Ingegneria del Software con Laboratorio (12 crediti) – Programma a.a. 2010-11
Prova Scritta del 20 giugno 2012

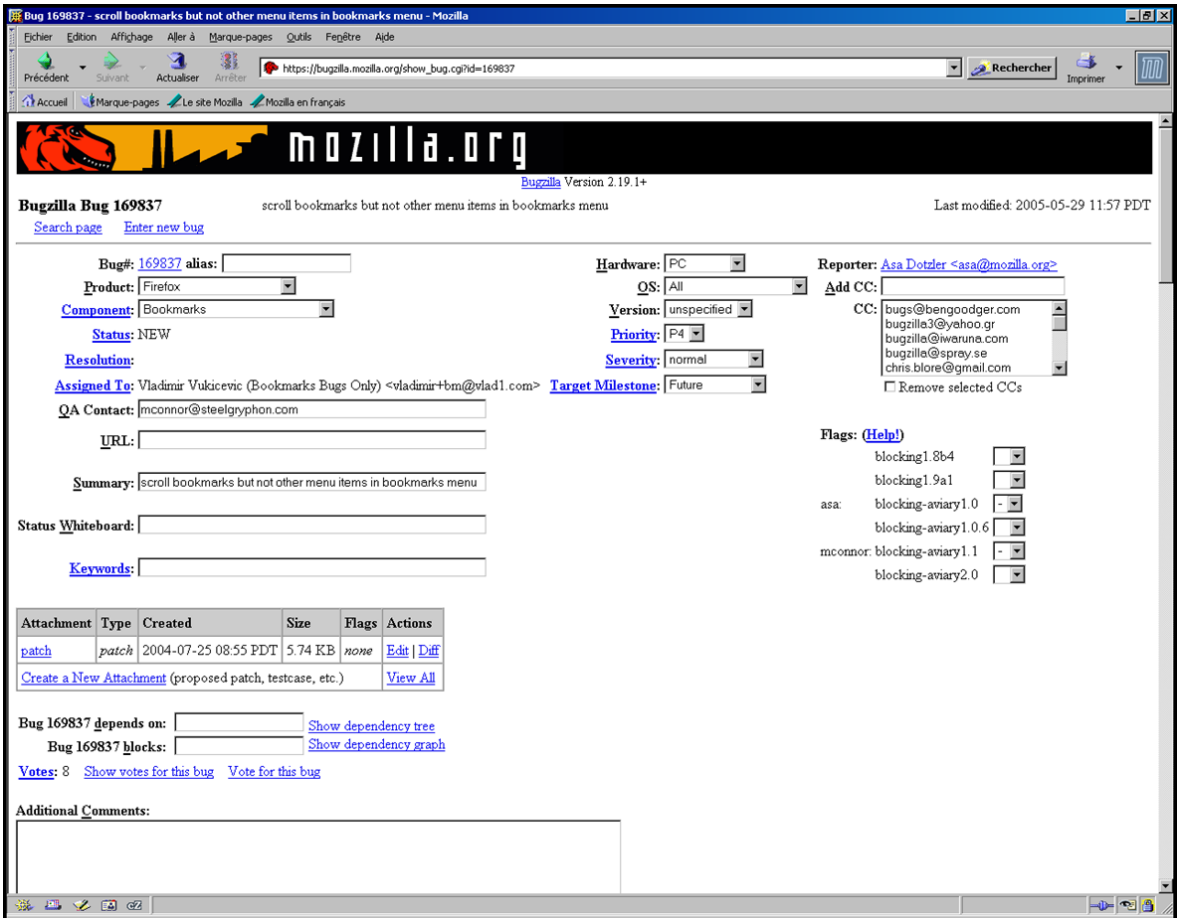
Tempo a disposizione: 3 ore (Solo parte generale: 1 ora e 30', solo UML: 1 ora e 30').

La parte generale oltre ai 2 esercizi prevede 1 domanda che verrà distribuita 30' prima della scadenza per la consegna – da quel momento non sarà più consentita la consultazione del materiale.

Svolgere parte UML e parte generale su fogli separati.

Esercizio 1 – Parte UML

Bugzilla è un bugtracker, cioè un software utilizzato per tenere traccia dei bug (difetti software) all'interno di un progetto software.



