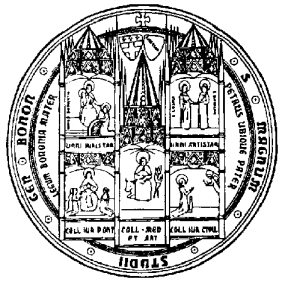


**Diploma di Laurea in Ingegneria Informatica**
**Corso di Ingegneria del Software**

# Esercitazione n°1

# Servizio di Prenotazione via Web

## Analisi e Specifica dei Requisiti

ing. Paolo Bellavista

e-mail: pbellavista@deis.unibo.it tel: (20) 93866

# Fasi del Processo di Sviluppo

- Analisi e Specifica dei Requisiti

  quale formalismo di rappresentazione?

- Progettazione

  quale metodologia?

- Implementazione e Codifica

  quali sistemi e quale linguaggio?

- Convalida e Verifica

  quale tipologia di test?

- Documentazione delle fasi precedenti + Manuale Utente

# Documento di Specifica dei Requisiti

- (Glossario)
- Definizione dei requisiti utente
- Specifica dei requisiti funzionali
- Specifica dei requisiti non funzionali (vincoli progettuali disgiunti da quelli di servizio)
- Diagramma dei casi d’uso
- Diagrammi di flusso dei dati
- Specifica del sistema in un linguaggio formale
- Piano di test del sistema

# Definizione preliminare del problema (1)

Si vuole creare un sistema di ***prenotazione dei posti di un teatro***. Le prenotazioni si devono poter fare ***tramite Internet***, usando un numero (non prefissato) di postazioni.

- Ogni cliente, usando una macchina opportunamente installata, può prenotarsi per un posto libero in uno spettacolo in programma (esistente).
- Il cliente può sbagliarsi o doversi trovare nelle condizioni di annullare una sua prenotazione. Chi ***amministra il sistema*** deve poter aggiungere nuovi spettacoli ed eliminare quelli vecchi.
- Il sistema deve essere ***affidabile***: deve essere sempre disponibile per gli utenti il servizio di prenotazione e non devono essere perse le prenotazioni fatte. Occorre prevedere anche la possibilità di sospendere il servizio (memorizzazione delle prenotazioni da ricaricarsi al riavvio).

# Definizione preliminare del problema (2)

Poiché sono *possibili guasti*, occorre predisporre più macchine (non necessariamente due soltanto - cliente/servitore) per la memorizzazione delle prenotazioni.

È richiesto un ***sistema dinamico***: le macchine devono poter essere in numero non prefissato; un amministratore deve avere sempre la possibilità di aggiungere o togliere macchine senza interrompere il servizio.

Il cliente può, infine, non conoscere gli spettacoli in programma: occorre prevedere una funzionalità di elencazione degli spettacoli disponibili. Analogamente un cliente deve poter sapere quali e quanti posti sono liberi per un dato spettacolo.

# 2) ANALISI E SPECIFICA DEI REQUISITI ('COSA')

## 2.1) DEFINIZIONE DEI REQUISITI

> Le condizioni iniziali sono enunciati relativi a sistemi particolari. È compito dello scienziato spiegare il mondo finché è possibile senza far ricorso a condizioni iniziali particolari.
> (da "la mente di Dio" di Paul Davies)

### 2.1.1) Distinzione fisica

Si vuole che le postazioni da cui si fanno le prenotazioni siano fisicamente distinte da quelle che svolgono la funzione di memorizzazione delle prenotazioni fatte.

### 2.1.2) Distanza fisica

Si richiede che le prenotazioni avvengano tramite Internet.

### 2.1.3) Quantità di postazioni di prenotazione

Il numero delle macchine da cui si fanno le prenotazioni non è prefissato, e si vuole che sia modificabile dinamicamente.

### 2.1.4) Quantità di postazioni di memorizzazione delle prenotazioni

Il numero di tali macchine non è prefissato e si vuole che in qualsiasi momento ne possa essere aggiunta o tolta una.

