HAPPY CODE INC

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

ANALISI DEI REQUISITI

Versione 1.3 - 6 dicembre 2007

Capitolato: "BR-jsys"

Data creazione:	18/11/2007
Versione:	1.3
Stato del documento:	Formale, esterno
Redazione:	Michele Bortolato, Alessia Trivellato
Revisione:	Marco Tessarotto
Approvazione:	Elena Trivellato

Lista di distribuzione

Elena Trivellato	Responsabile di progetto
Filippo Carraro	Progettista
Alessia Trivellato, Michele Bortolato	Analisti
Marco Tessarotto	Verificatore
Tullio Vardanega, Renato Conte	Committente
Zucchetti S.r.l	Azienda proponente

Diario delle modifiche

Versione	Data rilascio	Descrizione
1.3	03/12/2007	Correzioni grammaticali, correzioni strutturali.
1.2	28/11/2007	Inserimento casi d'uso.
0.2	22/11/2007	Aggiornamento requisiti.
0.1	18/11/2007	Stesura preliminare del documento.

Sommario

Il presente documento contiene una descrizione del prodotto oggetto del capitolato d'appalto designato dalla sigla BR-jsys. Tale descrizione comprende le sue funzionalità, la tipologia di utenti ed il contesto d'uso. Prosegue con la descrizione e l'analisi dei requisiti del prodotto in questione con l'ausilio di un formalismo grafico: i diagrammi use-case UML. Vengono infine elencati tutti i requisiti individuati dalla HappyCode inc.

Indice

Capitolo 1

Introduzione

1.1 Scopo del documento

Il presente documento viene redatto al fine di identificare i requisiti impliciti ed espiciti presenti nel capitolato d'appalto denominato BR-jsys e pubblicato dal committente per conseguire il raggiungimento dell'obiettivo richiesto e soddisfare le esigenze di tutti gli stakeholders coinvolti nella realizzazione del progetto di cui sopra.

1.2 Scopo del prodotto

Il prodotto che verrà realizzato va inserito in un progetto globale. Il suo obiettivo principale è quello di integrare il sistema di validazione dei dati destinati al database dell'applicazione principale attraverso la creazione di business rules utili a confutarne la validità ed evitare quindi l'introduzione di errori nel processo di acquisizione. Si tratta in sostanza di introdurre un nuovo strato applicativo che aiuti a realizzare la consistenza dei valori introdotti.

1.3 Glossario

Viene fornito come file esterno (Glossario_0_4.pdf).

1.3.1 Definizioni, acronimi, abbreviazioni

Per la visione dei suddetti contenuti si rimanda alla consultazione del glossario allegato.

1.4 Riferimenti

Per un miglior tracciamento dei requisiti abbiamo ritenuto opportuno consultarci con l'azienda proponente. Il contenuto di tale contatto è reperibile nel verbale allegato (Incontro2007_11_22.pdf).

1.5 Descrizione degli allegati

Oltre al presente documento l'offerta è accompagnata dai seguenti allegati:

- Piano di Progetto: riporta l'impegno orario totale aziendale, l'impegno orario dei singoli componenti ed il costo preventivato.
- Piano di Qualifica: definisce la strategia di verifica e validazione.
- Glossario: definisce i termini non chiari e che assumono un significato particolare per il fornitore.

Capitolo 2

Descrizione generale

2.1 Contesto di utilizzo

Il prodotto richiesto dal cliente rientra nell'ambito di un progetto globale di automazione nella generazione del codice per la produzione di software gestionale. Il risultato di questo prodotto si traduce in piattaforme applicative complesse in ambiente J2EE che interagiscono con database relazionali.

2.2 Funzioni del prodotto

Con la realizzazione del prodotto richiesto si vuole fornire uno strumento software idoneo a validare il processo di inserimento e di elaborazione dei dati che intercorre nel progetto globale. Verranno portate a termine alcune fasi intermedie:

- Definizione di un linguaggio semplice ed espressivo per definire le regole di validazione (business rules) e calcolo per classi business object;
- realizzazione di un repository di regole in formato XML;
- realizzazione di un validatore che verifichi staticamente la coerenza di suddette regole;
- realizzazione di un interfaccia (API) per eseguire le regole di validazione e ottenere il risultato della valutazione, sia come accettazione dei valori inseriti, che come messaggi di errore da presentare all'utente;
- realizzazione di un interprete dinamico che valuti le regole lette dal repository nel contesto della classe che realizza il business object e che indichi se è possibile proseguire con la transazione verso il database.

