Appalto concorso sistema GAWS

Data: 29 ottobre 2007 Autore: Fabio Faieta Azienda Proponente: Soluzioni Software srl	**Appalto concorso per la realizzazione di un modello software per la generazione automatica di un componente di comunicazione basato su *Web Services* per applicazioni gestionali** **GAWS**

Art. 1) Oggetto dell'appalto

Il presente capitolato consiste nell'affidamento, a fornitori terzi, della realizzazione di un sistema *software* prototipale per la generazione automatica di componenti di comunicazione basati su *Web Services* per applicazioni gestionali.
Il termine "GAWS" denota il prodotto *software* in oggetto.
Il termine "committente" denota i proff. Vardanega e Conte, in qualità di delegati rappresentanti dell'azienda proponente che ha commissionato il lavoro.

Art. 2) Caratteristiche e requisiti tecnici minimi

Premessa

Il progetto qui illustrato si intitola:

> Generazione automatica componenti di comunicazione basati su *Web Services*.
>
> In seguito denominato "**GAWS**".

Tale progetto si colloca nel quadro di un progetto aziendale strategico più ampio, nel seguito denominato "progetto globale".

Obiettivi del progetto globale

Gli obiettivi del progetto globale sono i seguenti:

- automazione nello sviluppo di sistemi informatici "gestionali" transazionali basati su *database* relazionali
- definizione delle specifiche applicative attraverso la definizione di un apposito sistema di regole espresse in modo dichiarativo e/o tramite *script* controllabili e manipolabili dall'utente
- generazione automatica del codice applicativo a partire dalle regole di cui sopra mediante l'utilizzo di opportuni modelli[1]
- utilizzo di modelli di generazione *multi-language*, capaci di generare codice sorgente in svariati linguaggi di programmazione e di aderire eventuali altre caratteristiche desiderate.

Un sistema dotato di tali caratteristiche otterrebbe i seguenti immediati benefici:

- maggiore **produttività** (il codice viene generato automaticamente e una regola contenuta in una sola riga di testo sostituisce decine o centinaia di righe di codice)
- maggiore **qualità** (ancora: il codice viene generato automaticamente, esonerando lo sviluppatore dal ricordare quando una certa regola di programmazione deve scattare e in quale sequenza all'interno di una particolare transazione)
- migliore **manutenzione** applicativa (poiché essa si sposta al livello delle regole e non più del codice scritto manualmente, e quindi per definizione auto-documentante e slegato dallo sviluppatore)

[1] I modelli sono porzioni di codice sorgente del linguaggio *target*, gestiti con un linguaggio proprietario di lettura e interpretazione delle regole definite nel *repository*. Si tratta di *file* di testo che contengono i metacomandi di traduzione (e nello specifico hanno estensione ETM).

- immediato **adeguamento** alle novità tecnologiche (l'aggiornamento periodico dei modelli consente la rigenerazione totale del codice con nuove caratteristiche applicative (p.es. gestione accessi) o tecniche (p.es. interfaccia verso un nuovo DBMS).

Lo schema riportato in Figura 1 indica lo schema a blocchi di funzionamento del progetto globale.

