HAPPY CODE INC

**BR-jsys** *business rules* per sistemi gestionali in architettura J2EE

# Piano Di Qualifica

---

Versione 1.4 - 16 febbraio 2008

## Capitolato: "BR-jsys"

| | |
|---|---|
| **Data creazione:** | 19/11/07 |
| **Versione:** | 1.4 |
| **Stato del documento:** | Formale, esterno |
| **Revisione RR** | |
| **Redazione:** | Luca Appon |
| **Revisione:** | Marco Tessarotto |
| **Approvazione:** | Elena Trivellato |
| **Revisione RPD** | |
| **Redazione:** | Elena Trivellato, Alessia Trivellato |
| **Revisione:** | Mattia Meroi, Marco Tessarotto |
| **Approvazione:** | Michele Bortolato |

## Lista di distribuzione

| | |
|---|---|
| HappyCode inc | Gruppo di lavoro |
| Tullio Vardanega, Renato Conte | Committenti |
| Zucchetti S.r.l | Azienda proponente |

## Diario delle modifiche

| Versione | Data rilascio | Descrizione |
|---|---|---|
| 1.4 | 05/02/2008 | Correzione del documento |
| 1.3 | 05/02/2008 | Completamento tabella tracciamento Requisiti-Test |
| 1.2 | 04/02/2008 | Correzione grammaticale documento |
| 1.1 | 28/01/2008 | Modifica al "Processo di ispezione" |
| 1.0 | 25/01/2008 | Aggiunta della tabella di tracciamento Requisiti-Test |
| 0.6 | 24/01/2008 | Modifica al capitolo "Pianificazione ed esecuzione del collaudo" |
| 0.5 | 23/01/2008 | Modifiche allo "Scopo del documento" |
| 0.4 | 22/01/2008 | Modifica al layout dei documenti |
| 0.3 | 21/12/2007 | Documento sottoposto a revisionamento automatico |
| 0.2 | 06/12/2007 | Correzione errori |
| 0.1 | 19/11/2007 | Stesura preliminare del documento |

# Indice

# Capitolo 1

# Introduzione

## 1.1 Scopo del documento

Nel presente documento illustreremo le strategie di verifica e validazione adottate al fine di garantire la qualità attesa del nostro prodotto. Si utilizzerà l'analisi statica, sotto forma di ispezioni del codice e dei documenti, al fine di individuare errori e difetti negli stessi. Verrà inoltre utilizzata l'analisi dinamica del codice, sottoforma di test, per soddisfare i requisiti quali funzionalità, affidabilità e usabilità del prodotto "BR-jsys".

- Funzionalità:
  Il prodotto soddisferà pienamente i requisiti descritti nel documento di "AnalisiDeiRequisiti".
- Affidabilità:
  Il sistema sarà privo di errori in quanto la verifica verrà effettuata attraverso opportuni strumenti di controllo (test e ispezioni).
- Usabilità:
  L'utente finale del sistema non dovrà necessariamente avere grandi competenze nel campo informatico.

## 1.2 Scopo del prodotto

Il prodotto richiesto verrà inserito nell'ambito di un progetto più ampio. Il suo scopo è quello di automatizzare il sistema di validazione dei dati in ingresso al database dell'applicazione principale.

## 1.3 Glossario

Viene fornito come documento esterno chiamato Glossario.1.8.pdf .

## 1.4 Riferimenti

- Capitolato d'appalto concorso per sistema "BR-jsys";
- Verbale dell'incontro con il proponente "Incontro2007-11-22.pdf";
- Verbale dell'incontro con il proponente "Incontro2008-02-05.pdf";
- "Ingegneria del Software" 8a edizione - Ian Sommerville;
- "Analisi dei Requisiti";
- "Norme di Progetto";
- "Piano di Progetto".

