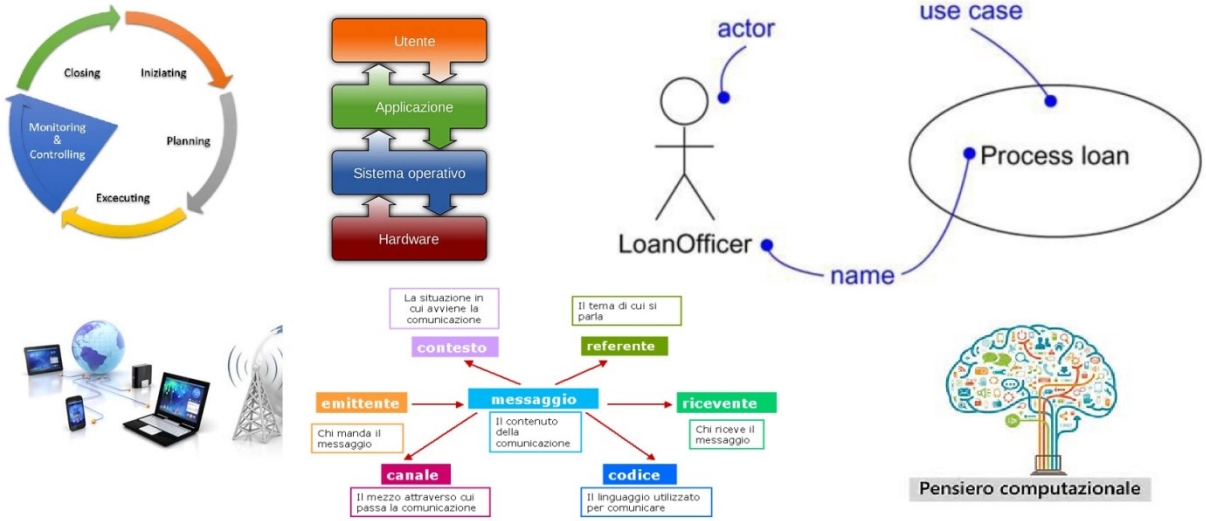
ITIS-LS “Francesco Giordani” Caserta

prof. Ennio Ranucci

a.s. 2020-2021

Astrazione - Comunicazione - Processi - Progetti - Documentazione - Pensiero Critico - Pensiero Computazionale



Astrazione

L'informatica è la scienza dell'astrazione.

Osserviamo l'evoluzione dell'informatica dagli anni 50 del secolo scorso ad oggi:

1. Il primo sistema operativo era costituito da alcune schede perforate in cui si indicava cosa fare delle schede successive (ad esempio eseguire le istruzioni contenute nelle prossime 10 schede). Successivamente il sistema operativo divenne software complesso che gestiva le risorse del computer e consentiva l'interazione con l'utente mediante comandi digitati da tastiera. Questa tipologia di interfaccia utente, detta a linea di comando, era caratterizzata da un'interazione di tipo testuale tra utente ed elaboratore: l'utente impartisce comandi testuali in input mediante tastiera alfanumerica e riceve risposte testuali in output dall'elaboratore mediante display o stampante. Oggi il sistema operativo adotta l'interfaccia grafica utente, nota anche come GUI (dall'inglese *Graphical User Interface*), un tipo di interfaccia che consente all'utente di interagire con la macchina controllando oggetti grafici convenzionali. La grafica è concepita tramite la metafora di un piano di lavoro rappresentato dallo schermo (detto *scrivania* o *desktop*), con le *icone* a rappresentare servizi vari disponibili e i file (di cui alcune a forma di cartellina per le directory) e le *finestre* a rappresentare le applicazioni. Il mouse rappresenta lo strumento principale per attivare i servizi indicati dalle icone. Nei tablet e negli smartphone la metafora della scrivania è stata sostituita dalla "schermata home" dove possono essere posizionate le icone delle applicazioni, che permettono appunto di avviare le differenti applicazioni, o i widget, cioè degli elementi che mostrano velocemente alcune informazioni, come le previsioni del tempo, l'ora, gli appuntamenti nel calendario, le ultime mail ricevute, etc. Si possono avere più schermate home, che si scorrono passandoci sopra con il dito, e servono per mettere a disposizione più spazio, dove mettere altre icone o widget. Questo tipo di sistema operativo è pensato per essere usato con le mani, tramite un touchscreen.
   Il filo conduttore nell'evoluzione dei sistemi operativi è l'eliminazione dei dettagli hardaware e software nella gestione delle risorse del computer e nella comunicazione con la macchina, sostituendoli, per astrazione, con concetti vicini al modo di pensare umano. La gestione dei componenti hardaware diventa trasparente (vedi RAM, ecc.) e le istruzioni per la macchina diventano icone da toccare o cliccare;
2. Per comunicare con la macchina si scrivevano le istruzioni in linguaggio macchina perforando una scheda. Nel linguaggio macchina le astrazioni dall'hardaware sono ridotte al minimo. Chi scrive in linguaggio macchina deve conoscere i codici operativi del processore che utilizza. Una maggiore astrazione si è ottenuta con il primo traduttore software che traformava il codice sorgente assembler in codice macchina. L'assembler sostituisce i codici operativi con codici mnemonici, con questa astrazione la parte ALU del processore diventa trasparente al programmatore che "vede" ancora registri e RAM. Con i linguaggi evoluti (Pascal, Cobol, Fortran, …) anche i registri e la Ram diventano trasparenti al programmatore sostituiti da identificatoridi costanti, variabili e altra strutture dati come record, array ecc. Con i linguaggi orientati agli oggetti anche i dati e i metodi incorporati possono diventare trasparenti al programmatore. Ad esempio in un ambiente di programmazione visuale la form è un oggetto che il programmatore utilizza anche senza conoscere i dettagli implementativi.

