Egoless Group
egolessgroup@googlegroups.com

Norme di progetto per C04/PQS

Versione 3.0

Stato del documento:
Formale ed
Esterno

Sommario :

In questo documento sono illustrate tutte le norme che devono essere seguite per la conduzione del progetto relativo al capitolato **C04**: "**Sistema *software* per l'informatizzazione della gestione di qualità a norma ISO 9000:2000 nelle scuole**".

Redazione

Nominativo	Ruolo	Data
Eric Miotto	Verificatore	20 novembre 2006
Margherita Collicelli	Amministratore	20 novembre 2006
Roberto Pordon	Verificatore	20 novembre 2006
Lucia Meneghello	Amministratore	10 gennaio 2007
Eric Miotto	Amministratore	14 febbraio 2007

Lista di Distribuzione

Nominativo	Ruolo
Tullio Vardanega	Committente
Renato Conte	Committente
Lucia Meneghello	Verificatore
Margherita Collicelli	Responsabile
Eric Miotto	Amministratore/Programmatore
Stefano Gazzola	Programmatore
Roberto Pordon	Verificatore
Lorenzo Daniele	Programmatore
Alberto Meneghello	Programmatore

Approvato da:

Versione	Nominativo	Data
3.0	Margherita Collicelli	16 marzo 2007

Registro delle Modifiche:

Versione	Autore	Data
3.0	Margherita Collicelli	16 marzo 2007
Approvazione documento per la Revisione di qualifica.		
2.46	Lucia Meneghello	16 marzo 2007
Corretti errori ortografici e verificato documento.		
2.45	Eric Miotto	14 marzo 2007
Aggiornata sezione sulle metriche (sezione 7.7).		
2.44	Lorenzo Daniele	09 marzo 2007
Aggiunta precisazione in 7.6.1 (Event Dispatching Thread).		
2.43	Lucia Meneghello	06 marzo 2007
Corretti alcuni errori ortografici.		
2.42	Lorenzo Daniele	05 marzo 2007
Rivisto capitolo sulle icone per l'interfaccia (7.6.3).		
2.41	Eric Miotto	03 marzo 2007
Puntualizzato che i nomi non devono avere accenti (sezione 7.2). Rilassate norme sui commenti Javadoc (sezione 7.5.2).		
2.4	Lorenzo Daniele	03 marzo 2007
Aggiunte norme attinenti le icone nell'interfaccia grafica (7.6.3).		
2.3	Lorenzo Daniele	02 marzo 2007
Aggiunta sezione sull'accessibilità dell'interfaccia grafica (7.6.2).		
2.25	Eric Miotto	02 marzo 2007
Aggiunte regole di codifica (sezioni 7.5.1 e 7.5.4). Introdotta sezione sugli idiom (sezione 7.6) per raccogliere buone regole di programmazione.		
2.2	Lorenzo Daniele	01 Marzo 2007
Integrate norme di codifica relativamente alla concorrenza (7.6.1) ed alla clonazione(7.6.4).		
2.1	Eric Miotto	27 febbraio 2007
Aggiornata procedura per la consegna dei documenti (sezione 5.6). Aggiunte altre regole di nomenclatura (sezione 7.2), inserite regole per l'indentazione (sezione 7.5.3).		
2.0	Lorenzo Daniele	23 febbraio 2007
Approvazione documento per la Revisione di Progetto Definitivo.		
1.3	Lucia Meneghello	22 febbraio 2007
Effettuata verifica del documento.		

1.2	Eric Miotto	21 febbraio 2007

Inserita sezione 7.2 sulla nomenclatura nella progettazione e nella codifica. Leggera correzione all'introduzione. Corrette sezioni 2.3, 3.1 e 5.4.

1.1	Eric Miotto	21 febbraio 2007

Eliminata regola che variabili e metodi privati hanno il nome che inizia con l'underscore (sezione 7.4.3). Inserita sezione sulle metriche (sezione 7.7).

1.06	Lucia Meneghello	21 febbraio 2007

Aggiornata lista di distribuzione.

1.05	Eric Miotto	19 febbraio 2007

Aggiornata sezione 5.3.

1.04	Eric Miotto	18 febbraio 2007

Aggiornata lista di distribuzione, aggiornata sezione 6.1 (aggiornate completamente le regole per BoUML) ed inserita sezione 7.5.2 sui commenti in Javadoc.

1.03	Eric Miotto	17 febbraio 2007

Aggiornate sezioni 2.1 (inserito link al wiki sugli strumenti), 6.1 (cambiamento editor usato) ,7.4.2 (spiegata meglio la transizione da ArgoUML a BoUML).

