Università degli Studi di Padova

KaleidosCode

SWEDesigner

Software per diagrammi UML

Verbale incontro con *Zucchetti s.p.a.* in data 23/02/2017

Informazioni sul documento

Data Redazione	23/02/2017
Redazione	Sovilla Matteo
Verifica	Pezzuto Francesco
Approvazione	Sanna Giovanni
Uso	Esterno
Distribuzione	*KaleidosCode*

kaleidos.codec6@gmail.com

Informazioni sulla riunione

- **Data riunione**: 23/02/2017
- **Luogo**: Sede di *Zucchetti s.p.a.* in via Giovanni Cittadella, 7, PD
- **Ora di inizio**: 16:30
- **Ora di fine**: 17:30
- **Partecipanti**:
  - Piccoli Gregorio (*Zucchetti s.p.a.*)
  - Bonato Enrico
  - Pace Giulio
  - Pezzuto Francesco
  - Sanna Giovanni

  Anche i gruppi "SWEetBIT" e "Digital Cookies" erano presenti alla riunione.

Primo incontro con *Zucchetti s.p.a.*.
Lo scopo della riunione era farsi un idea più dettagliata del problema da affrontare e cogliere i requisiti funzionali fondamentali. La comprensione iniziale, derivata solo dal capitolato proposto, era insufficiente. Non si sapeva esattamente cosa domandare al fine di una completa comprensione della richiesta, visto il breve periodo di preparazione per l'incontro.
La riunione si è svolta insieme agli altri gruppi interessati al capitolato.

Ordine del giorno

1. Qualità e complessità codice prodotto;
2. Dominio trattato;
3. Riusabilità;
4. Ruolo dell'UML;
5. Preferenze del committente;
6. Suggerimenti per progettare la soluzione.

Di seguito, il riassunto delle discussioni fatte.

1. Per il codice prodotto dal diagramma delle classi, la traduzione è banale e consiste nello scheletro delle classi insieme all'intestazione dei metodi; non ha senso parlare di complessità e qualità del codice.
   Per il diagramma delle attività, invece, la qualità del codice prodotto dipende in gran parte dalla completezza e grado di specializzazione (inteso come grado di raffinamento dell'attività rappresentata) del diagramma tradotto, oltre ovviamente alla qualità della logica di traduzione.

2. Le realtà che si possono modellare con i diagrammi prodotti dal disegnatore devono fare parte di un dominio specifico; sarebbe impensabile produrre software, sufficientemente completo, dalla traduzione di diagrammi che modellano qualsiasi realtà.
   E' gradito il dominio dei giochi da tavolo.
   E' necessaria un'analisi approfondita del dominio scelto.

3. È importante massimizzare la riusabilità dei componenti, per poterli sfruttare nel maggior numero di realtà modellate dai diagrammi (le quali apparterranno ad un unico dominio).
   Consigliato creare template: il focus su un dominio permette di creare modelli che si adattano a molti casi specifici di tale dominio.
   È stato esemplificato il dominio dei giochi da tavolo: molti giochi prevedono una scacchiera; bisogna, allora, prevedere un metodo che disegni la scacchiera, sarà poi il gioco specifico che deciderà come gestirla (ad esempio, dama userà tutte le caselle, Monopoli userà solo le caselle del contorno).

4. Nei progetti reali, l'UML ha un'utilità solo nel breve termine; un obiettivo è quello di renderlo importante per tutta la durata del progetto e non una rappresentazione iniziale che diventa inutilizzabile o addirittura inutile. A tal senso viene lasciata libertà di essere irriverenti con l'UML, pensando a soluzioni anche non convenzionali rispetto allo standard.

5. È gradito lo sviluppo del prodotto attraverso le tecnologie web, ma è anche data la possibilità di realizzarlo per desktop.
   Il server può essere locale, ma è lasciata la libertà di acquistare spazi web. In entrambi i casi bisogna simulare il costo dell'acquisto.
   Consigliato concentrarsi su uno specifico dominio.

6. È preferibile implementare l'applicazione lato server in Java, in quanto facilita il test (compilatore) e non bisogna simulare l'ereditarietà.
   Il problema principale della traduzione da UML a codice è dato dalla mancanza di continuità tra i vari diagrammi. Valutare la possibilità di creare diagrammi "ibridi", che abbiano elementi sia di struttura che di comportamento; in questo modo si risparmierebbe l'integrazione di codici derivati da diagrammi diversi.
   È gradita la possibilità di tenere aggiornato il diagramma creato, in seguito a modifiche del codice precedentemente prodotto.
   Le classi e le attività del software sono spesso molte e complesse per essere tracciate con chiarezza nei diagrammi; bisogna perciò prevedere più viste di uno stesso diagramma attraverso livelli di dettaglio: layer per i diagrammi delle classi, innesti per i diagrammi delle attività, zoom.