Quindi:

1. L'astrazione in informatica può essere applicata per sostituire la macchina reale che elabora 0 e 1 con macchine sempre più astratte (più vicine al modo di pensare dell'uomo) che eseguono i servizi indicati dall'utente (modelli teorici di hardware o software).
2. Nei linguaggi di programmazione l'astrazione consente di distaccarsi dalle istruzioni in linguaggio macchina e di scrivere istruzioni molto più vicine al linguaggio comprensibile all'uomo piuttosto che a quello comprensibile dalla macchina.
3. Il processo di astrazione consiste anche nell'isolare un particolare oggetto di indagine dall'ambiente, eliminando tutti quei dettagli, quei particolari e quelle variabili che non hanno alcun nesso sul fenomeno che si sta studiando.

L'evoluzione dell'informatica è un percorso di astrazione che sostituisce i modelli fisici delle informazioni con modelli concettuali sempre più vicini al pensiero umano.

Comunicazione

Il processo comunicativo, secondo il modello Shannon-Weaver, si basa su alcuni elementi fondamentali:

1. il sistema (animale, uomo, macchina) che trasmette (l'emittente);
2. un canale di comunicazione, necessario per trasferire l'informazione;
3. un contesto di riferimento in cui il processo si sviluppa;
4. il contenuto della comunicazione è contenuto nel messaggio;
5. il destinatario del messaggio comunicato (il ricevente);
6. l'informazione;
7. un codice formale mediante il quale viene data una forma linguistica all'informazione, cioè viene significata.

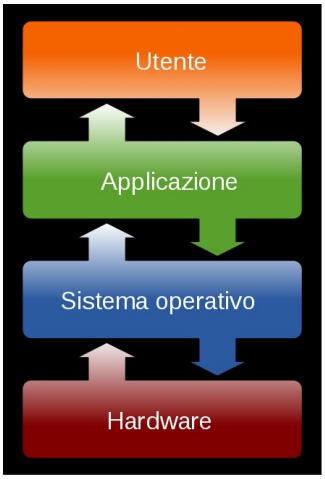
Un *protocollo di comunicazione*, in informatica, è un insieme di regole formalmente descritte che definiscono le modalità di comunicazione tra due o più entità. Queste regole a seconda delle entità interessate e del mezzo di comunicazione. Se le due entità sono remote, si parla di protocollo di rete.

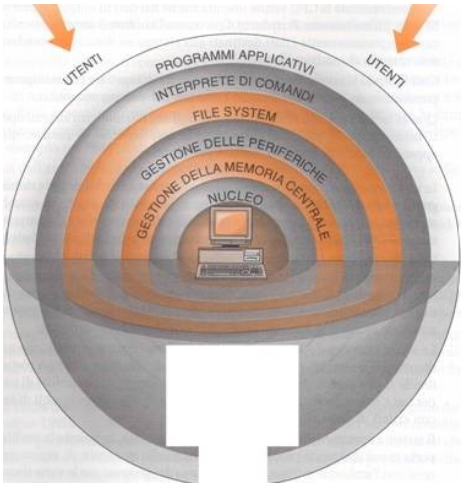
Comunicazione hardware

Lo scambio di dati tra la scheda madre o quelle inserite negli *slot* e il processore avviene tramite dei conduttori integrati nella scheda, il *bus* dei dati; il modo nel quale lo scambio avviene può seguire diversi standard:
ISA, *Industrial Standard Architecture*, nata con i computer IBM di tipo AT, microprocessore 80286;
MCA, *Micro Channel Architecture*, standard per i computer IBM della serie PS/2;
EISA, *Extended Industrial Standard Architecture*, adottata da molti computer DOS compatibili.
PCI, *Peripherical Component Interconnect*, progettato per velocizzare l'accesso alle unità periferiche.

