HAPPY CODE INC

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

Specifica Tecnica

Versione 0.5 - 12 febbraio 2008

Capitolato: "BR-jsys"

Data creazione:	18/11/2007
Versione:	0.5
Stato del documento:	formale, esterno
Redazione:	Michele Bortolato
Revisione:	Alessia Trivellato, Elena Trivellato
Approvazione:	Mattia Meroi

Lista di distribuzione

HappyCode inc	Gruppo di lavoro
Tullio Vardanega, Renato Conte	Rappresentanti del committente
Zucchetti S.r.l	Azienda committente

Diario delle modifiche

Versione	Data rilascio	Descrizione
0.5	05/02/2008	Aggiunta del nome del file nel modello di documento.
0.4	22/01/2008	Modifica al layout dei documenti.
0.3	22/12/2007	Aggiornamento requisiti.
0.2	21/12/2007	Documento sottoposto a revisionamento automatico.
0.1	18/12/2007	Stesura preliminare del documento.

Indice

Capitolo 1

Introduzione

1.1 Scopo del documento

Il presente documento si ripropone di descrivere il sistema software "BR-jsys", sulla base del documento di Analisi dei Requisiti, dal punto di vista architetturale. Lo strumento da noi adottato sarà il linguaggio UML, sottoforma di diagrammi delle classi. L'approccio adottato nell'illustrare il sistema sarà di tipo "Bottom-up" Fornirà inoltre una visione più dettagliata delle componenti da realizzare. Verrà definito il contesto d'uso del sistema e fornita la decomposizione di questo in componenti principali.

1.2 Scopo del prodotto

Per lo scopo del prodotto si faccia riferimento al documento di Analisi dei Requisiti.

1.3 Glossario

Viene fornito come documento esterno chiamato Glossario.1.4.pdf .

1.4 Riferimenti

1.4.1 Riferimenti normativi

- Capitolato d'appalto BR-jsys
- AnalisiDeiRequisiti
- NormeDiProgetto
- PianoDiProgetto

- PianoDiQualifica
- “Ingegneria del software” 8a edizione - Ian Sommerville
- “The Definitive ANTLR Reference”

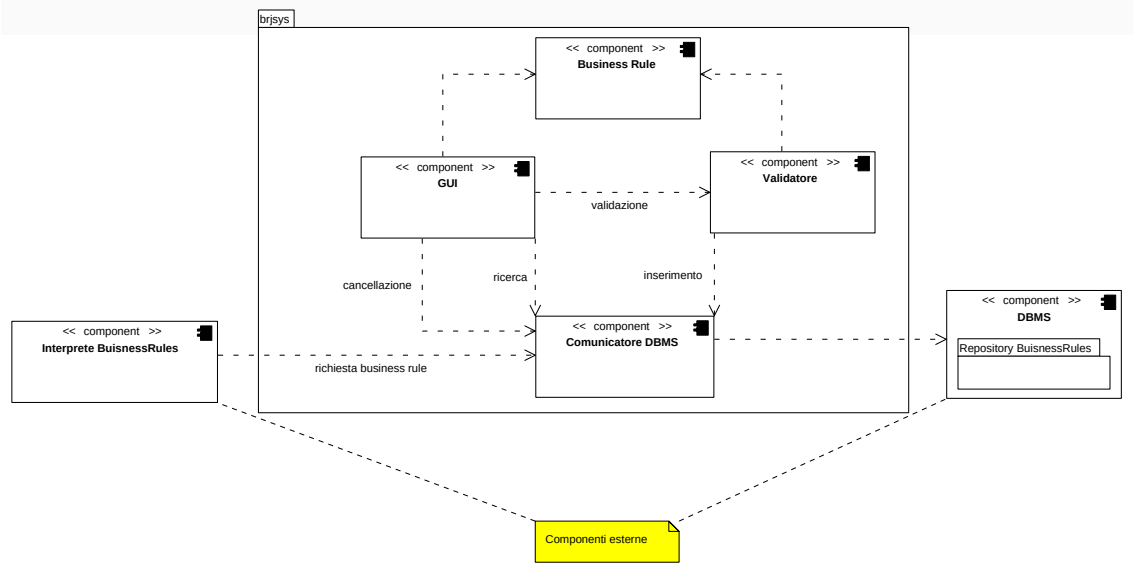
Capitolo 2

Definizione del prodotto

2.0.2 Metodo e formalismo di specifica

Abbiamo attuato la progettazione del prodotto con l'uso di diagrammi UML 2.1 in quanto linguaggio internazionalmente riconosciuto e standardizzato. Il software di cui lazienda ha fatto uso per la creazione dei diagrammi e' ArgoUML 0.24. L'utilizzo di tale software deriva dalla completezza di strumenti di cui e' fornito, i quali soddisfano in pieno le nostre necessita'.

