UNIVERSITA' DEGLI STUDI DI PADOVA

FACOLTA' DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

STUDIO DI UNA METODOLOGIA DI SVILUPPO DEL SOFTWARE ORIENTATA AI SERVIZI E BASATA SUL MODELLO

Laureando:

ANDREA CHIODI

Relatore:

Ch.mo Prof. SERGIO CONGIU

ANNO ACCADEMICO 2011/2012

Sommario

Indice delle figure

1 Introduzione

Questo elaborato ha lo scopo di studiare la metodologia di sviluppo del software ideata e sviluppata all'interno dell'azienda SEC Servizi, presentandone lo schema e le scelte architetturali su cui si basa (capitolo 2) ed in seguito illustrando le funzionalità e gli strumenti che formano il framework a supporto (capitolo 3). Al termine vengono tratte alcune conclusioni (capitolo 4).

1.1 Contesto di business

SEC Servizi è un consorzio con sede a Padova in Via Transalgardo, costituito in forma di Società Consortile per Azioni, fra istituti bancari sorto nel 1972 con la finalità di fornire servizi di operatività bancaria e tecnologica ai propri soci e clienti. Sec Servizi si è trasformata ed evoluta nel tempo grazie ad una forte cultura dell'innovazione e ad una profonda attenzione verso l'eccellenza tecnologica, che le hanno permesso di divenire una tra le principali realtà italiane nell'ambito dell'offerta di servizi informativi in outsourcing.

SEC Servizi assume il ruolo di fornitore di servizi ICT, amministrativi, ausiliari ed a valore aggiunto con responsabilità di unica interfaccia (diretta/coordinamento) verso i clienti.

Alla fine degli anni '90, il sistema bancario ha dovuto fronteggiare nuove sfide legate ai mutamenti imposti dall'incalzare delle nuove tecnologie. L'avvento di Internet e di nuovi dispositivi mobili ha portato ad un mutamento delle caratteristiche e del comportamento della clientela, e all'incremento dei canali distributivi di erogazione del servizio.

A seguito di questo impulso, nel 2001 SEC ha iniziato lo studio e l'implementazione di una nuova soluzione architetturale, elaborando la metodologia di sviluppo del software che è l'oggetto del presente studio.

1.2 Contesto architetturale

Nel contesto di evoluzione tecnologica a cui abbiamo accennato, SEC Servizi ha deciso di dotarsi di un Ufficio Architetture, il cui team è attualmente composto da circa 30 persone tra dipendenti e collaboratori esterni.

L'Architettura in SEC ha lo scopo di abilitare le richieste del business garantendo la standardizzazione della progettazione delle applicazioni del Sistema Informativo Bancario mediante processi, strumenti e servizi.

Questo obiettivo richiede la creazione di una piattaforma trasversale e condivisa, ma che nel contempo permetta un livello di personalizzazione, che uno degli asset più importanti di SEC nei confronti dei clienti. Tale piattaforma deve avere la robustezza necessaria a garantire la continuità del servizio, e nel contempo la versatilità nell'erogazione su diversi canali tecnologici.

Lo scenario architetturale iniziale consisteva in una quantità di soluzioni tra loro eterogenee. A livello dei dati esisteva una forte ambiguità, causata da diverse rappresentazioni di oggetti uguali, come ad esempio i codici identificativi degli istituti bancari, oppure il formato delle valute. La comunicazione tra gli attori del processo di sviluppo non era uniforme, mancava la capacità di astrazione e un linguaggio unificante. Infine, lo sviluppo delle prime soluzioni orientate ai servizi, essendo prive di una logica condivisa, aveva provocato la proliferazione di servizi simili o addirittura uguali.

2 Metodologia di sviluppo

Le esigenze che sono state evidenziate nel capitolo precedente hanno portato all'elaborazione di una metodologia di sviluppo del software basata sul modello e orientata ai servizi.

La metodologia si basa su diversi paradigmi architetturali, che hanno portato alla progettazione di un processo di sviluppo ben definito in ogni sua fase. Scopo di questo capitolo è di presentare e motivare tali paradigmi e le soluzioni adottate nei vari moduli del processo.

E' importante definire chi sono gli attori coinvolti nel processo:

- Analisti funzionali: nel contesto considerato, gli analisti funzionali si occupano dello studio del modello di business che l'applicazione deve supportare
- Referenti architetturali: sono le persone che supportano lo sviluppo del software definendo gli standard e offrendo soluzioni architetturali al contesto applicativo
- Sviluppatori: sono le persone con skill specifiche che utilizzano gli strumenti a disposizione per implementare le soluzioni applicative

Ciascuna figura, come vedremo, svolgerà il proprio compito in maniera armonica con il disegno generale: uno degli scopi principali della metodologia è di garantire una corretta comunicazione tra le parti in gioco.

2.1 Architettura basata sul modello e orientata ai servizi

I fondamentali pattern architetturali su cui si basa l'intera metodologia studiata, e che caratterizzano il titolo del presente studio, sono l'MDA e il SOA.

Il primo approccio architetturale adottato è l'MDA (acronimo di Model Driven Architecture). MDA è un paradigma di sviluppo del software che mette a disposizione un insieme di direttive per strutturare le specifiche dei dati, che sono espressi tramite modelli. L'approccio MDA definisce le funzionalità del sistema utilizzando un modello indipendente dalla piattaforma e un appropriato linguaggio DSL (Domain Specific Language, cioè un linguaggio specifico di dominio).

Questo approccio si sposa perfettamente con il contesto definito in precedenza, visto che siamo in presenza di applicazioni che insistono su uno stesso dominio (cioè il mondo bancario) e che nel contempo devono erogare servizio su canali diversi (quindi il modello che descrive l'applicazione deve essere indipendente dalla piattaforma specifica).

Solo in un secondo momento, grazie all'ausilio di mappature specifiche e trasformazioni automatiche, il modello PIM (Platform Independent Model) si tradurrà in un modello PSM (Platform Specific Model) per lo sviluppo della specifica componente software.

L'uso estensivo del concetto di modellazione, che viene adottato nella quasi totalità delle fasi di sviluppo, ha generato un bagaglio di competenze specifiche che gli sviluppatori devono acquisire. In determinate situazioni, alcuni sviluppatori si occupano esclusivamente della modellazione, andando ad aggiungere la figura del "modellatore" alla lista degli attori del processo di sviluppo che è stata presentata nel precedente paragrafo.

Il secondo approccio architetturale applicato è il SOA (acronimo di Service Oriented Architecture). SOA è un architettura software adatta a supportare l'uso dei servizi, per garantire l'interoperabilità tra diversi sistemi così da consentire l'utilizzo delle singole applicazioni come componenti del processo di business, e soddisfare i requisiti utente in modo integrato e trasparente. E' un paradigma per l'organizzazione e l'utilizzazione delle risorse distribuite che possono essere sotto il controllo di domini di proprietà differenti. Fornisce un mezzo uniforme per offrire, scoprire, interagire ed usare le capacità di produrre gli effetti voluti consistentemente con presupposti e aspettative misurabili (defizione OASIS).

In una architettura orientata ai servizi, essi sono funzionalità di business ben definite, costruite come componenti software che possono essere riutilizzati per differenti finalità.

L'architettura SOA utilizza lo standard XML per veicolare l'informazione, e lo standard WSDL per descrivere i servizi. Come architettura di esecuzione, si basa su una dorsale di pubblicazione dei servizi denominata ESB (Enterprise Service Bus).

Questa componente ha lo scopo di creare le connessioni tra le diverse componenti tecnologiche che rappresentano lo strato di back end, e fornisce alcune funzionalità basilari come il routing, il monitoraggio e la gestione delle eccezioni.

2.2 Processo di sviluppo

Il seguente schema rappresenta i moduli di cui è composto il processo di sviluppo del software, che saranno considerati e dettagliati nei seguenti paragrafi.

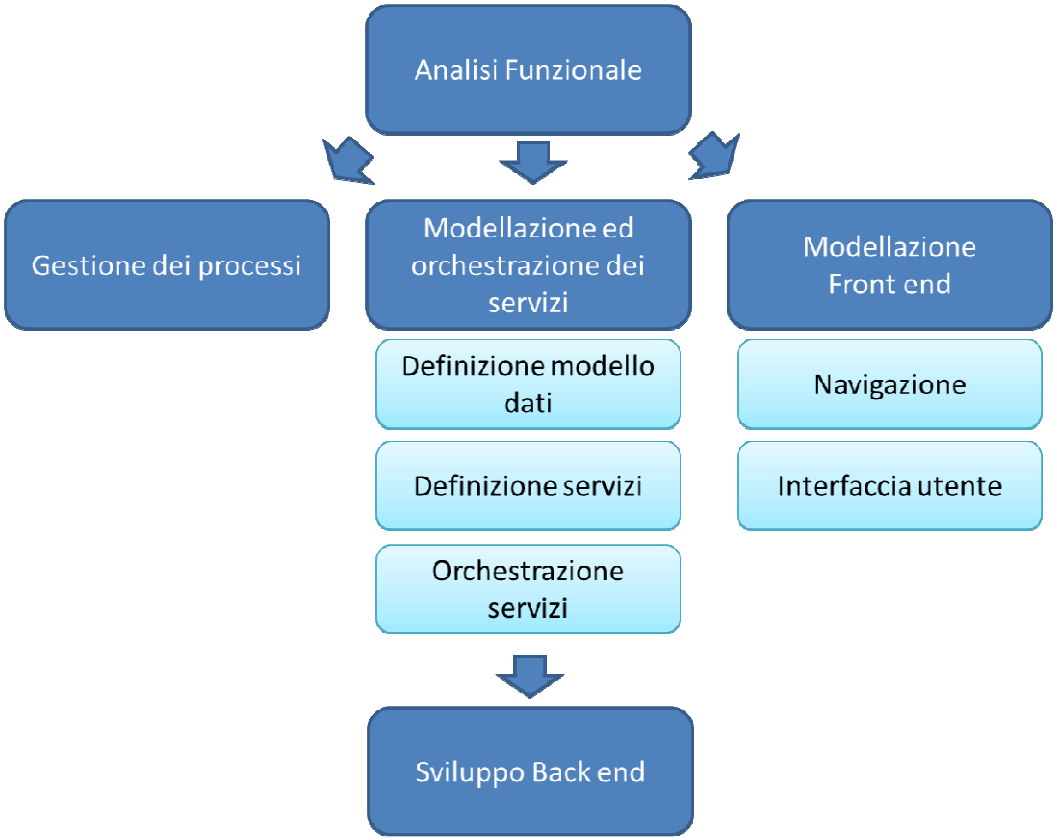

Figura 1 Processo di sviluppo

2.2.1 Analisi funzionale

La prima fase, di più alto livello, prevede la raccolta dei requisiti funzionali. L'analisi funzionale è quella fase del processo di sviluppo durante la quale vengono identificati e descritti i processi che compongono il sistema informativo analizzato.

Questa prima fase è ovviamente imprescindibile e propedeutica per il proseguo dello sviluppo, come evidenziato dallo schema del processo. Le fasi successive infatti, ed è questo un notevole punto di forza della metodologia studiata, sono quasi interamente parallelizzabili ed eseguibili da team diversi, ognuno con competenze specifiche. Questo garantisce la massima efficienza e rapidità nei tempi di sviluppo.