1.02	Eric Miotto	14 febbraio 2007

Aggiornata sezione 7 specificando che ora i nomi sono tutti in italiano.

1.01	Roberto Pordon	08 febbraio 2007

Integrazione norme di progettazione

1.0	Roberto Pordon	02 febbraio 2007

Approvazione documento per la Revisione Preliminare di Progetto.

0.7	Alberto Meneghello, Lorenzo Daniele	02 febbraio 2007

Effettuata Verifica.

0.6	Roberto Pordon	01 febbraio 2007

Aggiunta sezione sugli strumenti usati.

0.5	Alberto Meneghello, Lorenzo Daniele	01 febbraio 2007

Corretti lievi errori. Verificato l'intero documento.

0.4	Roberto Pordon	30 gennaio 2007

Rimosse norme per l'utilizzo del server (trasferite in [U]). Aggiornata parte per risoluzione di anomalie e discrepanze

0.3	Roberto Pordon	28 gennaio 2007

Aggiunte norme per utilizzo server. Correzione formattazione

0.23	Eric Miotto	14 gennaio 2007

Corretto punto sulla colorazione degli elementi dei diagrammi use case nella sezione 6.1.

0.22	Eric Miotto	28 dicembre 2006

Aggiornato documento secondo il modello.

0.21	Eric Miotto	17 dicembre 2006

Integrate norme per la stesura dei documenti, aggiunte regole per la gestione delle tabelle.

0.20	Eric Miotto	10 dicembre 2006

Aggiunta prima bozza sezione sulla verifica; aggiornate regole versionamento documenti..

0.16	Eric Miotto	10 dicembre 2006

Aggiunte sezione 5.4 sullo stile delle parole di glossario.

0.15	Lorenzo Daniele	10 dicembre 2006

Scritte norme sulla stesura dei casi d'uso (paragrafo 5.1).

0.14	Eric Miotto	10 dicembre 2006

Aggiunta regola per gestire i file compressi; inserite prime regole per l'uso della mailing list.

0.13	Eric Miotto	09 dicembre 2006

Aggiunta sezione su Google Groups; corretta qualche imprecisione e svista.

0.12	Eric Miotto	08 dicembre 2006

Conferma regole abbozzate; aggiunta regole per le date; aggiunta regola per segnalare la modifica di documenti; aggiornato lo stile secondo il modello.

0.11	Margherita Collicelli	04 dicembre 2006

Correzione tabella distribuzione e piccola correzione al formato del documento.

0.1	Eric Miotto	29 novembre 2006

Prima bozza del documento.

Indice

1 Introduzione

Questo documento si propone di illustrare tutte le norme che il gruppo seguirà nello svolgimento del progetto. Queste regole riguardano:

- la redazione di documenti;
- l'uso del gruppo su Google Groups;
- l'uso del server;
- l'uso degli strumenti;
- la conduzione dell'analisi dei requisiti, della progettazione, della codifica e della verifica.

1.1 Riferimenti

- [G] Glossario, Egoless Group
- [Q] Piano di qualifica, Egoless Group
- [U] Utilizzo del server, Egoless Group
- [UML] Specifica di UML, versione 2.0, OMG
  http://www.uml.org/
- [JC] Java Coding Style Guide, Achut Reddy, Sun Microsystems
  http://developers.sun.com/sunstudio/products/archive/whitepapers/java-style.pdf
- [ISB] Lucidi del corso di Ingegneria del software mod. B, prof. Renato Conte, Università di Padova, a.a. 2006/2007
  http://www.math.unipd.it/~conte
- [DP] Design Patterns, elements of reusable object-oriented software
  Gamma, Helm, Johnson, Vlissides, Ed. Addison-Wesley ISBN 0-201-63361-2
- [CC SEI] Cyclomatic Complexity – Software Technology Roadmap
  http://www.sei.cmu.edu/str/descriptions/cyclomatic_body.html
- [JLF] Java Look and Feel Design Guidelines
  http://java.sun.com/products/jlf

2 Regole generali

Queste regole valgono in ogni ambito (documentazione, programmazione, analisi, progettazione, eccetera...).

2.1 Strumenti usati

Per la conduzione del progetto vengono usati i seguenti strumenti:

- OpenOffice 2.0 per la redazione dei documenti;
- LaTeX con il package beamer per la redazione delle presentazioni;
- NetBeans 5.5 come IDE Java;
- Subversion per il versionamento di documenti e sorgenti e per la memorizzazione di altri file;
- Trac come *issue tracking system*;
- Google Groups per la mailing list;
- SciTE e Smultron come editor di testo avanzati;
- BoUML per la produzione di diagrammi UML.