2.0.3 Presentazione dell'architettura generale del sistema e identificazione dei componenti architetturali di alto livello



Abbiamo deciso di suddividere il prodotto in quattro macrocomponenti:

1. **GUI**:Fornisce all'utente un interfaccia grafica minimale consentendogli di effettuare le operazioni di cancellazione e interrogazione del repository in maniera user-friendly.

2. **Business Rule**: Componente che rappresenta la business rule che l'utente dichiara e che deve essere passata al validatore per compilarla.

3. **Validatore**: Componente in grado di accettare in input una business rule , di validarla, e di inserirla, se scritta correttamente, nel repository .

4. **Comunicatore**: Componente in grado di connettersi al DBMS esterno e di comunicare con esso tramite i formalismi del linguaggio XQuery. Sarà appunto tramite Query che il DBMS effettuerà operazioni di insermento, cancellazione o di ricerca nel repository qualora un componente lo richiedesse.

Questa suddivisione, per quanto minimale, consente di individuare le componenti che verranno ampliamente descritte nel successivo capitolo.

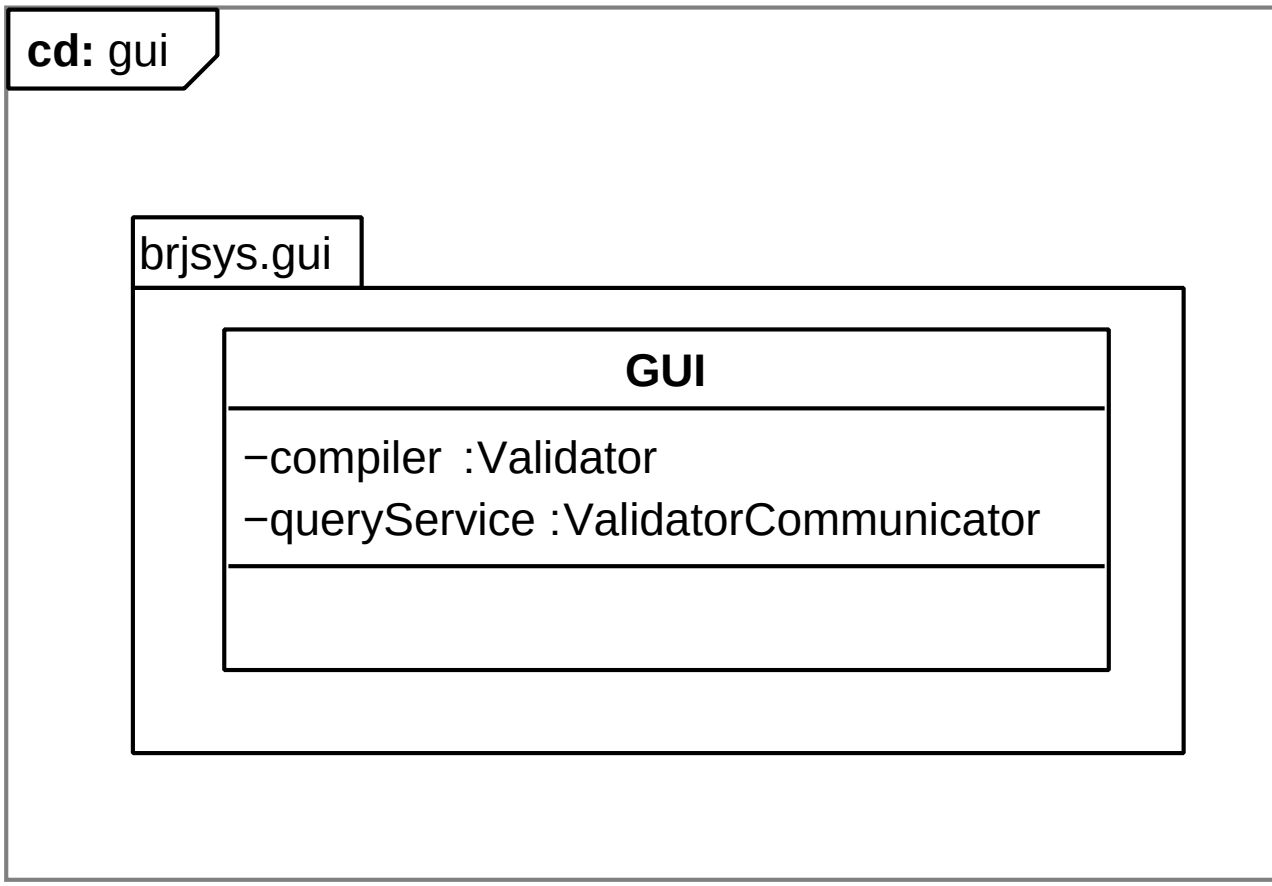
Capitolo 3

Descrizione dei singoli componenti

Qui di seguito analizzeremo separatamente le quattro macrocomponenti elencate il capitolo precedente e ne daremo una descrizione più approfondita.

