Configuration Management

Versione: 0.3.2 ***Data:*** 21/12/2018
Autore: Pellizzari Luca ***Responsabile:*** Di Benedetto Gianluca

Indice

1. Organizzazione del team

Il team GL^3 è composto da tre membri: Baradel Luca, Di Bendetto Gianluca, Pellizzari Luca.

Il team si riunisce due volte a settimana:

1) una subito dopo l'incontro con il cliente per fare un'attività di brainstorming quindi per cercare di chiarire gli eventuali dubbi e per individuare gli obiettivi comuni del lavoro;
2) una precedente al prossimo incontro con il cliente per implementare il lavoro pianificato nell'incontro precedente.

Nella prima delle due sedute inoltre si pianifica il lavoro individuale che i membri del team dovranno svolgere entro la data prevista per il secondo incontro in modo da poter discutere (ed eventualmente modificare) i risultati ottenuti prima di mostrarli al cliente.

Il conduttore scelto per il gruppo è Gianluca Di Benedetto.

Gli altri ruoli dei membri del team sono i seguenti:

- Baradel Luca: documentazione, gestione versioni, database;
- Di Benedetto Gianluca: responsabile del CM, gestore file, dati di test e relativa documentazione;
- Pellizzari Luca: codice, change request, software (tool) di supporto.

Lo strumento scelto dal team per gestire i file relativi al progetto è GitHub; inoltre è stato creato un gruppo Whatsapp per accordarsi sulle date degli incontri e per ulteriori comunicazioni.

2. Identificazione item

Gli item identificati in questa prima fase del progetto sono i seguenti:

1) Documentazione:
    a. Documento dei requisiti;
    b. Analisi dei requisiti;
    c. Casi d'uso UML;
    d. Manuale utente;
    e. Verbale di interazione con il cliente;
    f. Offerta in relazione al mandato;
    g. Documento di progetto;
    h. Documento di avanzamento del progetto;
    i. Configuration Management.
2) Change request;
3) Codice;
4) Gestione versioni;
5) Dati di test e relativa documentazione;
6) Database;
7) Gestore file;
8) Software (tool) di supporto.

Di seguito vengono elencate le caratteristiche principali degli item e il modo in cui questi item verranno gestiti dal team di sviluppo.

2.1. Documentazione

Tutta la sezione legata alla documentazione verrà salvata in una repository condivisa fra i membri del team su GitHub, per ogni documento vengono mantenute tutte le sue versioni (che corrispondono a file con indici di versione di primo o secondo livello diversi) in formato pdf e l'ultima versione di ogni documento viene salvata anche in formato docx in modo che sia facilmente modificabile per passare alla versione successiva.

Per ogni documento descritto in questa sezione è presente un template all'interno della repository del team su GitHub nella cartella Documentazione/Templates. In questo modo è possibile per ogni documento avere una versione pronta con intestazione (che comprende: data, autore, responsabile, numero di versione e logo del gruppo) e indice già scritti.

2.1.1. Documento dei requisiti

1) Il nome del file sarà DocRequisiti_data.pdf (esempio: DocRequisiti_2018-11-23.pdf);
2) I requisiti vanno classificati in base allo stakeholder (cliente, sviluppatori, utenti finali, eccetera) e divisi in funzionali e non funzionali;
3) Ogni requisito viene scritto in linguaggio naturale strutturato e deve essere inserito nel documento dei requisiti insieme ad un'eventuale motivazione;
4) Un requisito può essere associato ad una priorità che indica il suo valore di business.

2.1.2. Analisi dei requisiti

1) Il nome del file sarà AnalisiRequisiti_data.pdf (esempio: AnalisiRequisiti_2018-11-23.pdf);
2) Il documento di analisi dei requisiti deve contenere, per ogni requisito presente nel documento dei requisiti, una descrizione che ci permette di trasformare il requisito del cliente in un requisito del sistema. In altre parole, ogni azione descritta dal cliente deve essere mappata in un'azione che deve compiere il sistema;
3) Ad ogni documento di analisi dei requisiti deve essere associata una data e un codice che identifica la versione del documento (come descritto nella gestione delle versioni).

2.1.3. Casi d'uso UML

1) Il nome del file sarà CasiUso_data.pdf (esempio: CasiUso_27-11-2018.pdf);
2) Il file contiene un certo numero di rappresentazioni con sintassi UML che descrivono le azioni messe a disposizione dal sistema;
3) Il file è realizzato con il tool draw.io disponibile online; oltre al file in formato pdf verrà memorizzato anche un file in formato XML in modo che possa essere successivamente modificato/aggiornato.