# Capitolo 2

# Visione generale della strategia di verifica

## 2.1 Organizzazione, pianificazione strategica e temporale, responsabilità

La HappyCode ha pianificato delle fasi temporali per definire, sviluppare e validare il prodotto; tali fasi saranno prese in analisi e descritte nel dettaglio nei successivi paragrafi.

### 2.1.1 Ciclo di vita

Le attività seguiranno un modello di tipo evolutivo che permetterà di intrecciare le attività di specifica, sviluppo e convalida del software, e di apportare eventuali modifiche ai documenti in tempi diversi. L'obiettivo è comprendere al meglio le richieste del cliente e sviluppare dunque una migliore definizione dei requisiti del sistema. Per arrivare ad un prodotto finale sarà quindi indispensabile lavorare in stretto contatto con il cliente, sottoponendogli periodicamente versioni parziali o prototipi del prodotto finale. Adotteremo lo sviluppo esplorativo, concentrandoci dapprima sulle parti del sistema che sono ben chiare (requisiti ben compresi) e, solo successivamente verranno aggiunte nuove parti/funzionalità a fronte di chiarimenti da parte del cliente.

### 2.1.2 Pianificazione delle attività

In una prima fase gli analisti discuteranno e cercheranno di comprendere al meglio il problema da risolvere, grazie soprattutto alle varie comunicazioni e incontri con il cliente. Una volta chiariti e consolidati i vari requisiti sarà quindi possibile la stesura del documento intitolato "AnalisiDeiRequisiti", che sarà la base per la fase di progettazione seguente. Queste attività potranno essere eseguite in modo parallelo su diverse parti del sistema,

come anche le attività di progettazione e verifica, in modo da consentire la riduzione di tempi e costi di consegna del prodotto. Al termine delle attività di progettazione seguirà quella di programmazione, che produrrà il prototipo richiesto. L'ultima fase prima del collaudo sarà dedicata ad un attenta verifica di tutto il lavoro svolto nelle precedenti fasi. Il verificatore ed il responsabile di progetto sono le figure alle quali verranno affidate le responsabilità pertinenti alle attività di verifica. Per una descrizione più dettagliata della pianificazione si veda il documento "Piano di Progetto".

## 2.2 Risorse necessarie, risorse disponibili

Le attività di verifica necessitano di risorse umane e tecnologiche. Il gruppo è composto da sette membri, ognuno dei quali durante tutto il periodo di lavoro dovrà assumere, in periodi di tempo diversi (in base alle disponibilità e competenze di ciascuno), tutti i ruoli significativi per lo sviluppo del prodotto. L'amministratore del progetto sarà tenuto a supervisionare tutte le fasi di verifica e gestire le varie risorse necessarie per consentire un'attività di buon livello, senza sprechi od oneri eccessivi. Per la comunicazione tra i componenti è stato creato un gruppo Google, accessibile ai soli membri, raggiungibile all'URL: *http://groups.google.com/group/happycodeinc*. Essendo inoltre ogni componente provvisto di un account Gmail, verrà utilizzato il sistema di chat locale per la comunicazione interattiva. Per l'archiviazione dei file verrà invece utilizzato un server SVN (vedi il documento "Norme di Progetto").

## 2.3 Strumenti, tecniche, metodi

### 2.3.1 Analisi Statica

Verrà utilizzata durante la stesura del codice sorgente per rilevare errori, omissioni o anomalie, oltre ad incongruenze che possono sorgere tra il progetto ed i requisiti; sarà quindi applicata sulla struttura delle varie componenti del sistema "BR-jsys". Essa comprende le seguenti sottofasi:

- **Analisi del flusso di controllo:** Verificherà una corretta esecuzione del codice. In particolare si controllerà che non vi siano statement irraggiungibili, ossia istruzioni la cui condizione di accesso non può mai essere vera.
- **Analisi dell'uso dei dati:** Verificherà un corretto utilizzo delle variabili. In particolare si controllerà che non vi siano variabili utilizzate prima di essere inizializzate, variabili inutilizzate, variabili sempre vere o sempre false.

