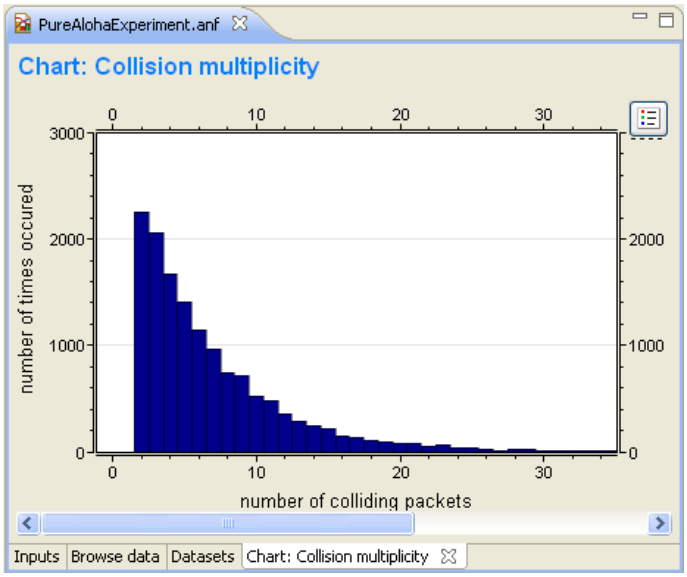
I diagrammi a linee possono essere utilizzati per visualizzare vettori di output. A ciascun vettore nel dataset corrisponde una linea nel diagramma. E' possibile specificare quali simboli utilizzare per rappresentare i dati (croci, punti, più, quadrati, triangoli o nessun simbolo), come connettere i punti e il colore delle linee. Le linee singole possono essere nascoste.



9.5. *ISTOGRAMMI*

Gli istogrammi possono visualizzare i dati degli istogrammi. Supportano tre differenti view:

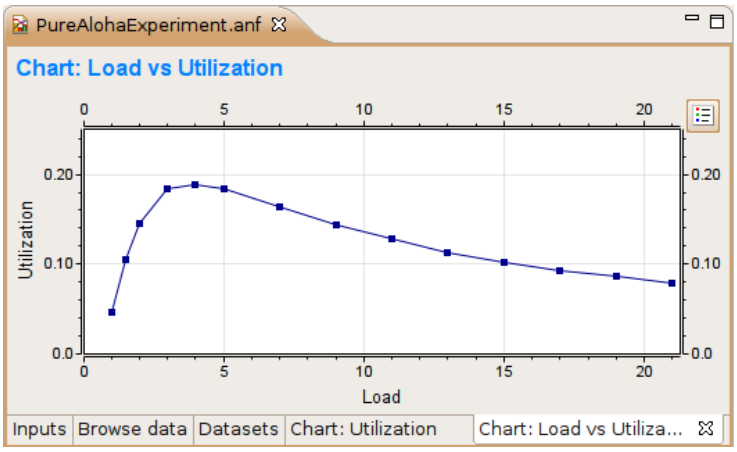
- Conta: il grafico mostra la somma registrata dei punti dei dati in ogni cella.
- Densità di probabilità: il grafico mostra la funzione densità di probabilità calcolata a partire dai dati dell'istogramma.
- Densità cumulativa: il grafico mostra la funzione di densità cumulativa calcolata a partire dai dati dell'istogramma.



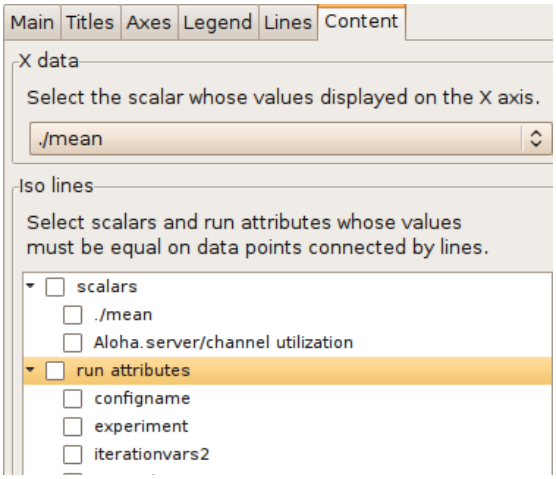
9.6. *GRAFICI A DISPERSIONE*

I grafici a dispersione possono essere creati a partire sia da dati scalari che vettoriali. È necessario selezionare una statistica per le coordinate x; mentre le voci degli altri dati danno le coordinate y. L'accoppiamento dei valori x e y cambia a seconda della presenza di scalari e vettori.

- **Scalari**: per ogni valore dello scalare x, i valori di y vengono selezionati dagli scalari nella stessa esecuzione.
- **Vettori**: per ogni valore del vettore x, i valori di y vengono selezionati dalla stessa esecuzione e dallo stesso tempo di simulazione.



Per default, ogni punto del dato che esce dallo stesso scalare y appartiene alla stessa linea. Però non è sempre ciò che si vuole perché questi valori potrebbero essere stati generati in esecuzioni aventi differenti impostazioni di parametro; quindi, non sono omogenei. È possibile specificare gli scalari per determinare le linee "iso" del diagramma a dispersione. Solo quei punti che hanno gli stessi valori di quegli attributi "iso" sono connessi da linee.


Main
Titles
Axes
Legend
Lines
Content
X data
Select the scalar whose values displayed on the X axis.
./mean
Iso lines
Select scalars and run attributes whose values must be equal on data points connected by lines.
scalars
./mean
Aloha.server/channel utilization
run attributes
configname
experiment
iterationvars2

CONCLUSIONE

Questa tesi ha l'obiettivo di proporre una panoramica generale sul tema della simulazione discreta e, in particolare, presentare il software di simulazione ad eventi discreti OMNeT++.

Si sono discussi gli elementi generali di tale software (campo di utilizzo, applicazioni, funzionamento, etc), al fine di garantire un'utile strumento per l'esecuzione del programma. Inoltre, per facilitare la comprensione di utilizzo, è stato inserito un esempio riguardante un Network di coda. Questo esempio ha permesso di rendere più chiaro ciò che OMNeT++ è in grado di fare.

Personalmente ho trovato questo software molto intuitivo e di facile utilizzo. La sua progettazione è stata molto curata e si nota un notevole miglioramento della versione 4.x rispetto alla precedente 3.x. Grazie alle numerose e intuitive interfacce grafiche, l'utente può facilmente creare i più disparati scenari di simulazione in campi anche molto diversi tra loro.

OMNeT++ è risultato, quindi, essere un ottimo software di simulazione ad eventi discreti. Anche se è nato principalmente come simulatore di Network di reti di comunicazione, lo consiglio anche per tutte le altre simulazioni ad eventi discreti.

Bibliografia

- **Carlucci D., Menga G., 1998, Teoria dei sistemi ad eventi discreti: UTET Libreria.**

- **Di Febbraro A., Giua A., 2002, Sistemi ad Eventi Discreti: McGraw-Hill.**

- **Martinoli B., 1993, Guida alla simulazione: FrancoAngeli.**

- **OMNeT++ Simulation Software (www.omnetpp.org)**

- **OMNeT++ User Guide 4.1** (Copyright © 2010 András Varga and OpenSim Ltd.)

- **OMNeT++ User Manual 4.1** (Copyright © 2010 András Varga and OpenSim Ltd.)