Politecnico di Milano

Project Planning 1.4

Progetto di Ingegneria del Software 2

SWIMv2: Small World Hypothesis Machine v2

Autori:
Bulla Jacopo
Caio Davide
Cappa Stefano

Professore:
Mottola Luca

2 Project Planning

Lo scopo di questo documento è mostrare il processo di pianificazione di un progetto chiamato SWIMv2, per ottenere un'equa suddivisione del carico di lavoro tra i singoli membri che ne fanno parte. Per raggiungere l'obiettivo prefissato è stato necessario stimare l'effort richiesto, in base alle esperienze precedenti.
Scelti gli elementi del team di sviluppo, tenendo conto di fattori sociali e conoscenze personali per ottimizzarne la produttività, è avvenuta la divisione in task.
Ad ognuno di essi è stato associato un membro del team, specificandone una stima dell'effort.

Indice

1. Introduzione

Il progetto consiste in un social network (chiamato SWIMv2) che permette di cercare e/o offrire aiuto per svolgere specifici lavori.
SWIMv2 è l'aggiornamento di un sistema limitato e unicamente su invito. Lo scopo di questa versione è di realizzarne una pubblica ed accessibile da tutti.

In tutto il documento saranno utilizzati i seguenti termini:

- **Visitatore: utente non registrato, cioè accede alla piattaforma senza inserire dati personali e registrarsi e di conseguenza non può utilizzare le funzionalità del sistema (a parte la ricerca per i visitatori**).
- **Utente**: utilizzatore registrato di SWIMv2 che ha creato un "profilo".
- Profilo: pagina personale di un Utente in cui vi sono informazioni personali.
- Registrazione: procedura con cui un Visitatore può diventare un Utente e di conseguenza avrà accesso a tutti i vantaggi del possedere un profilo personale, come ad esempio la ricerca di aiuto e il rilascio feedback.
- Social Network: quando si utilizzerà questo termine, ci si riferirà ad un generico sistema in cui vi sono diversi utenti che interagiscono tra loro.
- Feedback: è il metodo con cui gli utenti che hanno richiesto aiuto esprimono una valutazione (voto da 1 a 5 ed eventualmente un commento) verso chi l'ha fornito.
- **Insieme generale delle abilità: lista di abilità che gli Utenti possono selezionare in fase di Registrazione.** Tale insieme rappresenta tutte le possibili voci selezionabili.
- **Insieme personale delle abilità: lista di abilità associate ad un singolo Utente. Esse sono visualizzabili sul profilo personale dell'Utente** e costituiscono un sottoinsieme dell'insieme generale delle abilità (in casi particolari potrebbero coincidere).
- **Proposta abilità: richiesta all'amministratore dell'aggiunta di un elemento nell'insieme generale delle abilità**
- **Suggerimento: parola utilizzata per indicare gli utenti che il sistema mostra dopo aver stretto amicizia con un altro Utente. Questa funzione sarà definita meglio nel Design Document.**

I membri sono suddivisi in Utenti, Visitatori ed un solo Amministratore. Ad ogni categoria sono associate mansioni ben definite:

- Utenti: **devono scegliere almeno un'abilità** da inserire nel proprio insieme personale, cercare altri utenti, inviare/rispondere a richieste di amicizia o aiuto tra gli amici o nell'intera rete di utenti. Per far si che la qualità del servizio si mantenga sempre elevata, essi hanno la possibilità di fornire un feedback dopo aver ricevuto aiuto. Inoltre, **possono modificare il proprio profilo (di conseguenza anche l'insieme personale delle abilità).** Un'altra funzionalità è la possibilità di **proporre l'aggiunta di nuove abilità nell'insieme globale**. Infine, quando un utente stringe amicizia con un altro (senza aver ricevuto precedenti suggerimenti), entrambi visualizzano una lista di amici consigliati.
- Visitatori: possono solamente visualizzare la lista degli utenti e le loro abilità, senza poter accedere al profilo.
- Amministratore: ha il compito di definire l'insieme generale delle abilità ed eventualmente **confermare/rifiutare proposte di aggiunta di nuove abilità**.

