Verbale esterno 2018-03-29

Gruppo JurassicSWE · Progetto IronWorks

JurassicSWE@gmail.com

Informazioni sul documento

Redazione	Leo Moz
Verifica	Lidia Alecci
Approvazione	Gianluca Travasci
Uso	Esterno
Distribuzione	Prof. Tullio Vardanega Prof. Riccardo Cardin Gruppo JurassicSWE

Sommario

Tale documento riassume l'incontro del 2018-03-29 tra il gruppo JurassicSWE ed il proponente Zucchetti S.p.A..

1 Informazioni

1.1 Informazioni generali

- **Data**: 2018-03-29;
- **Luogo**: Via Giovanni Cittadella, 7, Padova;
- **Ora inizio**: 16:00;
- **Ora fine**: 17:00;

1.2 Partecipanti

- **Gruppo JurassicSWE**: Lidia Alecci, Marco Masiero, Leo Moz, Gianluca Travasci;
- **Zucchetti S.p.A.**: Gregorio Piccoli;
- **Altri**: nessuno;

1.3 Argomenti

L'incontro è servito per discutere la fase preliminare dell'*Analisi dei Requisiti*. Lo scopo principale è stato quello di comprendere quali componenti del prodotto siano essenziali al Proponente e quali invece potrebbero essere aree d'espansione a scopo sperimentale e conoscitivo. Si è colta l'occasione anche per porre alcune domande circa dettagli più specifici e dubbi generici emersi nel corso del colloquio.

2 Domande e Risposte

1. **Qual è stata la genesi del progetto proposto?**
   Tutto è partito riflettendo sulla possibilità di generare file di configurazione per un ORM_g esistente (es. $Hibernate_g$); poi l'idea è evoluta verso la richiesta di realizzare un $editor_g$ per $robustness\ diagram_g$ con successiva generazione del codice di classi $Java_g$ atte a descrivere le $entity_g$.
   Per ogni entity è necessario conoscere i campi dati per poter associare: o la generazione di $script_g$, per la creazione e manutenzione di un $database\ relazionale_g$, o semplicemente i file di configurazione per un ORM.

2. **Quali sono le richieste minime e le funzionalità strettamente necessarie?**
   Si richiede un sistema di disegno, molto semplice e basilare che preveda i quattro elementi del robustness diagram ($attore_g$, $boundary_g$, $control_g$ ed $entity_g$) collegati tra loro tramite frecce. Ci sono varie librerie che aiutano in simili implementazioni, ad esempio $JointJS_g$.
   Può essere utilizzata qualunque libreria, $framework_g$ o altro che possa aiutare il gruppo durante lo sviluppo del prodotto; cercando comunque di sviluppare un'interfaccia moderna ed accattivante. Una volta soddisfatte le richieste minime, sarebbe bello riflettere insieme per scoprire ambiti interessanti da approfondire.

3. **Avete specifiche direttive sul sistema il disegno? Per esempio $PlantUML_g$ genera un disegno dal testo scritto dall'utente, mentre altri sistemi permettono il "trascinamento" e il posizionamento degli elementi.**
   No. Preferiremo l'approccio tipico di tutti gli editor attuali (trascinamento e inserimento di elementi); l'approccio di PlantUML limita la libertà di disposizione del diagramma. Potete decidere di usare entrambi gli approcci in base alla funzionalità: il trascinamento per l'inserimento degli elementi, e quello descrittivo per la definizione dei campi dati delle entity (in alternativa alla più classica interfaccia); $UMLet_g$ segue quest'ultima filosofia per i diagrammi delle classi.

4. **Per quanto riguarda la generazione del codice delle classi Java e degli script per il $database_g$, avete richieste in particolare?**
   Anche se indicati come obbligatori nel capitolato, riflettendoci stiamo viran-

do più sul produrre i file di configurazione per l'ORM, quindi la generazione di codice passa un po' in secondo piano. Se si ottiene il file XML di configurazione da Hibernate non è necessaria né la produzione del codice Java né degli script per il mantenimento del database relazionale, potremmo considerare "codice" i file di configurazione.