3.1 GUI

3.1.1 Diagramma delle classi



3.1.2 Gui

Tipo, obiettivo e funzione del componente

Questa componente fornisce all'utente un interfaccia minimale consentendogli di effettuare le operazioni di cancellazione, e di querying sul repository . Nel caso di esecuizone di una query definita dall'utente verranno fornite inoltre informazioni relative ai tempi d'esecuzione. Questa componente viene realizzata tramite una singola classe java.

Relazioni d'uso di altre componenti

Questa componente utilizza:

- BusinessRule per dichiarare una nuova business rule da spedire al validatore;
- Validator per effettuare la validazione di una business rule ;
- GUICommunicator che verrà trattato successivamente.

Interfacce con e relazioni di uso da altre componenti

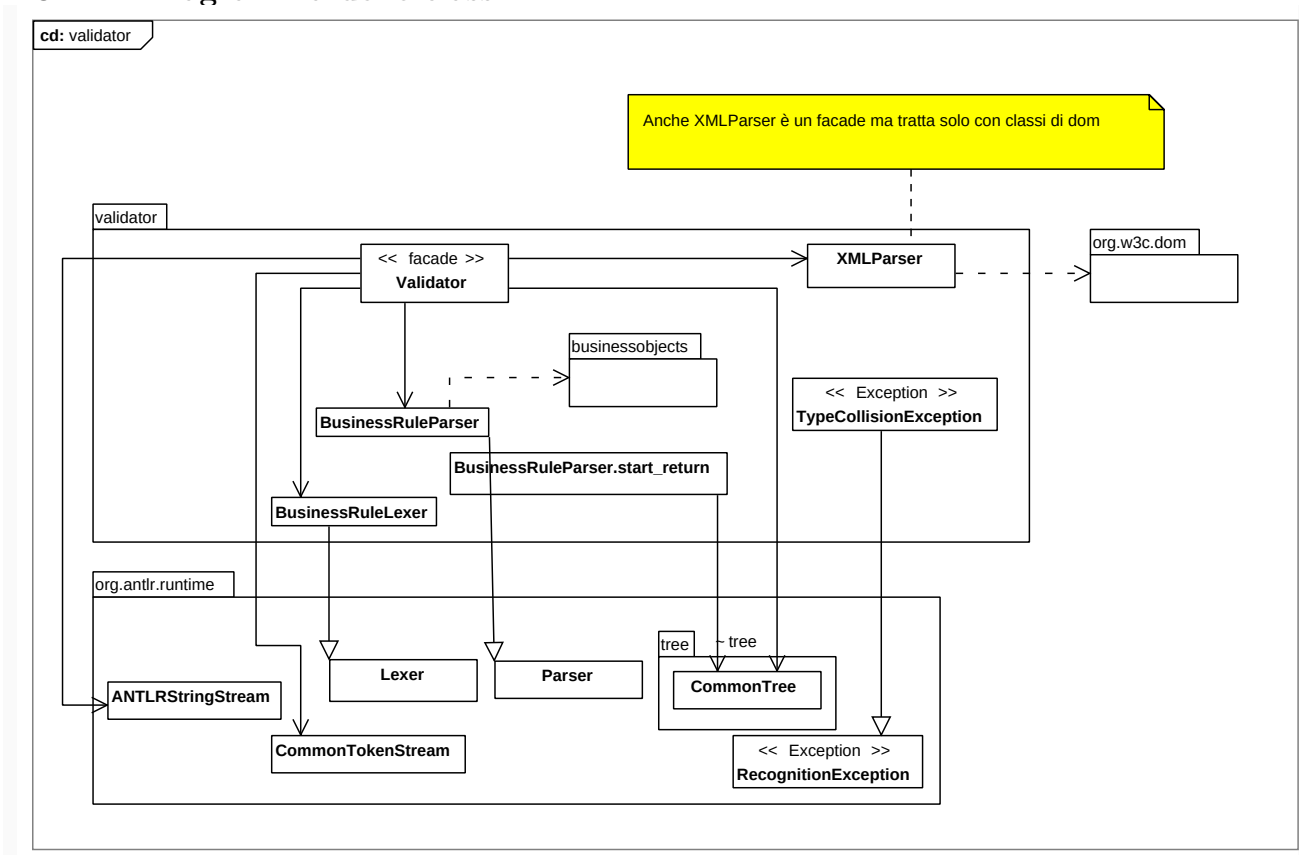
Nessuna.

Attività svolte e dati trattati

Per la GUI possiede i campi dato GUICommunicator e Validator che le forniscono le funzionalità che si devono visualizzare, entrambe verranno trattate in seguito.

3.2 Validatore

3.2.1 Diagramma delle classi



3.2.2 Validator

Tipo, obiettivo e funzione del componente

Questa componente effettua la validazione di una business rule senza far fare all'utilizzatore tutte le singole operazioni di compilazione.

Relazioni d'uso di altre componenti

Validator usa le componenti BusinessRuleParser , BusinessRuleLexer , XMLParser e ValidatorCommunicator che verranno descritte in seguito.