Comunicazione software

Un sistema operativo è il software di base che gestisce le risorse e la comunicazione con l'utente, con le periferiche e la Rete. È costituito da più programmi software che comunicano tra loro.





In informatica l'espressione comunicazione tra processi (in inglese **inter-process communication o IPC)** si riferisce a tutte quelle tecnologie software il cui scopo è consentire a diversi processi di comunicare tra loro scambiandosi dati e informazioni. I processi possono risiedere sullo stesso computer o essere distribuiti su una rete.
I processi sono programmi software in esecuzione.

Comunicazione Macchina Macchina e Uomo Macchina

Nel giro di poco più di mezzo secolo abbiamo assistito ad una rivoluzione totale delle nostre società, lo dimostra il fatto che oggi abitiamo in società informatizzate in cui viviamo il rapporto con la tecnologia in maniera abbastanza naturale. Nell'arco di qualche decennio siamo passati dalla costruzione di infrastrutture per le reti di telecomunicazione alla guida di veicoli autonomi che attraverso la rete Internet comunicano tra loro, alla dematerializzazione dei documenti cartacei grazie alla nascita del cloud computing e delle blockchain. Siamo arrivati alla possibilità di poter comunicare con chiunque vogliamo, in qualsiasi parte del mondo si trovi, in qualsiasi momento noi desideriamo e alla possibilità di connettere alla rete non solo oggetti di uso industriale, ma anche oggetti della nostra quotidianità per ottimizzare l'efficienza di termostati, impianti elettrici, e elettrodomestici eccetera.Tutto è stato reso possibile grazie all'intensificarsi dell'interazione uomo-macchina. Nel giro di mezzo secolo l'informatica oltre ad elaborare informazioni è entrata a pieno titolo anche nella comunicazione delle informazioni.

Anche se al giorno d'oggi un sistema operativo è fortemente legato alla sua interfaccia grafica (GUI), le cose non sono andate sempre così. Nell'era dei pionieri dell'informatica, i sistemi operativi non esistevano, ogni utente era anche programmatore e doveva occuparsi di scrivere i programmi per qualsiasi cosa. Col tempo, si affermarono i sistemi operativi Unix e MS-DOS, che utilizzavano interfacce a caratteri in cui i comandi dovevano essere digitati utilizzando la tastier; per esempio doveva scrivere il comando "date" per modificare la data e l'ora attuale oppure "change directory nomeCartella" per cambiare cartella.

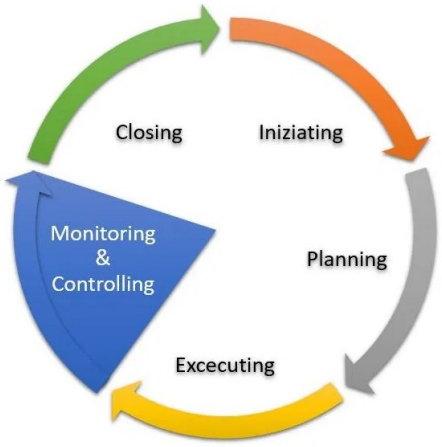
Processi e Progetti (fonte Lorenzi Colleoni "Gestione Progetto" Atlas)

Un processo è composto di attività svolte in modo continuativo per assolvere a specifici obiettivi e compiti.

Un progetto consiste, in senso generale, nell'organizzazione di azioni nel tempo per il perseguimento di uno scopo predefinito, attraverso le varie fasi di progettazione da parte di uno o più progettisti. Scopo finale è la realizzazione di un bene o servizio il cui ciclo di sviluppo è gestito tipicamente attraverso tecniche di project management.
Un progetto ha vincoli di tempo, costo, risorse.

Esempio: il processo logistico assolve al compito di ricevere i materiali in ingresso necessari per produrre e poi di gestire la consegna dei prodotti finiti.
Esempio: il progetto per la realizzazione di una centrale termica è un'iniziativa che ha un inizio e una fine precisi e richiede risorse per un arco di tempo limitato.