2.1.4. Manuale utente

1) Il nome del file sarà ManualeUtente_data.pdf (esempio: ManualeUtente_2018-11-23.pdf);
2) Tutte le possibili operazioni che l'utente può svolgere tramite il sistema software che verrà sviluppato devono essere descritte in un manuale;
3) Ogni aggiornamento del sistema che modifica le funzionalità messe a disposizione dallo stesso o che cambia il modo di effettuare alcune operazioni deve essere descritto nel manuale: fra questo e il sistema non devono esserci inconsistenze;
4) Ogni versione (non variante) del sistema deve essere associato ad una versione del manuale.

2.1.5. Verbale interazione con il cliente

1) Il nome del file sarà VerbInteraz_data.pdf (esempio: VerbInteraz_2018-11-23.pdf);

2) La struttura deve essere la seguente:
    a. Data (ed eventualmente luogo) dell'incontro;
    b. Presenti;
    c. Oggetto;
    d. Temi trattati;
    e. Decisioni prese;
    f. Prossimi passi;
    g. Prossimi incontri.
3) Questo verbale è gestito internamente dal team degli sviluppatori quindi il suo codice identificativo è a due livelli: il primo livello è diverso per ogni verbale relativo ad un incontro con il cliente e il secondo indica il numero di revisione interna.

2.1.5. Offerta in relazione al mandato

1) Il nome del file sarà Offerta_data.pdf (esempio: Offerta_2018-11-23.pdf);
2) L'offerta viene formulata in linguaggio naturale strutturato senza la presenza di termini tecnici legati all'implementazione;
3) Deve contenere la data dell'offerta, l'autore del documento, il codice di versione ed eventualmente un glossario con i riferimenti a regolamenti o fonti esterne;
4) Contiene una descrizione generale della soluzione proposta che descrive le funzionalità del sistema dal punto di vista dell'utilizzatore. Queste funzionalità dipendono dai requisiti forniti dal cliente e dalla loro successiva analisi svolta dal team di sviluppo.

2.1.6. Documento di progetto

1) Il nome del file sarà DocProgetto_data.pdf (esempio: DocProgetto_2018-11-23.pdf);
2) Il documento viene scritto in linguaggio naturale strutturato senza la presenza di termini tecnici legati all'implementazione;
3) Deve contenere la data, l'autore del documento, il codice di versione ed eventualmente un glossario con i riferimenti a regolamenti o fonti esterne;
4) A differenza del documento: "Offerta in relazione al mandato", descritto al punto precedente, questo documento contiene una proposta di soluzione, ovvero descrive l'idea progettuale emersa dall'analisi dei requisiti ma senza fare un'offerta esplicita al cliente. Se l'idea di soluzione proposta in questo documento viene approvata dal cliente allora nell'incontro successivo verrà fatta un'offerta.

2.1.7. Configuration Management

1) Il nome del file sarà CM_data.pdf (esempio: CM_2018-11-23.pdf);
2) Il file contiene il numero di versione del CM, la data, il nome dell'autore della versione corrente, il nome del responsabile e le varie sezioni che costituiscono il corpo del documento.

2.1.8. Documento di avanzamento del progetto

1) Il nome del file sarà DocAvanzamento_data.pdf (esempio: DocAvanzamento_2018-11-23.pdf);
2) Il file contiene il numero di versione del documento, la data, il nome dell'autore della versione corrente, il nome del responsabile e il corpo del documento.
3) Il "corpo del documento" citato al punto precedente consiste di un elenco di date con descrizione dei passaggi portati a termine in quelle date ad esempio: "Consegnata al cliente versione 0.1 del documento di progetto".
4) Dal momento che le consegne al cliente corrispondono ad un incremento di 1 nel valore dell'indice di versione di secondo livello (o di primo livello se la consegna comprende nuove funzionalità), è

fortemente consigliato indicare nel documento di avanzamento il numero di versione di un item ogni qual volta il suo indice di secondo livello (o di primo livello) viene incrementato.

2.2. Change request

1) Contengono: un codice univoco (numero della richiesta), autore della richiesta, data della richiesta, priorità, descrizione, moduli coinvolti (opzionale se esterna), tempo di lavoro stimato e successivamente data approvazione/rifiuto, data inizio modifiche;
2) Possono essere interne (esigenze degli sviluppatori) o esterne (esigenze del cliente);
3) Se sono esterne si pianifica la data di inizio delle modifiche (in base al tempo di lavoro stimato e alla priorità della modifica), se interne prima devono essere approvate (votazione a maggioranza fra gli sviluppatori);
4) Possono essere presentate in formato cartaceo durante gli incontri (fra gli sviluppatori o con il cliente) oppure in formato digitale (ad esempio pdf) ma in entrambi i casi devono contenere le informazioni elencate al punto 1.