All'indirizzo https://egoless.mine.nu/PQS_doc/wiki/GuidaAiTool sono presenti alcune pagine che dettagliano l'installazione e l'uso di alcuni degli strumenti impiegati. Queste pagine hanno valenza di documentazione.

2.2 Nomi

Il nome di una persona è sempre scritto nella forma “Nome Cognome”.

2.3 Date

Le date sono sempre riportate nel seguente modo (GG Mese AAAA):

- prima il numero del giorno sempre con due cifre;
- poi il nome del mese per esteso. La prima lettera può essere maiuscola o minuscola;
- poi l'anno con quattro cifre.

Alcuni esempi di date corrette:

- 08 dicembre 2006
- 11 Gennaio 2007

Alcuni controesempi significativi:

- 8 dicembre 2006
- 11 gennaio 07
- 11/01/2007

3 Uso di Google Groups

Il sito del gruppo è raggiungibile alla pagina

http://groups-beta.google.com/group/egolessgroup.

3.1 Redazione pagine

Le pagine vengono ora gestite su Trac. Viene comunque ancora regolamentata la redazione di pagine in Google Groups.

Una volta aggiunta/modificata una pagina, è necessario postare un messaggio nel quale spiegare brevemente cosa è cambiato nella pagina. Per fare ciò, basta semplicemente salvare la pagina e quando proposto scrivere il testo del messaggio.

3.2 Gestione file

N.B.: *Dal momento dell'attivazione del server i file presenti su Google Groups saranno reperibili nei repository ed è a quelli che si dovrà fare riferimento. Queste regole rimangono comunque valide in caso di uso di Google Groups.*

3.2.1 Gestione file relativamente a Google Groups

I file che non vengono più utilizzati vanno rinominati anteponendo una tilde (~) al nome.

Il nome dei documenti dovrebbe riportare la loro versione per rendere più immediato capire ai membri se il documento deve essere scaricato o meno.

E' fortemente sconsigliato inserire archivi compressi al cui interno sono contenuti più documenti: è preferibile postare singolarmente ogni documento. Nel caso sia necessario inserire file compressi (perché la dimensione è elevata), si utilizzi il formato ZIP.

3.3 Uso della mailing list

Nel rispondere ad un dato *thread*, è necessario valutare attentamente se includere i messaggi precedenti o meno, per contenere la dimensione dei messaggi da scaricare (specialmente per coloro che dispongono di connessioni ad Internet lente) e per rendere più comprensibile il messaggio.

- I messaggi del thread “Aggiornamento documenti” non devono mai includere i messaggi precedenti.
- Se l'inclusione dei messaggi precedenti viene ritenuta idonea per il messaggio, per limitare la leggibilità e la lunghezza del messaggio si ammettono al massimo due livelli di inclusione (si faccia riferimento all'illustrazione 1).

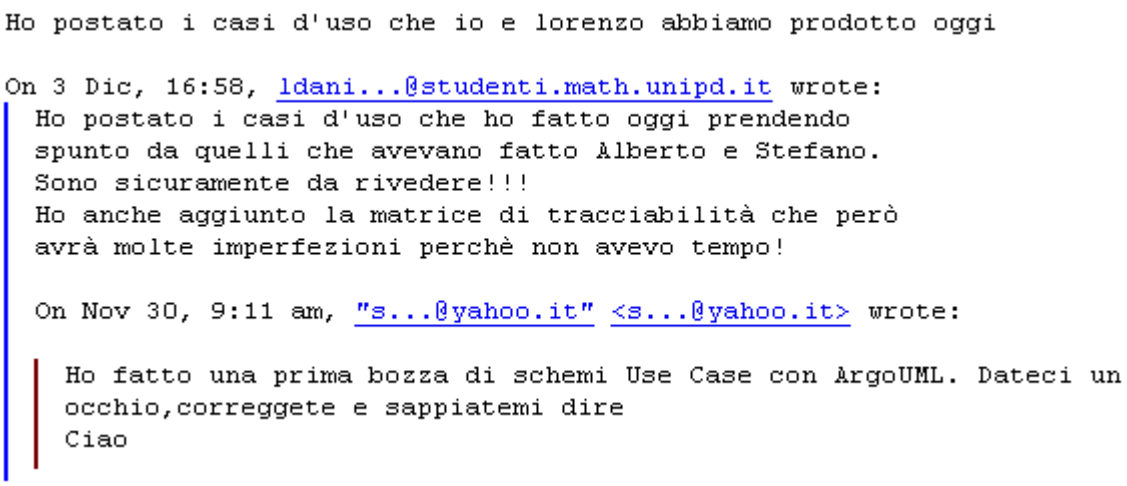

Illustrazione 1: Esempio di due livelli di inclusione. La risposta è a livello 0; il messaggio marcato con la linea blu laterale è a livello 1; il messaggio marcato con la linea marrone è a livello 2.