### 2.1.5) Affidabilità

Il sistema complessivo dev'essere affidabile, ossia occorre garantire (sotto certe ipotesi):

- che sia sempre possibile fare una prenotazione, da una qualsiasi postazione;
- che il sistema non perda tutte le prenotazioni che sono state fatte.

### 2.1.6) Conoscenza

Le postazioni di prenotazione devono conoscere quelle di memorizzazione, inoltre queste ultime devono conoscersi tra loro.

### 2.1.7) Concorrenza

Per una postazione di prenotazione è richiesto solamente che faccia una prenotazione e attenda il risultato, ossia mentre si attende il risultato non può essere fatta nessun altra prenotazione.
In una macchina di memorizzazione è richiesto che possa servire più richieste di prenotazione che possono avvenire contemporaneamente, anche da client diversi!

### 2.1.8) Guasto

I guasti sono possibili e, nei limiti del possibile, bisogna fare in modo che non compromettano il normale funzionamento del sistema.
Occorre prevedere cosa fare quando "cade":

- una postazione che ha fatto una richiesta;
- una postazione di memorizzazione mentre non sta facendo nulla;
- una postazione di memorizzazione che sta portando a termine una prenotazione.

## 2.1.9) Funzionalità

Si può notare che ognuna delle due categorie di “postazione” ha un suo particolare “utente” e perciò deve avere funzionalità differenti.
Sulle postazioni di prenotazione ci sono gli utenti che vogliono prenotarsi, mentre su quelle di memorizzazione (su una o più) ci sono gli amministratori del sistema (oppure gli stessi gestori del teatro) che mantengono al corrente il sistema sugli spettacoli del teatro.

### 2.1.9.1) Postazione di prenotazione

- permettere l’inserimento dei dati di una prenotazione, da parte dell’utente;
- inviare alle postazioni di memorizzazione la richiesta di una prenotazione;
- inviare la richiesta di cancellare una prenotazione;
- richiedere a una postazione di memorizzazione tutti gli spettacoli in programma;
- richiedere tutti i posti che sono ancora liberi in un particolare spettacolo

### 2.1.9.2) Postazione di memorizzazione

- accettare le prenotazioni che arrivano da qualsiasi postazione di prenotazione;
- accettare le richieste di informazioni da parte delle postazioni di prenotazione;
- portare a termine una prenotazione, accertando che tutte le postazioni di memorizzazione l’abbiano ricevuta e registrata;
- aggiunta di un nuovo spettacolo, da parte dell’amministratore della postazione;
- cancellazione di uno spettacolo (che è già stato rappresentato);
- possibilità di connettersi, come nuova macchina, a un sistema già funzionante;
- possibilità di disconnettersi dal sistema.

## 2.1.10) Collisione di Prenotazioni

Può accadere che, da differenti postazioni di prenotazione, due o più utenti tentino di prenotare uno stesso posto (per un certo spettacolo).
Al più, solamente una delle prenotazioni dovrà essere registrata da tutte macchine di memorizzazione; tutte le altre devono essere rifiutate (e chi le ha fatte dev’esserne informato).

# 2.2) SPECIFICA DEI REQUISITI

> Non è la velocità o la potenza del cervello a distinguere l'uomo dalla macchina bensì l'ironia.
> (Luciano De Crescenzo)

## 2.2.1) Distinzione fisica

Per distinguere tra le postazioni da cui si fanno le prenotazioni e quelle in cui vengono memorizzati i dati di tutte le prenotazioni, si deve prevedere una distinzione tra:

- i **client** (clienti), ossia le postazioni da cui ogni utente può inserire e richiedere una prenotazione (facendo domanda a un server);
- i **server** (servitori), ossia le macchine che servono da memoria dei dati di tutte le prenotazioni.

