Verbali Esterni

Gruppo SWEetBIT — ***Progetto SWEDesigner***

Informazioni sul documento

Versione	1.0.0
Redazione	Sebastiano Bertolin
Verifica	Da inserire
Approvazione	Da inserire
Uso	Esterno
Distribuzione	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Gruppo SWEetBIT

Descrizione

Questo documento traccia i verbali di tutte le riuioni esterne, con il committente od il proponente, svolte dal gruppo SWEetBIT.

Indice

1 Riunione 1

1.1 Informazioni sulla riunione

- **Data:** 23/02/2017
- **Luogo:** Zucchetti - sede di Padova , via Giovanni Cittadella 7
- **Ora:** 16:30
- **Durata:** 90 min
- **Argomento:** Chiarimenti dei requisiti sul *capitolato*$_G$
- **Partecipanti Interni:** Davide Santimaria - Fabio Massignan - Gianmarco Salmistraro - Malick Bodian - Salvatore Pilò - Sebastiano Bertolin
- **Partecipanti Esterni:** Gregorio Piccoli

1.2 Domande e Risposte

- **Come deve essere la qualità del *codice*$_G$ generato dai *diagrammi*$_G$ *UML*$_G$?**
  Per rispondere al meglio alla domanda, ci si vuole da prima focalizzare su un altro punto, ovvero il perché sia stato suggerito il tema dei giochi da tavolo su questo *capitolato*$_G$.
  Ebbene nel *dominio*$_G$ dei giochi da tavolo, la ripetitività è molto elevata; Basti pensare al gioco della dama o degli scacchi, entrambi condividono la stessa scacchiera di gioco, anche il Monopoli ha una scacchiera, che si diversifica per colori e tipi di caselle , ma è pur sempre una scacchiera. Focalizzarsi su di un specifico ambito, in questo caso i giochi da tavolo, aiuta a migliorare la qualità del *codice*$_G$ (vista anche la possibilità del suo riutilizzo) , in quanto se si decidesse di rappresentare ogni contesto, risulterebbe difficoltoso generare del *codice*$_G$ che si adatti al meglio in ogni situazione.
  Sfruttare il fattore del riutilizzo sicuramente aiuta, ma la scelta che incide, è il modo in cui si sceglie di disporre i *diagrammi*$_G$, ovvero se visualizzarli per *layer*$_G$, suddividendoli per categorie ad esempio, oppure disporre il tutto su un unico *layer*$_G$ ed utilizzare un sistema di evidenziazione o *zoom*$_G$ od altre tecniche. Tale scelta è importante perchè riflette quanto più complesso e specifico possa essere il progetto che l'*utente*$_G$ vuole realizzare, e da questo ne deriva anche la qualità del *codice*$_G$. La vera difficoltà sta nel generare *codice*$_G$ dai *diagrammi*$_G$ che rappresenteranno i *metodi*$_G$, per essi si suggerisce un approccio utilizzando l'*activity diagram*$_G$.

- **Nel nostro *designer*$_G$ dobbiamo includere dei *template*$_G$?**
  Si, a patto che venga scelto un *dominio*$_G$ su cui il *designer*$_G$ si basi.
  Mettere a disposizione ad esempio una scacchiera 8x8, evitando che l'*utente*$_G$ finale debba crearsela da zero. Ovviamente i giochi da tavolo non son l'unico *dominio*$_G$ su cui ci si può basare, la scelta potrebbe ricadere in altri settori, ma tale scelta deve esser fatta per poter avere dei *diagrammi*$_G$ e quindi del *codice*$_G$.
  Il *dominio*$_G$ fin ora citato è consigliato perchè offre molti approfondimenti sull'adattamento in *diagrammi*$_G$ ed inoltre garantirà una fase di testing più piacevole.