- **Verifica formale del codice:** Verificherà la correttezza del codice scritto. In particolare si constaterà la correttezza totale di ogni unità, in modo che non conduca mai in uno stato di non terminazione.

### 2.3.2 Analisi Dinamica

Verrà applicata durante la progettazione e la stesura del codice, al fine di verificare dinamicamente l'indipendenza delle singole unità rispetto all'integrazione del sistema. Verrà testato quindi il sistema in tutti i suoi possibili casi e verranno effettuate prove per verificarne l'integrità. L'analisi avverrà:

- Attraverso l'inserimento di nuove regole business, in un contesto di prova, effettueremo test sulla validazione e li confronteremo con i risultati attesi;
- tramite opportune query di prova effettueremo test sull'utilizzo del DBMS;
- attraverso opportuni driver da noi progettati e sviluppati, testeremo il corretto funzionamento del DBMS e del validatore;
- utilizzando stub per quanto riguarda la Gui.

# Capitolo 3

# Gestione amministrativa della revisione

## 3.1 Processo di ispezione

Le ispezioni del codice e dei documenti verranno eseguite da una squadra di almeno 4 persone che, analizzeranno sistematicamente il codice e ne individueranno i possibili difetti. Parteciperanno alla riunione di ispezione:

- L'autore del documento ispezionato;
- i membri del gruppo tenuti a ispezionare il codice;
- il moderatore capo;
- il segretario che prenderà nota degli errori scovati.

In fase iniziale l'autore del codice presenterà alla squadra di ispezione il funzionamento dello stesso. Ognuno dei membri della squadra sarà poi tenuto a studiare il codice, al fine di individuarne difetti ed errori; i difetti individuati verranno poi annotati dal segretario durante la riunione di ispezione.

## 3.2 Comunicazione e risoluzione di anomalie

Il documento redatto dal segretario dovrà elencare le segnalazioni in modo che sia possibile individuare il punto errato in modo semplice e non ambiguo, eventualmente citando le parti non corrette. In dettaglio il documento sarà costituito da una tabella contenente:

- Un descrizione che identifichi in modo univoco l'errore (riga del codice, nome e revisione del file);
- i passi che hanno portato al verificarsi dell'errore;

- la gravità dell'errore ed il tempo in cui deve essere corretto.

L'autore del codice sottoposto a ispezione, preso atto dei difetti individuati nello stesso, sarà tenuto a correggerli nei tempi indicati presentando un documento in cui indica le modifiche apportate.

## 3.3 Trattamento delle discrepanze

Se dovessero verificarsi discrepanze tra le necessità del cliente ed i requisiti risultanti dall'analisi, si provvederà ad un ulteriore analisi che avrà la priorità sulle altre attività. Verrà aggiornato quindi il documento relativo all' "Analisi dei Requisiti" e i documenti/pezzi di codice da esso dipendenti. In ogni caso, se fosse necessario un cambiamento, si dovrà prima di tutto tracciarne l'impatto sugli altri requisiti e sul progetto del sistema, valutando l'effetto della modifica proposta. Il costo della modifica verrà stimato in base a quanti cambiamenti dovranno essere fatti al documento dei requisiti e, se opportuno, al progetto del sistema e alla sua implementazione. Completata tale analisi, si dovrà decidere se procedere o meno con la modifica. Sarà quindi compito dell'analista, in collaborazione col progettista e l'amministratore, valutare questo impatto sul lavoro già svolto e comunicare ai membri interessati le variazioni.