2.3 Caratteristiche degli utenti

Il prodotto è rivolto ad una fascia di utenza che non necessità di particolari conoscenze informatiche ma esperta in ambito commerciale e gestionale.

2.4 Vincoli generali di utilizzo

La piattaforma che andremo a realizzare si appoggia su di un server di tipo J2EE 2.5. Il repository delle regole deve essere scritto secondo il linguaggio XML.

2.5 Assunzioni e dipendenze

- Le business rules sono divise in 3 tipi di classe fondamentali (rejector, projector, calculator);
- la virtual machine utilizzata per sviluppare gli oggetti richiesti è Java 1.5.

Capitolo 3

Tabella dei requisiti

3.1 Requisiti funzionali

3.1.1 Obbligatori

3.1.1.1 Il validatore statico deve poter verificare la coerenza delle regole: verificare cioè se esse costituiscono un insieme soddisfacilbile o se rigettano qualunque insieme di valori possibili per i business object. (R. ESPLICITO)

3.1.1.2 L'interprete dinamico associa classi java a regole. (R. ESPLICITO)

3.1.1.3 L'interprete valuta le regole lette dal repository nel contesto della classe che realizza il business object e indica se è possibile proseguire con la transazione verso il database. (R. ESPLICITO)

3.1.1.4 Deve essere possibile creare regole atte alla realizzazione di operazioni di calcolo a completamento di input parziali dell'utente. (R. ESPLICITO)

3.1.1.5 I business object devono poter gestire oggetti di tipo Master, Detail o Master/Detail. (R. ESPLICITO)

3.1.1.6 Deve essere possibile inserire e cancellare regole nel repository. (R. ESPLICITO)

3.1.1.7 Violazioni di regole da parte dei business objects verranno gestiti mediante appropriata gestione degli errori. (R. ESPLICITO)

3.1.1.8 Il contenuto del repository deve poter essere cambiato a run time senza dover procedere con nuova programmazione. (R. ESPLICITO)

3.1.2 Opzionali

3.1.2.1 Il prodotto dovrebbe essere dotato di un compilatore per tradurre le business rules in classi java, per non avere penalizzazioni in tempo di esecuzione. (R. ESPLICITO)

3.2 Requisiti di qualità

3.2.1 Obbligatori

3.2.1.1 Definizione di un repository di regole in XML. (R. ESPLICITO)

3.2.1.2 Realizzazione di un linguaggio di alto livello per definire le regole di validazione e calcolo per le classi java. (R. ESPLICITO)

3.2.1.3 Il sistema di regole utilizzerà java reflection per le classi di business objects di cui è costituito. (R. ESPLICITO)

3.2.1.4 Le business rules che implemeteremo saranno di tipo rejector. (R. ESPLICITO)

3.2.1.5 Il progetto verrà accompagnato da una completa documentazione che descriverà:

- il linguaggio di definizione delle regole;
- l'interazione con la classe java;
- il repository delle regole;
- l'API di attivazione delle regole;
- l'interprete e validatore delle regole;

(R. ESPLICITO)

3.2.1.6 Al prodotto finale verrà allegato un manuale tecnico e la documentazione di accompagnamento del codice sorgente.(R. ESPLICITO)

3.2.2 Desiderabili

Il sistema dovrebbe:

3.2.2.1 fornire elevata configurabilità ed estendibilità delle classi java; (R. ESPLICITO)

3.2.2.2 aiutare gli utenti meno esperti a non commettere errori grossolani attraverso la verifica formale delle regole. (R. ESPLICITO)