3.4 Invio di e-mail all'esterno del gruppo

Le seguenti regole si applicano quando è necessario inviare e-mail per conto del gruppo.

- Cercare di inviare il messaggio inserendo come mittente l'indirizzo e-mail del gruppo. Questo si ottiene usando un client di posta elettronica come Thunderbird: il metodo è quello di creare un nuovo account per una casella di posta fittizia, inserire le informazioni su utente e casella e-mail correttamente ed utilizzare questo account per inviare l'e-mail del gruppo.
- Questo non è però sempre possibile. Nei messaggi e-mail è presente un campo

che si chiama Reply-To, che indica l'indirizzo che il client di posta elettronica deve usare per inviare un messaggio di risposta usando il pulsante Rispondi o Reply. L'importante sarebbe valorizzare correttamente questo campo, in modo da avere la certezza che il destinatario del messaggio risponda all'indirizzo desiderato.

- I messaggi e-mail spediti per conto del gruppo vanno archiviati in Google Groups. A tal scopo è sufficiente inviare il messaggio per conoscenza al gruppo, avendo cura di usare la copia carbone nascosta (Ccn o Bcc), che spedisce la copia ma cancella dall'header questa informazione. Basta poi ricordarsi di controllare il gruppo ed autorizzare il post del messaggio (l'indirizzo e-mail del gruppo non è considerato membro ma esterno).

4 Server

Per le norme di utilizzo del server e degli strumenti che fornisce fare riferimento a [U].

5 Redazione di documenti

5.1 Repository

I documenti del gruppo sono memorizzati nel *repository* all'indirizzo https://egoless.mine.nu/svn/doc. L'area Trac ad esso associato è raggiungibile all'indirizzo https://egoless.mine.nu/PQS_doc/ , riferita nel seguito con PQS_doc.

5.2 Regole di versionamento

Le versioni dei documenti procedono secondo la seguente logica:

- si parte dalla versione 0.1;
- i numeri interi (1.0, 2.0, ...) si adottano solo per documenti verificati ed approvati;
- per correzioni si incrementa il numero di versione di 1/100 (per esempio da 0.1 a 0.11);
- per correzioni più estensive si incrementa il numero di versione di 1/10 (per esempio da 0.1 a 0.2).

Quando un file viene inserito in Google Groups, le versioni precedenti vanno rinominate anteponendo una tilde al nome.

5.3 Segnalazione nuove versioni

I cambiamenti ai documenti nel repository sono osservate da Trac e rese disponibili nel

feed disponibile nella Timeline di PQS_doc.

Per segnalare aggiornamenti di documenti che sono stati inseriti in Google Groups, si usi la discussione “Aggiornamento documenti” (o successive) nella mailing list.

5.4 Stile

Per la redazione dei documenti sono stati previsti i seguenti modelli:

- “Verbale.ott” per i verbali;
- “Lettera.ott” per le lettere;
- “Doc Standard.ott” per tutti gli altri documenti.

Alle parole che sono presenti nel glossario va applicato lo stile di carattere Enfasi.

Ai riferimenti di codice nel testo va applicato lo stile di carattere “Testo sorgente”. A frammenti di codice riportati nella documentazione va applicato lo stile [Eg]Codice.

5.4.1 Tabelle su più pagine

La redazione di tabelle richiede alcuni accorgimenti, specialmente se sono distribuite su più pagine:

- impedire che una riga sia a cavallo tra due pagine. Per evitare ciò, cliccare sulla tabella con il pulsante destro e selezionare la voce “Tabella...”. Selezionare il tab “Flusso di testo” e deselezionare l'opzione “Consenti l'interruzione delle righe tra pagine e colonne” (traduzione approssimata di “Consenti interruzioni di pagina e di colonna dentro una cella.”);
- ripetere l'intestazione della tabella su ogni pagina. Per fare ciò, cliccare sulla tabella con il pulsante destro e selezionare la voce “Tabella...”. Selezionare il tab “Flusso di testo” e selezionare l'opzione “Ripeti intestazione”, eventualmente indicando quante sono le righe iniziali da ripetere.

5.5 Redazione del glossario

La redazione del glossario è compito esclusivo di uno degli amministratori. Gli altri componenti del gruppo possono segnalare i termini da inserire (e proporne la definizione) utilizzando la pagina:
https://egoless.mine.nu/PQS_doc/wiki/ProposteGlossario

Una volta inserite le definizioni nel Glossario, l'amministratore le rimuoverà dalla pagina.