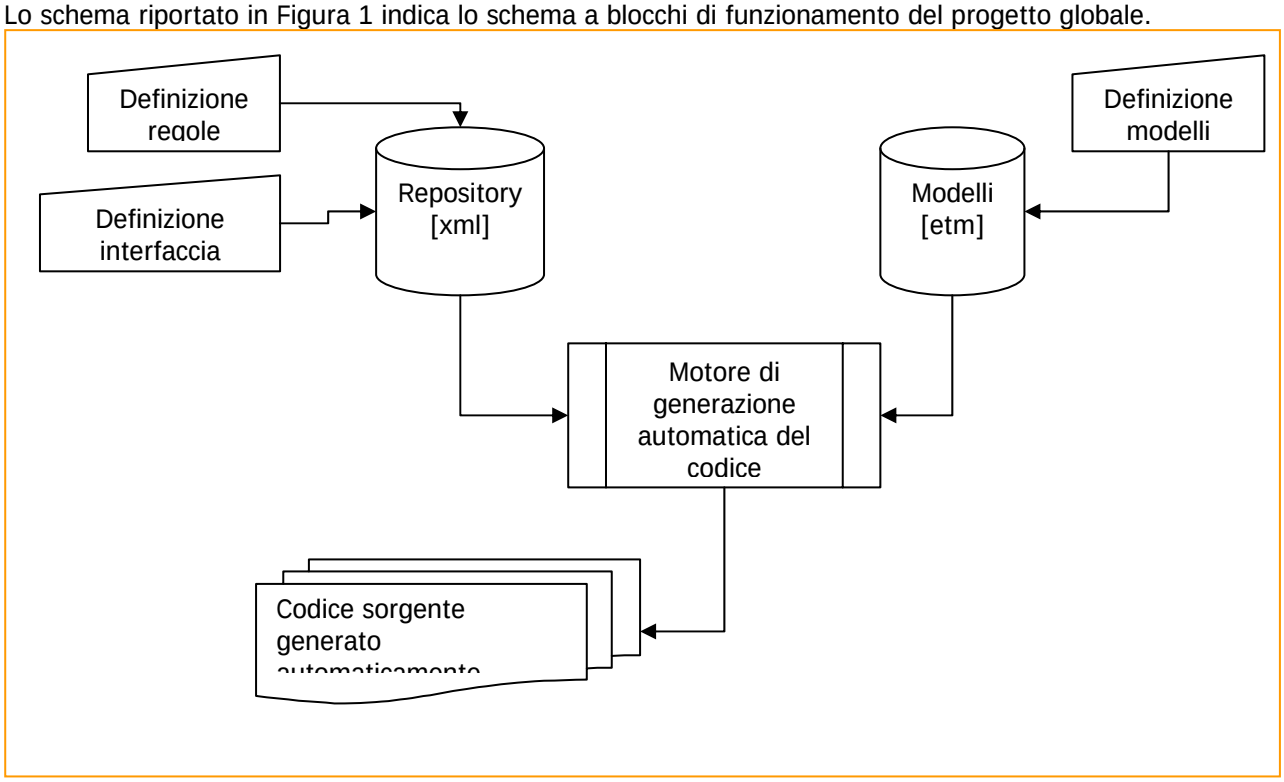

Figura 1 : Schema del sistema globale. Generazione del codice.

L'obiettivo del progetto globale è quello di automatizzare la produzione di applicazione di natura gestionale.

Stato dell'arte

Allo stato attuale il progetto globale consente la generazione di applicazioni gestionali di tipo *client-server* come riportato in Figura 2. Nella figura è evidenziato che a partire da particolari modelli di generazione specifici per ogni strato applicativo viene generato il codice relativo. Il risultato finale è un'applicazione *client-server* (indicata in figura all'interno del riquadro con sfondo colorato).