3.3 Requisiti di interfacciamento

3.3.1 Obbligatori

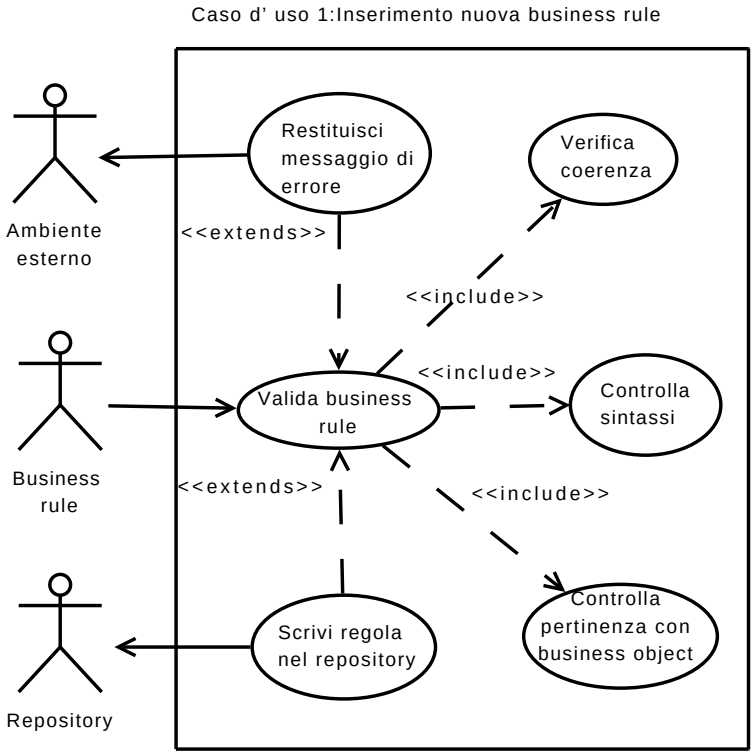
3.2.3.1 Il linguaggio scelto deve essere di alto livello; (R. ESPLICITO)

3.2.3.2 Il prodotto si deve integrare con le componenti già esistenti del progetto globale nell'ambito del quale verrà usato. (R. ESPLICITO)

Capitolo 4

Casi d'uso

4.1 Validazione nuova business rule.



- **Attori coinvolti:**
  - business rule.
  - repository.
  - ambiente esterno.

- **Scopo e decrizione sintetica:**
  Il validatore riceve in ingresso una nuova business rule, che dovrà essere validata per essere scritta nel repository.
- **Flusso di eventi:**
  - Il validatore carica la business rule.
  - Viene fatto un controllo sintattico per verificare che la business rule sia espressa secondo il linguaggio appropriato.
  - Si controlla che i campi ai quali vuole accedere la business rule combacino con quelli dei business objects.
  - Si effettua un controllo di coerenza con le altre regole presenti nel repository.
  - Si scrive la nuova business rule nel repository.
- **Precondizioni:**
  È stata immessa una nuova business rule.
- **Postcondizioni**
  La business rule viene scritta nel repository oppure viene segnalato l'errore all'ambiente esterno.

4.1.1 Scrivere business rule nel repository.

- **Attori coinvolti:**
  - business rule.
  - repository.
  - ambiente esterno.
- **Scopo e decrizione sintetica:**
  La business rule viene scritta nel repository in maniera da facilitarne un suo uso successivo.
- **Flusso di eventi:**
  - Tramite metodi opportuni la business rule verrà scomposta e posta nel repository per una sua successiva applicazione.
  - L'ambiente esterno viene informato dell'avvenuto inserimento.
- **Precondizioni:**
  La business rule ha passato il test sintattico, di coerenza con gli oggetti business coinvolti e di coerenza con le altre regole business precendentemente presenti nel repository.

- **Postcondizioni:**
  Il repository contiene una nuova business rule.

- **Specifiche supplementari:**
  Si dovrà porre la nuova business rule nel repository in maniera che non sussistano ambiguità con le altre già presenti. Si dovranno comunque rispettare i criteri programmazione XML.

4.1.2 Controllare sintassi.

- **Attori coinvolti:**
  - business rule.
  - ambiente esterno.

