Università degli Studi di Padova

KaleidosCode

SWEDesigner

Software per diagrammi UML

Verbale incontro con *Zucchetti s.p.a.* in data 17/03/2017

Informazioni sul documento

Data Redazione	17/03/2017
Redazione	Pace Giulio
Verifica	Pezzuto Francesco
Approvazione	Sovilla Matteo
Uso	Esterno
Distribuzione	*KaleidosCode*

kaleidos.codec6@gmail.com

Informazioni sulla riunione

- **Data riunione**: 17/03/2017
- **Luogo**: Sede di *Zucchetti s.p.a.* in via Giovanni Cittadella, 7, PD
- **Ora di inizio**: 17:30
- **Ora di fine**: 18:00
- **Partecipanti**:
  - Piccoli Gregorio (*Zucchetti s.p.a.*)
  - Bonato Enrico
  - Pace Giulio
  - Pezzuto Francesco
  - Sanna Giovanni
  - Sovilla Matteo

Ordine del giorno

1. Chiarimento riguardo lo scopo del progetto e descrizione dell'idea di editor del gruppo;
2. Verificare se l'idea di editor del gruppo è valida per il proponente;
3. Chiarimento sul come trattare il linguaggio UML nell'editor;
4. Chiedere se bisogna pensare alla possibilità di editare codice multithread;
5. Chiarimento sul possibile aggiornamento del diagramma in seguito a modifiche al codice precedentemente generato;
6. Chiedere consiglio su metodo di salvataggio da adottare se si decidesse di dare la possibilità all'utente di salvare i propri personali design pattern;
7. Chiarimento riguardo la possibile realizzazione del gioco da tavolo "Hex".

Di seguito, il riassunto delle discussioni fatte.

1. *KaleidosCode*: Vorremmo chiarimenti sul focus principale del progetto: non siamo sicuri se dobbiamo realizzare un software incentrato sullo sviluppo Java/Javascript mediante il diagramma UML o un editor UML che in più produce codice. In alcuni casi infatti dobbiamo scegliere se essere fedeli ai vincoli che UML impone o se "valicarli" per un maggior livello di completezza del codice.

   Zucchetti s.p.a.: La cosa che vogliamo controllare è la generazione del codice e per fare ciò può essere necessario modificare leggermente il diagramma UML. Ad esempio mettiamo caso che voi vogliate utilizzare un editor esterno per la generazione dei

diagrammi, come ad esempio ArgoUML, che mi salva l'xmi, poi leggendo l'xmi voi generate il codice. Potrebbe andare benissimo così, ma c'è un problema. Se dovete inventare un diagramma nuovo, non c'è lì dentro e allora bisogna capire come risolvere la situazione.
Avete quindi due opzioni: la prima è usare un disegnatore UML standard e richiedere che venga usato sotto certe condizioni. Allora quello che dovete fare in questo caso è controllare che venga usato secondo le vostre direttive e semmai segnalare un eventuale errore. Quindi potete mettere le restrizioni che volete all'UML a patto che il vostro controllore le riconosca e gestisca correttamente. Se invece intendete fare voi stessi il disegnatore avete più libertà sotto questo punto di vista.
Voi come pensavate di fare?

KaleidosCode: Per ora non ci siamo addentrati ancora nella progettazione, ma come idea di massima pensavamo di realizzare il disegnatore organizzandolo a livelli: il livello più esterno è un diagramma dei package. Clickando sui singoli elementi si passa a un diagramma delle classi, poi a una visualizzazione più dettagliata di una singola classe. Qui ogni metodo avrà associato il corrispondente diagramma delle attività. All'interno del diagramma delle attività pensavamo di specificare ulteriormente ogni attività con una sorta di diagramma di flusso modificato ad hoc. L'idea è quella di mettere a disposizione una serie di bolle organizzate in opportune librerie. Le bolle sono le unità di base del diagramma di flusso e sono già tradotte in codice. In questo modo, combinandole tra loro, il codice che viene generato dovrebbe verosimilmente essere privo di "buchi". Quindi fissando un dominio, che può essere, come da capitolato, quello dei giochi da tavolo, pensavamo di mettere a disposizione un insieme completo di bolle per essere in grado di garantire che, utilizzandole, l'editor possa tradurre il diagramma in codice senza errori.
Qualora l'utente voglia in qualche modo estendere le possibilità offerte dalle bolle pre-realizzate, pensavamo di considerare un bolla personalizzabile in cui l'utente stesso può inserire il proprio codice, ma in questo caso verrebbe notificata la mancanza di sicurezza nella sua "buona" generazione a partire dal diagramma.

Zucchetti s.p.a.: Va bene. A questo proposito, quello che stanno notando anche gli altri gruppi è che, visto il dominio relativamente ristretto, sono presenti molti pattern. Hanno pensato quindi di sfruttare e di far utilizzare i design pattern nel loro editor UML. Provate a considerare l'idea, magari potrebbe integrarsi bene con quello che avete già pensato. Tenete conto però che, aggiungendo anche i pattern, il diagramma UML potrebbe diventare un po' farraginoso e difficile da leggere se fate tante classi che magari usano tanti pattern.