Interfacce con e relazioni di uso da altre componenti

Validator è in relazione con la componente GUI per permettere la validazione.

Attività svolte e dati trattati

La componente contiene soltanto il metodo validate() che effettua la validazione della business rule .

3.2.3 BusinessRuleLexer

Tipo, obiettivo e funzione del componente

Questa componente, derivata da org.antlr.runtime.Lexer, non è altro che una classe wrapper per la stringa che rappresenta la business rule che aggiunge ad essa funzionalità necessarie per il successivo parsing.

Relazioni d'uso di altre componenti

Nessuna.

Interfacce con e relazioni di uso da altre componenti

La componente BusinessRuleParser necessita la componente BusinessRuleLexer che verrà approfondita successivamente.

Attività svolte e dati trattati

BusinessRuleLexer mette a disposizione vari metodi per la lettura del testo della business rule , nonchè per la gestione di eventuali eccezzioni avvenute in fase di validazione.
Nota:*Questa classe è stata creata utilizzando uno strumento automatico per la generazione di parser data in input la specifica di una grammatica.*

3.2.4 BusinessRuleParser

Tipo, obiettivo e funzione del componente

Questo componente, derivata da org.antlr.runtime.Parser, effettua il parsing della stringa che rappresenta la business rule . Effettua il controllo sintatico della regola, il controllo semantico facendo un controllo sui tipi dei dati siano essi costanti oppure campi dati di business objects . Mentre effettua la validazione BusinessRuleParser genera l'albero di pasing secondo le specifiche presenti nel controllo semantico. Infine è in grado di dare informazioni accurate riguardo eventuali errori in fase di validazione.

Relazioni d'uso di altre componenti

Questa componente necessita di un TokenStream fornitogli indirettamente da BusinessRuleLexer. Per effettuare il controllo sui tipi per il business object associato deve poi riferirsi alla componente BusinessObjects ed infine ha bisogno della componente TypeCollisionException per trattare gli errori in fase di validazione.

Interfacce con e relazioni di uso da altre componenti

BusinessRuleParser viene messa in relazione con XMLParser che verrà trattata successivamente. È utilizzata inoltre dalla componente Validator.

Attività svolte e dati trattati

BusinessRuleParser mette a disposizione vari metodi per effettuare il parsing e per i test semantici, nonchè dispone di numerosi campi dato per la ricognizione dei token.
Nota:*Questa classe è stata creata utilizzando uno strumento automatico per la generazione di parser data in input la specifica di una grammatica.*

3.2.5 TypeCollisionException

Tipo, obiettivo e funzione del componente

Questa componente, derivata dalla classe org.antlr.runtime.RecognitionException, permette di ricavare informazioni sugli eventuali errori avvenuti in fase di parsing della business rule , essa in definitiva aggiunge alla sua superclasse la possibilità di riportare informazioni su errori di tipo.

Relazioni d'uso di altre componenti

Nessuna.

Interfacce con e relazioni di uso da altre componenti

La componente TypeCollisionException è utilizzata da BusinessRuleParser per sollevare eccezioni da errori sul controlo dei tipi.

Attività svolte e dati trattati

Viene ridefinito il metodo printStackTrace() e messo a disposizione un costruttore per avere informazioni sui tipi che si sono rivelati incompatibili.

3.2.6 XMLParser

Tipo, obiettivo e funzione del componente

La componente XMLParser si occupa di effettuare la conversione dell'albero sintattico prodotto dalla componente BusinessRuleParser in un elemento XML rappresentante la business rule , l'elemento XML risultante conterrà:

- il nome della business rule , che deve essere univoco;
- il nome del business object associato a quela regola;
- la struttura del albero sintattico della business rule scritta secndo la metodologia elemento-attributo tipica di XML;
- la struttura del albero sintattico della business rule scritta lineramente secondo la notazione prefissa rappresentata da un singolo attributo XML;
- la business rule effettivamente digitata dall'utente;
- eventuali commenti associati dall'utente alla business rule ;

Relazioni d'uso di altre componenti

XMLParser ha bisogno dell' albero di parsing prodotto da BusinessRuleParser , nonchè necessita della componente business rule per ricavare le informazioni da inserire nel'elemento XML.

Interfacce con e relazioni di uso da altre componenti

La componente ValidatorCommunicator ha bisogno dell'elemento XML prodotto da XMLParser per avviare la procedura di inserimento nel repository .

Attività svolte e dati trattati

XMLParser contiene le operazioni per scorrere l'albero di parsing ed effettuare il traduzione in XML.Inoltre dispone di una tabella Hash statica che serve per associare i valori numerici che il parser

3.2.7 businessobjects

Tipo, obiettivo e funzione del componente

La componente businessobjects si occupa di offrire un namespace comune a tutti i business objects che durante la validazione di una business rule vengono interpellati richiedendo accesso ai suoi campi o sottocampi.