Gli **Stakeholder** sono individui o gruppi di individui coinvolti nel progetto in forme più o meno strette: ad esempio, capo progetto, soci di maggioranza e minoranza, lavoratori dipendenti, collaboratori autonomi, clienti, ...
Il progetto si sviluppa seguendo un percorso ciclico:



Ciclo di vita di un progetto : La fase di avvio
Durante questa prima fase di avvio, viene **identificato l'obiettivo**, o il "bisogno" del progetto; questo può essere, ad esempio, la risoluzione di un problema aziendale o l'analisi e creazione in concreto di un'opportunità.
Viene condotto quindi uno studio di fattibilità per verificare se ciascuna opzione è in linea all'obiettivo e viene determinata una soluzione finale.
Lo **studio di fattibilità** si pone domande sulla realizzabilità del progetto. Domande come "possiamo svolgere il progetto? Abbiamo le risorse per farlo?".
Inoltre, si effettua anche una fase di **studio di giustificazione del progetto**, rispondendo per esempio alla domanda "è necessario il progetto per questo obiettivo?".
Una volta svolte queste analisi e considerato il progetto fattibile e necessario, questo viene ufficialmente fatto partire e, nel caso in cui non sia stato giá individuato, viene nominato un

project manager.Successivamente viene identificato e coinvolto il **team di progetto** che inizia così a prendere forma.
Il documento redatto alla fine di questa fase è **business plan**

Per una efficace comunicazione è utile utilizzare nell'ambito del business plan utilizzare le metodologie SMART e 5W 1H

- **S** = Specific (Specifico)
- **M** = Measurable (Misurabile)
- **A** = Achievable (Raggiungibile)
- **R** = Realistic (Realistico)
- **T** = Time-Based (Temporizzabile)

***W**hat? **W**hy? **W**ho? **W**here? **W**hen? and **H**ow?*

Un business plan dovrebbe avere i seguenti contenuti:

- la descrizione del progetto;
- la descrizione del prodotto/servizio;
- la storia dell'azienda e della sua struttura organizzativa;
- presentazione degli stakeholder;
- il prospetto degli aspetti amministrativi e degli investimenti;
- l'analisi del know how e prospetto delle risorse umane necessarie (team);
- la fattibilità tecnica e legale;
- indicazione dei rischi;
- piano di progetto comprendente lo sviluppo temporale e le tappe principali di verifica.

IL PIANO DI PROGETTO

È un modello previsionale del progetto dal punto di vista dei risultati.
Il software per rappresentare ed aggiornare il piano di progetto viene denominato:
Project management
Per rappresentare un piano di progetto si usa: la **WBS (work break down structure)** un modello gerarchico degli obiettivi.

Sw: Project di Microsoft, ProjectLibre open source

Modello Business Plan Startup (http://www.business-plan.it/business-plan-startup.htm)

Business Plan guida al piano industriale

Cominciamo dalla parte descrittiva , spesso trascurata perchè è diffusa l'opinione che ritiene il Business Plan composto solo dai 'numeri', cioè dalle previsioni sulle vendite, sugli investimenti, sui costi, ecc.

Questa sezione, invece, è fondamentale se vogliamo chiedere l'intervento di un Investitore nel capitale di rischio.

1) SINTESI DEL PROGETTO IMPRENDITORIALE

a) l'impresa e i suoi fondatori: quando é sorta, cosa fa o si appresta a fare. Descrizione accurata dell'imprenditore e dei suoi collaboratori chiave. Obiettivi economici e sociali

b) opportunità offerte dal mercato: descrizione della concorrenza, quote e tasso di crescita del mercato

c) prodotto e tecnologia: in cosa si distingue con riguardo al prodotto, alla tecnologia disponibile, al patrimonio intangibile dell'impresa

d) proiezioni finanziarie: relative ai 2-3 anni successivi all'ipotesi di finanziamento/equity

e) proposta di finanziamento: entità del finanziamento/equity, modalità di utilizzo, vantaggi per il venture capitalist (ipotesi di remunerazione del capitale, altri vantaggi a seconda che la matrice sia o meno industriale).

