NUMERO: 2198A ANNO: 2017

A P P U N T I

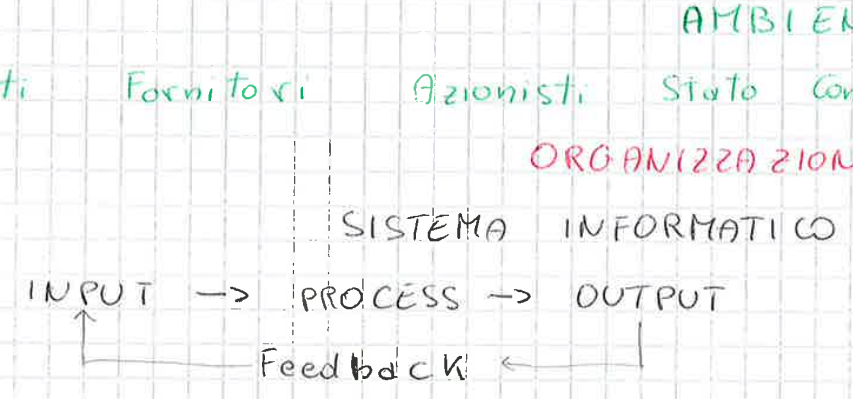
STUDENTE: Bellone Marco

MATERIA: Sistemi informativi industriali - Teoria + esercizi - Prof. Corno

DEFINIZIONI

29/09/15

Il Sistema Informativo è un sistema che serve a immagazzinare e processare informazioni usate dall'azienda. In senso lato è l'insieme delle persone e degli strumenti che lo utilizzano e trattano le informazioni.
Da questo insieme ricaviamo un sottoinsieme del sistema informativo che è il sistema informatico che comprende solo la parte computerizzata. È un sistema che memorizza ed elabora le informazioni solo dal punto di vista informatico.
Solitamente si cerca di specificare il sistema informatico compreso nel sistema informativo
La definizione di sistema informativo del libro (LAUDON) afferma si tratti di un insieme di componenti correlati che lavorano assieme per raccogliere, elaborare, conservare e distribuire informazioni a supporto delle operazioni decisionali di coordinazione, di controllo, di analisi e di visualizzazione all'interno di una organizzazione.
L'organizzazione è una struttura che gestisce processi.



per l'azienda e può essere usata all'infinito. Ha un costo marginale di produzione nullo. In un'organizzazione coesistono sia scarsezza che eccesso di informazioni e i "problemi" riguardano il ciclo di vita (acquisizione, immagazzinamento, ...).
le organizzazioni stanno diventando sempre più dipendenti dalle informazioni.

1/10/2015

PUNTI DI VISTA SUL SISTEMA INFORMATIVO

Posso dire l'evoluzione di un sistema informativo che cambia per adeguarsi ai cambiamenti della società e dell'informatica. Bisognerà sempre gestire più "viste" sapendo che definiscono la stessa cosa.
Classifichiamo i modelli secondo 3 punti di vista, a seconda del punto di vista ci sarà una descrizione diversa del modello del sistema informativo:
- i modelli tecnologici descrivono COME è fatto in termini di, hardware, software, server, ...
- i modelli funzionali che sono un insieme di modelli che ci aiutano a descrivere CHE COSA FA il modello informativo
- i modelli organizzativi rivolti a CHI li usa o tiene questi sistemi informativi.

Per meglio descrivere il modello tecnologico, possiamo distinguere la parte più applicativa da quella tecnologica: la seconda è la parte

L'UML usa una semantica particolare che assegna significati precisi a quello che si scrive. CLASS DIAGRAM
Dobbiamo cercare di scrivere i concetti che il sistema informativo dovrà gestire e, quindi, fargli capire cosa sa del mondo che deve elaborare. Questi modelli devono essere comprensibili a persone diverse, perciò deve utilizzare uno stesso dominio applicativo.
Il "problema" principale è descrivere schematicamente un dominio di conoscenza.
La prima fase della raccolta dei requisiti è catturare ed identificare i concetti principali, le caratteristiche e i dati associati ai concetti, le relazioni tra i vari concetti.
Definiamo i livelli di astrazione e concretezza

ASTRATTO	CONCRETO
Concetto	Istanza
Entità	Oggetto
Classe	Item
Categoria	Esempio
Tipo	Occurrence

Il sistema informativo lavora sempre su dati concreti. La difficoltà è scrivere in senso astratto qualcosa che opera a livello concreto.
Una classe è un'insieme di entità che condividono le stesse caratteristiche.
Devo trovare oggetti da mettere nel modello che sono caratterizzati da identità, attributi, operazioni che può fare e messaggi che può ricevere.
La classe si identifica col bordo spesso mentre l'oggetto con quello fine. [Object : class]

Prova scritta del 2/09/2015

Tempo a disposizione: 2 ore. Non è permesso consultare testi o appunti.

Parte 1

Si consideri il seguente scenario:

In un sistema di trasporti urbani vengono utilizzati dei biglietti Chip-on-Paper che utilizzano la tecnologia RFID per effettuare la convalida. In tali biglietti, un piccolo chip è integrato nel biglietto cartaceo, e può essere letto appoggiandolo agli opportuni lettori installati a bordo dei mezzi.