> Poi bisogna tenere conto che le distanze calcolate con il compasso sulla carta non coincidono esattamente con quelle misurate dalle gambe.
> (da "tre uomini a zonzo" di J.K.Jerome)

## 2.2.2) Distanza fisica

L'uso di Internet permette di annullare la distanza fisica tra l'utente e il teatro.
Bisogna che i client e i server comunichino tra loro usando i protocolli di Internet.

## 2.2.3) Quantità di postazioni di prenotazione

Il numero di client non è prefissato e dev'essere variabile dinamicamente.
Occorre prevedere la possibilità di inserimento di un nuovo client.
In particolare, si deve metterlo a "conoscenza" dell'esistenza di un server.

## 2.2.4) Quantità di postazioni di memorizzazione delle prenotazioni

Anche il numero di server non è prefissato e dev'essere variabile dinamicamente.
Ovviamente sotto l'ipotesi che nel sistema siano presenti almeno due macchine server, altrimenti verrebbe compromessa l'affidabilità del sistema.
Bisogna prevedere dei protocolli per l'inserimento e la rimozione di un server.
L'inserimento è la fase più critica perché il nuovo server inserito, prima di poter funzionare, deve portarsi in uno stato "consistente" con quello di tutti gli altri server del sistema.
Egli dovrà necessariamente copiarsi lo "stato" di uno degli altri server, perciò occorre fare in modo che tale "stato" non venga modificato durante la copia.
Come per il client, un nuovo server deve "conoscere" almeno uno dei server (a cui chiedere lo stato).

> I computer sono inaffidabili, ma gli uomini ancora di più.
> (da "la legge di Murphy" di Arthur Bloch)

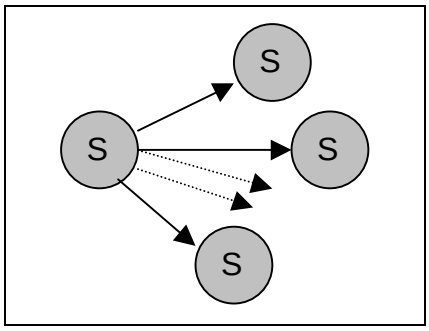
## 2.2.5) Affidabilità

Se si vuole che ogni client sia sempre in grado di fare una prenotazione, è necessario che ogni client "conosca" più di un server.
Infatti, se ogni client fosse a conoscenza dell'esistenza di un solo server, nel momento in cui tale server si guastasse, tutti i client che lo riferivano non saranno più in grado di fare alcuna prenotazione!
Per non perdere i dati delle prenotazioni fatte si deve necessariamente ricorrere alla replicazione dei dati all'interno d'ogni server.
L'idea è di avere più un server, ognuno con una sua copia locale dei dati.
Tutti i server sono delle **copie attive**, ossia ognuno di essi può ricevere e servire una richiesta di prenotazione da parte di un qualsiasi cliente!
Occorre che i server si coordinino per mantenere lo stato locale identico per tutti.
**Ipotesi di affidabilità:** i server devono essere almeno due!

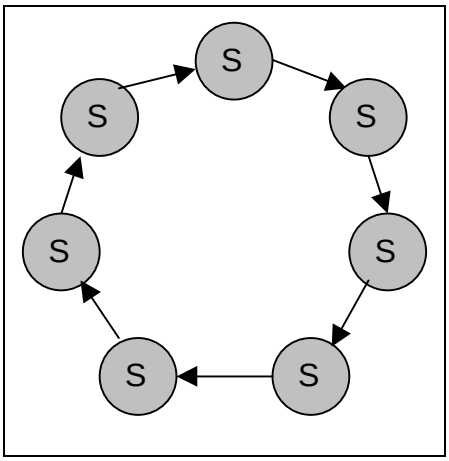
## 2.2.6) Conoscenza

> Conosci te stesso.
> (Chilone di Sparta)