La fase di analisi, eseguita dagli analisti funzionali, produce essenzialmente tre output:

- La progettazione di uno o più **processi di business**, cioè le procedure, più o meno complesse, che implementano le funzionalità applicative
- La definizione del **data dictionary**, cioè l'insieme degli oggetti, e delle loro relazioni, che sono utilizzati dai processi precedentemente considerati
- La definizione delle **logiche di navigazione** e il layout dell'**interfaccia utente**, cioè le specifiche per lo sviluppo del front end dell'applicazione

Con l'ausilio dei referenti architetturali, vengono inoltre effettuate alcune scelte a livello di configurazione applicativa: a seconda delle esigenze riscontrate, possono essere infatti utilizzati alcuni strumenti messi a disposizione dalla piattaforma.

2.2.2 Gestione dei processi

Se i processi di business individuati durante l'analisi dovessero essere particolarmente complessi, o comunque ci sia la necessità di gestirne e coordinarne l'esecuzione, la soluzione architetturale prevista è l'utilizzo di un motore di gestione dei processi (BPM, Business Process Management).

In questo caso, è necessario un effort supplementare, rappresentato da una fase dedicata nel processo di sviluppo.

Un processo è un insieme di attività tra loro correlate finalizzate alla realizzazione di un risultato definito e misurabile; ragionando in ottica MDA, è il modello che descrive una sequenza di azione ripetibili.

Lo scopo della piattaforma di gestione dei processi è quello di standardizzare la progettazione di flussi operativi complessi, e di agevolare l'interazione tra più applicazioni su diversi canali tecnologici.

Inoltre, una caratteristica della piattaforma BPM è la capacità di migliorare costantemente i processi, tanto da poter essere definito un processo di ottimizzazione dei processi.

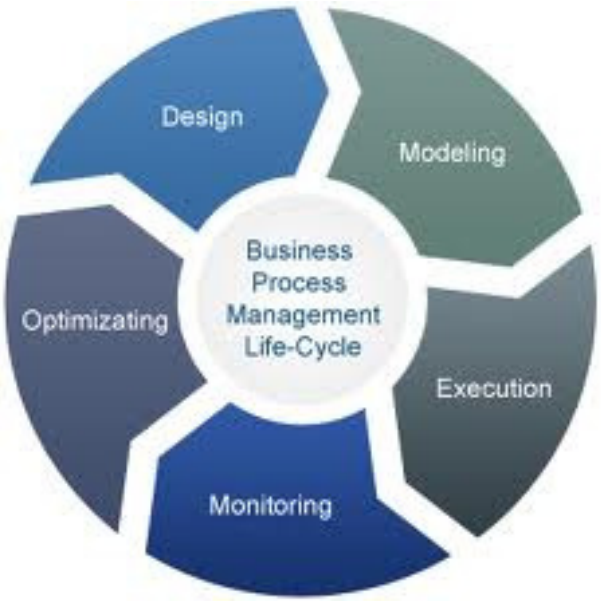

Figura 2 Ciclo di vita BPM

La figura illustra il ciclo di vita della gestione dei processi, che attraversa le seguenti fasi:

- **Disegno** : il disegno dei processi mira a identificare e rappresentare il flusso delle attività
- **Modellazione** : la modellazione concretizza il disegno dei processi e introduce le variabili che ne condizionano l'esecuzione
- **Esecuzione** : l'esecuzione del processo può essere integrale o parziale, a seconda degli strumenti e degli scopi
- **Monitoraggio** : il monitoraggio ha lo scopo di raccogliere dati sul processo in maniera più fruibile possibile, secondo diverse metriche specifiche
- **Ottimizzazione** : l'ottimizzazione tende ad eliminare le inefficienze riscontrate nel processo, dando valore al ciclo di vita BPM

2.2.3 Modellazione e orchestrazione dei servizi

La fase di modellazione e orchestrazione dei servizi è la più importante e rappresenta il core layer del processo di sviluppo. Questa fase implementa le scelte architetturali più importanti, che sono come abbiamo visto i pattern SOA e MDA.

Lo scopo principale della modellazione è il disaccoppiamento tra la vista funzionale e quella tecnica, garantendo l'indipendenza dalla piattaforma tecnologica.

La fase è composta da 3 diverse operazioni:

- Definizione del modello dei dati
- Definizione dei servizi
- Orchestrazione dei servizi

Ciascuna operazione viene effettuata attraverso la modellazione, cioè la rappresentazione di oggetti e funzioni attraverso strutture schematiche e semplificate.

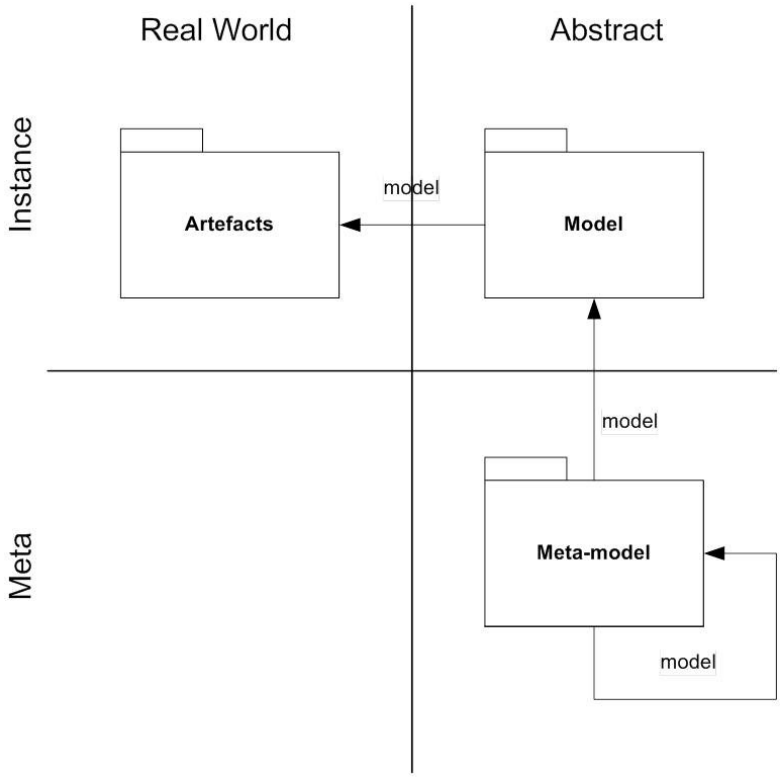

Figura 3 Modellazione

Lo schema illustra i livelli con cui la modellazione viene progettata ed eseguita. Se il modello è l'astrazione di un oggetto nel mondo reale, il metamodello è un ulteriore astrazione, e definisce la struttura e gli elementi basilari del modello.

2.2.3.1 Definizione del modello dei dati

La prima operazione consiste nella traduzione del data dictionary, definito in sede di raccolta dei requisiti, in un sistema di rappresentazione dei dati astratto e condiviso, cioè un modello ad oggetti.

Un concetto base che questa soluzione architetturale persegue è la riusabilità. Di conseguenza nel disegno di un modello per un nuovo processo un operazione propedeutica è la verifica dei modelli già esistenti, in modo tale da ottimizzare il lavoro. Nel nuovo modello, quindi, è sufficiente importare oggetti e relazioni già definiti che possono essere riutilizzati (esiste, come vedremo, la possibilità di caratterizzarle) limitandosi a modellare le nuove entità, che nel caso ottimale potrebbero addirittura non sussistere. Questo meccanismo è particolarmente efficace nel caso in cui i contesti delle varie applicazioni siano omogenei: è questo il caso studiato, in quanto le applicazioni gravitano tutte nel campo finanziario.

2.2.3.2 Definizione dei servizi

La successiva operazione consiste nella modellazione dei servizi, in ottica SOA.

Un servizio rappresenta una funzionalità operativa elementare che si appoggia al modello ad oggetti definito in precedenza.

Il servizio è un entità astratta che prescinde dalla tecnologia di implementazione, ma è una rappresentazione che ne descrive il comportamento (in termini essenzialmente di input/output). Alla stessa maniera degli oggetti, la rappresentazione dei servizi avviene in maniera schematica e semplificata attraverso la modellazione di diagrammi UML. Per

uniformare e semplificare lo sviluppo ci si avvale di un modello standard del servizio definito in maniera condivisa, secondo lo schema visto in precedenza.

2.2.3.3 Orchestrazione dei servizi

L'ultima operazione che compone la fase è l'orchestrazione dei servizi. Abbiamo visto come i servizi rappresentino funzionalità atomiche o comunque elementari; i processi di business però sono tipicamente costituiti da operazioni complesse, che utilizzano al loro interno svariate funzionalità. Questa funzione di orchestrazione dei servizi elementari è svolta da un ulteriore componente, denominato service process. Un service process, quindi, rappresenta schematicamente un processo di esecuzione di servizi elementari, e si avvale anch'esso di un modello standard, definito in maniera condivisa sulla piattaforma di sviluppo, che ne guida la modellazione.

Alla luce del fatto che le interfacce del service process sono le componenti esposte dal service bus, si può affermare che i service process (e non i servizi descritti nel paragrafo precedente) sono gli elementi che rappresentano i servizi definiti dal paradigma SOA, nonostante un infelice ambiguità nella denominazione.

E' evidente il vantaggio di utilizzare un doppio livello di astrazione per l'architettura SOA: la soluzione garantisce una perfetta modularità nella composizione dei servizi di alto livello, estendendo il concetto di riusabilità utilizzato per il modello ad oggetti e facilitando la comprensione e la manutenzione delle strutture software.

2.2.4 Sviluppo back end

Tutte le fasi finora analizzate risultano, come più volte sottolineato, indipendenti dalla tecnologia di implementazione. Durante la fase di integrazione del back end, viene ovviamente effettuata una specifica scelta tecnologica, ed il team di sviluppatori con le competenze specifiche procede secondo l'implementazione prescelta per ciascun servizio.

L'unico vincolo imposto agli sviluppatori è rappresentato dal rispetto dell'interfaccia di I/O del servizio modellato. Come vedremo, uno dei punti forza del framework è dato proprio dall'assistenza che esso fornisce agli sviluppatori, per cui questo vincolo risulta facilmente sostenibile.

Visto che questa fase è l'unica legata ad una specifica tecnologia, è importante sottolineare come il processo di sviluppo studiato sia flessibile e permetta di migrare agilmente a tecnologie più efficienti e performanti. Se vi fosse infatti questa esigenza, basterebbe quindi replicare unicamente la fase di integrazione.

Nel contesto di business analizzato, accade piuttosto frequentemente che tra i requisiti dell'applicazione vi sia l'utilizzo di una struttura documentale. In questo caso, la soluzione che l'architettura mette a disposizione è una piattaforma ECM (Enterprise Content Management, cioè la gestione dei contenuti).

Questa piattaforma può fornire diverse funzionalità, ma essenzialmente garantisce la gestione strutturata e flessibile del ciclo di vita del documento, oltre ovviamente a facilitarne l'indicizzazione e la conservazione.

2.2.5 Modellazione front end

L'ultima fase che analizzeremo nel processo è lo sviluppo dello strato di front end dell'applicazione.

L'architettura di front end si basa sul pattern MVC (Model-View-Controller). Il pattern MVC è imperniato sulla separazione dei compiti tra i componenti software, che si suddividono in tre strati:

- **Model**: fornisce i metodi per accedere ai dati utili all'applicazione.
- **View**: visualizza i dati forniti dallo strato Model e si occupa dell'interazione con gli utenti.
- **Controller**: elabora i comandi dell'utente (tipicamente attraverso lo strato View) e reagisce invocando gli altri due strati, secondo le logiche di navigazione impostate.