1.1. Requisiti e specifiche

Non vi sono particolari restrizioni sulla progettazione, a meno delle tecnologie da utilizzare. Infatti, il software dovrà essere realizzato con:

- JEE (Java Enterprise Edition)
- EJB (Enterprise Java Bean)
- Interfaccia utente come "Web application" oppure come "Java application"

Deliverables	Consegna
27/10/2012	Project Planning
24/11/2012	RASD
22/12/2012	Design Document (DD)
26/01/2013	Implementazione software
07/02/2013	Testing

Inoltre, è necessario seguire una precisa Deadline (vedi tabella) rilasciando delle "Deliverables" in date prestabilite. Ad ogni consegna, il committente riceverà il materiale richiesto secondo il piano prestabilito. Lo scopo è mantenerlo aggiornato sui progressi e sulle scelte progettuali.

2. Fase preliminare

Prima di procedere alla pianificazione è necessario organizzare nel modo più efficiente il team di sviluppo e analizzare gli strumenti a disposizione.

2.1. Team

E' composto da tre persone provenienti da aree geografiche differenti, ma con in comune lo stesso luogo di studio, cioè il "Politecnico di Milano". Nessuno di essi ha problemi rilevanti di trasporto e per questo l'incontro dei tre membri a Milano può avvenire senza problemi in qualunque momento.
Tutte e tre i membri si conoscono da anni, perché studenti negli stessi corsi, facilitandone l'integrazione e i rapporti. Durante la formazione del team non è stato considerato solo l'aspetto sociale, nonostante sia molto rilevante, ma anche quello delle esperienze passate.

2.2. Esperienze personali

Come già anticipato, i membri del team hanno esperienze molto simili grazie al luogo di studio in comune e allo sviluppo degli stessi progetti negli anni precedenti.

A seguire una breve descrizione delle esperienze acquisite da ogni membro:

- Bulla Jacopo: studente d'ingegneria informatica che ha sviluppato il progetto d'ingegneria del software 1, ha conoscenze nella progettazione di software tramite diagrammi UML, sviluppo in java e di applicazioni web grazie al corso di "Hypermedia Application", dedicandosi particolarmente sulla parte logica. Ha competenze in Java, Javascript, ASP, C, C# e in questi IDE: Eclipse, Visual Studio.
- Caio Davide: studente d'ingegneria informatica che oltre aver svolto il progetto d'ingegneria del software 1, ha prodotto con Jacopo il progetto di "Hypermedia Application" focalizzandosi soprattutto sulla progettazione del sistema, lato client, server e database. Per questo motivo è il membro più adatto nel design di questo progetto. Ha conoscenze in UML, Java, VisualBasic e C e in questi ambienti di sviluppo: VisualStudio ed Eclipse.
- Cappa Stefano: come i primi tre membri, ha sviluppato il progetto d'ingegneria del software suddividendo il lavoro in modo equo tra logica e grafica (in swing). Ha prodotto in modo indipendente alcuni software, dedicandosi soprattutto alla GUI in Swing e ha conoscenze in UML, Java, C, C#, Objective-C, Android e Windows Phone, usando soprattutto IDE come: Visual Studio, Eclipse e NetBeans.

2.3. Tecnologie

Le tecnologie a disposizione sono le seguenti:

- Computer: Asus Zenbook, Samsung Serie 9 e un MacBook Pro
- Sistemi operativi: Windows, Mac, Linux
- Connessioni a internet: fibra ottica 10MB
- Browser: Internet Explorer, Mozilla Firefox, Apple Safari, Google Chrome
- Strumenti di comunicazione: tre Smartphone, Skype, Facebook
- Servizi di archiviazione dati: Dropbox
- IDE: Eclipse Juno e NetBeans
- Repository online di Google Code con Subversion (SVN)

2.4. Stima di effort

La stima di effort è stata realizzata considerando il progetto "Ingegneria del Software 1" sviluppato un anno fa. Poiché tutti i membri hanno lavorato in team differenti, è stata considerata una media.
E' da tenere presente che quel progetto era il primo che i membri hanno dovuto sviluppare, inoltre era anche con tecnologie differenti, quindi l'effort richiesto deve essere considerato in modo obiettivo.