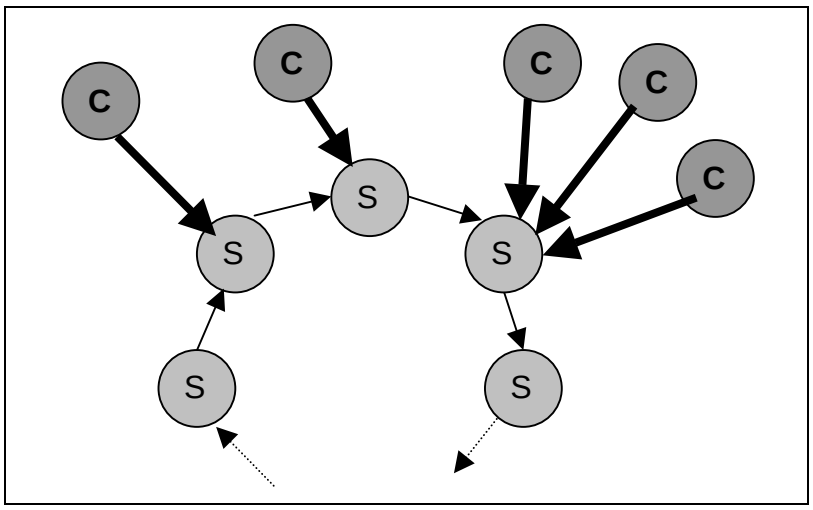
Poiché è richiesto che il sistema sia "dinamico" e che le comunicazioni avvengano in Internet, nasce il problema che ogni client deve conoscere l'indirizzo di almeno uno dei server (non ho trasparenza all'allocazione dei server).
Si può pensare che tale "conoscenza" sia fornita dall'utente (per l'esattezza, da chi configura il sistema) al momento dell'installazione.
Basta un solo indirizzo di un server, poiché ulteriori indirizzi possono essere richiesti a lui.
Un discorso analogo vale per l'inserimento nel sistema di un nuovo server.
Ogni server deve per forza conoscere tutti gli altri?




Si può pensare che ogni server mantenga in una tabella gli indirizzi di tutti i server del gruppo, così da poter comunicare con tutti.
Ma questa struttura ha uno svantaggio: c'è un forte sovraccarico di comunicazione da parte di un solo server. Si può benissimo pensare che il sistema all'inizio abbia solo due server o pochi di più e che quindi i client comunichino con essi.
Se successivamente si inseriscono altri server, poiché i client continuano a conoscere solo quelli iniziali, i nuovi venuti serviranno solo le comunicazioni con gli altri server.
Il risultato è che i vecchi server saranno sovraccaricati (sia dalle comunicazioni con i client sia da quelle con i nuovi server).




Ho pensato di usare una struttura a **anello.**
In un anello ogni server conosce solamente due server: il suo successore e il suo predecessore nella catena.
Ovviamente è un **anello virtuale**, poiché ogni server, se volesse, potrebbe comunicare con qualsiasi altro server… ma non può perché non ne conosce l'indirizzo!



## 2.2.7) Concorrenza

Il client sarà un programma monoprocesso, ossia esso passa tra i suoi due stati (quello di inserimento della prenotazione da parte dell'utente e quello di richiesta verso un server di fare la prenotazione, con conseguente attesa) in maniera sequenziale.
D'altronde prima di poter fare una nuova prenotazione bisogna avere la certezza che quella appena fatta è stata registrata.
Il server invece deve poter servire più richieste in parallelo; sarà perciò un programma multiprocesso.
L'idea di base è di avere un processo che attende l'arrivo di un messaggio; non appena ne arriva uno, viene generato un nuovo processo che legge il messaggio e lo serve (cioè fa le azioni che il particolare messaggio dice di fare).
Mentre il nuovo processo fa ciò, il processo principale torna subito a accettare un nuovo messaggio.