Proseguendo nell'applicazione del concetto di modellazione alle varie fasi dello sviluppo, è stato ideato un particolare modello per il disegno della navigazione logica del front end.

In seguito, per continuare invece con il concetto di generazione di ambienti di runtime standardizzati, è stato messo a punto uno strumento di assistenza nel disegno dell'interfaccia utente.

2.2.5.1 Modellazione della navigazione

Per rappresentare la logica di navigazione si è scelto ancora una volta di adottare il paradigma MDA ed utilizzare la modellazione. A partire dalla definizione delle regole elaborate durante la fase di raccolta dei requisiti, viene modellata infatti una macchina a stati, che rappresenta schematicamente il flusso della navigazione.

La macchina a stati è un modello utile a rappresentare il comportamento dinamico di un sistema. Gli elementi essenziali di una macchina a stati sono:

- **Stato**: condizione che identifica una particolare fase di vita del sistema.
- **Evento**: avvenimento significativo nel ciclo di vita del sistema.
- **Transizione**: il passaggio tra due stati in risposta ad un evento.

Come abbiamo visto, si è scelto di applicare tale modello al sistema costituito dal flusso di navigazione di un applicazione. La definizione degli elementi essenziali viene adeguata di conseguenza:

- **Stato**: insieme delle componenti e dei dati che rappresentano l'interfaccia visuale in un determinato istante della navigazione.
- **Evento**: avvenimento generato dall'interazione utente durante la navigazione.
- **Transizione**: il passaggio tra due stati, implica tipicamente una generica chiamata ai servizi erogati dalla piattaforma, per reperire i dati necessari alla visualizzazione dello stato di arrivo.

La modellazione di una macchina a stati, oltre ad offrire la possibilità di definire un flusso di navigazione in termini astratti permette di tradurre il diagramma in codice che può essere sfruttato per implementare la parte dell'interazione utente.

2.2.5.2 Modellazione interfaccia utente

Per supportare la modellazione dell'interfaccia utente è stato sviluppato un tool grafico denominato Visual Editor. Tale strumento offre allo sviluppatore una serie di funzionalità, che vanno dall'integrazione con lo strato di sicurezza, all'internazionalizzazione.

E' importante sottolineare, in ottica di parallelizzazione delle attività, l'interazione che si viene a creare con la modellazione dei servizi e dei processi: come abbiamo visto, la definizione dei service process e di conseguenza la generazione delle interfacce, può essere effettuata prima dell'implementazione del servizio di back end. Si capisce quindi come, una volta terminata la fase di modellazione ed orchestrazione dei servizi, lo sviluppo delle integrazioni di back end e di front end possano essere eseguite in modo parallelo, e siano indipendenti tra loro.

3 Framework JMC

In questo capitolo sarà presentato il framework JMC (acronimo di Jet Modeling Composer) che è la struttura che implementa e supporta il processo di sviluppo definito nel precedente capitolo.

JMC è un framework di esecuzione OLTP basata su standard di mercato e prodotti open source che permette di far interagire sistemi e soluzioni eterogenee (gestione del repository aziendale dei processi, dei servizi e del modello dei dati), come abbiamo visto studiando il contesto, garantendo la consistenza ed il governo.

OLTP (OnLine Protocol Processing) è una tecnologia che permette agli utenti di accedere in modo rapido ed intuitivo ad una elevata mole di dati. OLTP si integra perfettamente con l'architettura SOA e si basa sulla transazionalità dei componenti.

Oltre ad implementare i concetti e le scelte architetturali illustrati nel precedente capitolo, uno degli aspetti più interessanti del framework è la separazione, a livello di approccio, tra il disegno delle componenti e l'implementazione degli stessi. A livello di design time, infatti, si è scelto di personalizzare pesantemente gli strumenti, fornendo strumenti flessibili e ricchi di funzionalità. L'obiettivo a cui questa scelta tende è quello, d'altra parte, di standardizzare il più possibile il codice di runtime, rendendolo governabile e mantenibile in maniera semplice e centralizzata, nonostante il crescere delle applicazioni e delle tecnologie supportate.

Partendo dal presupposto che l'intera architettura si basa sui modelli, questo concetto si traduce nello sviluppo di meccanismi di generazione automatica del codice a partire dal modello. Lo sforzo quindi nell'elaborare questi strumenti rende anche più semplice il lavoro degli sviluppatori, limitandone al minimo (idealmente azzerandolo) il lavoro di scrittura del codice di esecuzione.

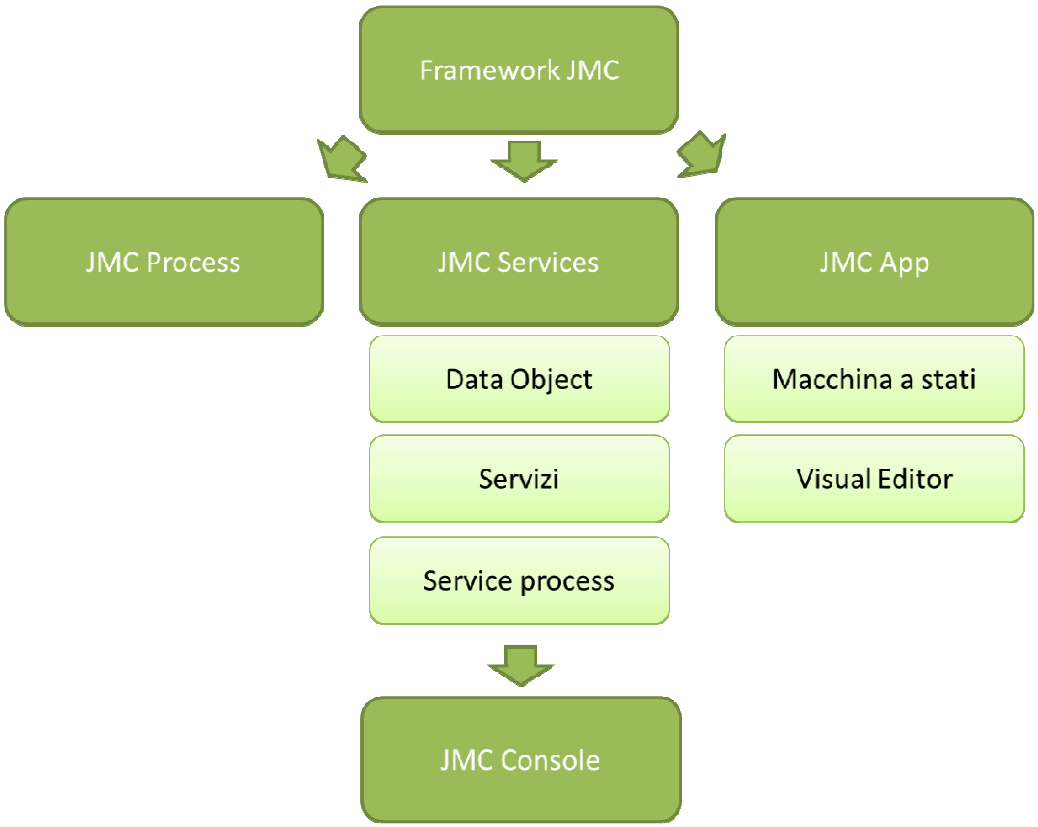

Figura 4 Framework JMC

In figura è descritta la struttura degli strumenti offerti dal framework: come si può facilmente notare, essa ricalca quasi fedelmente il processo di sviluppo descritto nel primo capitolo.

E infatti il framework è studiato appunto per fornire a ciascuna fase del processo uno strumento adeguato agli scopi e ai principi descritti.

L'unica incongruenza è rappresentata dall'integrazione con il back end, cui JMC non ha un vero e proprio strumento, ma lascia agli sviluppatori la libertà di utilizzare l'IDE a loro più congeniale, mettendo comunque a disposizione l'ambiente di sviluppo standard.

Al posto della fase di integrazione del back end, dunque, è invece riportato uno strumento di Governance, denominato JMC Console, che mette a disposizione varie funzionalità a supporto sia della fase di disegno che di esecuzione del software.

3.1 Strumenti tecnici

Lo strumento di modellazione e sviluppo è Rational Software Architect (RSA) prodotto da IBM e basato sull'IDE Eclipse.

Attualmente viene utilizzata in SEC la versione 7.6, che però non supporta l'integrazione con alcuni application server (in particolare JBoss). Per cui temporaneamente lo sviluppo su tali server viene effettuato con MyEclipse. Il problema sarà risolto con la migrazione alla versione 8 di RSA, che sopperisce ai limiti della versione precedente e diventerà quindi la piattaforma di sviluppo standard per tutte le fasi del processo.

Gli application server in cui le applicazioni sono installate sono in questo memento IBM WebSphere e JBoss.

Lo strumento per il versionamento del software (sia la parte dei modelli che le implementazioni) utilizzato è Rational Team Concert (RTC) prodotto da IBM e integrato su RSA tramite un plugin. RTC, oltre ad offrire i normali servizi di sincronizzazione, mette a diposizione anche un ambiente centralizzato per la compilazione del codice. Tale ambiente è costantemente allineato con quello di produzione e garantisce quindi la compatibilità delle librerie generate con tale ambiente.

Infine, RTC consente anche di effettuare il deploy delle librerie nei vari ambienti, gestendone in maniera automatica la schedulazione nel rispetto delle politiche aziendali.

Per quanto riguarda la gestione dei processi, lo strumento utilizzato è Metastorm BPM nella versione 9.1.

3.2 Sviluppo dei servizi (JMC Services)

Lo strumento messo a disposizione dal framework JMC per supportare la fase di modellazione ed orchestrazione dei servizi è JMC Services, che è a sua volta un framework sviluppato per la piattaforma di sviluppo RSA.

Esso fornisce un ambiente per lo sviluppo di servizi, perseguendo alcune principali finalità:

- Astrazione del servizio, attraverso un linguaggio DSL
- Standardizzazione del servizio, secondo le specifiche SOA
- Realizzazione del servizio, tramite la generazione automatica del codice

Il linguaggio DSL prescelto è l'UML (Unified Modeling Language). Per sua natura, il linguaggio UML è un linguaggio di natura generica, e non possiede le caratteristiche necessarie ad un linguaggio DSL. D'altra parte, l'UML può essere esteso tramite i profili UML, rendendolo adeguato per descrivere un dominio specifico, conservando le proprie qualità intinseche, che sono la soprattutto semplicità di comprensione e la disponibilità di svariati tool per l'analisi e l'esecuzione.

In particolare, i profili UML consistono in:

- **Stereotipi** : sono tag che estendono la semantica di un oggetto
- **Proprietà** : sono informazioni aggiuntive associate agli stereotipi
- **Vincoli** : sono regole che definiscono quali sono gli elementi UML permessi

Riprendendo la Figura 3 , che illustra i livelli di modellazione adottati nell'architettura, possiamo quindi affermare che sia il metamodello che il modello vengono rappresentati tramite il linguaggio UML così esteso.

Il metamodello, in particolare, è stato costruito dagli sviluppatori del plugin JMC Services mentre il modello viene costruito dai modellatori nella corrispondente fase del processo di sviluppo.

Abbiamo visto come la fase di modellazione ed orchestrazione dei modelli si suddivida in 3 operazioni distinte, cioè la definizione del data model e la modellazione di servizi e service process. Analizziamole dunque alla luce della struttura UML a disposizione.