L'amministratore può decidere di non inserire i termini proposti, o proporre una modifica della definizione.

5.6 Consegna dei documenti

Una volta completati e approvati i documenti necessari per una revisione, si proceda come segue:

- dentro la cartella PDF del repository doc creare una cartella di nome C04_RXX_EgolessGroup, dove RXX è la sigla della revisione;
- generare i PDF di ogni documento dentro la cartella appena creata;
- comprimere la cartella (**non** direttamente i file) accertandosi che il file compresso abbia lo stesso nome della cartella. In questo modo decomprimendo l'archivio viene creata automaticamente una cartella che contiene tutti i PDF, come da richiesta del committente.

Se la consegna avviene con utilizzo di usb flash disk si provvederà a che il supporto contenga esclusivamente i file da consegnare. In particolare si ricordi di eliminare eventuali file nascosti.

6 Regole per l'analisi dei requisiti

6.1 Stesura dei casi d'uso

- Verranno redatti utilizzando BoUML;
- I vecchi diagrammi dei casi d'uso possono essere lasciati in ArgoUML. Se nasce l'esigenza di modificarli, vanno rifatti in BoUML;
- Tutti gli “ovali” dovranno trovarsi posizionati sopra un rettangolo di sfondo;
- Gli attori si troveranno al di fuori di tale rettangolo di sfondo;
- Gli ovali avranno colore giallo, mentre il rettangolo colore azzurro;
- All'interno degli ovali il titolo del caso d'uso specifico va scritto tutto in maiuscolo. Notare che BoUML permette di posizionare il testo rispetto all'ovale;
- L'identificativo del ruolo dell'attore va scritto sotto l'attore stesso con solo la prima lettere in maiuscolo, le altre in minuscolo;
- Il titolo dello use case sarà scritto all'interno di un piccolo rettangolo di colore bianco posizionato nell'angolo superiore destro del rettangolo di sfondo. Attenzione: dovete prima fare un rettangolo con lo sfondo bianco e poi creare un testo, dato che il testo non ha ne' sfondo ne' bordo.
- All'interno di esso il titolo sarà scritto nella forma CU ID – TITOLO tutto in maiuscolo con font “bold”.

7 Regole di progettazione e codifica

7.1 Documento di riferimento

Avendo scelto Java come linguaggio di sviluppo, si utilizzeranno come norme generali di riferimento per la progettazione e la codifica quelle definite dal documento [JC]. Di seguito sono indicati alcuni adattamenti per il nostro caso specifico. In caso di conflitto tra le norme indicate in [JC], valgono sempre quelle del presente documento.

7.2 Nomenclatura

I nomi delle classi, degli oggetti, dei metodi, delle variabili devono essere in lingua italiana. Sono ammesse commistioni tra italiano ed inglese nel caso di nomi caratteristici (in ogni caso documentati più avanti).

Non sono ammessi accenti nei nomi. Se il nome dovrebbe avere degli accenti, semplicemente si omettono.

Per denominare riferimenti a controlli Swing, si utilizza la forma

[prefisso][nome significativo in italiano]

dove [prefisso] è una sigla di 2, 3, 4 o 5 caratteri che identifica il tipo di controllo. I prefissi individuati sono elencati e mantenuti alla pagina https://egoless.mine.nu/PQS_doc/wiki/PrefissiSwing .

Il nome di una classe JavaBean deve finire con il suffisso “Bean”.

Il nome di una classe che deriva da `JFrame` deve finire con il suffisso “Frame”.

Il nome di una classe che deriva da `JPanel` deve finire con il suffisso “Panel”.

Il nome di una classe che deriva da `JDialog` deve finire con il suffisso “Dialog”.

Il nome di una classe che deriva da `JWindow` deve finire con “Window”.

Il nome di una classe che implementa un Web Service deve iniziare con il prefisso “WS”.

Il nome di una classe che deriva da `Exception` deve finire con il suffisso “Exception”.

Il prefisso dei *getter* e dei *setter* rimane in inglese anche se la proprietà ha il nome in italiano. Sono quindi corretti `setNome` e `getNome`.

7.3 Progettazione

Nella progettazione si dovrà fare uso di almeno tre *design patterns* e si dovrà cercare di ottenere alta coesione e basso accoppiamento. Per la definizione e le strategie d'uso

fare riferimento a [ISB] e [DP].

7.4 Diagrammi delle classi

7.4.1 Linguaggio

Come linguaggio per la rappresentazione si utilizzerà UML 2.0 (fare riferimento a [UML]).