## 2.2.8) Guasto

> Sopporta di essere danneggiato un poco dal tuo vicino.
> (Pittaco di Mitilene)

I guasti sono possibili, ma non devono compromettere l'intero servizio. Se un server si guasta deve essere tolto dal sistema, ossia si deve fare in modo che tutti gli altri server riescano a riportarsi in una configurazione di normale funzionamento.
Sarà compito del server caduto di riunirsi (aggiungersi) al sistema.
Chi controlla? Il controllo se un server è "caduto" è lasciato a chi sta cercando di comunicare con esso. Un server può accorgersi che il suo successore è caduto e quindi per comunicarlo agli altri può solo inviare un messaggio nel senso opposto!
Invierà un messaggio di **riaggancio** che conterrà l'indirizzo del server caduto e il proprio; il server che riconoscerà come suo predecessore il server caduto, invierà al server che ha rilevato l'errore un messaggio con il suo indirizzo (può farlo perché la catena è virtuale!) e aggiornerà il riferimento al suo predecessore; analogamente farà il server che ha rilevato l'errore con il riferimento in avanti.
Nel frattempo, tutti i messaggi che arrivano a chi ha rilevato l'errore vanno fermati (fintantoché non si ripristinerà l'anello).
**Ipotesi di guasto singolo**: non possono verificarsi due guasti contemporaneamente!
Occorre prevedere cosa fare quando "cade":

- un client che ha fatto una richiesta (e sta aspettando la risposta che la prenotazione è stata fatta o che è stata annullata);
- il server a cui il client ha appena fatto una richiesta (non ha ancora fatto circolare la prenotazione)
- lo stesso server, ma mentre la prenotazione è in circolo
- uno qualsiasi degli altri server
- idem, ma tale server ha appena ricevuto il messaggio di prenotazione (viene perso il messaggio!)

## 2.2.9) Funzionalità

> Il povero non sa che la sua funzione nella vita è permetterci l'esercizio della generosità.
> (Jean-Paul Sartre)

### 2.2.9.1) Lato Client

- inserimento dei dati di una prenotazione, da parte dell'utente;
- inviare a un server la richiesta di una prenotazione e attendere la risposta;
- inviare la richiesta di cancellare una prenotazione e attendere la risposta;
- richiedere a un server l'elenco di tutti gli spettacoli in programma;

- richiedere tutti i posti che sono ancora liberi in un particolare spettacolo.

#### 2.2.9.2) Lato Server

- accettare le richieste di prenotazione che possono arrivare da qualsiasi client;
- accettare le richieste di informazioni (spettacoli, posti liberi) da parte dei client;
- portare a termine una prenotazione, inviandola al suo successore e attendendo che ritorni a sé in uno dei due versi.
- aggiunta di un nuovo spettacolo, da parte dell'amministratore del server;
- cancellazione di uno spettacolo (che si è concluso);
- possibilità di connettersi, come nuovo server, a un sistema già funzionante;
- possibilità di disconnettersi dal sistema.

> Bello, quando sul mare si scontrano i venti e la cupa vastità delle acque si turba... guardare da terra un naufrago lontano e rallegrarsi dello spettacolo dell'altrui rovina. (Lucrezio)

## 2.2.10) Collisione di Prenotazioni

Quando due o più utenti inviano una richiesta di prenotazione per uno stesso posto (per uno stesso spettacolo) occorre stabilire come deve comportarsi ogni server.
Se le richieste arrivano tutte su uno stesso server, poiché il data base è acceduto in mutua esclusione, solamente una di loro sarà registrata (e quindi inviata agli altri server).
Se invece le richieste arrivano su server differenti, poiché trovano il posto libero nel data base locale, vengono registrate e inviate in circolo nell'anello.
Entro breve tempo una di loro arriverà a uno degli altri server interessati; ma in esso il posto è già stato prenotato.
Non potendo scriversi nel data base, la nuova prenotazione dev'essere annullata (ossia si rimanda indietro nell'anello un messaggio di cancellazione della prenotazione).
E così succederà anche per le altre.
Ma in questo è molto probabile che tutte le prenotazioni falliscano (dicendo all'utente che il posto è già stato occupato) ma in realtà il posto non è occupato!
Occorrerà prevedere quest'eventualità e fare in modo che non tutte le prenotazioni che collidono vengano rifiutate, ma anzi che una e soltanto una venga accettata da tutti i server (ovviamente, dev'essere la stessa per tutti).