## 3.4 Procedure di controllo di qualità di processo

Il controllo della qualità del software interessa l'intero processo di sviluppo del prodotto in questione. A tale scopo sono stati adottati degli standard di documentazione che regolano la struttura e la presentazione dei documenti, nonchè degli standard di codifica e di processo. Quest'ultimi definiscono i processi da seguire durante tutto lo sviluppo software; includono definizione dei processi di specifica, progettazione e convalida, oltre ad una descrizione dei documenti che dovrebbero essere scritti durante l'esecuzione di questi processi. Il controllo di qualità prevede il monitoraggio del processo software, per garantire che le procedure e gli standard di qualità siano seguiti. Esso comprende inoltre il miglioramento del processo e la soluzione degli eventuali problemi. Quest'ultimo consiste nel comprendere i processi esistenti e modificarli per aumentare la qualità del software e/o diminuire i costi e i tempi di sviluppo. Il suo obiettivo principale è quello di concentrarsi sul perfezionamento, per migliorare la qualità del prodotto e, in particolare, per ridurre il numero di difetti nel software consegnato. Una volta ottenuto ciò, gli obiettivi primari diventeranno la riduzione dei costi e dei tempi. Per avere un esteso controllo degli errori, le modifiche apportate ai documenti, in seguito a inconsistenze riscontrate durante la fase di verifica e stesura, verranno notificate secondo alcune convenzioni interne illustrate nel documento "NormeDiProgetto". Queste modifiche si troveranno all'inizio di ogni

documento nella sezione “Diario delle modifiche”. I verificatori sono tenuti a controllare parallelamente la documentazione sul lavoro svolto.

# Capitolo 4

# Resoconto attività di verifica

## 4.1 Tracciamento componenti-requisiti

Per una completa e chiara tracciabilità componenti-requisiti, ogni componente, classe o metodo del prodotto "BR-jsys" è stato ideato come risposta ad ogni singolo requisito. Al fine di tenere traccia di tali corrispondenze è stata redatta una tabella, consultabile nel documento "Specifica Tecnica", che identifica per ogni componente quali sono i requisiti che la interessano. Ciò faciliterà la fase di verifica; si attesterà quindi se ogni componente soddisfi pienamente i requisiti ad esso associato.

## 4.2 Descrizione prove sui Requisiti

In questa sezione intendiamo elencare la descrizione dei vari test che successivamente effettueremo sui requisiti.

**Requisito F1**

| | |
|---|---|
| **Obiettivo Prova:** | Creare un linguaggio per le business rules |
| **Dipendenze:** | nessuna |
| **Descrizione Prova:** | Rappresentazione di una business rule attraverso il linguaggio creato |
| **Input:** | Business rule in linguaggio naturale |
| **Output:** | Business rule scritta nel linguaggio adottato |

**Requisito F2**

| | |
|---|---|
| **Obiettivo Prova:** | Testare la correttezza sintattica di ogni business rule |
| **Dipendenze:** | F1 |
| **Descrizione Prova:** | Inserimento di business rules d'appoggio (sintatticamente corrette o meno), per testarne il loro stato di accettazione |
| **Input:** | Business Rule di appoggio |
| **Output:** | La stringa accettata o meno dal validatore |

**Requisito F3**

| | |
|---|---|
| **Obiettivo Prova:** | Testare il corretto inserimento delle business rules nel repository |
| **Dipendenze:** | F2 |
| **Descrizione Prova** | Inserimento di varie business rules d'appoggio già validate e controllo all'interno del repository dell'avvenuto inserimento |
| **Input:** | Business Rule (validata) di appoggio |
| **Output:** | Lo stato dell'inserimento della business rule attraverso la notifica rilasciata (positivo: inserimento avvenuto con successo, negativo: eccezione) |

**Requisito F4**

| | |
|---|---|
| **Obiettivo Prova:** | Informare l'utente dell'avvenuta o meno validazione delle business rules appena inserite e del tempo trascorso, attraverso una notifica il più possibile chiara ed esaustiva |
| **Dipendenze:** | F2, F3, NU3 |
| **Descrizione Prova:** | Inserimento di varie business rules d'appoggio (corrette o meno) e controllo della corretta notifica di risposta alla richiesta di inserimento (attraverso timer nel codice) |
| **Input:** | Business Rules (corrette o meno) di appoggio |
| **Output:** | Una notifica per ogni business rule che si è provato ad inserire, con il relativo messaggio riguardante il tempo impiegato |