5. **È necessario poter esportare/importare il diagramma? Se si, ci sono preferenze sul formato?**
   Sarebbe gradito. La forma non ha importanza, potrebbe essere un *JSON*$_g$ visualizzato a schermo e che quindi va manualmente copiato, o un file da scaricare.
   Sarebbe interessante poter riprendere il lavoro, ancora in corso, direttamente all'accesso. Tuttavia, salvando i dati sul client si risulterebbe poi legati a quella specifica macchina, invece, salvandoli sul server si potrebbe riprendere il lavoro da qualunque terminale.

6. **Richiedete necessariamente il salvataggio del diagramma server-side?**
   Sarebbe gradito, ma siamo aperti ad altre proposte.

7. **Sarebbe possibile sviluppare il prodotto serverless, ovvero con un server che fornisca i file necessari a disegnare e poi produrre il codice tutto client-side?**
   A primo acchito si potrebbero riscontrare problemi ad uscire dalla *sandbox*$_g$ del browser (ad esempio per salvare i dati). Inoltre, data la natura didattica del progetto, forse sarebbe meglio sviluppare almeno un paio di procedure server: una che accetti una POST dei dati inviati dal client, ed eventualmente li salvi da qualche parte, ed una che legga quanto salvato.

8. **Avete intenzione di integrare il prodotto nei vostri sistemi (così come visti in occasione della precedente riunione)?**
   No, chiedere lo sviluppo di un *plug-in*$_g$ per i nostri sistemi avrebbe comportato l'imposizione di vincoli troppo stringenti.
   Siamo più interessati ad una dimostrazione di fattibilità con le tecnologie indicate, e ad esplorare funzionalità e problematiche aggiuntive che potrebbero emergere e rivelarsi interessanti. Ad esempio:

   - la necessità di inserire alcuni dati aggiuntivi per garantire la validazione dei dati richiesti all'utente;

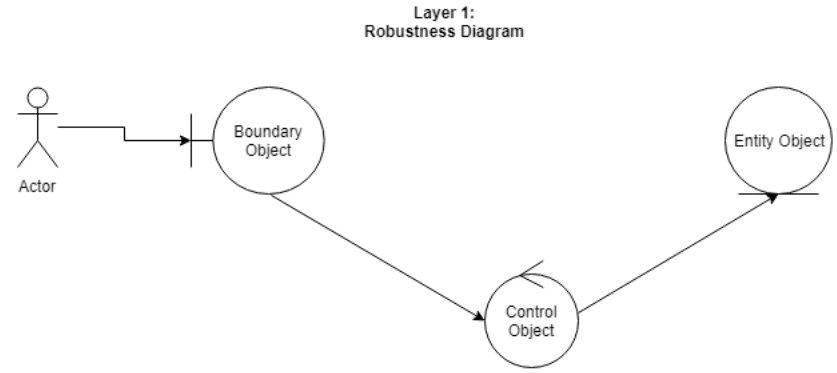
- per restare all'interno del prodotto richiesto, data una entity con i suoi campi dati, l'ORM genera i file di configurazione "grezzi", sarebbe interessante poter opzionalmente specificare altre informazioni (ad esempio sulle tabelle o sugli indici) che conducano alla produzione di file di configurazione più efficienti, tali informazioni sarebbero appunto "in più" e non descrivono strettamente l'entity;
- approfondire il concetto di entity, a prima vista si potrebbe pensare che ad una entity corrisponda una tabella del database, ma potrebbe essere descritta in modo più ricco (ad esempio tramite il formalismo di *Wernier-Orr*$_g$, o specificando alcune cardinalità) e quindi non corrispondere più, nel database, ad una sola tabella.

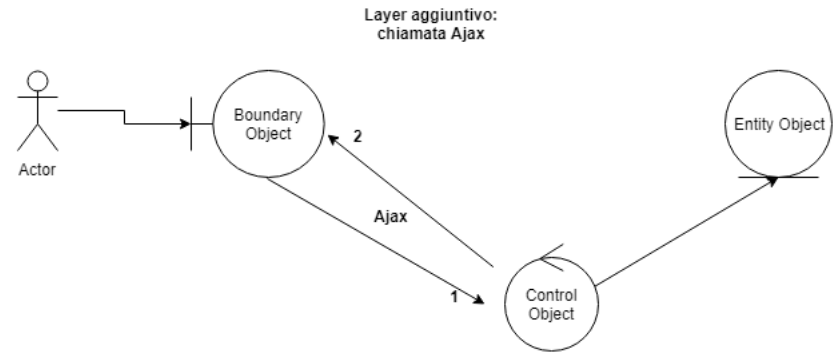
9. **Quale freccia prevista dall'UML collega gli elementi del robustness diagram?**
   Il robustness diagram vero e proprio prevede una freccia piena (non tratteggiata) come quella dell'associazione nel class diagram.