3.2.1 Data Model

Il data model JMC è una collezione strutturata di classi UML stereotipate, appartenenti alle seguenti tipologie:

- DATA OBJECT : rappresenta un entità consistente, oppure un raggruppamento di attributi affini
- FINDER : è un particolare data object utilizzato per rappresentare le chiavi di ricerca
- DATA SERVICE : rappresenta una struttura dati utile al funzionamento interno di un servizio

Ciascun elemento descritto può essere arricchito di attributi primitivi, ciascuno con la propria molteplicità, che ne caratterizzano la persistenza.

E' inoltre possibile utilizzare delle associazioni per mettere in relazione elementi diversi, anch'esse con la propria molteplicità. Esistono vari tipi di associazione, ma le più utilizzate sono l'associazione diretta, che modella un oggetto come un attributo non primitivo di un altro oggetto, e la generalizzazione, che invece permette ad un oggetto di ereditare gli attributi di un oggetto diverso.

A valle dell'analisi funzionale, quando ci si approccia alla modellazione di un servizio e del suo relativo data model, bisogna innanzitutto svolgere un analisi dei dati tesa ad individuare quale struttura degli elementi UML sia più consona a rappresentarli.

In seguito, nel rispetto del concetto di riusabilità alla base della filosofia MDA, bisogna effettuare una ricerca del dizionario condiviso per capire se eventualmente alcuni oggetti sono già stati (anche parzialmente) creati. Per fare questo esiste uno strumento apposito messo a disposizione da JMC Console, che è la piattaforma di governance del framework e le cui funzionalità saranno dettagliate in seguito.

Questo dizionario dei dati è basato su un metamodello costruito con EMF (Eclipse Modeling Framework) strutturalmente più semplice del metamodello utilizzato per la modellazione degli oggetti, in quanto deve descrivere solamente le proprietà e le relazioni degli oggetti stessi, senza ulteriori tipizzazioni.

Eseguite queste operazioni preliminari, si passa alla modellazione vera e propria degli oggetti precedentemente descritti.

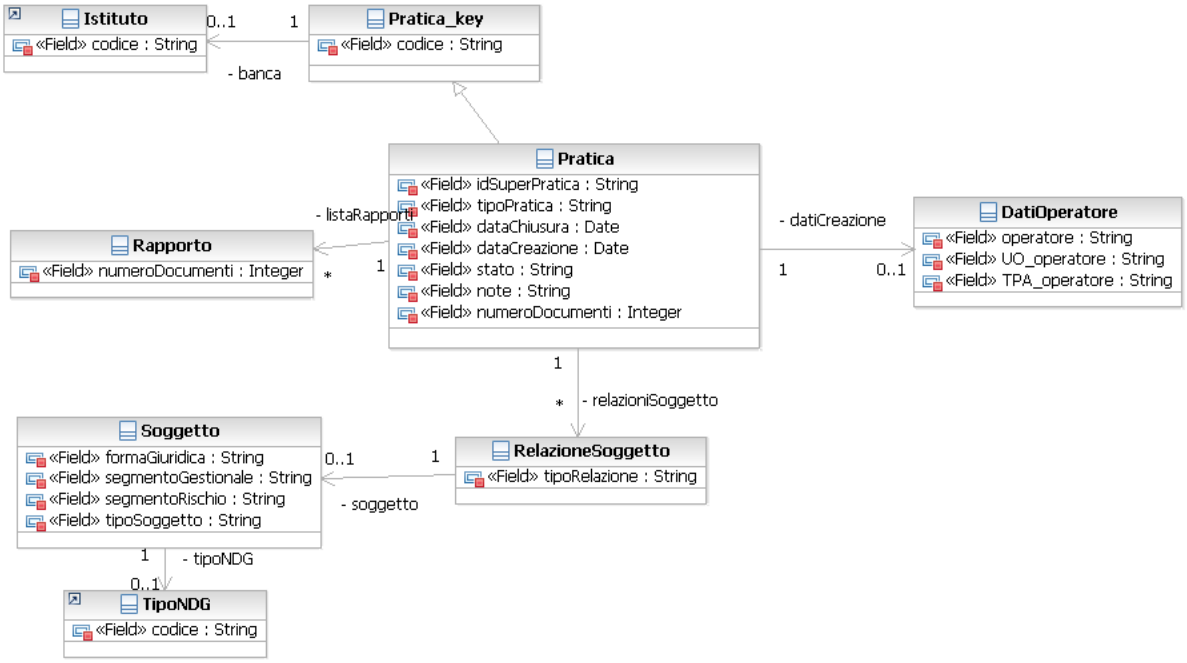

Figura 5 Modellazione del data model

3.2.2 Servizi

Una volta che abbiamo il data model a disposizione, l'operazione successiva è la modellazione dei servizi.

Un servizio JMC è costituito da una classe UML sterotipata con uno stereotipo di tipo "service" che ne caratterizza la tipologia. Tra gli stereotipi attualmente a disposizione i più utilizzati sono ECI, CobolFlow, DB2, Oracle, SqlServer, WebService, Generic, Http.

Ciascuno stereotipo associa un insieme di proprietà al servizio, specifiche della tipologia prescelta, tipicamente relative alla connessione verso il back end (un servizio Oracle, per esempio, avrà una proprietà che specifica i nomi del package e della stored procedure da chiamare, che ne rappresenta il back end).

Viene inoltre associata alla classe che rappresenta il servizio una operazione, che rappresenta l'azione che il servizio stesso andrà ad eseguire.

All'elemento servizio così creato sarà possibile, al pari delle classi UML del data model, aggiungere degli attributi primitivi oppure aggiungere degli attributi non primitivi utilizzando le associazioni, utilizzando quindi gli oggetti precedentemente modellati.

E' importante, in questo caso, oltre a specificare la molteplicità di tali attributi, specificare anche se essi costituiscono dati di input o di output al servizio (è possibile anche selezionare entrambe le possibilità) tramite la valorizzazione di un apposita proprietà.

E' possibile infine implementare per il servizio una specifica gestione degli errori. Tramite una funzionalità del plugin, infatti, si può associare al servizio un errore già definito in un dizionario apposito, oppure procedere alla definizione di un nuovo errore.

A questo punto la modellazione del servizio è terminata, e le operazioni da effettuare sono la validazione e la trasformazione.

La validazione è un utilità del plugin che effettua i controlli di integrità del modello UML costruito, segnalando al modellatore eventuali anomalie, indirizzandolo nella risoluzione delle stesse.

La trasformazione del servizio è un operazione che elabora la struttura UML appena creata, la traduce in codice Java e produce le librerie e i file di configurazione necessari per il deploy sulla dorsale di pubblicazione dei servizi.

Oltre a questo, a seconda della tipologia di servizio selezionata, produce anche le strutture della tecnologia specifica pronte per l'implementazione (ad esempio, per i servizi di tipo Oracle viene creata la struttura della stored procedure, compresa la definizione delle variabili di input/output).

Infine, un ulteriore funzionalità è la generazione automatica della documentazione: tale funzionalità si basa su un metamodello ad hoc, costruito con EMF in maniera analoga al metamodello relativo al dizionario dei dati.

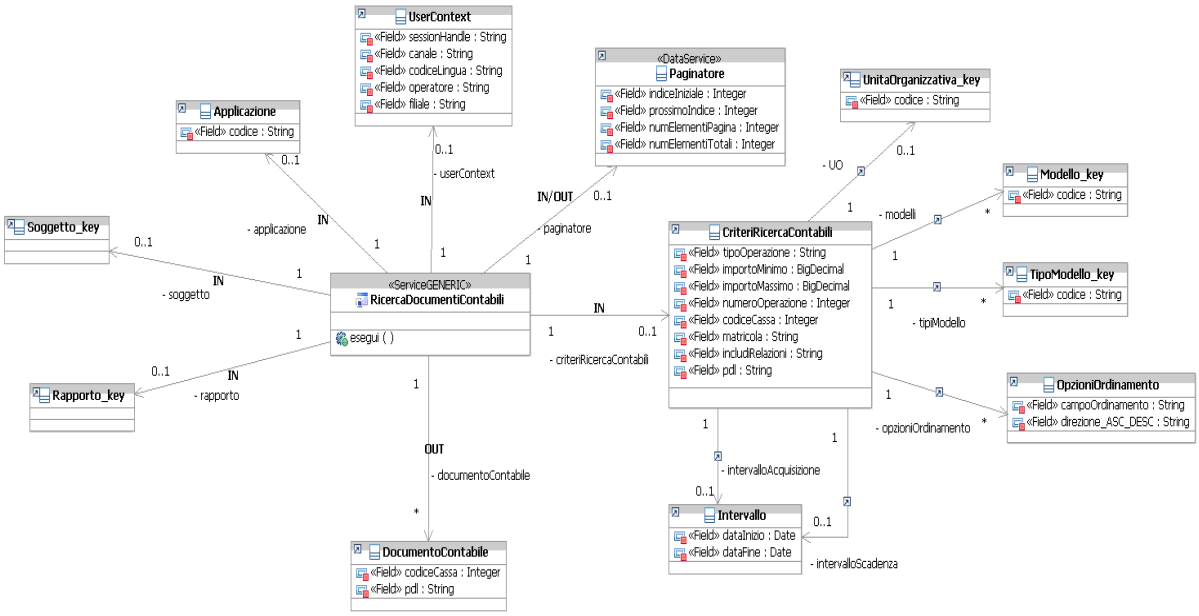

Figura 6 Modellazione di un servizio

3.2.3 Service process

L'ultima operazione da compiere in questa fase dello sviluppo riguarda l'orchestrazione dei servizi, che si traduce nella modellazione dei service process.

Il service process è un componente più complesso dei servizi, ed è costituito da diversi elementi:

- Classe UML che rappresenta il service process
- Classi UML di request e response
- Sequence diagram
- Mappatura

Tipicamente, la creazione di un service process è guidata da un wizard presente nel plugin: la prima operazione richiesta è la scelta dei servizi che il processo dovrà orchestrare.

A questo punto le classi di request e di response, che rappresentano l'interfaccia del service process verso il mondo esterno, sono generate automaticamente dal processo di creazione e sono formate dalla somma degli attributi di input ai servizi (classe di request del sp) e dalla somma degli attributi di output ai servizi (classe di response del sp).

L'orchestrazione è un processo dinamico, ed è gestita tramite la mappatura e il sequence diagram.

La mappatura viene eseguita tramite un tool grafico, e consente di collegare gli attributi degli oggetti di request e response con gli attributi di input/output dei servizi orchestrati.

Il sequence diagram, infine, permette di disegnare il flusso temporale di esecuzione dei singoli servizi, ed è possibile anche condizionarlo attraverso apposite funzioni di controllo OCL.

Al termine della modellazione, come nel caso dei servizi, le operazioni da effettuare sono la validazione e la trasformazione, che hanno il medesimo significato, e che quindi producono le librerie e i file di configurazione utili al deploy dei componenti sul service bus.

Esiste la possibilità di definire particolari service process, denominati template, che contengono delle orchestrazioni standard di servizi, dei processi cioè utilizzati trasversalmente dai processi applicativi (come ad esempio delle chiamate di autenticazione o tracciatura). I template così definiti possono essere inseriti, in fase di creazione, nel sequence diagram di un

determinato service process, evitando così di replicare il lavoro ad ogni orchestrazione modellata.