Fase	Effort
Pianificazione e analisi requisiti	5
Progettazione	50
Implementazione	400
Testing	20

3. Tasks

Lo sviluppo del progetto è stato suddiviso, secondo il "Modello a Cascata", in cinque fasi.
Ognuna di esse prevede una Deadline precisa ed una suddivisione in sotto-tasks.

Tasks:
1. Project Planning
2. Analisi dei requisiti
3. Design
4. Implementazione
5. Testing

3.1. Project Planning

Deadline: 27/10/2012

E' la fase iniziale in cui è redatto il documento di Project Planning.
Esso contiene: la descrizione dei principali tasks con relativa suddivisione in sotto-tasks, la stima del tempo di sviluppo di ogni fare tramite i "Diagrammi di Gantt" e la revisione finale per verificarne la congruenza.

3.2. Analisi dei requisiti

Deadline: 24/11/2012

Fase in cui sono analizzati gli obiettivi e raccolte le informazioni per costruire i modelli e stendere il Requirements Analysis and Specification Document (RASD).

La suddivisione decisa è la seguente:
1. Raccolta delle informazioni al fine di comprendere meglio il dominio applicativo
2. Analisi degli obiettivi e dei vincoli di progetto
3. Creazione degli scenari, casi d'uso e modelli (UML e Alloy)
4. Revisione

3.3. Design

Deadline 22/12/2012

Task molto importante in cui è definito il funzionamento dell'intero sistema. La suddivisione è stata realizzata per separare al meglio le attività che ne fanno parte, permettendo un controllo più preciso e mirato, fino alla redazione del Design Document (DD).

Sotto-tasks:
1. Architettura di sistema
2. Persistenza dei dati (database)
3. Modello di navigazione
4. Diagrammi BCE e di sequenza
5. Revisione

3.4. Implementazione

Deadline: 26/01/2013

E' il momento in cui i vari membri del team si occupano di scrivere il codice, seguendo la pianificazione e le tempistiche definite in precedenza.
Poiché la stima iniziale è solitamente imprecisa, questa fase comprende anche le eventuali revisioni della documentazione.

Suddivisione in sotto-tasks:

1. Scrittura della logica in Java
2. Scrittura della grafica in Java
3. Documentazione (javadoc)
4. Redazione del documento d'installazione
5. Redazione del manuale d'uso
6. Definire in modo opportuno i casi di test
7. Testing e Revisione

3.5. Testing

Deadline: 02/02/2013

Fase in cui è eseguito il test del progetto in questione e di quello di un altro team di sviluppo.
Testare il prodotto finale è un'attività cruciale per lo sviluppo di un software stabile e ben realizzato.
Per questo motivo è stato distribuito il lavoro in modo ottimale, dividendo il Testing nei seguenti sotto-tasks:

1. Controllare il corretto funzionamento e individuare i problemi
2. Scrittura documentazione sui risultati ottenuti

4. Distribuzione del carico di lavoro

La suddivisione del carico di lavoro per ogni membro del team di sviluppo è rappresentata dalla tabella che segue. Per evidenziare meglio l'organizzazione generale e l'effort richiesto, sono stati stimati i carichi di lavoro individuali per ogni sotto-tasks. Inoltre, il **"Diagramma di Gantt" allegato a questo documento**, mostra nel dettaglio l'allocazione temporale dei sotto-tasks, evidenziandone anche le dipendenze (freccia che collega una barra ad un altra).