Relazioni d'uso di altre componenti

Nessuna.

Interfacce con e relazioni di uso da altre componenti

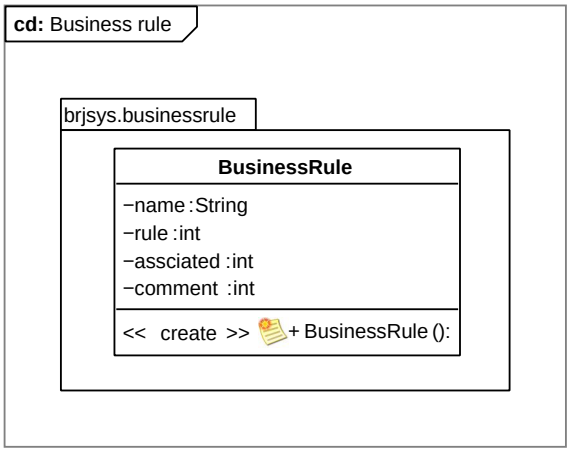
BusinessRuleParser necessita di questa componente per effettuare il controllo dei tipi qualora fosse necessario.

Attività svolte e dati trattati

Il componente businessbjects offre le classi java che rappresentano i business objects .

3.3 Business Rule

3.3.1 Diagramma delle classi



3.3.2 BusinessRule

Tipo, obiettivo e funzione del componente

La componenete BusinessRule rappresenta la business rule che l'utente inserisce e vuole validare.

Relazioni d'uso di altre componenti

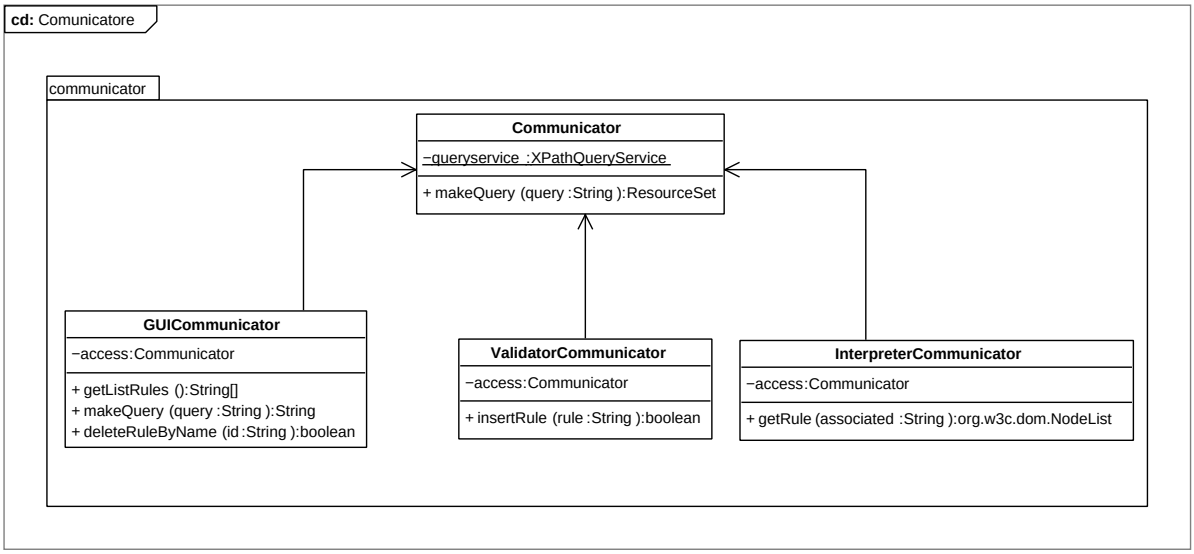
Nessuna.

Interfacce con e relazioni di uso da altre componenti

La componente GUI utilizza BusinessRule nel caso in cui l'utente voglia inserire una nuova business rule . La componente Validator effettua la validazione di un istanza di BusinessRule tramita la chamata del suo metodo validate().

Attività svolte e dati trattati

La componente contiene i campi dato Stringa per rappesentare una business rule ossia name, associatedObject, rule e comment, quest'ultima pu non essere istanziata, vengono inoltre messi a disposizione il costruttore e il metodo ridefinito toString().



3.4.2 Communicator

Tipo, obiettivo e funzione del componente

Communicator fornisce alle componenti che la utilizzano la possibilità di effettuare query di qualsiasi tipo al repository presente nel DBMS.

Relazioni d'uso di altre componenti

Communicator necessita di interagire con la componente esterna DBMS per interrogare il repository .

Interfacce con e relazioni di uso da altre componenti

Le componenti GUICommunicator, InterpreterCommunicator e Validator-Communicator necessitano di Communicator per potergli passare Query specifiche, le loro descrizioni sono trattate successivamente.

Attività svolte e dati trattati

Quando viene istanziata per la prima volta inizializza la connessione al DBMS tramite la definizione di una variabile statica che successivamente consente di interrogare il repository direttamente passandogli la stringa che rappresenta la query composta secondo le specifiche XQuery.