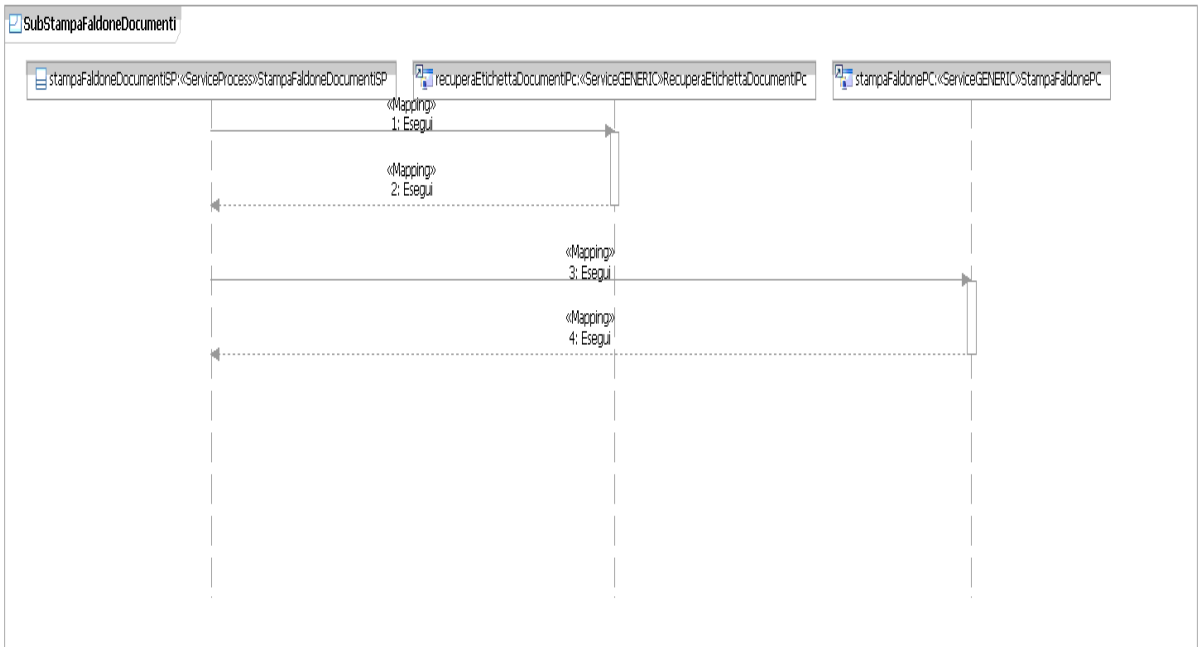

Figura 7 Modellazione di un service process

3.3 Sviluppo front end (JMC App)

Per quanto concerne lo sviluppo del front end, cioè l'interazione con l'utente, il framework JMC mette a disposizione una piattaforma di strumenti e funzionalità che prende il nome di JMC App.

Come abbiamo visto, l'architettura di front end si basa sul pattern MVC: JMC App implementa il paradigma adottando la tecnologia Java Server Faces (JSF) nella versione 2.0, e conglobando inoltre la componente RichFaces nella versione 4.1.

I tre strati quindi vengono implementati nel seguente modo:

- MODEL : lo strato di modello non è interamente conglobato come da prassi, in realtà l'integrazione con lo strato di persistenza avviene attraverso le interfacce dei service process esposte sul service bus
- VIEW : lo strato di presentazione è rappresentato da facelets XHTML (eXtensible Hyper Text Markup Language) che coniugano le proprietà del linguaggio XML con le caratteristiche dell'HTML, integrandosi in maniera efficiente con JSF
- CONTROLLER : lo strato di controller viene implementato secondo le specifiche JSF, quindi attraverso i managed beans

Lo sviluppo del front end passa innanzitutto attraverso la modellazione della navigazione, e viene successivamente implementata attraverso le funzionalità offerte da un tool grafico denominato Visual Editor.

3.3.1 Macchina a stati

La macchina a stati è un metamodello costruito utilizzando, analogamente alla modellazione dei servizi, il linguaggio UML esteso dai profili UML.

I profili sono stati applicati agli stati ed alle transizioni, e sono di due tipi:

- SMProfile : profilo che definisce le proprietà relative agli elementi della macchina a stati che sono indipendenti dalla piattaforma e fanno riferimento alla personalizzazione JMC delle proprietà della macchina a stati
- PSMProfile : profilo che definisce le proprietà relative agli elementi della macchina a stati che sono dipendenti dalla piattaforma di esecuzione

Il diagramma della macchina a stati permette di definire il flusso relativo all'interazione utente dell'applicazione sviluppata.

Gli stati che popolano il diagrmma possono appartenere alle seguenti tipologie:

- Stato Iniziale : tipologia primitiva UML per indicare lo stato di partenza che è unico
- Stato Finale : tipologia primitiva UML per indicare lo stato di terminazione della macchina a stati. Possono essere presenti più stati finali
- Stato : tipologia primitiva UML specializzata tramite gli stereotipi
    a) NON RENDERIZZABILE : non ha corrispondente visuale
    b) RENDERIZZABILE : lo stato definisce gli elementi che andranno a comporre la parte di "presentation"
- Sottomacchina : equivale a inserire la macchina a stati al posto di un normale stato. Permette il riutilizzo di macchine a stati in contesti diversi da quelli per cui è stata realizzata in un'ottica di riusabilità dei modelli
- Nodo Decisionale: è uno stato di tipo non renderizzabile al quale sono associate delle transizioni pilotate da una condizione detta "di guardia". Tale tipologia di stato permette di rappresentare un flusso condizionato nell'interazione utente.
- Nodo di passaggio: è uno stato di tipo NON RENDERIZZABILE in cui sia le transizioni di ingresso che quelle di uscita rappresentano delle chiamate a back end.

Il diagramma è completato dalle transizioni: una transizione può essere vista come un evento associato all'interazione utente che può portare la macchina ad assumere un diverso stato rispetto a quello di partenza, o a ritornare al medesimo stato dopo aver eseguiro però una chiamata verso il back-end.

Tale evento può essere sottoposto al controllo di una "guardia", ovvero una condizione che permette, se verificata, alla macchina a stati di percorrere la transizione fino allo stato di arrivo.

E' possibile pilotare una chiamata a back end direttamente dallo stato, senza utilizzare una transizione, tramite una cosiddetta chiamata tecnica. Questo meccanismo è utilizzato per recuperare dei dati di servizio utili nelle gestione dello stato, non direttamente legati alla logica di navigazione.

Quando si parla di chiamata a back end, si intende l'interazione con i servizi esposti dall'architettura, quindi con le interfacce di request e response dei service process (che, come abbiamo detto, compongono lo strato di modello dell'architettura MVC).

Terminata la modellazione della macchina a stati, si procede alla trasformazione del modello, che produce i componenti necessari alla successiva fase di disegno dello strato di interazione utente.

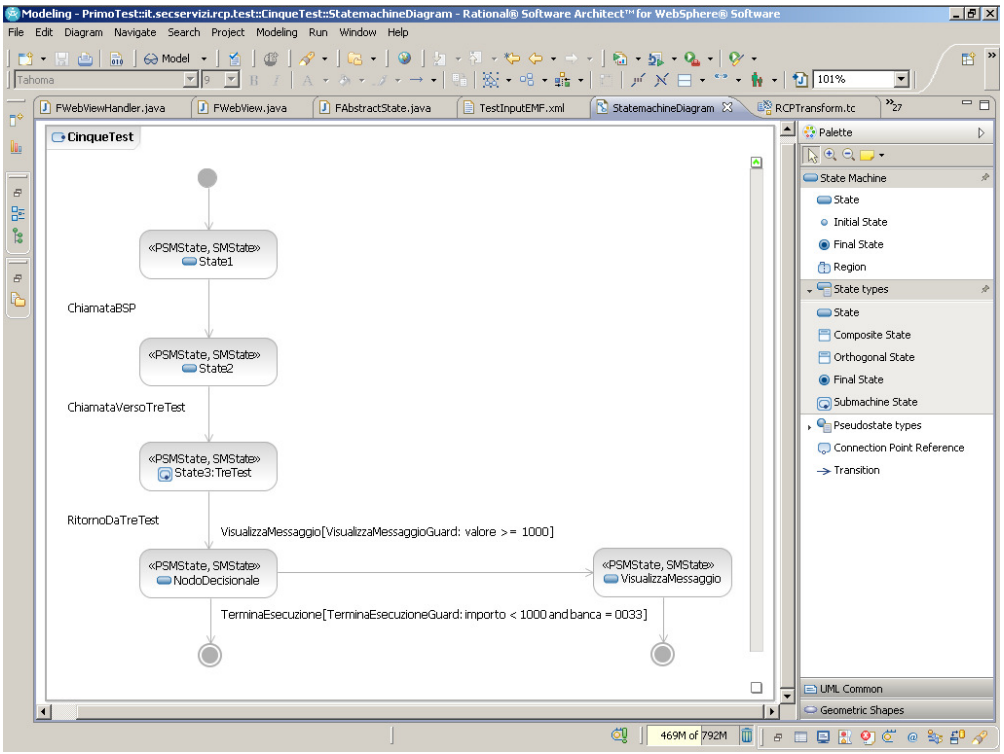

Figura 8 Modellazione macchina a stati

3.3.2 Visual Editor

Il Visual Editor è un tool grafico utilizzato per la modellazione dell'interfaccia visuale e per la generazione del codice che la implementa. E' stato costruito come estensione del progetto Visual Editor di Eclipse.

Il Visual Editor si basa su di un metamodello visuale XMI, che definisce le componenti visuali che lo sviluppatore può modellare. Quando il modello sarà terminato, l'operazione di trasformazione genererà il codice XHTML della pagina.

Nel momento in cui si aggiungono le varie componenti (widget) alla pagina, lo strumento offre diverse funzionalità:

- Autenticazione : in maniera automatica, vengono recuperate le informazioni dell'utente verificate dall'agente di autenticazione (SiteMinder)
- Autorizzazione : con un meccanismo analogo, recupera le condizioni di autorizzazione dell'utente, e attraverso la definizione di opportune regole può condizionare la renderizzazione di determinati componenti a tali CDA
- Internazionalizzazione : gestisce in modo centralizzato un dizionario di etichette (denominate k-code)
- Comunicazione : garantisce l'accesso in maniera trasparente al service bus ed ai servizi pubblicati, tracciando tutte le chiamate
- Centralizzazione : alcuni componenti standard sono reperibili direttamente sull'http server, garantendo una rapida renderizzazione

E' inoltre possibile per gli sviluppatori creare dei nuovi componenti grafici personalizzati, sfruttando lo standard JSF. Se tali componenti risultano utilizzabili in maniera trasversale, possono essere pubblicati nel tool e resi disponibili per lo sviluppo di altre applicazioni.