2.3. Codice

1) Il codice sorgente legato al progetto viene gestito tramite GitHub, ogni membro del team può leggerlo e modificarlo a condizione che gli altri membri siano d'accordo (se necessario va effettuata una change request per poter modificare il codice);
2) Ogni volta che uno degli sviluppatori effettua una modifica o aggiunge nuovo codice deve registrare i cambiamenti tramite la procedura per la gestione delle versioni.

2.4. Gestione versioni

È stato scelto di gestire le versioni delle componenti del sistema nel seguente modo:

1) Ogni versione è identificata da un codice numerico a tre livelli (solo i primi due quando presentata al cliente);
2) Il terzo livello contiene il numero della revisione interna su un certo item, partendo da zero;
3) Il secondo livello contiene il numero della consegna al cliente (può essere zero solo in fase di sviluppo, quindi la prima consegna di un certo oggetto avrà 1 come codice di secondo livello);
4) Il codice di primo livello identifica un oggetto che mantiene obiettivi e caratteristiche fondamentali per tutte le sue consegne, questo codice è zero fino a che non si ottiene una versione completa, funzionante e utilizzabile dal cliente.
5) Esempio:
    a. 0.2 è la seconda consegna di un prototipo con alcune funzionalità minori ma che non può ancora essere considerato un sistema;
    b. 0.2.9 è la nona revisione interna sul prototipo sopra citato;
    c. 1.0.3 è la terza revisione interna della versione completa e funzionante del sistema non ancora rilasciata;
    d. 1.1 è la prima versione completa e funzionante del sistema consegnata al cliente;
    e. 3.1.15 è la quindicesima revisione interna della prima consegna della terza versione completa e funzionante del sistema.

2.5. Dati di test e relativa documentazione

2.5.1. Tipi di test e responsabili nominati

I possibili tipi di test che verranno eseguiti sul sistema sono i seguenti:

1) Black – box (funzionale);

2) White – box (strutturale);
3) Regression testing (prima di ogni consegna al committente);
4) Interface testing;
5) Stress testing;
6) Testing con tool di supporto (tool da definire);
7) Inspection.

Per ogni tipo di test è stato nominato un responsabile che si incarica di effettuare il testing seguendo le regole specificate successivamente e documentando in modo opportuno i risultati ottenuti. I responsabili per i diversi tipi di test sono:

- Baradel Luca: black – box, white – box;
- Di Benedetto Gianluca: regression testing, interface testing;
- Pellizzari Luca: stress testing, testing con tool di supporto, inspection.

2.5.2. Passi da seguire durante il testing

Per quanto riguarda i test di tipo dinamico (dal numero 1 al numero 6 compreso della sezione precedente) i passi da seguire sono i seguenti:

1) Specificare qual è il tipo di test che sta per essere effettuato;
2) Specificare data, autore e numero progressivo del test;
3) Elencare quali sono le componenti coinvolte;
4) Specificare quali sono gli input;
5) Specificare quali sono gli output previsti;
6) Specificare se ci sono ulteriori risorse utilizzate;
7) Descrivere i risultati ottenuti, quindi il comportamento del sistema (errore oppure comportamento corretto);
8) Aggiungere eventuali note.

I risultati del testing vanno riportati in un documento che deve contenere tutti i dati citati negli otto passaggi sopra descritti.

Per quanto riguarda il testing di tipo statico (inspection) questo può essere eseguito singolarmente oppure in gruppo. Se viene eseguito da una sola persona questa deve:

1) Registrare data, autore e numero progressivo del test;
2) Descrivere quali sono le componenti che verranno analizzate;
3) Descrivere in modo preciso quali sono i problemi riscontrati e dove si trovano.

Nel caso in cui questo processo venga eseguito in gruppo (da due o più persone) è possibile assegnare dei ruoli ai membri del gruppo: ad esempio una persona può leggere il codice, una può prendere nota dei problemi riscontrati, una può fare da “moderatore” dell’incontro, eccetera.

L’inspection è un processo che può essere utilizzato non solo per rilevare eventuali errori nel codice ma anche per controllare la correttezza di documenti come ad esempio UML, documenti di progetto, analisi dei requisiti, eccetera. In questi casi i passi da seguire sono gli stessi rispetto a quelli descritti per l’inspection del codice: registrare data, autore e numero progressivo dell’inspection, identificare l’elemento che si vuole analizzare e registrare i problemi riscontrati.

2.6. Database

2.7. Gestore file

2.8. Software di supporto

2.8.1. GitHub

Uno dei tool di supporto utilizzati dal team GL^3 è GitHub, usato come repository per tutta la documentazione collegata al progetto. Il responsabile di questo item deve assicurarsi che la repository contenga tutti e soli i file indicati nel CM e che tutti questi file abbiano la struttura stabilita dal team e descritta nel CM.