2) L'IMPRESA

a) storia

b) forma e composizione societaria

c) presenza di legami con altre imprese

3) L'IMPRENDITORE O IL NUCLEO IMPRENDITORIALE

a) caratteristiche generali

b) esperienze passate

b1) affini al business

b2) non affini al business

c) motivazioni

d) ruolo svolto all'interno della iniziativa

4) IL TEAM IMPRENDITORIALE

a) i componenti del gruppo

b) esperienze dei componenti del gruppo

c) ruoli chiave nel progetto/impresa

c1) soggetti preposti per i ruoli chiave

c2) prospettive future di assetto del team

5) IL MERCATO DI SBOCCO

a) descrizione del mercato e dei suoi segmenti

b) dimensioni e prospettive di sviluppo della domanda

c) risultati delle ricerche di mercato, se si sono svolte

d) potere contrattuale dei clienti

e) dimensioni e prospettive di sviluppo del/i segmento/i di mercato in cui si opera (tasso di crescita, determinanti della crescita, stagionalità / ciclicità)

6) LA CONCORRENZA

a) descrizione della struttura dell'offerta

b) situazione e grado di turbolenza tecnologica del settore

c) profilo dei principali concorrenti

d) grado di competitività del settore

e) perché i prodotti della concorrenza non soddisfano pienamente le esigenze del mercato

f) come si pensa di superare le barriere all'entrata del settore

g) definizione delle barriere all'uscita

h) barriere all'entrata nei confronti dei concorrenti potenziali

i) identificazione dei prodotti/servizi sostitutivi

7) I MERCATI DI APPROVVIGIONAMENTO

a) identificazione delle principali fonti di approvvigionamento
b) descrizione delle principali caratteristiche delle fonti di approvvigionamento (costanza dell'offerta, affidabilità dei fornitori, ecc.)
c) fonti chiave di approvvigionamento
d) potere contrattuale dei fornitori

8) IL PRODOTTO/SERVIZIO

a) descrizione del bisogno che si intende soddisfare
b) descrizione del prodotto/servizio
c) presenza di eventuali brevetti o licenze
d) modalità di utilizzo ed elementi di interesse
e) fase dello sviluppo in cui si trova (crescita, maturità, declino)
f) tempi, modalità e costi per la messa a punto del prodotto/servizio nuovo (nel caso di avvio o di programmi di sviluppo)

9) LA COMMERCIALIZZAZIONE

a) la filosofia di marketing adottata
b) le scelte di prezzo
c) il piano di comunicazione
d) i canali distributivi prescelti
e) la rete di vendita
f) eventuali accordi di commercializzazione
g) budget delle vendite
h) i costi di commercializzazione

10) IL PATRIMONIO TECNICO-INDUSTRIALE

a) politiche di acquisizione dei brevetti, know-how
b) accordi a livello produttivo
c) le scelte di produzione interna o di acquisizione presso terzi
d) le modalità di approvvigionamento
e) la struttura produttiva
f) i tempi, i modi e i costi per la predisposizione o per l'adeguamento della struttura produttiva
g) composizione e natura dei costi di produzione
h) la struttura di ricerca e sviluppo
i) accordi di ricerca e sviluppo
l) i tempi, i modi e i costi per la predisposizione o per l'adeguamento della struttura di ricerca e sviluppo
m) il controllo della qualità

11) NETWORK

a) sintesi delle alleanze e degli accordi già raggiunti con altre aziende
b) identificazione di possibili alleanze future
c) le relazioni industriali poste in essere

12) LE PROIEZIONI ECONOMICO FINANZIARIE

a) conti economici previsionali
b) stati patrimoniali previsionali
c) flussi finanziari previsionali
d) indici di sviluppo, di redditività, liquidità e solidità
e) analisi del punto di pareggio finanziario (Ricavi totali=Costi totali)
f) i rischi finanziari connessi

13) RAPPORTI CON L'INVESTITORE

a) cosa si offre
b) cosa si chiede in cambio

14) ALLEGATI

a) curricula del team imprenditoriale e del nucleo
b) descrizioni dettagliate/schede tecniche del prodotto o del processo produttivo
c) risultati dettagliati delle indagini di mercato poste in essere
d) tutto il supporto informativo contabile.

Utilizziamo gli strimenti sopra descritti per documentare gli esercizi e le esercitazioni di laboratorio.

Business Plan Semplificato

esercizio “Calcolatrice html javascript”