- **Se viene modificato il *codice*$_G$ generato, il *diagramma*$_G$ deve aggiornarsi anch'esso?**
  È un aspetto sicuramente interessante se si riuscisse ad implementare. Solitamente dopo la creazione dei *diagrammi*$_G$ ed il rispettivo *codice*$_G$, le successive modifiche vengono apportate solo al *codice*$_G$; in quanto il *diagramma*$_G$ ha lo scopo principale da fungere da linea guida, rappresentando solo le *classi*$_G$ principali, tralasciando nella visualizzazione quelle di supporto. Rappresentare tutte le *classi*$_G$ che costituiscono il progetto,potrebbe ridurre la leggibilità del *diagramma*$_G$; una soluzione potrebbe essere quella di celare o inserire in un *layer*$_G$ diverso le *classi*$_G$ di supporto.
  Visualizzare solo lo scheletro della *classe*$_G$ senza l'implementazione dei *metodi*$_G$ può esser una soluzione alla domanda posta.

- **L'applicazione deve essere solo *desktop*$_G$ o deve essere anche una *Web-App*$_G$?**
  È preferita la web-application, ma la scelta non è vincolante. Lo scopo principale è entrare nell'ottica di fare un progetto usando molto i *diagrammi*$_G$.

- **Il *codice*$_G$ prodotto deve esser in formato *Java*$_G$ o *JavaScript*$_G$?**
  Nella fase di disegno dei *diagrammi*$_G$, in particolare a quelli dei *metodi*$_G$, si deve procedere in modo astratto , ovvero tracciando solo l'algoritmo necessario al *metodo*$_G$ in esame. Procedendo con quest'ottica si può generare *codice*$_G$ in entrambi i linguaggi. Proseguire in quest'ottica di pensare per specifiche risulta interessante ed una buona sfida , ma se si decidesse di pensare per programma, iI consiglio è quello di procedere inizialmente con *Java*$_G$, il quale è un linguaggio più controllato e verificato.

- **L'applicazione dovrà esser disponibile su di un server oppure è sufficiente in locale?**
  Non è necessario l'acquisto di uno spazio su cui ospitare l'applicazione, eventualmente si possono utilizzare dei server gratuiti a tempo limitato come ad esempio

Amazon, Heroku.

- **Considerando che il numero di gruppi che aderiscono a tale *capitolato*$_G$ è aumentato, alcuni dei requisiti opzionali son diventati obbligatori?**
  I requisiti opzionali rimangono invariati, un aspetto su cui focalizzarsi è lo studio dell'*UML*$_G$. Il progetto consiste nel creare un disegnatore che abbia il *diagramma*$_G$ della *classe*$_G$ con i rispettivi *diagrammi*$_G$ dei *metodi*$_G$; la soluzione di come rappresentare il collegamento tra questi *diagrammi*$_G$, è sicuramente il punto su cui ci si deve concentrare, proponendo anche modelli ibridi dei *diagrammi*$_G$.

- **Ci devono esser dei controlli nella costruzione del *diagramma*$_G$? e come devo restituire il *codice*$_G$ generato?**
  Ciò che viene restituito può essere un pacchetto contenente i *file*$_G$ oppure una visualizzazione testuale del *codice*$_G$; l'importante è che venga visualizzato il risultato che l'*utente*$_G$ si aspetta. Per quanto riguarda i controlli, son un lavoro di cortesia, evitare che l'*utente*$_G$ durante la fase di "disegno" faccia degli errori come ad esempio dimenticarsi il nome della *classe*$_G$ è sicuramente apprezzato. Ovviamente se si desse la possibilità di aggiungere nel *diagramma*$_G$ un rettangolo dove si possa aggiungere del *codice*$_G$ manualmente, in quella situazione risulterà impegnativo gestire automaticamente l'errore.