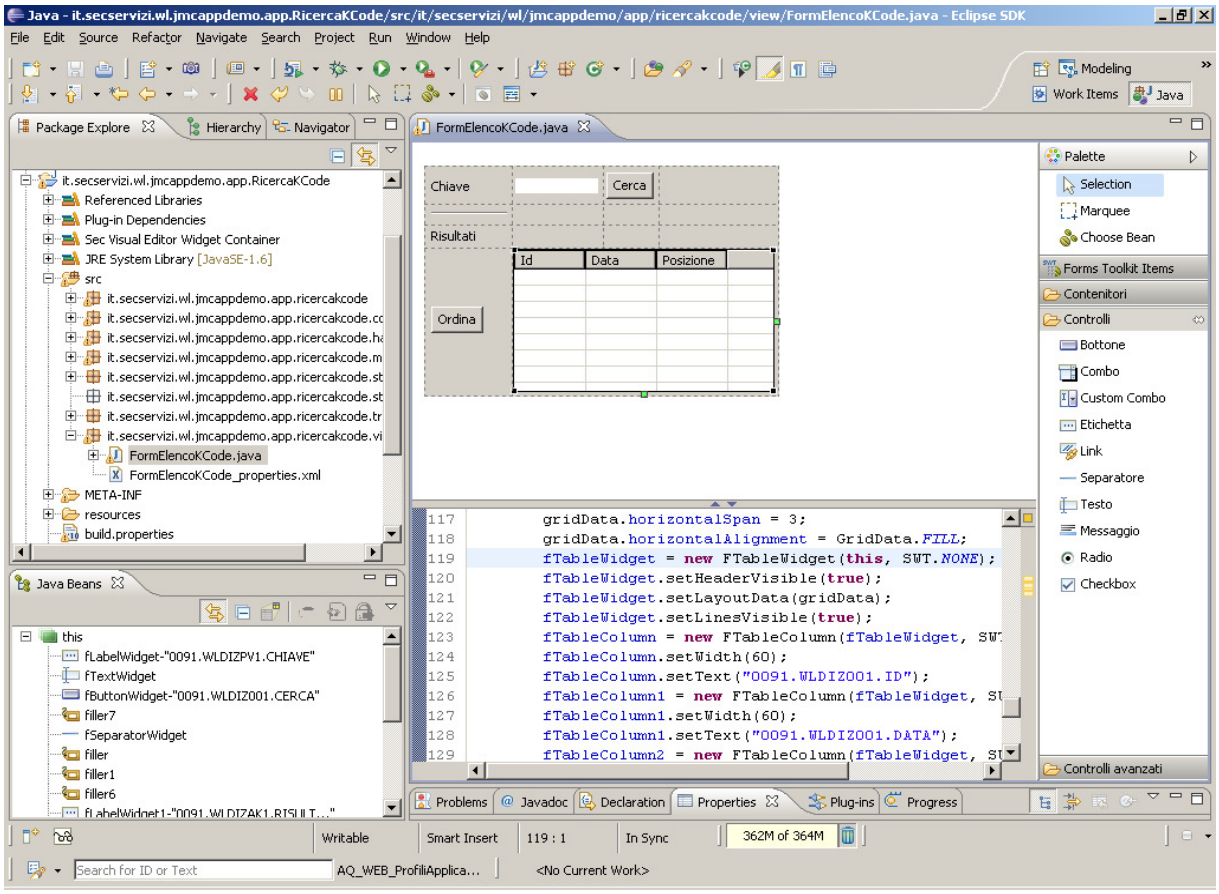

Figura 9 Modellazione View con Visual Editor

3.4 Architettura di esecuzione

Nella seguente figura è rappresentata l'attuale architettura di esecuzione presente in SEC.

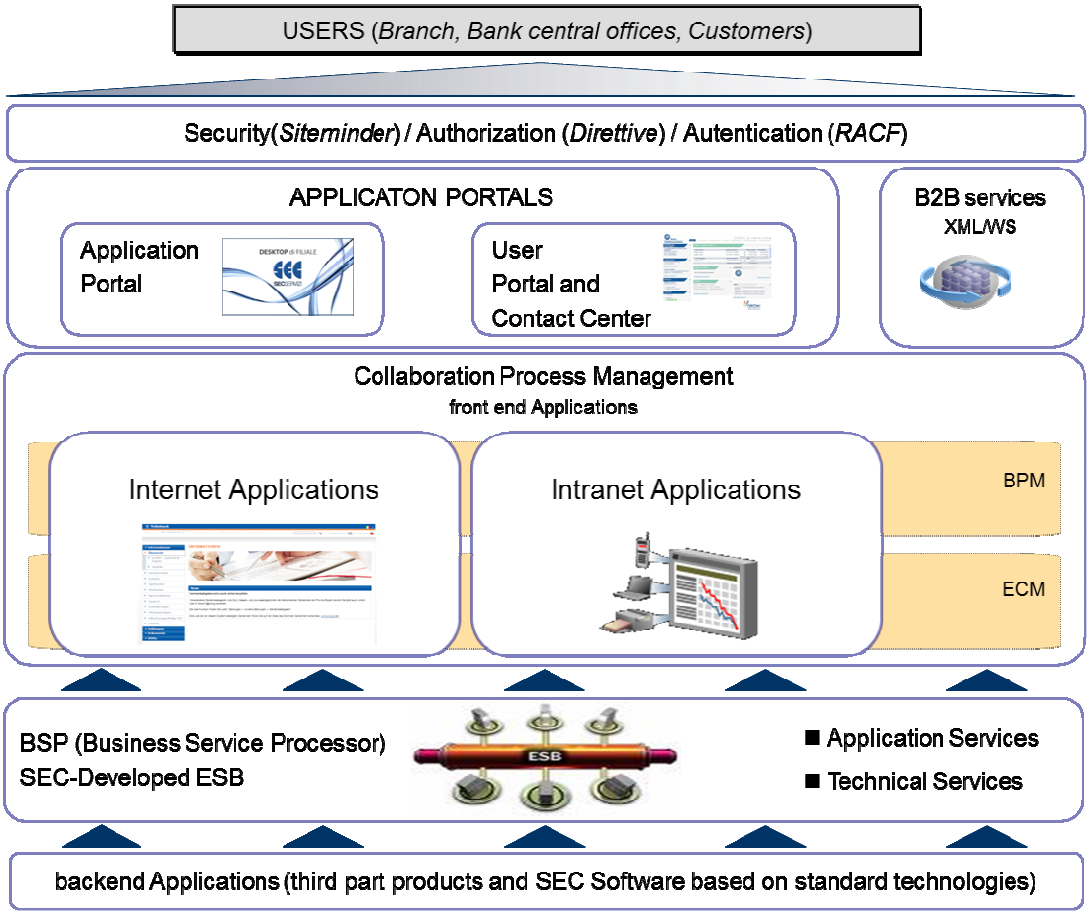

Figura 10 Architettura di esecuzione

Oltre a dare un utile visione generale, dallo schema si evince come la componente centrale e fondamentale dell'intero sistema sia il service bus, che in ambito JMC è denominato BSP (Business Process Processor).

Durante le comunicazioni tra le varie componenti a cui è connesso, il BSP attua un meccanismo di deserializzazione degli oggetti. Questo meccanismo è sostenibile grazie alla generazione automatica delle classi che rappresentano gli oggetti di input/output, in quanto tali classi contengono nativamente i metodi necessari ad effettuare la deserializzazione.

Il BSP è un Enterprise Java Bean di tipo stateless, che adotta l'architettura J2EE Connector.

Questo si traduce nell'implementazione di due fondamentali componenti:

- il transaction manager, che gestisce le transazioni
- il resource adapter, che gestisce in maniera normalizzata le chiamate ai connettori verso il back end

Il transaction manager è di tipo bean-managed, cioè la transazionalità è definita dalla tipologia del servizio, su un apposito file di configurazione.

Il resource adapter, denominato JMC Connector, è un connettore di connettori che offre il notevole vantaggio di esporre ai servizi un interfaccia standard verso le risorse di back end.

3.5 Governance (JMC Console)

JMC Console è un applicazione web che rappresenta un contenitore di strumenti di governance in continua evoluzione. Mano a mano che il framework JMC si arricchisce di funzionalità e espande i suoi campi di utilizzo, infatti, si rafforza l'esigenza di strumenti sempre più efficienti di controllo e governo sulle attività.

Attualmente, JMC Console supporta strumenti afferenti alle seguenti aree di interesse:

- Analisi d'impatto
- Testing
- Monitoraggio

Le tre aree descritte sono molto diverse tra loro: l'analisi d'impatto è uno strumento a cui abbiamo accennato descrivendo la modellazione dei dati, ed è quindi un supporto utile a design time. L'ambiente di test invece è uno strumento molto utile durante lo sviluppo dei servizi e dei service process, durante quello che possiamo chiamare develop time. Il monitoraggio, infine, è una funzionalità prettamente legata al run time.

3.5.1 Analisi d'impatto

Come abbiamo visto, esiste un dizionario (a sua volta modellato con EMF) che può essere consultato per verificare l'esistenza di oggetti. Oltre a questo, c'è la possibilità di navigare le gerarchie tra gli oggetti nel dizionario: questa funzionalità è utile per capire quale impatto abbiano sul modello le modifiche ad uno specifico elemento (da cui il nome dello strumento).

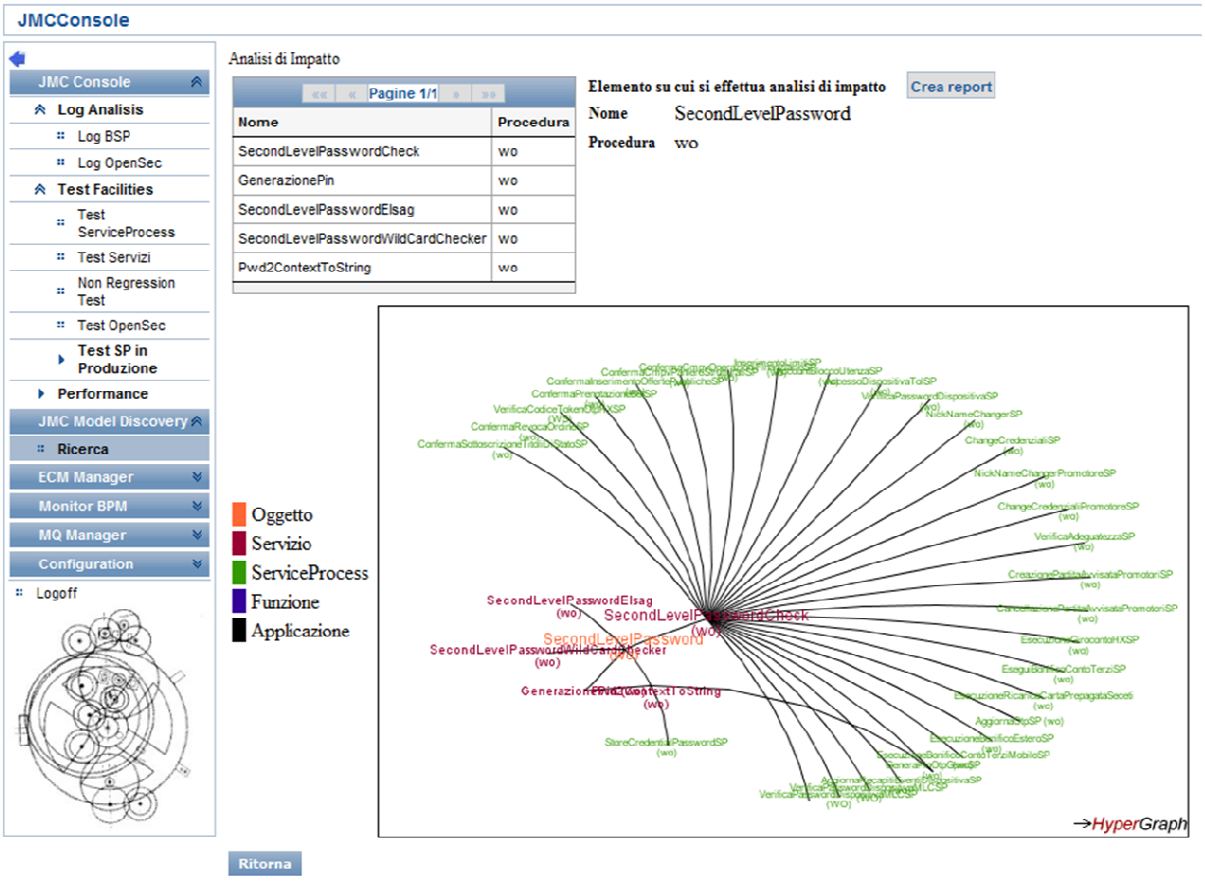

Figura 11 Analisi d'impatto su JMC Console

3.5.2 Testing

Il test dei servizi e dei service process è un attività di fondamentale importanza durante lo sviluppo, utile per verificare il comportamento delle componenti, sia dal punto di vista del corretto funzionamento, che dal punto di vista delle performance.