7.4.2 Editor

Per realizzare i diagrammi delle classi va usato BoUML.

I diagrammi realizzati con ArgoUML vanno rifatti in BoUML quando c'è l'esigenza di aggiornarli.

7.4.3 Aspetto

Il rettangolo che definisce un pacchetto deve essere di colore azzurro, i rettangoli delle classi di colore giallo.

7.5 Codifica

7.5.1 Intestazione file

Ogni file prodotto deve avere la seguente intestazione:

```
/*
 * Nome file: __NAME__.java
 * Data creazione: __DATE__
 * Info svn: $Id$
 */
```

dove:

- __NAME__ è il nome del file;
- __DATE__ è la data e ora di creazione;
- Id è una keyword di Subversion che viene espansa ed aggiornata ad ogni commit del file.

7.5.2 Commenti Javadoc

Oltre ai tag consigliati in [JC], è opportuno usare anche il tag `{@link}` che permette di inserire link alla documentazione di altre classi e/o metodi. Quando si usa il tag `{@link}` è opportuno riportare tale link anche nel tag `@see`.

I commenti Javadoc devono essere in italiano. Possono essere in inglese i commenti generati automaticamente da NetBeans per le proprietà dei JavaBean.

7.5.3 Indentazione

L'ampiezza di una tabulazione per l'indentazione è di 3 spazi.

E' necessario configurare gli editor che manipolano il codice in modo che la pressione del tasto TAB non produca una vera tabulazione ma la simuli con gli spazi. Questo per essere sicuri che ovunque venga visualizzato o copiato il codice si mantenga un aspetto coerente ed indipendente dalla larghezza decisa dall'ambiente per la tabulazione.

7.5.4 Ordine di dichiarazione

Non è necessario che siano ordinate in ordine lessicografico:

- la lista delle eccezioni che un metodo può lanciare;
- la lista di metodi e/o membri all'interno di ogni blocco definito in [JC 6.1];
- la lista di interfacce che una classe o un'interfaccia implementano.

7.6 Idiom

In questa sezione verrano raccolte una serie di buone regole per usare le tecnologie impiegate a livello di programmazione.

7.6.1 Uso della concorrenza per rendere l'interfaccia reattiva

Per garantire la reattività dell'interfaccia è bene seguire alcune linee guida indicate in http://java.sun.com/docs/books/tutorial/uiswing/concurrency/index.html .

Riassumendo:

Initial Thread
L'initial Thread, ovvero il main, dovrebbe occuparsi solamente della generazione dell'interfaccia oppure della creazione del thread che creerà la GUI, ponendolo nell'event dispatching thread (in seguito e.d.t.). NetBeans applica proprio il secondo metodo delegando la costruzione ad un thread che richiama initComponent() il quale viene accosato nell'e.d.t.;

Event Dispatching Thread
Ogni task che richiede un basso costo in termini temporali e computazionali e che coinvolge attivamente la grafica o avvii di thread, dovrebbe essere posto nell'e.d.t.. Questo garantisce che non vengano interrotti eventuali refresh o lanci di eventi da

parte della View.

L'event dispatching thread non è altro che una coda di job i quali vengono avviati in sequenza.
Inoltre, è molto importante il suo utilizzo per garantire la sicurezza della sincronizzazione tra altri thread quando questi dovessero chiamare concorrentemente qualche metodo “unsafe”; infatti molti metodi della libreria Swing, per non venir appesantiti ulteriormente, sono stati progettai in modo da essere non sincronizzati o thread unsafe. Solo pochissimi metodi quali JTextComponent.setText() sono sicuri di fronte ad una modifica da parte di un thread (quando un metodo swing è “safe”, tale proprietà è scritta chiaramente nella documentazione delle Java API). E' possibile, per i metodi unsafe, garantire la sequenzialità dello svolgersi delle invocazioni semplicemente ponendo i relativi oggetti Runnable uno dopo l'altro nell'e.d.t.. L'e.d.t. non deve essere creato, viene istanziato automaticamente da Swing.

Il codice per accodare un task nell'e.d.t. è il seguente:

```
SwingUtilities.invokeLater(new Runnable() {
    public void run() {
        //codice da eseguire
    }
}
```

Notiamo che essendo i task in questione di breve durata solitamente saranno composti da relativamente poche righe di codice. Per cui è conveniente utilizzare una classe anonima come nell'esempio.
E' possibile utilizzare anche il metodo invokeAndWait(Runnable) il quale però blocca il thread chiamante finchè la run non ha concluso con il rischio di rendere l'interfaccia poco reattiva (in pratica invokeLater è asincrona rispetto l'e.d.t. mentre invokeAndWait è sincrono (riferirsi alla documentazione ufficiale Java oppure a http://www-128.ibm.com/developerworks/java/library/j-ebb0917b.html dove la differenza è spiegata abbastanza bene).