2 Riunione 2

2.1 Informazioni sulla riunione

- **Data:** 15/03/2017
- **Luogo:** Zucchetti - sede di Padova , via Giovanni Cittadella 7
- **Ora:** 11:30
- **Durata:** 20 min
- **Argomento:** Consulenza sui *diagrammi*$_G$
- **Partecipanti Interni:** Davide Santimaria - Fabio Massignan - Gianmarco Salmistraro - Malick Bodian - Salvatore Pilò - Sebastiano Bertolin
- **Partecipanti Esterni:** Gregorio Piccoli

2.2 Domande e Risposte

- **Abbiamo pensato di rappresentare i *diagrammi*$_G$ dei *metodi*$_G$ collegando tra loro un *flow chart*$_G$ ed un *activity diagram*$_G$,come le sembra questa soluzione?**
  L'idea di sfruttare questi *diagrammi*$_G$ è apprezzata, così come la possibilità di scegliere quale dei due *diagrammi*$_G$ mettere in primo piano.

- **Data la scelta del *dominio*$_G$ dei giochi da tavolo, possiamo aggiungere dei *template*$_G$ di *classi*$_G$ e funzioni legate al *dominio*$_G$?**
  L'*UML*$_G$ non ha un grande supporto dei *template*$_G$, solitamente si usano le collaborazioni, che son un ente che descrivono un insieme statico di relazioni tra istanze, e i ruoli che queste istanze svolgono in queste relazioni. Questa soluzione è abbastanza scomoda; suggerisco di approfondire i *pattern*$_G$ che utilizza l'*UML*$_G$ per capire da sé che inizialmente l'*UML*$_G$ è nato per disegnare *classi*$_G$ e quindi con l'uso dei *pattern*$_G$ c'è contrasto. Ci son due rappresentazioni di *pattern*$_G$ quella della sua definizione e quella del suo uso; implementare la possibilità all'*utente*$_G$ di crearsi i propri *template*$_G$. Quando l'*utente*$_G$ crerà il suo *template*$_G$, le *classi*$_G$ che ne fanno parte dovranno avere una etichetta che ne definisce il ruolo e la funzionalità del cambio del nome; altrimenti se in un *diagramma*$_G$ si richiama più volte lo stesso *template*$_G$ darà luogo ad errori visto che il nome è un attributo univoco. Un esempio può essere un videogioco di calcio, dove per rappresentare la testa dei giocatori si usa un *pattern*$_G$, che a sua volta può esser un contenitore di altri *pattern*$_G$ come occhi, e naso. Già questo *pattern*$_G$ si può utilizzare 22 volte per il numero di giocatori in campo ed ognuno di essi avrà delle caratteristiche differenti, ma seguono tutti lo stesso modello.

- **Il passaggio tra *codice*$_G$ e disegno è un requisito richiesto?**
  No non è necessario questa caratteristica, si potrebbe valutare eventualmente la sincronizzazione , o meglio dare la possibilità a due persone di lavorare allo stesso *diagramma*$_G$, ma anche questo è un requisito che può esser inserito nei nei requisiti opzionali. Ci si potrebbe orientare su un disegnatore collaborativo eventualmente supportate con *JSON*$_G$ e con librerie come *diff-match-Patch* che permette di far lavorare in maniera collaborativa grazie allo scambio di *file*$_G$ in formato *JSON*$_G$; resta comunque una caratteristica che diventa un progetto all'interno del progetto attuale e quindi, come già detto, è un requisito opzionale.

- **È obbligatorio fornire l'auto compilazione del *codice*$_G$?**
  Se si intende che il *codice*$_G$ generato debba compilare questo è banalmente ovvio. La compilazione automatica sarebbe gradita.

- **Dobbiamo aggiungere la presenza di un *utente*$_G$ amministratore?**
  Credo che la mole di lavoro sia sufficiente e che quindi aggiungere nuove funzionalità al momento si possa esculdere, basta concentrarsi sui requisiti già definiti. Avete accennato alla possibilità di aggiungere dei gruppi, ma preferisco che vi concentriate sull'inserimento dei *template*$_G$ anzichè di questa funzionalità.