**Requisito F5**

| | |
|---|---|
| **Obiettivo Prova:** | Testare l'efficienza in termini di tempo di risposta del DBMS di appoggio al repository |
| **Dipendenze:** | F2, F3, F4 |
| **Descrizione Prova:** | Inserimento/eliminazione/modifica di varie business rules d'appoggio (sintatticamente corrette o meno) e valutazione del tempo di risposta impiegato dal DBMS. Tale valore viene visualizzato dalla Gui. |
| **Input:** | Business Rules (corrette o meno) di appoggio |
| **Output:** | Tempo impiegato |

**Requisito F6**

| | |
|---|---|
| **Obiettivo Prova:** | Il sistema deve essere in grado di interfacciarsi con l'interprete esterno, fornendo ad esso le business rules da eseguire |
| **Dipendenze:** | nessuna |
| **Descrizione Prova:** | Recuperare dal repository tutte le business rules associate ad un determinato business object |
| **Input:** | Business object (in stringa) |
| **Output:** | Le business rules associate |

**Requisito F7**

| | |
|---|---|
| **Obiettivo Prova:** | Testare la correttezza della cancellazione di una business rule dal repository e controllo dei risultati ottenuti |
| **Dipendenze:** | nessuna |
| **Descrizione Prova:** | Richiesta al DBMS di cancellare varie business rules date in input dal repository. Si testerà la correttezza dell'eliminazione fatta. Ogni eliminazione riuscita o meno dovrà essere accompagnata da una chiara notifica di evento |
| **Input:** | Business Rules da eliminare |
| **Output:** | Notifica di eliminazione (positivo: cancellazione avvenuta con successo, negativo: eccezione) |

**Requisito F8**

| | |
|---|---|
| **Obiettivo Prova:** | Creare interfaccia per l'inserimento/rimozione di Business Rules e verificare il corretto funzionamento |
| **Dipendenze:** | F3, F4, F7 |
| **Descrizione Prova:** | Si creerà una semplice interfaccia in linguaggio Java, con le componenti essenziali per permettere l' inserimento/rimozione di una business rule nel repository e la successiva notifica di tale evento |
| **Input:** | Business Rules da eliminare/inserire |
| **Output:** | Notifica di eliminazione o inserimento (avvenuta o meno) |

**Requisito F9**

| | |
|---|---|
| **Obiettivo Prova:** | REQUISITO DEPRECATO |
| **Dipendenze:** | |
| **Descrizione Prova:** | |
| **Input:** | |
| **Output:** | |

**Requisito F10**

| | |
|---|---|
| **Obiettivo Prova:** | Consentire al validatore di accedere ai campi del business object associato alla business rule da validare |
| **Dipendenze:** | F11 |
| **Descrizione Prova:** | Inserimento di varie business rules d'appoggio, che si riferiscono ai campi dati del business object associato, e testare la correttezza della validazione effettuata |
| **Input:** | Business rules d'appoggio |
| **Output:** | Esito della validazione |

**Requisito F11**

| | |
|---|---|
| **Obiettivo Prova:** | Testare che ad ogni business rule sia associato un business object |
| **Dipendenze:** | nessuna |
| **Descrizione Prova:** | Nella Gui ogni business rule ha un campo dati per il business object associato. Se tale campo ("associated Object") viene lasciato vuoto verrà lanciata un'eccezione dal sistema |
| **Input:** | Business rule di appoggio (con campo dati "associated Object" pieno o vuoto) |
| **Output:** | Una eccezione se il campo dati è stato lasciato vuoto, altrimenti niente |

**Requisito NU1**

| | |
|---|---|
| **Obiettivo Prova:** | Creare un linguaggio di alto livello, facile da capire per un utente con scarse conoscenze informatiche |
| **Dipendenze:** | F1 |
| **Descrizione Prova:** | Creazione di business rules complesse attraverso semplici inserimenti da parte dell'utente |
| **Input:** | Business rule di appoggio |
| **Output:** | Business rule scritta nel linguaggio adottato |