# 2.3) SPECIFICA DEL SOFTWARE

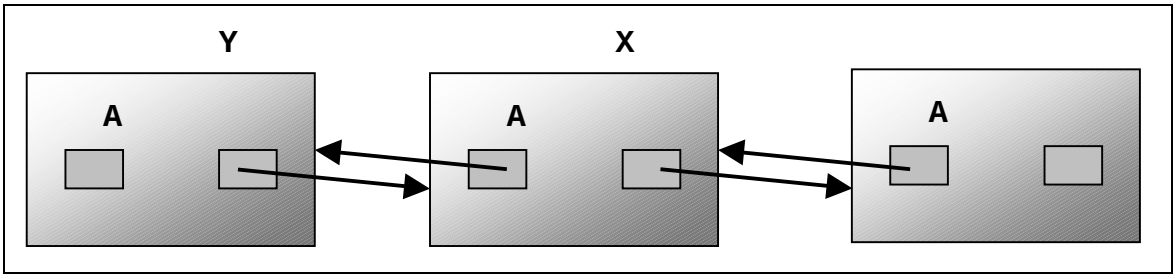
## 2.3.1) Anello Virtuale

> L'invenzione, secondo me, deriva direttamente da un certo ozio, forse addirittura da una certa pigrizia.
> (Agatha Christie)

Per realizzare l'anello virtuale, ogni nodo dovrà mantenere l'indirizzo del server che lo segue nella catena; aggiungiamo anche l'indirizzo del suo predecessore (che servirà per gestire i fallimenti e i guasti).
I server possono scambiarsi messaggi in ambo i sensi:

- quelli "**in avanti**" saranno messaggi di registrazione e di cancellazione;
- quelli "**indietro**" saranno messaggi di cancellazione e di controllo.

Ogni server dovrà mantenere in una **coda** tutti i messaggi di prenotazione (che riceve dai client) fintantoché tutti i server non hanno fatto la prenotazione.




## 2.3.2) Funzionamento "normale"

FASE 1:
Un client crea e invia una richiesta di prenotazione a uno dei server.
Tale server controlla se la prenotazione è corretta e sia inseribile.
Se non lo è, lo segnala al client.

> In tutto l'universo il nostro pianeta è probabilmente l'unico in cui esistono le galline. E l'universo è sconfinato! Quindi, come possiamo dire che una gallina è 'normale'?
> (da "c'è nessuno?" di Jostein Gaarder)

Se tutto è a posto, esegue la prenotazione nella sua copia locale, invia un messaggio di prenotazione al suo server successore e inserisce nella coda i dati sul messaggio (che devono contenere il relativo cliente).
Il messaggio farà tutto il giro e tornerà al server che l'ha inviato.
FASE 2:
All'arrivo di un tale messaggio, ogni server controlla che non sia stato generato da lui (basta che controlli la sua coda delle prenotazioni da fare).
In tal caso, il messaggio va fermato, si estraggono dalla coda i dati relativi al messaggio e si dice al client che la prenotazione è stata fatta.
Altrimenti si esegue la prenotazione e si passa avanti il messaggio.
NB: le due fasi sono separate! Tra una e l'altra passa necessariamente un po' di tempo (il messaggio deve fare il giro dell'anello) e quindi il server deve mettersi in attesa.

## 2.3.3) Start-Up: aggiunta di un nuovo server

Per aggiungere un nuovo server occorre sapere l'indirizzo di almeno uno dei server già attivi. Il nuovo server deve inserirsi nella catena perciò occorre modificare i collegamenti "avanti" e "indietro" di due dei server della catena (quello dato e il suo successore).
Occorre anche copiare lo stato locale del server attivo in quello nuovo, ma bisogna prima impedire la circolazione (e esecuzione) dei messaggi in ambo i sensi (poiché altererebbero lo stato dei server mente lo si sta copiando su un altro).