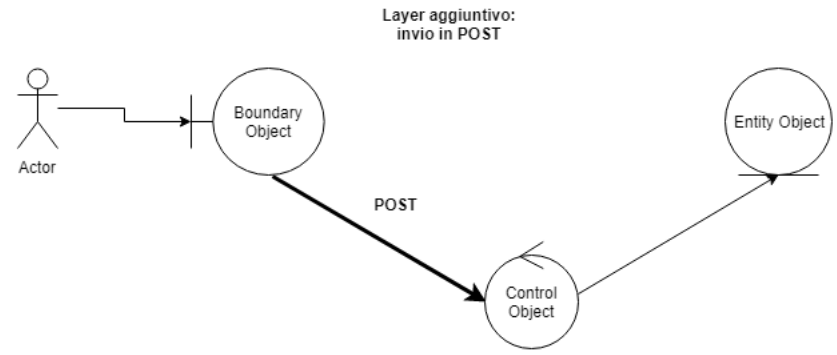
10. **In generale quali best practice andrebbero seguite quando si estende un diagramma UML?**
    Un buon approccio per l'estensione dei diagrammi è riconducibile al concetto di *layer*$_g$ dei software grafici. Il concetto per cui vari strati informativi sono attivabili e disattivabili per fornire diverse "viste" del diagramma può essere un approccio interessante. Inoltre per ogni layer si potrebbe aggiungere contenuto informativo diverso, in base al livello preso in esame.
    Si potrebbe pensare ad un "layer" di base con il robustness diagram semplice così come proposto da Ivar Jacobson, e poi dei layer aggiuntivi soprastanti che aggiungono funzionalità o informazioni. Come ad esempio: frecce che vanno e vengono tra boundary e control per descrivere chiamate *Ajax*$_g$ (magari anche numerate ad indicarne la successione) o una freccia singola più marcata ad indicare una trasmissione dati in POST, che blocca la boundary fino a computazione conclusa; rimuovendo questo layer il diagramma è quello di base, con una freccia singola, attivandolo invece si aggiunge informazione che induce poi ulteriore potenziale. È evidente che se in un control entra una freccia di tipo "POST", ci dev'essere una POST uscente verso un'altra boundary, altrimenti il client resterà bloccato in eterno, queste informazioni non sono deducibili dal layer di base, ma lo sono nel momento in cui si attiva quello soprastante.

Riassumendo esempi di layer potrebbero essere:

- un layer del diagramma vero e proprio;
- un layer delle comunicazioni (quanto discusso nell'esempio esteso qui sopra, in prestito dal communication diagram);
- un layer che identifichi a colpo d'occhio la parte client e la parte server dell'applicazione modellata (informazione mutuata dal deployment diagram), tramite due aree tenuemente colorate o mediante gli "scatolotti" previsti dal deployment diagram (sconsigliati, perché poi gli elementi aumentano e fuoriescono dagli "scatolotti");
- un layer che specifichi un elenco di controlli da fare sui dati passati.


Layer 1:
Robustness Diagram
Actor
Boundary Object
Entity Object
Control Object


Layer aggiuntivo:
chiamata Ajax
Actor
Boundary Object
2
Ajax
1
Entity Object
Control Object


Layer aggiuntivo:
invio in POST
Actor
Boundary Object
POST
Entity Object
Control Object

11. **Se strutturando il diagramma a layer insorgessero delle difficoltà nel mantenere coerenza e collegamenti tra le parti in presenza di trascinamenti o modifiche?**
    Sarebbe gradito che il layer del diagramma base mantenga la sua coerenza automaticamente, ad esempio spostando per trascinamento un elemento non si perda il collegamento tramite frecce. Per tutti gli altri layer la coerenza può essere eventualmente demandata all'utilizzatore, sarà lui in caso a cancellare o spostare elementi residui che risultino fuori posto dopo tali azioni, è meno importante insomma. Si noti che "layer" è solo un concetto, non c'è l'obbligo di disegnare elementi aggiuntivi effettivamente su un layer separato, molte cose potrebbero essere specificate mediante proprietà aggiuntive degli elementi; sarà poi compito dell'utente attivarne la visualizzazione o meno, tramite dei filtri.