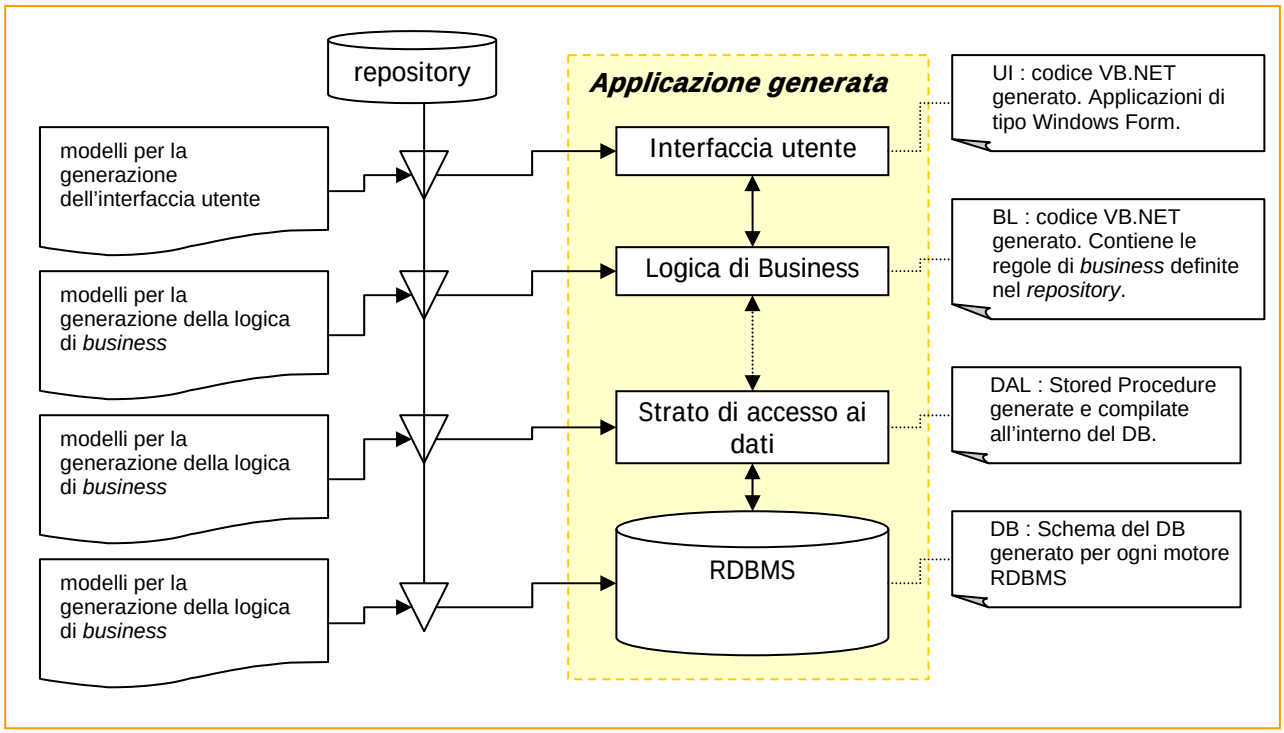

Figura 2 : Stato attuale schema di generazione ("as is")

Obiettivo del progetto specifico

Partendo dagli obiettivi del progetto globale come precedentemente illustrati ed alla luce dello stato attuale, l'obiettivo primario dello specifico progetto è illustrato nella Figura 3.

Le aree di intervento sono quelle in azzurro in Figura 3 e riguardano la comunicazione tra il *client* e i *Web Services*.

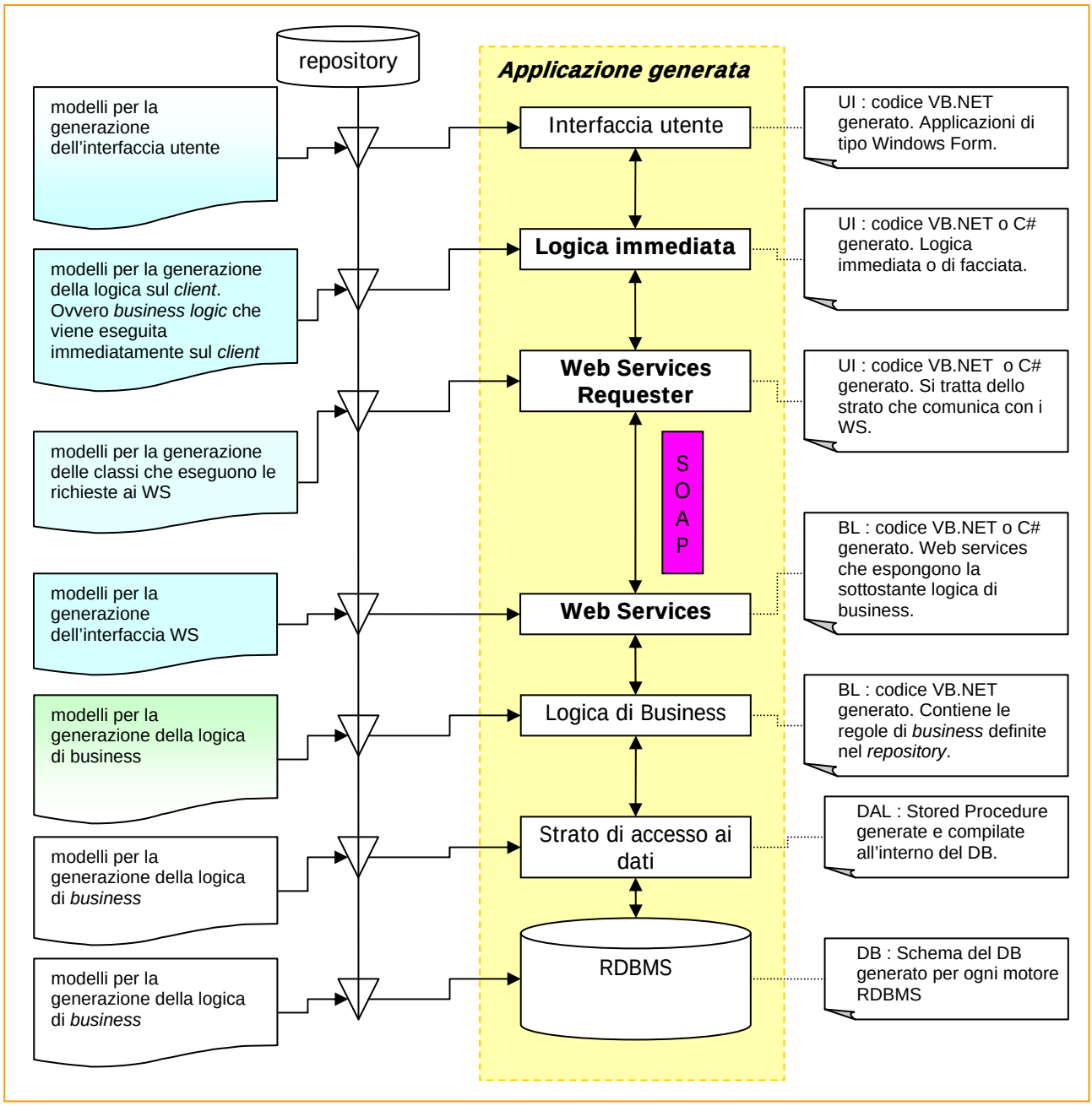

Figura 3 : Obiettivi del progetto GAWS. ("to be")