Storia dell’azienda e della sua struttura organizzativa

Istituto Tecnico Industriale e Liceo Scientifico “Francesco Giordani” Caserta
L'Istituto iniziò la sua attività il 1° ottobre 1961 utilizzando i locali di palazzo Catemario, nei pressi di piazza Marconi e fu intitolato a Francesco Giordani, già titolare della cattedra di Chimica all'Università di Napoli.
L'anno seguente furono utilizzati i locali di Villa Rosa, sull'Appia, ove fu istituito il primo laboratorio tecnologico, con attrezzature all'avanguardia, tant'è che i professori dell'Università di Napoli, come il prof. Cittadini, se ne servivano per eseguire prove sui materiali.
L'Istituto intanto cresceva e furono utilizzati alcuni locali del Liceo Scientifico "A. Diaz" per collocare i reparti di Macchine utensili, Saldature ed Aggiustaggio. Inoltre veniva aperta una sede staccata a Carinola.
L'Istituto industriale, sorto per formare Periti Meccanici, nel 1966 ampliò l'offerta formativa inaugurando la specializzazione per Elettrotecnici. A Villa Rosa fu allestita una Sala Macchine Elettriche e un laboratorio di Misure Elettriche. Furono utilizzati il Palazzo Vescovile, sul Corso Trieste, e il Palazzo Landolfi in Corso Giannone, per installare i Laboratori di Macchine Idrauliche e quelli di Aggiustaggio. Nel contempo si lasciava Villa Rosa, ormai fatiscente, e nuovi locali furono presi in locazione in via Ricciardelli e in via Tevere. Poi gli edifici dell'ex ACI divennero la sede principale dell'Istituto. In questi anni sedi staccate erano: Carinola, Marcianise, Capriati al Volturno, Alife, Piedimonte e Capua e fu istituito anche un corso serale.
Dal 1973 al 1985, l'Istituto aprì la specializzazione di Elettronica. Si chiuse l'esperienza del corso serale e, finalmente, venne assegnata la nuova sede di via Laviano, progettata nel 1962. A questo punto l'Istituto si aprì all'interesse delle nuove tecnologie con l'Istituzione della specializzazione di **Informatica**.
Negli anni dal 1985 al 1996 fu istituita la specializzazione di Chimica Industriale e fu portata a termine un'altra ala dell'Istituto di via Laviano.

Presentazione degli stakeholder

prof Ennio Ranucci e prof Margherita Puca, docente di Tecnologie di Progettazione di Sistemi Informatici, compagni di classe.

Il prospetto degli aspetti amministrativi e degli investimenti

Gli esercizi e le esercitazioni sono valutate anche dal punto di vista del costo prevedendo un costo orario di 23€ per il capo progetto, 16€ per l’analista e 10€ per il programmatore.

La descrizione del progetto e La descrizione del prodotto/servizio

ITIS-LS F.Giordani Caserta
Anno scolastico 2020/2021
Classe 3^ sez.B spec. Informatica
Data: 17/09/2020

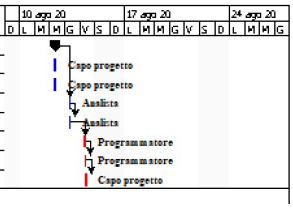
Numero es: 1
Versione:1.0
Programmatore/i: Lezione collettiva
Sistema Operativo:Windows 10
Compilatore/Interprete: Code::Blocks Release 17.12 rev 11256
Obiettivo didattico: L'alunno e' in grado di caricare e leggere un file di records ad accesso diretto;
Obiettivo del programma: Caricare e leggere un file di records con ricerca ad accesso diretto;

L'analisi del know how e prospetto delle risorse umane necessarie (team)

HTML: i principali tag e CSS
Javascript: strutture di controllo del flusso delle istruzioni e sottoprogrammi

piano di progetto comprendente lo sviluppo temporale e le tappe principali di verifica

	Nome	Durata	Avvio	Termine	Predecessori	Nome risorsa
1	**definizione e pianificazione**	**0,062 giorni**	**12/08/20 8.00**	**12/08/20 8.30**		**Capo progetto**
2	documentazione	0,031 giorni	12/08/20 8.00	12/08/20 8.15		Capo progetto
3	project management	0,031 giorni	12/08/20 8.15	12/08/20 8.30		Capo progetto
4	progettazione design	0,031 giorni	13/08/20 8.00	13/08/20 8.15	1	Analista
5	progettazione soluzione	0,031 giorni	13/08/20 8.15	13/08/20 8.30	4	Analista
6	codifica html	0,062 giorni	14/08/20 8.00	14/08/20 8.30	5	Programmatore
7	codifica javascript	0,062 giorni	14/08/20 8.30	14/08/20 9.00	6	Programmatore
8	monitoraggio e controllo del prodotto	0,021 giorni	14/08/20 9.00	14/08/20 9.10	7	Capo progetto



Documentazione

- Documentazione di processo
- Documentazione tecnica di progetto IEEE 830 e IEEE 829
- Documentazione del codice sorgente
    - Documentazione delle scelte implementative
    - Documentazione dell'interfaccia di programmazione
    - Test unitari
- Documentazione per l'utente
    - Installazione
    - Configurazione
    - Utilizzo
    - Risoluzione di problemi
    - Diritti d'autore
    - Contatti