3.4.3 GUICommunicator

Tipo, obiettivo e funzione del componente

La componente contiene metodi necessari per modellare query di cancellazione(tramite operazioni XQuery Update) e di query di semplice interrogazione.

Relazioni d'uso di altre componenti

GUICommunicator utilizza la componente Communicator per accedere al repository ed interrogarlo.

Interfacce con e relazioni di uso da altre componenti

GUICommunicator viene messa in relazione con la componente GUI dalla quale riceve richieste di cancellazione e di querying.

Attività svolte e dati trattati

GUICommunicator fornisce le operazioni:

- deleteRuleByName():che dato il nome di una regola, provvede ad eliminarla dal repository nel caso questa regola fosse presente;
- makeQuery():che permette l'esecuzione di una semplice query purche non implichi modifiche strutturali al repository ;
- getListRules():che ritorna, per ogni business rule nome, business object associato e sua struttura.

3.4.4 InterpreterCommunicator

Tipo, obiettivo e funzione del componente

InterpreterCommunicator contiene metodi necessari per rispondere a richieste di business rule da parte di un interprete esterno.

Relazioni d'uso di altre componenti

InterpreterCommunicator utilizza la componente Communicator per accedere al repository ed interrogarlo.

Interfacce con e relazioni di uso da altre componenti

InterpreterCommunicator viene messa in relazione con l'interprete esterno dal quale riceve richieste di business rules associate ad un determinato business object .

Attività svolte e dati trattati

InterpreterCommunicator offre il metodo getRule() per richiedere le business rule associate al business object .

3.4.5 ValidatorCommunicator

Tipo, obiettivo e funzione del componente

ValidatorCommunicator contiene metodi necessari per effettuare l'inserimento di una business rule validata alla quale è stato effettuato il parsin in XML.

Relazioni d'uso di altre componenti

ValidatorCommunicator utilizza la componente Communicator per accedere al repository ed interrogarlo.

Interfacce con e relazioni di uso da altre componenti

ValidatorCommunicator viene messa in relazione con la componente Validator per effettuare l'inserimento.

Attività svolte e dati trattati

ValidatorCommunicator offre il metodi insertRule() per inserire la business rule nel repository . L'inserimento avverrà soltanto se la business rule ha un nome che non è ancora presente nel repository .