Tasks	Nome	Descrizione	Membro 1 Bulla	Membro 2 Caio	Membro 3 Cappa	Effort	Deadline
	Analisi requisiti		10	9	10	29	24/11/2012
1.1	Raccolta delle informazioni	Analisi dei requisiti e degli obiettivi per la stesura del RASD	1	1	3	5	
1.2	Analisi degli obiettivi e dei vincoli		2	3	1	6	
1.3	Creazione modelli		6	4	5	15	
1.4	Revisione		1	1	1	3	
	Design		38	40	38	116	22/12/2012
2.1	Architettura del sistema	Progettazione del software e stesura del DD	5	5	5	15	
2.2	Persistenza dei dati		17	17	17	51	
2.3	Modello di navigazione		2	4	2	8	
2.4	Diagrammi BCE e di sequenza		4	1	3	8	
2.5	Revisione		10	13	11	34	
	Implementazione		59	59	59	177	26/01/2013
3.1	Scrittura codice logica	Implementazione, realizzazione della documentazione e definizione "casi di test"	10	42	34	86	
3.2	Scrittura codice grafica		38	6	14	58	
3.3	Documentazione		2	2	2	6	
3.4	Redazione documento installazione		0	3	0	3	
3.5	Redazione manuale d'uso		4	0	5	9	
3.6	Definizione "casi di test"		4	5	3	12	
3.7	Testing e Revisione		1	1	1	3	
	Testing		30	29	30	89	02/02/2013
4.1	Test funzionale e ricerca bug	Testing del software di un altro team di sviluppo	20	20	20	60	
4.2	Scrittura documentazione		10	9	10	29	
		TOTALE per membro	**137**	**137**	**137**	**411**	

Dopo la fase di Project Planning, distribuita in quattro giorni e svolta da tutti i membri in modo uniforme, è stato rilasciato questo documento in data 27/10/2012 che mostra l'effort e la ripartizione delle mansioni dei vari membri del team di sviluppo.

Il primo Task nella tabella è l'analisi dei requisiti, in cui si è cercato di suddividere il carico di lavoro in modo efficiente, per permettere lo sviluppo parallelo del RASD da parte dei tre membri del team.

A seguire vi è la fase di Design che vede una distribuzione dell'effort più uniforme nelle prime due fasi. Questa scelta è motivata da fattori sociali e puramente qualitativi. Più precisamente, dai progetti precedenti, si è notata una collaborazione più stretta tra i membri che hanno avuto modo di confrontarsi già dalle prime fasi di progettazione, cioè nel momento in cui si gettano le basi.
Il fattore qualitativo, oltre ad essere legato all'interazione sociale dei membri, evidenzia che lo scambio d'idee conduce a diverse proposte di progettazione, facendo sì che il gruppo possa scegliere la migliore tra un insieme molto più vasto di alternative.
Inoltre, è importante evidenziale che è stato allocato un valore elevato di effort per la progettazione del sistema e la stesura del Design Document (DD) perché un software ben progettato richiede un minore sforzo d'implementazione, soprattutto in caso di future modifiche.
Questo conduce a un notevole risparmio di tempo e a una maggiore probabilità di rispettare le scadenze imposte, senza sottovalutare il minor costo del prodotto finale.

La fase d'implementazione presuppone un'ottima progettazione del sistema. Ipotizzando tale condizione come risolta nel task di Design, è stato suddiviso il carico di lavoro in modo da permettere lo sviluppo parallelo delle varie parti del software. Questa scelta consente di ridurre notevolmente i tempi di scrittura del codice, mantenendo anche una qualità superiore, poiché il singolo programmatore svolge totalmente il compito prefissato per creare un modulo del prodotto finito che poi si andrà ad integrare con gli altri. Ovviamente, tutto ciò riduce la probabilità di scrittura/modifica contemporanea della stessa parte del software.
La stessa tecnica è stata adottata anche nella produzione della documentazione.

Per migliorare la stabilità del software, i membri del team di sviluppo realizzeranno indipendentemente dei test in JUnit della "logica del programma" con lo scopo d'individuare eventuali bug durante l'implementazione.
Al termine di tale fase, l'intero team si occuperà di verificare tutta la documentazione scritta nei task precedenti, ma soprattutto di realizzare "casi di test" in base agli scenari che saranno definiti durante la stesusa del RASD (deliverable 2).
Ovviamente, conclusa la procedura, sarà redatto un documento esaustivo per mostrare nel dettaglio ogni caso studiato e ogni bug individuato.

Per garantire una maggiore qualità dei prodotti finiti, si prevede sempre un certo numero di ore di testing, svolto da un altro gruppo su un progetto diverso. In questo caso il team di sviluppo si occuperà di esaminare un altro software ancora non rilasciato, mentre questo progetto sarà testato da un altro team. Anche questa fase terminerà con una documentazione esaustiva dei casi considerati e dei difetti individuati.