Gli interventi richiesti si possono riassumere nel seguente modo (osservare gli elementi in azzurro della Figura 3) :

1. Modifica alla *business logic*
   - 1.1 Comprensione della logica di *business* in un ambiente di sviluppo dichiarativo.
   - 1.2 Modifica e sezionamento della logica di *business* per ritagliare le funzionalità esponibili ai *Web Services*.
   - 1.3 Realizzazione delle classi che implementano la *business logic* funzionante attraverso la modifica delle classi esistenti.
   - 1.4 Realizzazione del modello per la generazione automatica delle classi viste al punto 1.3.

2. Realizzazione dei WS.
   - 2.1 Definizione del messaggio basato su protocollo SOAP utilizzando il linguaggio WSDL.
   - 2.2 Realizzazione dei *Web Services* con l'obiettivo di esporre la logica sottostante
   - 2.3 Realizzazione del modello per la generazione automatica del codice realizzato al punto 2.2

3. Modifica all'interfaccia utente:
   3.1 Modifica del modello di generazione dell'interfaccia utente per consentire la chiamata attraverso *Web Service*
   3.2 Modifiche alla logica di presentation per limitare / evitare le chiamate al *server* ove possibile.

4. Introduzione della *business logic* immediata. codice di supporto alla logica di *business* che viene eseguita sul *client* per limitare il traffico di rete e il numero di *round trip*[2] verso il db. Di fatto l'introduzione della logica immediata è semplicemente funzionale alle prestazioni generali del sistema e resta un paradigma accettabile solo perché operiamo in un contesto di codice "generato" e non scritto a mano. In ogni caso l'introduzione di questo strato è strettamente sottoposto ad un'attenta valutazione (come indicato dal primo punto di dettaglio).
   4.1 Valutazione della fattibilità e delle criticità inerenti l'introduzione della logica immediata.
   4.2 Realizzazione delle classi che offrono supporto alla logica immediata.
   4.3 Realizzazione dei modelli per la generazione automatica del codice relativo alle classi al punto 4.2.

5. Realizzazione dei modelli per la generazione delle classi che interrogano i *Web Services* dall'interfaccia utente
   5.1 Realizzazione delle classi che interrogano i *Web Services*
   5.2 Realizzazione del modello che genera automaticamente le classi indicate al punto 5.1

In generale, in tutte le fasi del progetto, gli studenti avranno accesso a una linea diretta via e-mail con un referente designato per tutte le necessità e sarà poi eventualmente valutato nel merito, caso per caso, il tipo di intervento.

Verranno poi definite riunioni con cadenza almeno quindicinale per la verifica dello stato avanzamento lavori e per l'assunzione di decisioni fondamentali per il prosieguo del progetto (in particolare nella fare di prototipo iniziale).

Gli studenti avranno a disposizione il prodotto alla data (*repository*, *front-end*, modelli realizzati, codice già generato), corredato da tutta la documentazione disponibile e avranno facoltà di libera installazione sui sistemi che riterranno opportuni; è possibile anche prevedere un corso iniziale diretto di due giornate.

Art. 3) Documentazione

Con la consegna del sistema GAWS, il fornitore assumerà anche l'onere di fornire i manuali e ogni altra documentazione tecnica idonea per assicurare il soddisfacente funzionamento e manutenzione del prodotto, compresi i sorgenti del codice commentato secondo le norme.

La documentazione, in particolare, comprende i manuali e le istruzioni stabilite dall'impresa fornitrice, con i dettagli concernenti le caratteristiche dei programmi di base e le procedure per il loro utilizzo (avviamento, fermi, interventi per guasti, operazioni consentite in fase di elaborazione, ecc.).

Art. 4) Rinvio

Per tutto quanto non previsto nel presente capitolato, sono applicabili le vigenti disposizioni di legge.

[2] Con il termine "*round trip*" si indicano tutte le comunicazioni andata e ritorno dal *client* verso il *server*.