**Requisito NU2**

| | |
|---|---|
| **Obiettivo Prova:** | Il linguaggio deve fornire funzioni primitive per le operazioni base tra i dati, come la somma, la media aritmetica e la negazione logica |
| **Dipendenze:** | F1 |
| **Descrizione Prova:** | Creazione di business rules contenenti diverse operazioni base. Se non vengono accettate tali operazioni viene lanciata un'eccezione |
| **Input:** | Business rule di appoggio contenenti operazioni base |
| **Output:** | Business rule scritta nel linguaggio adottato |

**Requisito NU3**

| | |
|---|---|
| **Obiettivo Prova:** | Verificare la chiarezza e completezza dei messaggi d'errore |
| **Dipendenze:** | nessuna |
| **Descrizione Prova:** | Si tenterà di validare vari business objects (con errori interni e di ogni genere possibile), valutando l'efficacia del messaggio d'errore |
| **Input:** | Business object (in stringa) |
| **Output:** | Messaggio d'errore comprensibile |

**Requisito NPo1**

| | |
|---|---|
| **Obiettivo Prova:** | Il formato di salvataggio delle business rules e del repository dovrà essere XML. |
| **Dipendenze:** | nessuna |
| **Descrizione Prova:** | Creazione e salvataggio di business rules in formato XML |
| **Input:** | Business rule di appoggio in XML |
| **Output:** | L'inserimento della business rule in formato XML nel repository |

**Requisito NPr1**

| | |
|---|---|
| **Obiettivo Prova:** | Testare la velocità di interrogazione del DBMS |
| **Dipendenze:** | nessuna |
| **Descrizione Prova:** | Popolamento del repository attraverso inserimento di business rules d'appoggio e interazioni con esso. Si valuterà l'efficienza, in termini di tempo di risposta, del DBMS usato |
| **Input:** | Business rule |
| **Output:** | Valutazione del tempo impiegato in ogni interrogazione |

**Requisito NPr2**

| | |
|---|---|
| **Obiettivo Prova:** | Rendere univoca l'identificazione delle business rules all'interno del repository |
| **Dipendenze:** | nessuna |
| **Descrizione Prova:** | Inserimento di varie business rules d'appoggio nel repository e controllo dell'unicità della loro indicizzazione |
| **Input:** | Business object (in stringa) |
| **Output:** | L'esito del controllo |

**Requisito NQ1**

| | |
|---|---|
| **Obiettivo Prova:** | Testare tutte le operazioni consentite dal linguaggio creato e la loro validazione |
| **Dipendenze:** | F1, F3, F7 |
| **Descrizione Prova:** | Si applicheranno a business objects esistenti tutte le operazioni che il linguaggio permette e si valuteraà l'esito della validazione di tali operazioni, confrontandoli con i risultati aspettati. |
| **Input:** | Business Objects |
| **Output:** | L'esito della validazione delle operazioni tra i business objects dati in input |

**Requisito NQ2**

| | |
|---|---|
| **Obiettivo Prova:** | Testare che il manuale descriva il linguaggio di definizione delle business rules |
| **Dipendenze:** | F1, F8 |
| **Descrizione Prova:** | Creazione di un manuale utente di facile comprensione per qualsiasi tipo di utente |
| **Input:** | Richieste di informazioni |
| **Output:** | Manuale utente completo e allo stesso tempo semplice di ogni componente del linguaggio di definizione delle business rules e delle funzioni della Gui |

**Requisito NQ3**

| | |
|---|---|
| **Obiettivo Prova:** | Descrivere le API relative al validatore e al DBMS |
| **Dipendenze:** | nessuna |
| **Descrizione Prova:** | Creazione di una descrizione dettagliata delle API relative al validatore e al DBMS |
| **Input:** | Richieste di informazioni |
| **Output:** | Il documento completo e semplice con la descrizione delle API |