LABORATORIO DI INGEGNERIA DEI SISTEMI SOFTWARE

Introduction

Our motto:
there is no code without a project, no project without problem analysis and no problem without requirements.

Requirements

Design and build a software system that makes a robot is able to walk along the boundary of a rectangular, empty room.

Requirement analysis

MAIN GOALS:

1. Project and deliver a software system that makes a robot something that moves in the four directions along the border of a rectangolar and epty room and with which the users can interface with HTTP/POST request.
2. A functional test of the product, in this case of the robot, can be carried out through the use of an algorithm that simulates the four movements (forward, backward, right, left).

Problem analysis

MAIN GOALS:

1. The main problems that can be encountered are that the system is a distributed application because use a HTTP protocol to send commands,the programmers must be aware of the API to interface with robots.
2. evaluate the **abstraction gap** and give a **quantitative measure of the effort/resources** necessary to build the system
3. The model and logical architecture is based on Java application using an API which sends commands to the robot by the http protocol.
4. refine the set of **functional TestPlan**
5. (with reference to **SCRUM**) define a (first) **product backlog** and a possibile set/sequence of **SPRINT**

WARNING: expressions like *'we have chosen to ...'*, *'I decided ...'*, etc. are **forbidden** here. Rather, this section should include sentences like ***'this (aspect of the) problem implies that ...'*** or ***'the usage of this (legacy) component requires that ...'***, etc.

Test plans

MAIN GOALS:

1. with reference to the **logical architecture** of the system, write a program (e.g. by using **JUnit**) that defines the set of **functional TestPlans** that the software must satisfy.

Project

MAIN GOALS:

1. by starting from the **logical architecture** of the system, define the **concrete/k> architecture of the system and the behavior of each component .**

Testing

MAIN GOALS:

1. complete the **functional TestPlans** according to the project code and execute them.

Deployment

MAIN GOALS:

1. prepare the software application to run and operate in a specific environment.

It involves installation, configuration, testing and making changes to optimize the performance of the software. It can either be carried out manually or (hopefully) through automated systems.

Maintenance

MAIN GOALS:

1. modify the software product after delivery to correct faults, to improve performance or other attributes.

A common perception of maintenance is that it merely involves fixing defects. However, one study indicated that over 80% of maintenance effort is used for non-corrective actions.

UN QUADRO DI INSIEME (da esperienza 2019/2020)

Il nostro obiettivo è costruire software impostando (in modo 'agile', ***non water-fall***, cioè con possibilità di modifiche dopo un ciclo di sviluppo):

1. l'analisi dei requisiti
2. l'analisi del problema
3. la progettazione e lo sviluppo della applicazione
4. il deployment (in docker)

L'obiettivo finale è partire fin dalla analisi dei requisiti con la definizione di MODELLI ESEGUIBILI e PIANI DI TESTING che si estendono/modificano nelle varie fasi.

Per questo fine può essere opportuno definire un 'progetto preliminare' da inserire in ***UN SOLO workspace*** (denominato ad esempio **xxxISS2021**, con **xxx**=nome del Team). In questo workspace andranno inseriti altri progetti, da quando inizia la fase 3.

L'analisi del problema dovrebbe produrre un modello che esprime la architettura logica del sistema e pone in luce l'***Abstraction Gap*** (se esiste) rispetto alle tecnologie di riferimento per la implementazione (per noi Java/Kotlin).

Per applicazioni relative a ***sistemi distribuiti eterogenei***, partiamo dalla constatazione che esiste 'di base' un Abstraction Gap rispetto a Java/Kotlin che ci porta a definire e realizzare un insieme di layer di supporto a livelli di astrazione crescente che culminerà della definizione del ***metamodello QAK*** e della relativa piattaforma di supporto (Plugin eclipse).

Va sottolineato che ogni specifico problema di norma può indurre un analista/progettista a individuare altri Abstraction Gap rispetto al metamodello QAK stesso, in accordo a un 'processo evolutivo' non predicibile e senza fine.

Lo sviluppo 'agile'

L'articolazione dello sviluppo in SPRINT (Scrum) inizia dopo l'analisi del problema, e si basa sulla opportunità di affrontare i requisiti in modo incrementale.

Di solito seguiamo un approccio ***'dal semplice al complesso'***, introducendo un opportuno insieme di ASSUNZIONI. Ma ovviamente questo non è l'unico criterio possibile nè quello sempre più opportuno.

Ad esempio il primo SPRINT della applicazione del TemaFinaleBo20.html può essere volto alla costruzione di un prototipo relativo al caso di un solo cliente che entra, ordina, consuma ed esce (senza incorrere nel problema 'maxstaytime').

Una volta arrivati al ***'done'*** (terminologia SCRUM) di uno SPRINT, per decidere come impostare il successivo SPRINT vanno inooltre sempre considerati anche aspetti di tipo

economico-organizzativo, in modo da assicurare che il processo di sviluppo si possa compiere nei tempi stabiliti e con costi contenuti.

Ogni SPRINT dovrebbe includere opportuni test (a partire dai *Piani di Testing* già delineati/definiti nella fase di analisi dei requisiti) da eseguire al termine dello SPRINT. Inoltre vanno considerate le fasi di ***SPRINT-review*** e ***SPRINT-retrospective*** che potrebbero essere condotte dopo un meeting con il committente (cioè con il docente).

Dunque il **numero e le finalità degli SPRINT sono definiti dal Team** di sviluppo, dopo le interazioni suddette con con il committente.

Come detto, ogni SPRINT andrebbe inserito in un apposito nuovo progetto nel workspace **xxxISS2020** e salvato in un apposito GIT repo, il cui indirizzo viene inviato al committente al momento della consegna.

History e non documentazione ex-post

E' importante che un progetto relativo ad uno SPRINT contenga un folder (**userDocs**) che include un documento organizzato secondo il template HTML proposto in questo documento.

Nel documento vanno riportati, per ogni sezione, i PUNTI-CHIAVE (come se fosse un presentazione da illustrare in un meeting), facendo riferimento, se si ritiene opportuno, ad altri documenti per descrizioni/discussioni più dettagliate.

Ovviamente lo SPRINT/N potrà fare riferimento con opportuni link - se e' il caso - a documenti dello SPRINT/N-1 o precedenti.

By Antonio Franzese email:

antonio.franzese4@studio.unibo.it