IronWorks

Utility per la Costruzione di Software Robusto

Verbale esterno 2018/04/03

Versione	-
Redattori	Mirko Gibin
Verificatori	Sharon Della Libera
Responsabili	Francesco Sacchetto
Uso	Esterno
Distribuzione	Gruppo Swear on Code
	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Gregorio Piccoli, Zucchetti S.p.A.

Descrizione

Verbale di incontro per il progetto **IronWorks** tra i componenti del gruppo Swear on Code ed il proponente Gregorio Piccoli, Zucchetti S.p.A.

1 Informazioni Generali

1.1 Informazioni Incontro

- **Data:** 2018/04/03;
- **Luogo:** Padova, via Cittadella 7;
- **Ora d'inizio:** 16:00;
- **Ora di fine:** 17:00;
- **Partecipanti del gruppo:**
  - Anna Poletti;
  - Antonio Moz;
  - Francesco Sacchetto;
  - Mirko Gibin;
  - Sharon Della Libera;
  - Stefano Nordio.
- **Partecipanti esterni:**
  - Gregorio Piccoli, Zucchetti S.p.A.

1.2 Ambiguità

Al fine di dipanare qualsiasi dubbio o ambiguità relativa al linguaggio impiegato nel documento viene fornito il *Glossario v1.0.0*, documento contenente la definizione di tutti i termini scritti in corsivo e marcati con una 'G' pedice.

1.3 Riferimenti

1.3.1 Informativi

- *Glossario v1.0.0*.

2 Ordine del Giorno

Chiarimenti inerenti il progetto **IronWorks** e la sua implementazione. Il *proponente*$_G$ fornisce idee e risposte circa le seguenti domande:

- Per quanto riguarda l'*editor*$_G$ dei *diagrammi di robustezza*$_G$, si preferisce l'implementazione del *drag and drop*$_G$, *point and click*$_G$ o qualche altra particolarità?
- Cosa si intende per codice di manutenzione delle tabelle?
- Per importare, salvare od esportare un progetto definiamo un *file*$_G$ personalizzato per l'*editor*$_G$ dei diagrammi?
- Per quanto riguarda il punto obbligatorio n.8, non è ancora chiaro cosa si intende per *regole per sviluppare manualmente il codice degli oggetti boundary*$_G$ *e control*$_G$. Potrebbe spiegarci meglio cosa si intende?

2.1 Per quanto riguarda l'*editor*$_G$ dei *diagrammi di robustezza*$_G$, si preferisce l'implementazione del *drag and drop*$_G$, *point and click*$_G$ o qualche altra particolarità?

Il *drag and drop*$_G$ è una valida soluzione, ma non c'è nessuna particolare richiesta in merito.

Interessante potrebbe essere lo sviluppo del *diagramma di robustezza*$_G$ a *layer*$_G$: man mano che si ottengono informazioni aggiuntive, queste si aggiungono al diagramma a strati. In questo modo non si modifica la base e si rispetta lo *standard*$_G$ previsto, ma si arricchisce l'informazione che dà.

Gli oggetti di tipo *control*$_G$ potrebbero prevedere una tecnica di *zoom-in*$_G$/*zoom-out*$_G$ per osservarli in modo più o meno dettagliato; ciò potrebbe tornare utile qualora fossero composti da più parti che si ritiene necessario distinguere.
Gli oggetti di tipo *entity*$_G$ potrebbero essere arricchiti con i diagrammi di *Warnier-Orr*$_G$, in modo che modifiche ai campi dati comportino l'aggiornamento solo di questi ultimi. Tramite questi diagrammi è possibile mostrare la gerarchia delle tabelle risultanti, e con la tecnica del *folding*$_G$ nascondere o mostrare le informazioni.

Tecniche simili permettono la realizzazione del *diagramma di robustezza*$_G$ a partire direttamente dai *casi d'uso*$_G$, e arricchendolo con le informazioni disposte a strati si rende possibile la creazione di un collegamento con il *diagramma delle classi*$_G$.

Eventuali modifiche future alle informazioni che si possiedono porterebbero all'aggiornamento dei *layer*$_G$, senza toccare il diagramma alla base. Ciò permetterebbe una progettazione iniziale migliore.

Potrebbe essere interessante anche giocare con i colori delle informazioni, ad esempio distinguendo tra *server*$_G$ e *client*$_G$.

2.2 Cosa si intende per codice di manutenzione delle tabelle?

Lo scopo è quello di creare delle tabelle contenenti le informazioni delle classi. Se in un futuro fosse necessario un aggiornamento ai loro campi dati, è auspicabile poter modificare le tabelle del *database*$_G$ creato in modo poco invasivo, senza dover alterare i dati presenti, e solo se necessario: prima di creare una nuova tabella o aggiungere un campo dati, è bene controllare che non esistano già.
Qualora dovessi aggiungere un nuovo campo dati, a livello di *diagramma di robustezza*$_G$ sarebbe sufficiente modificare il relativo diagramma di *Warnier-Orr*$_G$, e *Hibernate*$_G$ può aiutare molto nella fase successiva: è sufficiente modificare il relativo *file*$_G$ *XML*$_G$.

2.3 Per importare, salvare od esportare un progetto definiamo un *file*$_G$ personalizzato per l'*editor*$_G$ dei diagrammi?

Si, il *file*$_G$ può essere creato come meglio si crede e deve poter salvare lo stato del *diagramma di robustezza*$_G$ in un dato momento e ricaricarlo quando necessario.

2.4 Non è ancora chiaro cosa si intende per *regole per sviluppare manualmente il codice degli oggetti boundary*$_G$ *e control*$_G$. Potrebbe spiegarci meglio?

Le classi create devono essere utilizzate in modo semplice dagli *utenti*$_G$: devono essere presenti i metodi di get e set necessari, e per eventuali altri metodi deve essere chiaro il loro funzionamento.
Una classe ben definita può essere usata in modo semplice all'interno degli oggetti *boundary*$_G$ e *control*$_G$ che devono essere scritti a mano.

3 Tracciamento decisioni

ID	Decisione
VE_20180403.1	Valutazione sovrapposizione delle informazioni in *layer*$_G$
VE_20180403.2	Valutazione distinzione delle informazioni tramite colori
VE_20180403.3	Valutazione di *Hibernate*$_G$ come strumento
VE_20180403.4	Valutazione di *Warnier-Orr*$_G$ come diagramma di rappresentazione delle tabelle
VE_20180403.5	Valutazione di tecniche di *zoom-in*$_G$/*zoom-out*$_G$ e *folding*$_G$
VE_20180403.6	*File*$_G$ personalizzati per salvare, esportare e importare diagrammi
VE_20180403.7	Possibile implementazione di *drag and drop*$_G$

Tabella 1: Decisioni riunione esterna del 2018/04/03