Software per la documentazione tecnica

Subversion: strumento client/server per la gestione del codice sorgente
TortoiseSVN: client SVN per comunicare con il server SVN

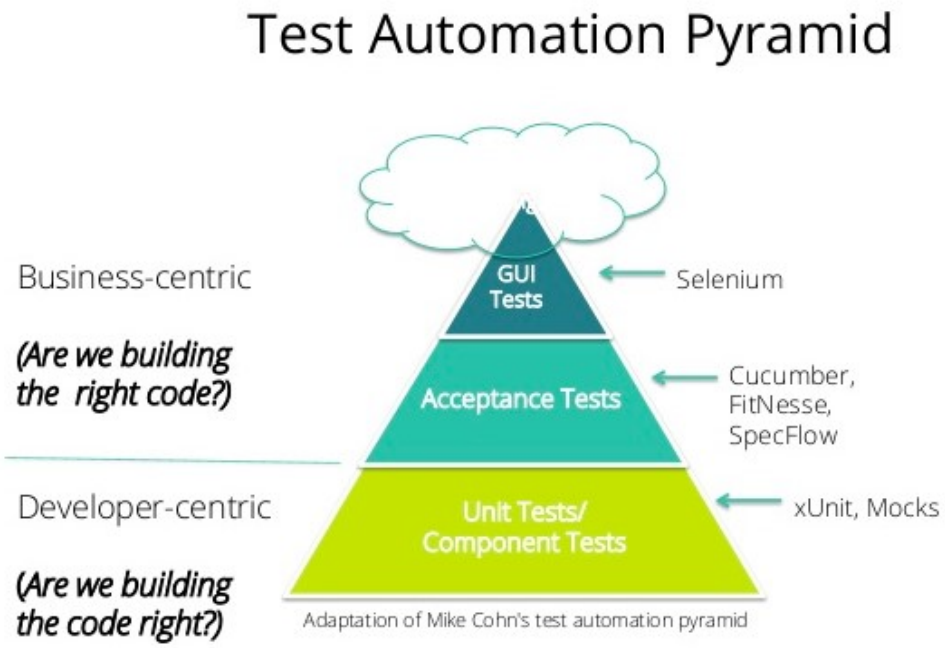
Wlink: strumento per la registrazione di una sessione utente che descrive la modalità di utilizzo, utile per la creazione di video tutorial

SourceMonitor: strumento per la verifica della qualità del codice sorgente

Il **Testing** è un modo per assicurarsi che un software non presenti anomalie e che le funzionalità che ci interessano performino come previsto. L'obiettivo principale è quello di **intercettare bug** (magari introdotti accidentalmente in una nuova versione) e di assicurarsi che bug già trovati e risolti rimangano tali, e inoltre che non vengano a crearsi delle regressioni (i componenti che in precedenza risultavano funzionanti devono continuare a farlo, possibilmente nello stesso modo con le stesse performance, se non addirittura migliori).

Teorema di Dijkstra (1969):
"il test di un programma può rilevare la presenza di malfunzionamenti, ma mai dimostrarne l'assenza".

Mike Cohn- Piramide della test automation:



Lo scopo dello unit testing è quello di verificare il corretto funzionamento di parti di programma permettendo così una precoce individuazione dei bug. Uno "unit testing" accurato può dare una prova abbastanza certa che un pezzo di codice funziona correttamente, con importanti vantaggi:

1. Lo unit testing facilita la modifica del codice del modulo in momenti successivi (refactoring) con la sicurezza che il modulo continuerà a funzionare correttamente;
2. Lo unit testing semplifica l'integrazione di moduli diversi perché limita i malfunzionamenti a problemi di interazione tra i moduli e non nei moduli stessi;
3. Lo unit testing fornisce una documentazione "viva" del codice, perché è intrinsecamente un esempio di utilizzo dell'API del modulo.

Limiti

In generale il testing non riesce ad identificare tutti gli errori in un programma e lo stesso vale per lo Unit Testing che, analizzando per definizione le singole unità, non può identificare gli errori di integrazione, problemi legati alla performance e altri problemi legati al sistema in generale. Lo unit testing è più efficace se utilizzato in congiunzione con altre tecniche di testing del software.
Come ogni forma di testing, anche lo Unit Testing non può certificare l'assenza di errori, ma può solo evidenziarne la presenza.