- **Scopo e decrizione sintetica:**
  Si controlla che la nuova business rule sia formulata secondo le specifiche del linguaggio scelto.

- **Flusso di eventi:**
  - Controllo sintassi per la nuova business rule.

- **Precondizioni:**
  È stata immessa una nuova business rule.

- **Postcondizioni:**
  La business rule può avanzare al controllo di coerenza con gli oggetti business oppure viene segnalata un'anomalia all'ambiente esterno.

4.1.3 Controllare pertinenza con business object.

- **Attori coinvolti:**
  - business rule.
  - ambiente esterno.

- **Scopo e decrizione sintetica:**
  Si controlla che la nuova business rule si riferisca a campi dato pertinenti agli oggetti business coinvolti.

- **Flusso di eventi:**
  - Contollo pertinenza con oggetti business coinvolti.

- **Precondizioni:**
  La nuova business rule ha superato positivamente la validazione sintattica.

- **Postcondizioni:**
  La nuova business rule può avanzare al controllo di coerenza con gli altri oggetti business presenti nel repository oppure viene segnalato un errore all'ambiente esterno.

- **Specifiche supplementari:**
  Saranno necessarie tecniche specifiche di accesso agli oggetti business coinvolti.

4.1.4 Controllare coerenza.

- **Attori coinvolti:**

  - business rule.
  - ambiente esterno.

- **Scopo e decrizione sintetica:**
  Si controlla la coerenza della nuova business rule con le altre regole business presenti nel repository.

- **Flusso di eventi:**

  - Controllo coerenza con le altre regole business presenti nel repository.

- **Precondizioni:**
  La nuova business rule ha superato positivamente il controllo di pertinenza con gli oggeti business coinvolti.

- **Postcondizioni:**
  La nuova business rule viene scritta nel repository oppure viene segnalata un'anomalia all'ambiente esterno.

- **Specifiche supplementari:**
  Le regole business presenti nel repository possono essere molte, ma non tutte si riferiscono agli stessi oggetti business, il controllo di coerenza dovrà essere intelligente, dovrà fare il controllo solo sulle regole che possono effettivamente andare in conflitto con la nuova,infine la business rule dovrà essere posta nel repository in modo da permettere un analisi il più efficiente possibile.

4.1.5 Restituire messaggi di errore.

- **Attori coinvolti:**

  - business rule.
  - ambiente esterno.

- **Scopo e decrizione sintetica:**
  In caso di fallimento nella validazione verrà avvisato in maniera opportuna l'ambiente esterno.

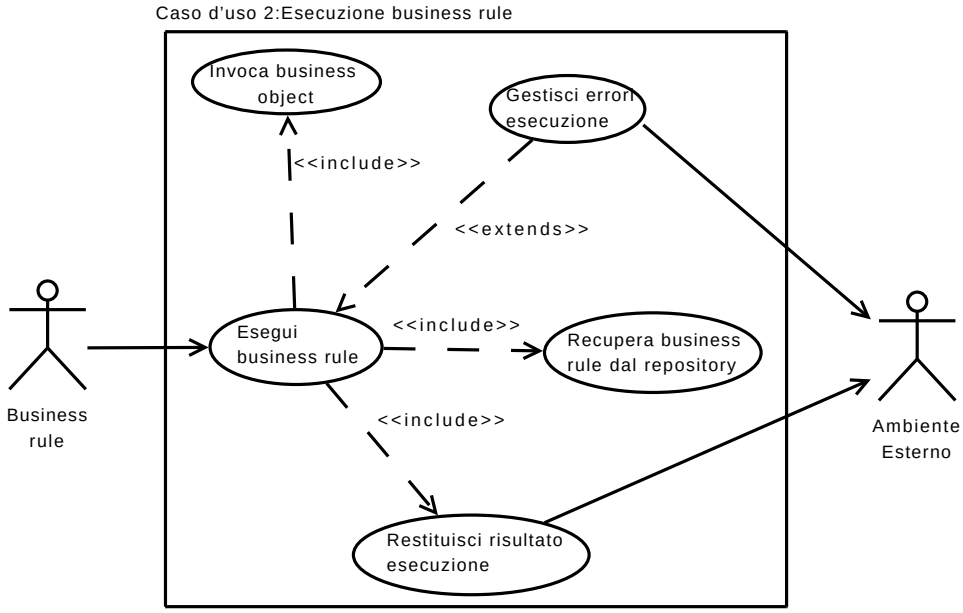
- **Flusso di eventi:**

  - Invia all'ambiente esterno un messagio che mostri in che punto della business rule si è verificato l' errore.