12. **Quindi dobbiamo fare tutto ciò di cui stiamo discutendo?**
    No. Le richieste minime sono appunto l'editor di robustness diagram. Tutto il resto sorge dall'esplorazione del prodotto. Le idee cardine sono la "*singleness*$_g$" (tradotta con "continuità"), la riduzione della distanza che c'è tra un diagramma UML e l'altro, rendere effettivamente utile il diagramma durante le varie fasi realizzative (dall'analisi alla progettazione), e permettere l'introduzione di più "semantica" possibile. Ogni proposta in tal direzione (ad esempio colorare gli elementi, o i layer nominati precedentemente) è ben accetta.

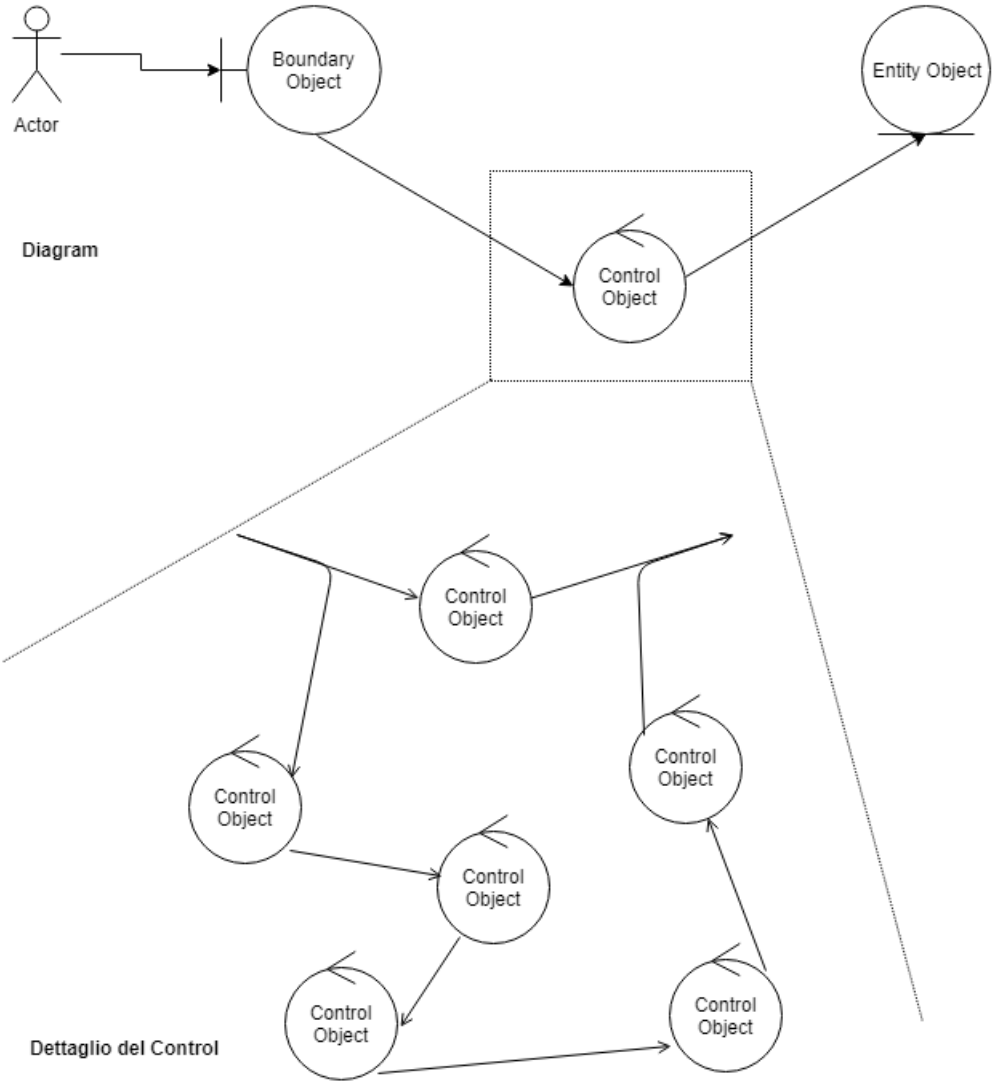
13. **Qualche consiglio per perseguire questa "continuità"?**
    Oltre ai già citati layer si potrebbe ad esempio pensare a:

