11 Diff 1.0-1.1

Rispetto alla versione 1.0 sono cambiati i seguenti punti:

- aggiunti ritorno carrello nelle grammatiche dei messaggi. La grammatica completa a questo punto é la seguente:

```
<ReqMsg> ::= SERVERFARM <ReqOp> \n <ReqBody> END \n
<ReqOp>  ::= SERVICE | SERVICEN
<ReqBody>::= <Task> \n | <Task> <ReqBody>
<Task>   ::= FUNCTION <FName> \n DATA <Data>
<AnsMsg> ::= SERVERFARM <AnsType> NUM <Integer>\n <AnsBody> \n END \n
<AsnType>::= OK | NOK
<AnsBody>::= <Ans> \n | <Ans> <AnsBody>
<Ans>    ::= <Summary> RESULT <Data>
           | <Summary> ERROR <Data>
<Summary>::= ANSWER <FName> <Data> HOST <Hostaddress>
<CtlMsg> ::= SERVERFARM CTL <CtlOp> \n END
<CtlOp>  ::= STATS | SHUTDOWN | RESET
<CtlAns> ::= SERVERFARM <CtlATy> \n <CtlBody> \n END \n
<CtlATy> ::= REPORT | OK
<CtlBody>::= <Data> \n | <Data> <CtlBody>
```

- é stato aggiunto, nei messaggi di tipo controllo che il client puó mandare al server farm, il messaggio di tipo `RESET`
- é stato aggiunto un esempio di messaggio di tipo `REPORT`

```
SERVERFARM REPORT
Host arruolati: 3
Task calcolati: 6034
Tempo medio per task: 2.2333 msec
Utilizzo: 23\%
Host: fujih12.cli.di.unipi.it attivo da Thu Dec  7 08:41:50 CET 2006  2014 task (tempo medio 2.34 msec)
Host: fujih14.cli.di.unipi.it attivo da Thu Dec  7 08:40:33 CET 2006  1997 task (tempo medio 2.14 msec)
Host: fujih12.cli.di.unipi.it attivo da Thu Dec  7 08:40:18 CET 2006  2023 task (tempo medio 2.22 msec)
END
```

- si é specificato che i nomi della classi e delle variabili indicati nel progetto devono essere utilizzati alla lettera
- si é specificato che la relazione non deve essere piú di 10 pagine, formato di stampa A4
- si sono aggiunte delle note al paragrafo che spiega come debbono essere effettivamente implementate le “funzioni” compreso un esempio di funzione “pesante”.

```
import serverFarm.Compute;
public class Inc implements Compute {
  /** tempo di "calcolo" della funzione */
  long msecs = 0L;
  /** @param secs numero di secondi da spendere nel calcolo della funzione<br>attenzione: si puo'
   * usare la sleep solo perche' si assume che un RemoteServer serva una richiesta alla volta ... */
  public Inc(int secs) {msecs = secs * 1000L; }
  /** metodo che calcola la funzione
   * @param in valori interi in ingresso. Anche se ne passiamo uno solo dal momento che e'
   *  un metodo con numero di parametri int variabile, il parametro si accede come in[0] ...
   * @return il valore calcolato dalla funzione "Int" (nome della claase)    */
  public int exec(int ... in) {
    try {Thread.sleep(msecs);} catch (InterruptedException e) {} // posso non fare nulla in questo caso
    return(in[0]+1);
  }

}
```