- **Precondizioni:**
  La nuova business rule non ha passato uno dei tre controlli di validazione.

- **Postcondizioni:**
  L'errore è stato segnalato correttamente.

4.2 Esecuzione business rule.



4.2.1 Eseguire business rule.

- **Attori coinvolti:**

  - business rule.
  - ambiente esterno.

- **Scopo e decrizione sintetica:**
  Le business rules presenti nel repository dovranno interagire con i busi-

ness objects necessari. Successivamente si dovrà segnalare all'ambiente esterno l'esito della validazione.

- **Flusso di eventi:**
  - La business rule viene recuperata dal repository.
  - Vengono invocati gli oggetti business necessari.
  - La business rule viene fatta eseguire.
  - Viene restituito il risultato del checking.
- **Precondizioni:**
  È stata invocata una business rule.
- **Postcondizioni:**
  Il checking è stato effettuato.
- **Specifiche supplementari:**
  Il checking può dare esito positivo o negativo; possono tuttavia verificarsi degli errori eccezzionali che dovranno essere segnalati opportunamente.

4.2.2 Recuperare business rule dal repository.

- **Attori coinvolti:**
  - business rule.
  - ambiente esterno.
- **Scopo e decrizione sintetica:**
  Data la richiesta d'uso di una business rule,la si dovrà cercare e recuperare dal repository XML.
- **Flusso di eventi:**
  - Rintraccia la business rule nel repository.
- **Precondizioni:**
  È avventuta la richiesta d'uso di una business rule.
- **Postcondizioni:**
  La business rule viene recuperata dal repository XML.
- **Specifiche supplementari:**
  Nel caso in cui la business rule non fosse presente, un opportuno avviso verrebbe segnalato all'ambiente esterno.

4.2.3 Invocare business objects necessari.

- **Attori coinvolti:**
  - business rule.
  - ambiente esterno.
- **Scopo e decrizione sintetica:**
  Data la business rule dovrò invcare gli opportuni oggetti business.
- **Flusso di eventi:**
  - Rintraccia gli oggetti business necessari.
  - Invoca gli oggetti business necessari.
- **Precondizioni:**
  È stata recuperata una business rule dal repository.
- **Postcondizioni:**
  I business objects opportuni sono stati invocati.
- **Specifiche supplementari:**
  Nel caso in cui un business(oppure parte di esso) non fosse presente, verrebbe segnalata l'anomalia all'ambiente esterno.

4.2.4 Restituire risultato del checking.

- **Attori coinvolti:**
  - business rule.
  - ambiente esterno.
- **Scopo e decrizione sintetica:**
  Verrà restituito all' ambiente esterno il risultato del check.
- **Flusso di eventi:**
  - Restituire all'ambiente esterno il risultato del check.
- **Precondizioni:**
  L'opportuna business rule è stata recuperata. Gli opportuni oggetti business sonon stati invocati.
- **Postcondizioni:**
  L'ambiente esterno ha ricevuto l'esito del checking.

4.2.5 Gestire errori esecuzione.

- **Attori coinvolti:**
  - business rule.
  - ambiente esterno.
- **Scopo e decrizione sintetica:**
  Segnala anomalie nelle funzionalitá precedentemente descritte.
- **Flusso di eventi:**
  - Segnala anomalie avvenute in una delle funzionalità sopra citate.
- **Precondizioni:**
  Immessa la richiesta di esecuzione di una business rule.
- **Postcondizioni:**
  L'ambiente esterno è stato informato dell'anomalia.
- **Specifiche supplementari:**
  L'anomalia segnalata dovrà segnalare precisamente le caratteristiche di quest'ultima.