Il modulo unittest fornisce un ricco insieme di strumenti per costruire ed eseguire dei test. Questa sezione mostra come un piccolo sotto insieme di tali strumenti sia sufficiente a venire incontro alle necessità della maggior parte degli utenti.
Ecco un breve script per testare tre funzioni del modulo random:

```
import random
import unittest

class TestSequenceFunctions(unittest.TestCase):

    def setUp(self):
        self.seq = range(10)

    def testshuffle(self):
        # assicura che la sequenza rimescolata non perda degli elementi
        random.shuffle(self.seq)
        self.seq.sort()
        self.assertEqual(self.seq, range(10))

    def testchoice(self):
        element = random.choice(self.seq)
        self.assert_(element in self.seq)

    def testsample(self):
```

```
        self.assertRaises(ValueError, random.sample, self.seq, 20)
        for element in random.sample(self.seq, 5):
            self.assert_(element in self.seq)

if __name__ == '__main__':
    unittest.main()
```

Una testcase viene creata derivandola da una classe unittest.TestCase. I tre test singoli sono definiti tramite dei metodi il cui nome inizia con le lettere "test". Questa convenzione sul nome informa il test runner di quali sono i metodi che rappresentano dei test.

Il punto cruciale di ogni test è una chiamata ad assertEqual() per controllare l'esattezza del risultato; al metodo assert_() per verificare una condizione, oppure ad assertRaises() per verificare che un'eccezione venga sollevata come ci si aspetta. Questi metodi sono usati in luogo di un'istruzione assert così che il test runner possa accumulare tutti i risultati dei vari metodi e produrre un resoconto finale.

Quando viene definito un metodo setUp(), il test runner avvierà questo metodo prima di ogni test. Allo stesso modo, se viene definito un metodo tearDown(), il test runner invocherà tale metodo dopo l'esecuzione di ogni test. Nell'esempio precedente, setUp() veniva usato per creare, per ogni test, una nuova sequenza.

Il blocco finale mostra un modo semplice per eseguire i test. unittest.main() mette a disposizione un'interfaccia a riga di comando per testare lo script. Quando viene eseguito da riga di comando, lo script precedente produce un output di questo tipo:

```
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```

Invece di unittest.main(), esistono altri modi per eseguire i test con un livello di controllo più raffinato, un output meno conciso e senza la necessità di eseguirli da riga di comando. Per esempio, le ultime due righe possono essere sostituite con:

```
suite = unittest.makeSuite(TestSequenceFunctions)
unittest.TextTestRunner(verbosity=2).run(suite)
```

Eseguendo il nuovo script dall'interprete o da un altro script, si ottiene il seguente output:

```
testchoice (__main__.TestSequenceFunctions) ... ok
testsample (__main__.TestSequenceFunctions) ... ok
testshuffle (__main__.TestSequenceFunctions) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.110s

OK
```

```
import unittest
def quoziente_resto(dividendoPar, divisorePar):
  quoziente = 0
  resto = 0
  dividendo = dividendoPar
  divisore = divisorePar
  if (divisore == 0):
    quoziente = "indefinito"
    resto = "indefinito"
  elif(divisore > dividendo):
    quoziente= 0
    resto = divisore
  else:
    while (divisore <= dividendo):
      dividendo = dividendo - divisore
      quoziente = quoziente + 1
      resto = dividendo
```

```
  print("    Il quoziente e il resto di", dividendoPar, "/", divisore, "è", quoziente, "e", resto,
end=".\n")

  return {"quoziente":quoziente, "resto":resto}


class TestQuozienteResto(unittest.TestCase):


    def test_zeroOverOne(self):

        qr = quoziente_resto(0, 1)

        self.assertEqual(0, qr["quoziente"])

        self.assertEqual(1, qr["resto"])


    def test_divideByZero(self):

        qr = quoziente_resto(1, 0)

        self.assertEqual("indefinito", qr["quoziente"])

        self.assertEqual("indefinito", qr["resto"])


    def test_dividesEvenly(self):

        qr = quoziente_resto(10, 5)

        self.assertEqual(2, qr["quoziente"])

        self.assertEqual(0, qr["resto"])


    def test_dividesWithRemainder(self):

        qr = quoziente_resto(10, 6)

        self.assertEqual(1, qr["quoziente"])

        self.assertEqual(4, qr["resto"])
```

```
    def test_secondLarger(self):
        qr = quoziente_resto(6, 10)
        self.assertEqual(0, qr["quoziente"])
        self.assertEqual(10, qr["resto"])
if __name__ == '__main__':
    unittest.main()
```

PYTEST

main

```
def square(x):
  return x ** 2
```

.replit

```
run="pytest testrunner.py "
```

testrunner.py

```
import pytest
from main import square
def test_square_4():
  assert square(2) is 4
test_square_4()
```

Selenium