KaleidosCode: Ma in che modo dobbiamo utilizzare questi pattern nel momento di passare a codice?

Zucchetti s.p.a.: Quando si sta in un dominio così formato si vengono a creare delle soluzioni. Queste soluzioni alle volte si riescono a riciclare come ereditarietà, alle volte come altro, fatto sta che capita saltino fuori pattern che sono l'unione di più pezzi che collaborano tra loro. Soprattutto c'è anche un altro aspetto: una stessa classe può partecipare a più pattern. Quindi la cosa interessante è che una stessa classe può "giocare in più ruoli". In ogni caso è solo una possibilità che vi presento, se volete seguirla bene, altrimenti va bene lo stesso.

2. *KaleidosCode*: Potremmo pensare di implementare questa idea magari con un sistema di etichette, magari colorate, per dare un'idea di questi pattern. In ogni caso volevamo un feedback per capire se stiamo andando verso la giusta direzione o se abbiamo interpretato male qualcosa.

   Zucchetti s.p.a.: No, state procedendo correttamente.

3. *KaleidosCode*: Una discussione che era nata tra di noi riguardava il fatto che trattandosi di UML magari il professor Vardanega tende ad essere rigido riguardo l'aderenza agli standard, mentre da quanto abbiamo capito dal primo incontro è apprezzata un po' di "irriverenza" nei confronti dell'UML.

   Zucchetti s.p.a.: È giusto. Il discorso è che se voi modificate l'UML con "arroganza", ovvero senza motivare la vostra scelta allora il professor Vardanega avrebbe tutti i motivi di contestarvi. Se vi inventate qualcosa di nuovo deve essere fortemente giustificato e deve avere una logica alla base. In quel caso sono sicuro che anche per lui non ci siano problemi. Comunque ricordatevi che non dovete essere "cattivi" con lo standard. Dovete semplicemente utilizzarlo come vi fa comodo.

4. *KaleidosCode*: Dovremo occuparci di trattare anche codice multithread?

   Zucchetti s.p.a.: No, senza dubbio no. C'è già tanto da dire così, l'importante è che sia chiaro fin dall'inizio. In ogni caso non lo richiedo assolutamente.

5. *KaleidosCode*: Un altro aspetto del quale non siamo sicuri riguarda l'aggiornamento del diagramma in seguito a modifiche al codice precedentemente generato.

   Zucchetti s.p.a.: No no no, non provate a farlo. Cioè se ci riuscite bene, ma guardate che è difficilissimo.

   KaleidosCode: Il dubbio ci è venuto perché nel capitolato viene richiesto un aggiornamento del diagramma in seguito a "piccole modifiche al codice".

   Zucchetti s.p.a.: Si, ma visto che voi avete parlato di bolle nel caso in cui per esempio innestiate delle bolle l'una dentro l'altra la complessità di controllo diventa molto alta; ad esempio, diventa difficile per un "controllore" capire, se innestate una bolla if dentro un altro if, a quale if ci si riferisce.

   KaleidosCode: L'idea più che altro derivava dal fatto che nel momento di aggiungere un elemento al diagramma pensavamo di creare un oggetto nascosto che contenesse le informazioni necessarie a popolare sia il diagramma stesso che il codice. Da qui l'idea, magari mediante colorazioni diverse, di rendere possibili piccole modifiche.

   Zucchetti s.p.a.: Allora dovete essere più liberali sulle bolle. Se invece volete essere più rigidi su quelle dovete rinunciare a questo aspetto.

6. *KaleidosCode*: Se decidiamo i dare la possibilità all'utente di salvare i propri personali design pattern potremmo considerare di gestirlo tramite cookies o saremmo costretti ad adottare altre soluzioni come ad esempio un salvataggio in remoto?

   Zucchetti s.p.a.: Un pattern avrebbe senza dubbio dimensioni troppo grandi per un cookie. Poi dipende dal formato che usate, ma è difficile che ci stia. Non ha senso usare cookies, piuttosto un local storage è molto meglio. Il cookie inoltre viene

passato ogni volta che viene contattato il server. Se fosse di dimensioni generose, si genererebbe molto traffico.

7. *KaleidosCode*: Pensavamo di inserire tra gli obiettivi desiderabili lo sviluppo del gioco da tavolo "Hex": vorremmo sapere se la realizzazione di quest'ultimo va fatta esclusivamente con l'ausilio dell'applicativo che realizzeremo o se l'applicativo deve darci la "base" del codice che poi possiamo anche integrare manualmente per conto nostro.

   Zucchetti s.p.a.: State dicendo una cosa molto semplice, che mi sembra ragionevole: pensate che con il vostro applicativo riuscirete a coprire una certa "quota" dell'insieme. È possibilissimo, quindi va bene. Magari create, ad esempio, il pattern turno, scacchiera, ecc. e poi inserite una bolla dove scrivete il rimanente, poiché non siete ancora arrivati a fare uno strumento così raffinato.

   KaleidosCode: Va bene. Grazie della disponibilità.

   Zucchetti s.p.a.: Bene, fatemi sapere se avete ulteriori dubbi.