Notiamo che tutti i metodi chiamati dal lancio di eventi (actionPerformed...) sono automaticamente accodati nell'e.d.t..

Quindi bisogna prestare attenzione a non inserire nell'e.d.t. un task che chiami il metodo invokeAndWait perchè ciò causerebbe un deadlock.

Swing Worker

Per quanto riguarda i task onerosi , essi dovrebbero essere delegati ad appositi thread implementati tramite SwingWorker<T1, T2> (incluso nelle API da Java 6

http://java.sun.com/javase/6/docs/api/javax/swing/SwingWorker.html#get()). Solitamente sono azioni che non coinvolgono aggiornamenti diretti dell'interfaccia grafica.

Semplicemente si dovrebbe ereditare dalla classe SwingWorker facendo overriding del metodo doInBackground() il quale ritorna un oggetto di tipo T1 a cui si può accedere in seguito col metodo get. Inoltre il metodo done viene chiamato in automatico dopo la fine dell'esecuzione del metodo precedente. All'interno di tale thread, coi metodi get(timeout) oppure get() è possibile ottenere l'oggetto ritornato in precedenza (happens before relationship http://java.sun.com/docs/books/tutorial/uiswing/concurrency/simple.html).

Questo è utile se ad esempio l'oggetto ritornato deve essere condiviso con altri thread. (Attenzione che la versione di get senza argomento rimane bloccata in attesa del termine di doInBackground). Il tipo T2 identifica il tipo di eventuali oggetti ritornati ad interim (durante l'esecuzione del task) cioè temporanei. E' un altro modo utile per far comunicare e.d.t. e thread di tipo swing worker (tramite i metodi publish e process di SwingWorker). In codice:

```
SwingWorker worker = new SwingWorker<T1, T2>() {
    @Override
    public T1 doInBackground() {
            codice1;
        }
        return ObjectOfTypeT1;
    }

    @Override
    public void done() {
        try {
            T1 obj = get();
        } catch (InterruptedException ignore) {
            //Se il thread viene interrotto
          }
        catch (java.util.concurrent.ExecutionException e) {
            //Se doInBackground lancia un' eccezione
        }
    }
};
...
worker.execute(); //avvia il thread
```

In linea generale i task che si occupano di recuperare dati tramite Web Service dovrebbero essere implementazione di tale tipo di thread. Il metodo doInBackground

dovrebbe occuparsi del recupero dei dati pubblicati (recuperandoli interamente una volta sola con il valore di ritorno oppure incrementalmente con publish) che verrebbero poi recuperati in una volta con una get oppure incrementalmente tramite process.

Inoltre SwingWorker può memorizzare tramite una variabile interna il suo progresso (utile per implementare le barre di progresso) tramite setProgress(1...100) ed int getProgress().

Da notare che, al contrario di quando non era nelle API, ora Swing Worker non può essere riutilizzato, nel senso che ogni oggetto istanza di una classe derivata da SwingWorker può eseguire il suo codice solo una volta.

> "instances of the old `SwingWorker` class were reusable, while a new instance of `javax.swing.SwingWorker` is needed for each new background task."
>
> "public final void **execute**()
>
> Schedules this `SwingWorker` for execution on a *worker* thread. There are a number of *worker* threads available. In the event all *worker* threads are busy handling other `SwingWorkers` this `SwingWorker` is placed in a waiting queue.
>
> Note: `SwingWorker` is only designed to be executed once. Executing a `SwingWorker` more than once will not result in invoking the `doInBackground` method twice.";

Ovvero:

```
SwingWorker worker = new DerivazioneDiSwingWorker();
worker.execute();
worker.get();  //attende il termine di doInBackground;
worker.execute(); //non ha effetto, non viene rieseguito!
```

7.6.2 Accessibilità dell'interfaccia grafica

Per rendere i form di input e le funzionalità dell'interfaccia accessibili tramite tastiera si dovrebbe:

- per ogni bottone inserire un *mnemonic* (scorciatoie contestuali o locali indicate tramite una lettera sottolineata). Se ad esempio la scritta di un bottone è "Chiudi" allora premendo alt+C il bottone verrà premuto in automatico. In JavaBeans è sufficiente andare su proprietà->mnemonic ed immettere la lettera desiderata. (Attenzione al fatto che utilizzare più volte la stessa lettera renderà alcuni elementi inaccessibili).
- nel caso di una coppia label-textfield analogamente ai bottoni è bene settare un

mnemonic per permettere l'accesso diretto da tastiera. Si opera associando la label al testfield. Poi si setta il mnemonic per la label. In NetBeans è possibile modificare entrambi i campi dal pannello delle proprietà.