Lo strumento infatti permette di sottomettere al servizio una request opportunamente valorizzata, e di lanciarne l'esecuzione (ovviamente su un server in cui la componente sia stata deployata) verificando la response ottenuta, i log (tutti quelli presenti ai vari livelli dell'architettura) e il tempo di esecuzione. E' possibile salvare a DB diverse request relative ad un determinato servizio, facilitando così il ciclo di test.

Una interessante funzionalità, infine, è quella di poter schedulare test massivi (utilizzando le request salvate su DB) per simulare la reazione del sistema a carichi elevati.

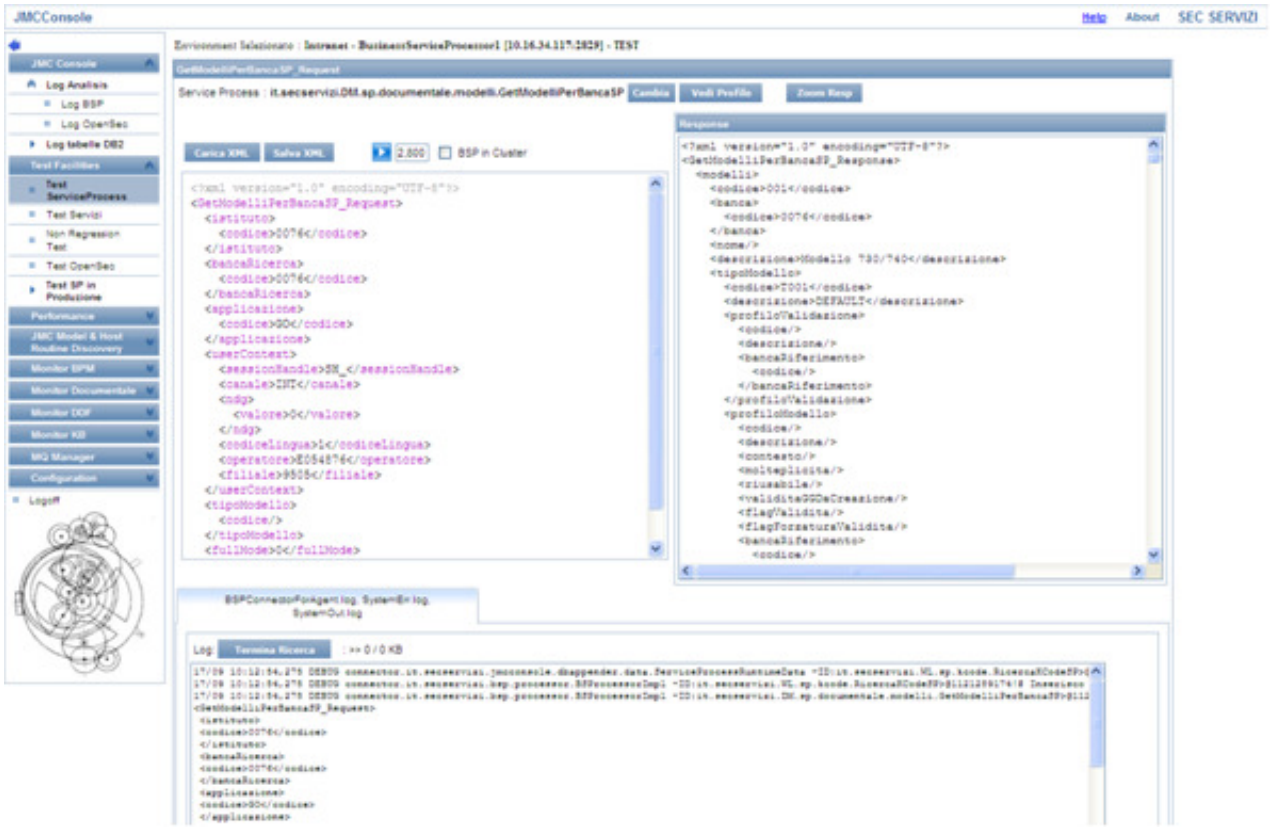

Figura 12 Testing su JMC Console

3.5.3 Monitoraggio

L'ultima funzione considerata è il monitoraggio. In un sistema informativo complesso ed in continua espansione, com è la piattaforma di esecuzione JMC, è essenziale monitorare in maniera precisa ed immediata le performances di tutte le componenti coinvolte.

Se consideriamo poi che il contesto di business in cui si lavora è quello bancario, e che una parte significativa dei servizi è garantita per un funzionamento 24x7 (quindi senza interruzioni) si capisce come l'immediato rilevamento di qualsiasi anomalia, sia essa anche un rallentamento, è un obiettivo cruciale.

JMC Console fornisce svariate rappresentazioni delle performances, con livello di dettaglio via via più granulare, fondamentalmente suddivise per applicazione.

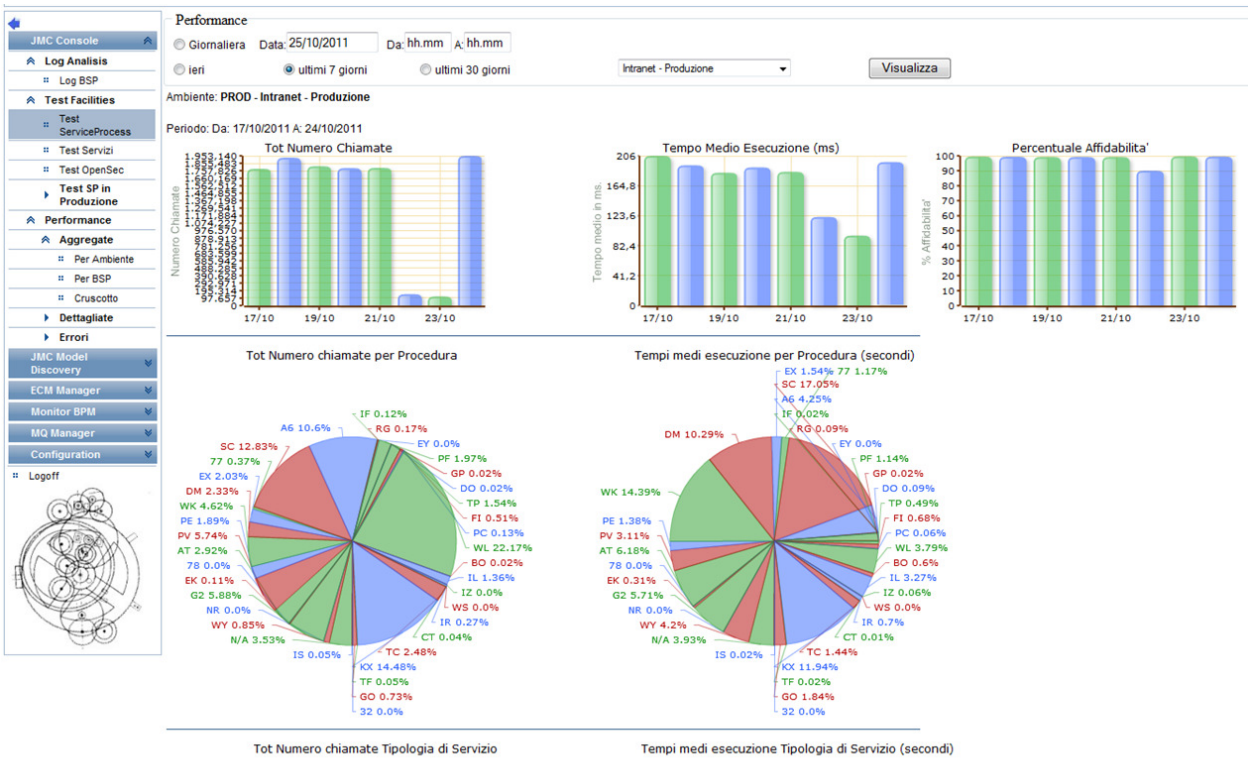

Figura 13 Monitoraggio su JMC Console

3.6 Gestione dei processi (JMC Process)

Il processo di sviluppo prevede, se necessario, una fase di gestione strutturata dei processi.

La figura seguente schematizza quelle che sono le fasi del ciclo di vita del processo nella concezione JMC, che ha come finalità la costante ottimizzazione.

Figura 14 Ciclo di vita di un processo

Per ciascuna delle fasi il framework mette a disposizione degli strumenti,contenuti in una piattaforma di gestione del processo denominata JMC Process:

- ANALYZE : la fase di analisi del processo si basa sull'utilizzo di Metastorm Provision
- DEPLOY : questa fase coincide essenzialmente con il disegno dei processi, basata su Metastorm Process Designer
- EXECUTE : la piattaforma esecutiva è rappresentata dal prodotto Metastorm BPM
- MONITOR : gli strumenti di monitoraggio sono di varia natura

3.6.1 Analisi dei processi

La fase di analisi del processo è implementata dal prodotto Metastorm Provision BPA (Business Process Analysis). Uno strumento BPA ha come scopo l'identificazione delle esigenze di un applicazione, individuando e descrivendo i processi necessari per la realizzazione del relativo business.

Uno dei principali obiettivi del BPA è, una volta individuati tutti i processi concorrenti alla realizzazione del business aziendale, riconoscere quali sono fondamentali e quali di sostegno.

Per ogni processo si individuano i criteri idonei per la valutazione della performance mediante la definizione e misurazione di parametri di efficienza e di efficacia degli stessi. L'obiettivo è la definizione di processi snelli, cioè processi ottimi senza sprechi. In tal senso la metodologia adottata può essere schematicamente sintetizzata come segue:

- razionalizzazione ed ottimizzazione di tutte le attività significative
- determinazione della sequenza logica delle attività necessaria alla realizzazione del processo, evitando soluzioni di continuità spazio-temporali (l'esecuzione delle varie attività deve fluire ad un passo ben definito e regolare)
- individuazione dei ruoli, cioè le figure che hanno le capacità di eseguire un determinato numero di attività componenti il processo stesso, conformemente agli standard di qualità attesi
- caratterizzazione puntuale di tutti gli input/output e dei relativi parametri qualitativi per consentire un attento monitoraggio delle performance
- pianificazione temporale e stima dei costi al fine di misurare l'efficienza del processo
- analisi dei feed-back cliente per la valutazione dell'efficacia del processo
- rilevazione tempestiva di tutte le situazioni di non conformità, cioè di deviazione del reale dall'ideale, ed individuazione delle azioni correttive eventualmente da intraprendere

Un processo snello deve quindi avere caratteristiche di semplicità, di compattezza, di economicità e di facilità d'uso. L'attenzione deve sempre essere maggiormente rivolta all'efficacia piuttosto che all'efficienza, che dovrebbe essere appena sufficiente per riuscire a generare l'output richiesto al processo.