Capitolo 4

Componenti esterne

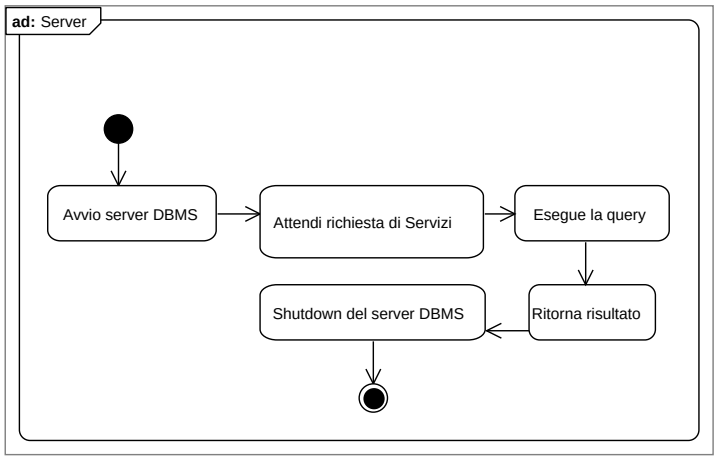
4.1 Interprete

4.2 DBMS

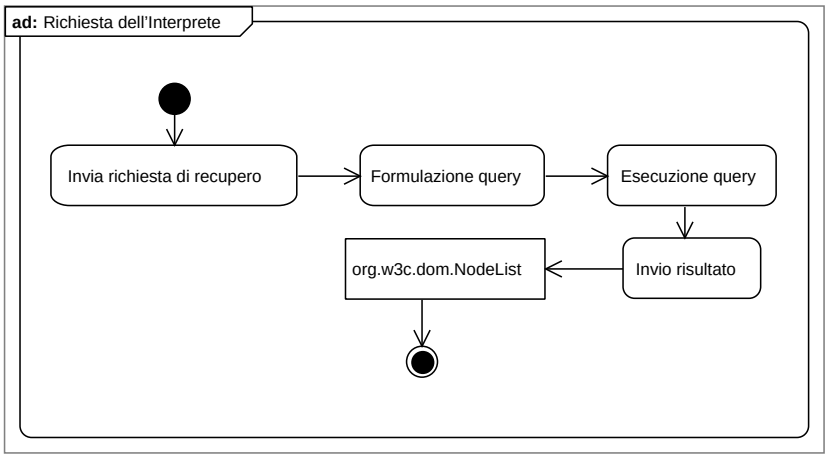
Capitolo 5

Diagrammi di attività

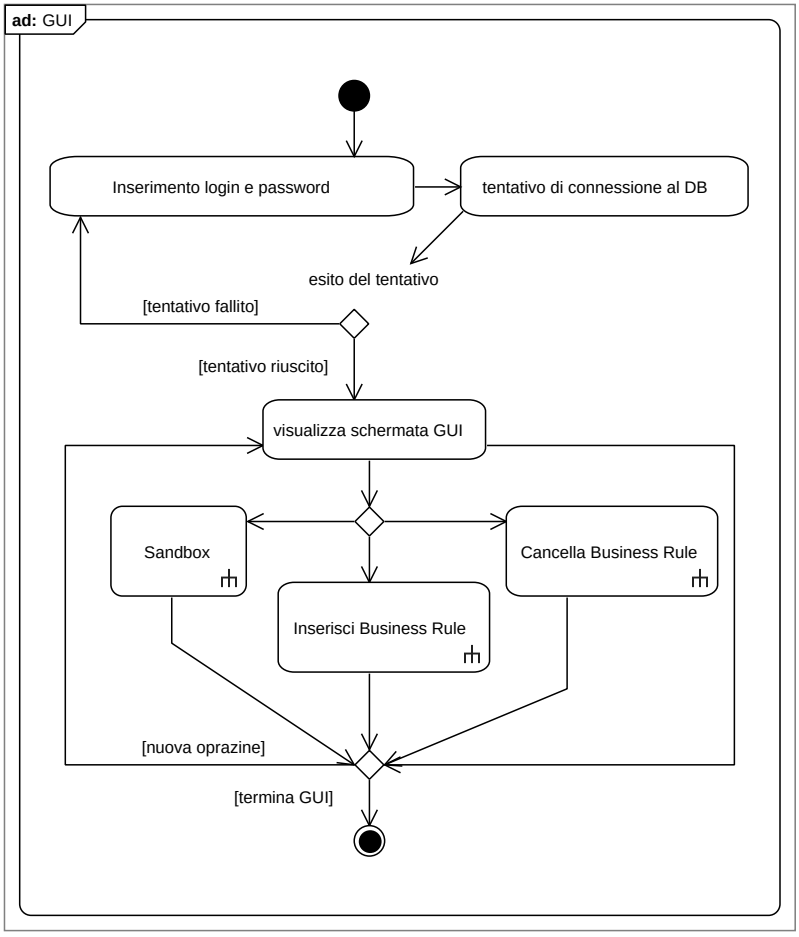
5.1 Server



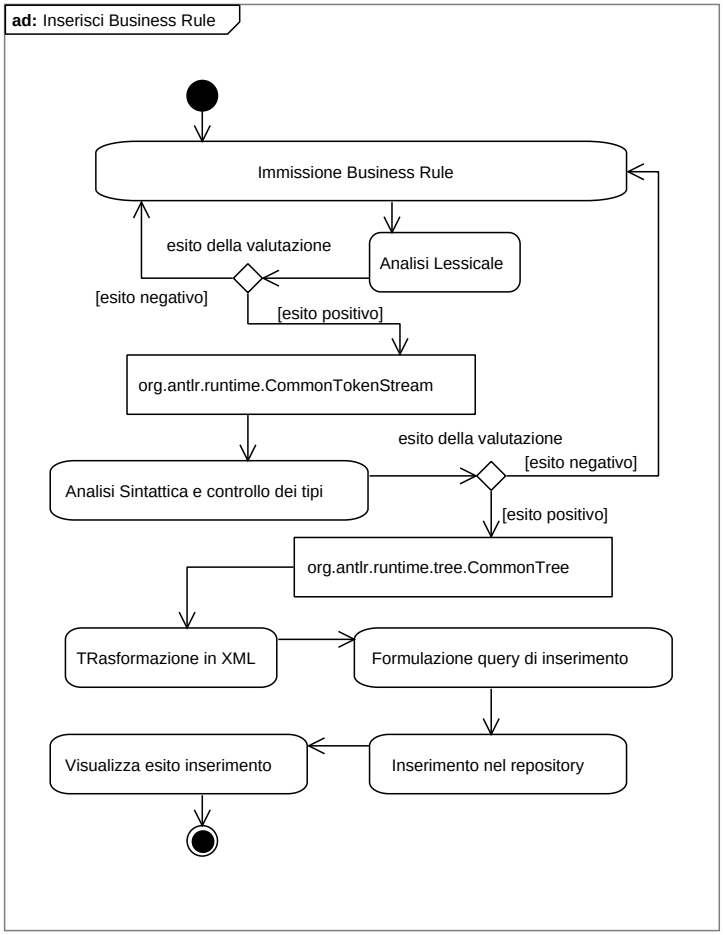
5.2 Richiesta dell'interprete



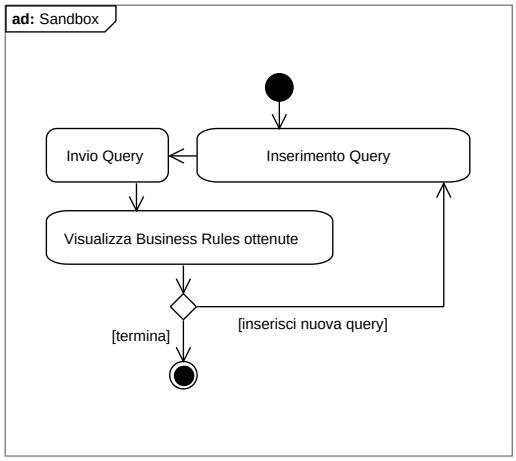
5.3 GUI


ad: GUI
Inserimento login e password
tentativo di connessione al DB
esito del tentativo
[tentativo fallito]
[tentativo riuscito]
visualizza schermata GUI
Sandbox
Cancella Business Rule
Inserisci Business Rule
[nuova oprazine]
[termina GUI]

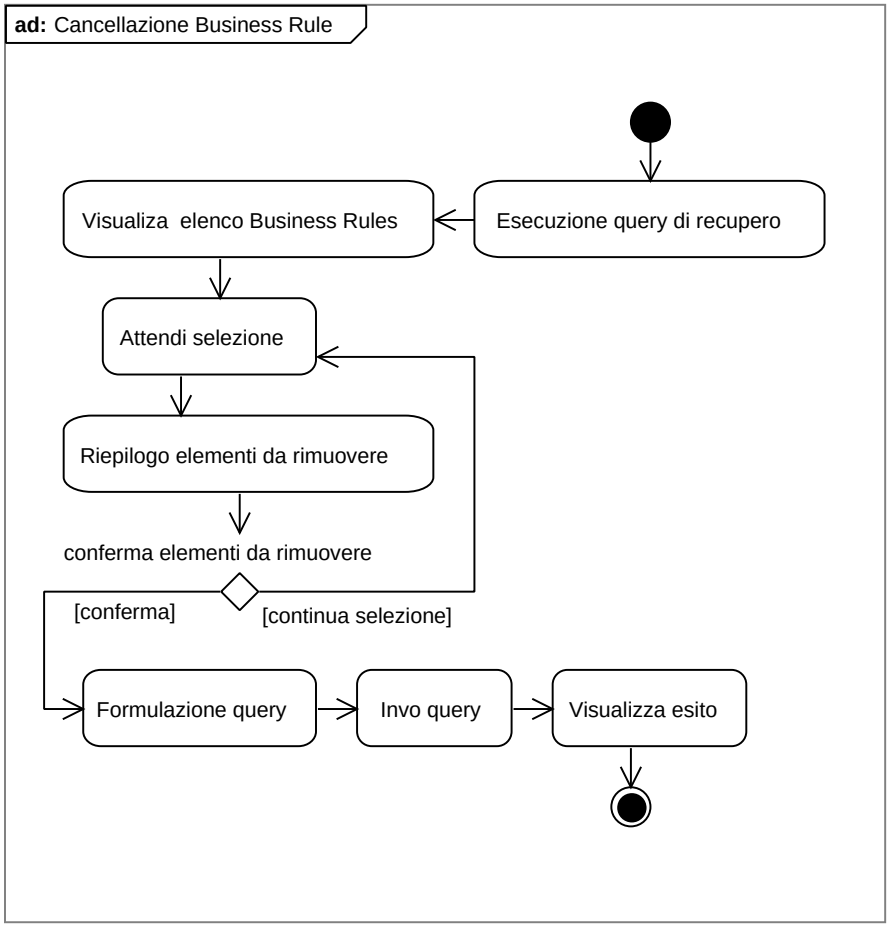
5.4 Inserisci business rule



5.5 Sandbox


ad: Sandbox
Invio Query
Inserimento Query
Visualizza Business Rules ottenute
[termina]
[inserisci nuova query]

5.6 Cancellazione business rule



Capitolo 6

Stime di fattibilità e di bisogno di risorse

Dopo aver analizzato il problema attraverso schemi progetturali sono state individuate le risorse necessarie per la realizzazione del prodotto. Attraverso l'utilizzo di software open source siamo riusciti a contenere i costi e contemporaneamente a rendere disponibili tutte le risorse necessarie. Le risorse necessarie ai nostri componenti per affrontare le varie problematiche di comunicazione, sviluppo del codice, gestione degli archivi, verifica dei documenti e del sistema sono state descritte nel documento Piano di Qualifica. Nella fase di specifica tecnica e successivamente di progettazione verrano utilizzati diagrammi UML delle classi e degli oggetti realizzati con Dia. Lo sviluppo avrà luogo singolarmente o in piccoli gruppi a seconda della natura della problematica da affrontare. Il documento/codice avrà in ogni modo un unico proprietario incaricato di renderlo pubblico tramite server SVN.

Capitolo 7

Tracciamento della relazione componenti-requisiti