Ovviamente è possibile accedere ad ogni elemento tramite il tast tab e settare i mnemonics manualmente tramite gli appositi metodi delle classi.

- Ad ogni oggetto del menu principale dovrebbe essere associato uno *shortcut* (scorciatoie globali indicate tramite testo non editabile a fianco del comando).

Inoltre in generale si dovrebbero usare accomodamenti quali i *tooltips* (non sempre altrimenti possono diventare fastidiosi) per rendere più facile l'apprendimento dell'interfaccia

7.6.3 Icone dell'interfaccia

Le icone utilizzate nell'interfaccia vanno salvate all'interno della cartella img. Rispetto alla radice della cartella che si trova in working_copy/src/GUIDidattica contenente il progetto (branch, tags o trunk) essa è situata in src/nu/mine/egoless/didattica/app/gui . Per garantire che il recupero delle icone per il loro utilizzo sia indipendente dal S.O. e FS utilizzato, da NetBeans si deve scegliere l'opzione classpath (e non File) nella schermata Properties->Icon relativa all'oggetto in questione
Quando possibile le icone dovrebbero essere scelte tra quelle presenti nel repository di Java Look and Feel([JLF] Volume I, Appendice B).

7.6.4 Clonazione di un JavaBeans

La clonazione di un oggetto deve avvenire tramite il metodo clone() di Obect. Esso crea solo una copia lazy senza scendere in profondità (cioè copia solo i reference in caso di variabili oggetto). Per cui bisogna farne l'overriding e ci si deve assicurare di richiamare anche il relativo clone() della sottoparte dell'oggetto rappresentante la sovraclasse. E' necessario che la classe implementi l'interfaccia Cloneable che funge semplicemente da tag ed evita il lanci dell'eccezione CloneNotSupportedException. In codice:

```
public class Clona implements Cloneable{
      int i=10;
      Integer x=new Integer(50);
      Clona(){};
      Clona(int _i, Integer _x){i=_i; x=_x;}
      public Object clone() throws CloneNotSupportedException{
            Clona t=(Clona)super.clone();
            t.i=this.i;
            t.x=new Integer(this.x);
            return t;
```

```
        }
}
```

7.7 Metriche

Verranno ora descritte le metriche che verranno valutate durante la codifica (e la progettazione quando questo è possibile) e i valori che dovrebbero assumere nel prodotto sviluppato.

7.7.1 Complessità ciclomatica

Per una introduzione alla *complessità ciclomatica* riferirsi al documento [CC SEI].

Per ogni classe sviluppata, la complessità ciclomatica deve essere compresa tra 1 e 10. Per ora non sono stati individuati casi in cui sono ammesse complessità ciclomatiche più elevate.

7.7.2 Afferent/Efferent couplings

La misura Efferent couplings deve rientrare nell'intervallo [0, 15]. Questo per evitare un eccessivo accoppiamento tra componenti.

La misura Afferent couplings deve rientrare nell'intervallo [0, 30]. La limitazione superiore è più alta di quella per EC ed è fissata per evitare che un componente abbia troppa responsabilità.

7.7.3 NCSS

I limiti sono:

- 50 per i metodi;
- 1500 per le classi;
- 2000 per i file.

7.7.4 Statement e Branch Coverage

Nell'esecuzione dei test su un singolo modulo, il branch coverage deve essere del 100%.

Uno statement coverage del 100% è sicuramente desiderabile ma non sempre raggiungibile. E' tollerata copertura inferiore se gli statement non coperti riguardano situazioni non facilmente riproducibili, come:

- eccezioni che è impossibile provocare (vedi `DatatypeConfigurationException`).

8 Conduzione della verifica

8.1 Risoluzione di discrepanze

Chiunque rilevi una discrepanza deve aprire una discussione su Google Groups per proporre come risolvere la discrepanza. Dopo aver deciso come procedere la discussione verrà chiusa e si provvederà a verbalizzare la decisione presa utilizzando la pagina https://egoless.mine.nu/PQS_doc/wiki/DiscrepanzeRisolte.

Se necessario si procederà ad un aggiornamento delle norme di progetto.

8.2 Segnalazione di anomalie

Le anomalie vanno segnalate inserendo un *ticket* in Trac. Per le norme sull'assegnazione dei ticket fare riferimento a [U].