3.6.2 Disegno dei processi

La fase di disegno del processo è implementata dal prodotto Metastorm Process Designer.

Il processo è costituito da un insieme di mappe, a loro volta formate dai seguenti elementi:

- Stage : stato di processo
- Action : transizione da uno stato ad un altro
- Folder : istanza del processo all'interno di una mappa
- Pratica : insieme di folder, possono essercene più d'uno perché ci potrebbero essere mappe in flussi paralleli
- Form : insieme di dati relativi al folder, legati ad azioni
- Role : ruolo operativo che abilita una lista di utenti ad effettuare operazioni sulla mappa

Ogni stage contiene dei folder, ovvero lo stage descrive lo stato dove il folder è fermo. Ogni modifica del folder prevede un'azione.

L'elaborazione BPM si basa su due fondamentali elementi: la TODO LIST, cioè l'elenco dei folder che sono in carico ad un determinato utente, e la WATCH LIST, cioè l'elenco dei folder su cui l'utente ha visibilità. Il calcolo delle liste viene fatto sulla base del ruolo assegnato all'utente ed allo stage in cui si trova il folder.

In fase di disegno del processo, è possibile modellare i seguenti tipi di stage:

- **User stage**: interazione utente, ovvero l'utente vede il folder nella to do list.
- **System stage**: i folder non si fermano sullo stage di sistema, e l'unico modo per uscire da questo stage sono le azioni condizionali. La prima azione che verifica la condizione viene eseguita e quindi comporta il cambio di stato
- **Subprocedure stage :** stage utile per creare flussi paralleli : un folder può sostare in un unico stage, quindi per gestire tali flussi devono essere create sotto mappe
- **Archive stage**: stage di fine corsa, ovvero quando il folder è concluso. Le pratiche sono fuori dal processo, quindi non c'e' una to do list associata.

E’ infine possibile modellare le seguenti tipologie di azione:

- **Manual Action**: azione che un utente può compiere
- **Timed Action**: azione che esegue una notifica, un sollecito o un aggiornamento quando un folder è sosta su uno stage per un certo tempo.
- **Conditional action**: caratterizzate da una condizione, tipicamente applicate ai system stage e user stage. La priorità è un attributo che prevede una sequenza di elaborazione delle regole
- **Flagged action**: legate ad un flag/evento, che ne determina l’esecuzione. La differenza rispetto ad una azione condizionale è il fatto che è associata ad un listener. Il flag è un segnale trigger di azione su un folder (listener) lanciato da un altro folder (raiser), e l’esecuzione dell’azione è asincrona rispetto al flag stesso, utilizzando un meccanismo di code.

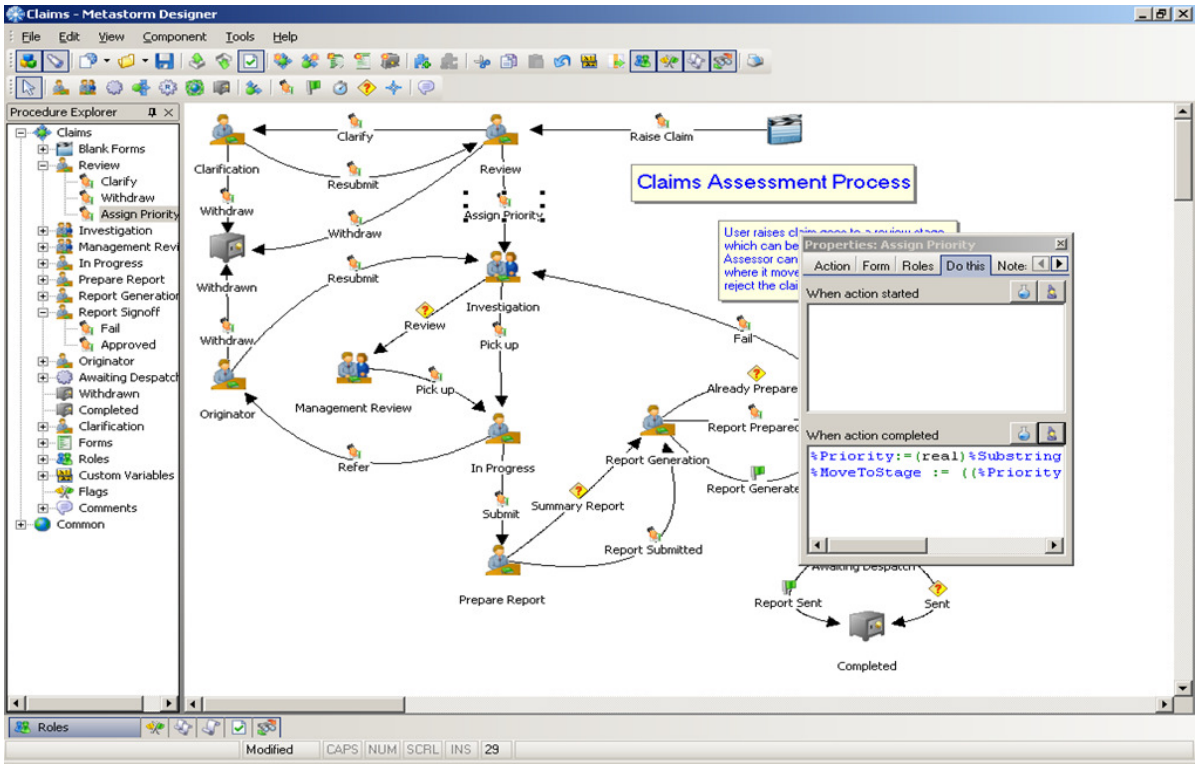

Figura 15 Disegno dei processi

3.6.3 Esecuzione dei processi

La figura rappresenta l'architettura di esecuzione di Metastorm BPM:

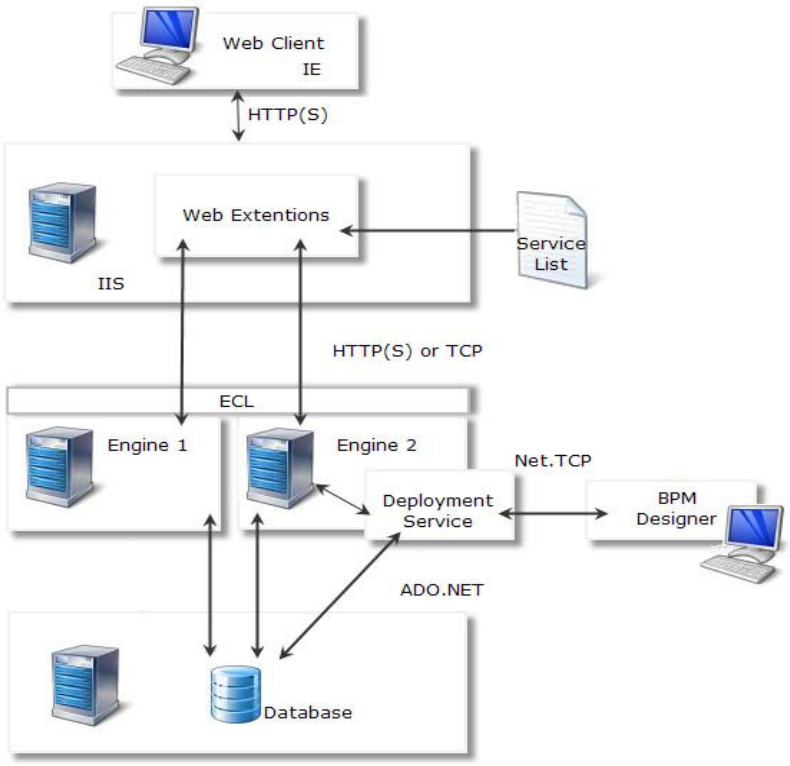

Figura 16 Architettura Metastorm BPM

Gli elementi costitutivi l'architettura sono:

- **Database** : è un database Oracle condiviso tra diversi engine, quindi gestisce il bilanciamento e la ridondanza
- **Process Engine** : è il motore che soddisfa le richieste del client web, autenticandone l'utenza ed eseguendo la logica di gestione dei processi
- **Lista dei processi** : è la lista dei servizi BPM che sono disponibili per l'esecuzione, e la mappatura degli engine che hanno la capacità di eseguirli
- **Web Extension** : è la componente applicativa che gestisce la comunicazione con l'engine
- **Web Client** : è il client che si interfaccia con la web extension e presenta i dati di ritorno all'utente

Nella declinazione JMC Process, la parte di Web Extension è rappresentata da un applicazione web (IIC) che utilizza la logica dei servizi, integrandosi con la suite di servizi presenti sul service bus.

3.6.4 Monitoraggio dei processi

Il monitoraggio dei processi viene svolto da diversi strumenti: dato che ad alto livello sono pubblicati i service process, su di essi è attivo il monitoraggio offerto dallo strumento standard JMC Console, che traccia i tempi di esecuzione e i log relativi ai servizi.

Data la complessità della piattaforma, è stato sviluppato in SEC uno strumento di diagnostica, denominato Log Monitor, che fornisce informazioni dettagliate a livello di motore BPM.

E' in corso di attivazione, infine, uno strumento di tipo BAM (Business Activity Monitor) che permette di eseguire un analisi di stampo funzionale sul processo, utilizzando metriche come il carico di lavoro svolto dagli utenti.

4 Conclusioni

La metodologia di sviluppo del software oggetto del presente studio è il frutto di una mentalità aperta all'evoluzione e tesa al continuo rinnovamento. La filosofia alla base, che ha poi indirizzato le scelte architetturali e i paradigmi adottati, è la ricerca di uniformità e di flessibilità.

Relativamente ai due principali pattern architetturali, che realizzano entrambe questa filosofia di base in maniera molto efficace, vanno comunque fatte considerazioni separate.

Adottare un architettura orientata ai servizi è stata una scelta premiante, vista che questa scelta è stata largamente condivisa dall'evoluzione delle architetture web che hanno dominato il mercato nell'ultimo decennio. E' stata particolarmente premiante per l'azienda SEC Servizi, considerato il contesto di business in cui opera: la centralizzazione offerta dalla piattaforma a servizi permette un governo più efficiente e garantisce una maggior continuità nell'erogazione del business.

D'altra parte, estendere il concetto di modellazione alla quasi totalità del processo di sviluppo, ha richiesto uno sforzo notevole e molti anni di lavoro. Ma i risultati ottenuti sono ancora più notevoli: il moltiplicarsi delle applicazioni che entrano a far parte della piattaforma applicativa di SEC Servizi, ciascuna con propri requisiti e con diverse esigenze tecnologiche, è assorbita dalla robusta struttura costruita, che è in grado di supportare in maniera agile e flessibile queste richieste. La standardizzazione garantita dalla generazione automatica del codice permette di aumentare la qualità del software e di facilitare enormemente la manutenzione e il rilevamento degli errori.

Un ultimo fattore, determinante per un azienda, è dato dalla diminuizione dei costi: l'automazione del processo produttivo, la separazione delle responsabilità, la maggior efficacia della comunicazione sono elementi che portano ad un miglior dimensionamento dei team di sviluppo, ad una diminuizione degli skill richiesti e a una riduzione dei tempi di sviluppo.