    - zoom: un po' come succede nel software di presentazione *Prezi*$_g$: ad alto livello si vedono le varie bolle o i vari elementi, poi effettuando uno zoom si entra nel dettaglio di ognuna di queste bolle. Nel caso concreto, ad esempio una entity è rappresentata dal suo simbolo, quando viene effettuato uno zoom sufficiente compaiono i suoi dettagli (campi dati, proprietà); in alternativa i dettagli possono essere visualizzati in una schermata o una sezione dedicata nel momento in cui viene selezionata l'entity.

    - *folding*$_g$: ripiegamento, come negli editor testuali che permettono di "comprimere" tutto il corpo di una funzione qui si potrebbe pensare di "ripiegare" o comprimere un insieme di elementi su cui si è finito di lavo-

rare o che in questo momento non sono di specifico interesse o un'intera parte marcata in uno specifico modo (ad esempio "parte server").

- raggruppamento: ci si è resi conto che spesso un control è composto, o comunque si serve, di svariati control: potrebbe essere interessante che ad alto livello la zona sia rappresentata come un solo elemento control ma quando si effettua lo zoom o si entra nel dettaglio questo control sia in realtà una composizione di svariati (sub)control (boundary ed entity tendenzialmente non presentano questa divisione in sottoparti).



14. **Può essere utile una versione mobile dell'applicazione?**
    No, non pare ragionevole che un'applicazione di sviluppo venga usata su dispositivi mobili.

15. **Avete particolari vincoli o richieste da un punto di vista prestazionale?**
    No, anche perché sarebbero di difficile misurazione e verificabilità. A titolo informativo tenete però presente che il prodotto dev'essere ragionevolmente utilizzabile, quindi, ad esempio, non può presentare vistosi rallentamenti in un diagramma con meno di 10 elementi.

3 Consigli e suggerimenti del Proponente

In questa sezione trovano spazio affermazioni o consigli che il Proponente fornisce ai presenti non necessariamente in seguito ad una domanda specifica.

1. Dividetevi bene i compiti e fate in modo che tutti i componenti del gruppo lavorino e svolgano una parte del progetto, è spiacevole per voi, per noi e per il Professore individuare o avere a che fare con studenti poco motivati o che non si impegnano.
2. Nome del gruppo "Jurassic SWE"? Avete pensato di cambiarlo?

4 Riepilogo delle decisioni

Codice	Descrizione
VE_2018-03-29.1	La richiesta minima è fondamentalmente un editor per robustness diagram e una parte di generazione di codice
VE_2018-03-29.2	Il codice generato può essere quello inizialmente previsto dal capitolato o semplicemente i file di configurazione per l'ORM scelto
VE_2018-03-29.3	Rispettando i vincoli tecnologici, possono essere usate tutte le librerie o framework che risultino utili, consigliato JointJS
VE_2018-03-29.4	Preferibile avere una parte client ed una parte server (anche minimale)
VE_2018-03-29.5	Il Fornitore è libero di individuare aree di espansione che risultino interessanti o di valutare lo sviluppo di alcune di quelle discusse
VE_2018-03-29.6	Preferibile avere un editor di testo con inserimento per trascinamento piuttosto che generare disegno da testo
VE_2018-03-29.7	Si consideri la possibilità di fornire funzionalità di import/export o ancora meglio il salvataggio dello stato del disegno lato server
VE_2018-03-29.8	Gli elementi di un robustness diagram sono collegati da una freccia continua, come quella dell'associazione nel class diagram
VE_2018-03-29.9	Nel caso si aggiungano funzionalità, si consideri di implementarle mediante un meccanismo a "layer"
VE_2018-03-29.10	Il diagramma di base deve poter essere modificato mantenendo la sua integrità
VE_2018-03-29.11	Non è strettamente necessario garantire l'integrità di eventuali layer successivi
VE_2018-03-29.12	Non sembra ragionevole la presenza di una versione mobile dell'applicazione
VE_2018-03-29.13	Non sono previsti particolari vincoli prestazionali

Tabella 1: Decisioni prese nella riunione esterna del 2018-03-29