Un bug è composto da un **id**, un **titolo** e da un **messaggio**, un po' come una email. Al primo messaggio, che normalmente descrive il problema, possono essere inseriti (uno o più) **commenti** in successione, allegati diversi **file** ed un **URL** esplicativo. Ogni bug ha un livello di **severity** (bloker, critical, major, normal, minor, trivial, enhanchement) ed una **priority** (da 1 a 5). Il valore di severity non è di certo l'unica proprietà fondamentale di un bug.
L'aspetto forse più importante è lo **stato** di un bug. Un bug è infatti associato ad uno **stato** che ne descrive la situazione attuale del bug e determina la vita del bug stesso, ovvero un "bug's life cycle"**.** Alcuni possibili stati di un bug sono*: new, assigned, resolved, fixed* e *closed.*

Ogni bug "b" può avere una lista di altri bug che devono essere fissati prima che "b" possa essere fissato (**depends on**). Simmetricamente, ogni bug b può avere una lista di altri bugs che possono essere fissati solo dopo che b è stato fissato (**blocks**)

a) **S**cegliere il diagramma UML più opportuno e rappresentare il "modello" dei bug descritto sopra
b) Estendere il modello precedente con le persone. L'insieme di persone coinvolte include il **reporter** del bug, lo sviluppatore che è stato incaricato a fissarlo (**assignedTo**), il software tester (chiamato **qa;** che è un abbreviazione di quality assurance) che testa la soluzione e una lista di persone che sono interessate a ricevere le notifiche relative al bug (**cc**). Aggiungere gli attributi che ritenete necessari.
c) **S**upponendo di dovere implementare un bugtraker e basandosi sul modello realizzato al punto b, scrivere in pseudocodice l'algoritmo che stampa per ogni sviluppatore tutti i bug a lui associati (cioè assigned to) non ancora closed in ordine di priorità

Esercizio 2 – Parte UML

Un bug ha una vita propria che comincia nel momento in cui il bug viene inserito nel bugtraker: il nuovo bug viene automaticamente associato allo stato di **new**. In questo caso il bug è stato inserito ed aspetta un volontario che lo prenda in gestione. A seconda delle configurazioni un bug può essere inserito di default come **unconfirmed**, in attesa che qualcuno ne confermi l'esistenza ad esempio tentando di riprodurlo sulla propria versione del programma. Una volta che il bug viene confermato come tale può essere assegnato **(assigned)**. Questo significa che qualcuno è ora incaricato di valutare l'entità del problema ed attivarsi per risolverlo. Una volta assegnato il bug può essere a sua volta riassegnato ad un altro sviluppatore o contrassegnato come risolto (**resolved**). Risolvere un bug può significare diverse casistiche. Il bug può essere contrassegnato come risolto in quanto non sussiste (**invalid**), non è riproducibile (**worksforme**), è un duplicato di uno esistente (**duplicate**) oppure è incompleto (**incomplete**). Ovviamente un bug può essere semplicemente risolto in quanto è stata applicata la correzione (**fixed**) necessaria. Un bug risolto, una volta rilasciata la versione, può essere chiuso (**closed**) ma allo stesso tempo potrebbe "resuscitare" in futuro (**reopened**).

a) In quale casi un bug potrebbe essere **reopened** da uno sviluppatore? Commentare.
b) Quale è il diagramma UML più opportuno per rappresentare il ciclo di vita di un bug così come descritto sopra?
c) Con il diagramma scelto al punto b) **rappresentare** il ciclo di vita di un bug (la descrizione è stata presa da internet e siamo coscienti del fatto che in certi casi è ambigua. Risolvere le ambiguità è parte dell'esercizio)

Esercizio 3 – Parte generale

Considerare l'operazione *Caricamento foto* di un social network descritta come segue:

L'utente seleziona il file dal proprio HD e clicca su condividi (la foto deve essere in formato JPEG o BMP e deve avere una dimensione inferiore a 5 MB). L'utente può annullare in ogni momento il caricamento della foto premendo il pulsante chiudi X
Una volta caricata, il sistema mostra la foto sulla bacheca dell'utente. In particolare, la foto diviene visibile nella bacheca dell'utente ai suoi amici a seconda delle seguenti impostazioni

- Tutti: visibile a tutti
- Solo amici: visibile solo agli amici
- Amici degli amici: visibile agli amici fino al secondo livello

a) Specificare il caso d'uso "Caricamento foto", corrispondente all'operazione descritta. Considerare sia lo scenario principale che gli scenari alternativi.
b) Modellare tramite tabelle di decisione il comportamento atteso dell'operazione.
c) Utilizzare il caso d'uso/le tabelle per ottenere casi di test per l'operazione. Motivare le scelte effettuate.

Esercizio 4 – Parte generale

Per il seguente problema

> *Filtraggio di E-mail.* Un sistema di e-mail filtra i messaggi in arrivo utilizzando una whitelist (i messaggi i cui mittenti sono nella whitelist vengono accettati), una blacklist (i messaggi i cui mittenti sono nella blacklist vengono cancellati), e uno Spamassassin tool (i messaggi che non passano questo controllo vengono contrassegnati come spam).

descrivere l'architettura di sistema nel seguente modo

a) Specificare il nome dello stile architetturale che ritenete più appropriato per il sistema.
b) Disegnare un diagramma che descrive l'architettura del sistema in accordo a tale stile, commentando brevemente a parole il funzionamento.
c) Evidenziare i principali punti di forza (vantaggi) e di debolezza (svantaggi) dell'uso di tale architettura.

Domanda 5 – Parte generale

Discutere lo sviluppo a fasi, mettendo in evidenza analogie e differenze nonché vantaggi e svantaggi rispetto allo sviluppo a cascata.