I biglietti contengono un numero predefinito di corse (5, 10 o 15) e possono essere convalidati a bordo degli autobus. Ad ogni convalida il numero di viaggi residui del biglietto viene decrementato ed il sistema conosce in ogni istante lo stato dei biglietti.

Ad ogni convalida di un biglietto, il sistema informativo registra anche le informazione relative: al dispositivo di validazione, all'autobus su cui è installato, all'ora di partenza, alla fermata di partenza, ed alla linea percorsa dall'autobus.

Il ciclo di vita dei biglietti prevede che essi vengano emessi dall'azienda di trasporti, successivamente i rivenditori (edicole, tabaccai, etc.) li attivano prima di venderli al pubblico caricando il numero di corse previsto. Gli utenti, in possesso di un biglietto attivato, possono convalidarlo un numero di volte pari alle corse concesse.

A fine mese, ogni venditore conferma l'elenco dei biglietti venduti (ovvero attivati) e lo invia all'azienda di trasporti urbana che emette, dopo aver fatto una verifica interna ed eventualmente richiesto una rettifica, una richiesta di pagamento per il venditore (il quale dovrà versare l'importo dei biglietti venduti meno una provvigione fissa). La procedura si conclude quando il sistema riceve dalla banca l'informazione che il pagamento richiesto è stato effettuato.

Nel contesto dello scenario delineato sopra, si definisca:

1. Il modello informativo concettuale (diagramma delle classi UML).
2. Il modello del processo (diagramma delle attività UML), comprendente sia il ciclo di vita dei biglietti che la procedura di emissione e pagamento.
3. I tre KPI ritenuti più rilevanti per la valutazione del processo, in particolare facendo riferimento alla *convenienza dell'uso dei biglietti per l'azienda di trasporti urbana*.
4. Il modello dei casi d'uso (diagramma dei casi d'uso UML) a livello *user goal*.

NB: è necessario modellare esclusivamente gli aspetti direttamente rilevanti per il sistema informativo.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 9 di 190

devo imporre l'aggiornamento annuale, altrimenti è meglio usare la data di nascita e derivare l'età.

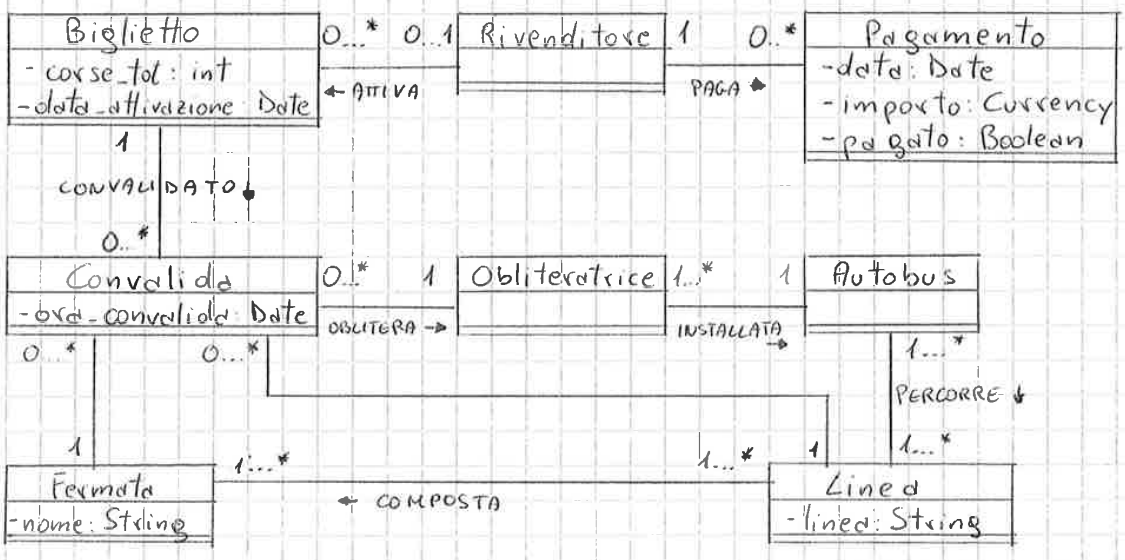
TEMA ESAME 2/9/2015

In primis bisogna trovare i concetti principali, le relazioni, la numerosità e gli attributi.

"Biglietto" viene attivato da "Rivenditore". Dall'informazione di uno dei due posso sapere qualcosa sull'altro. Con "Convalidato" sappiamo quale biglietto è stato usato su quale obliteratrice.
Un Rivenditore può attivare 0...* Biglietti e un Biglietto può essere attivato da 0...1 Rivenditore.
L'Obliteratrice può convalidare 0...* Biglietti e un Biglietto può essere convalidato da 0...* Obliteratrice (perché un Biglietto può avere più viaggi).
Un'Obliteratrice può essere installata su 1 Autobus e un Autobus può installare 1...* Obliteratrici.
Una linea è percorsa da 1...* Autobus e un Autobus percorre 1...* Linee.

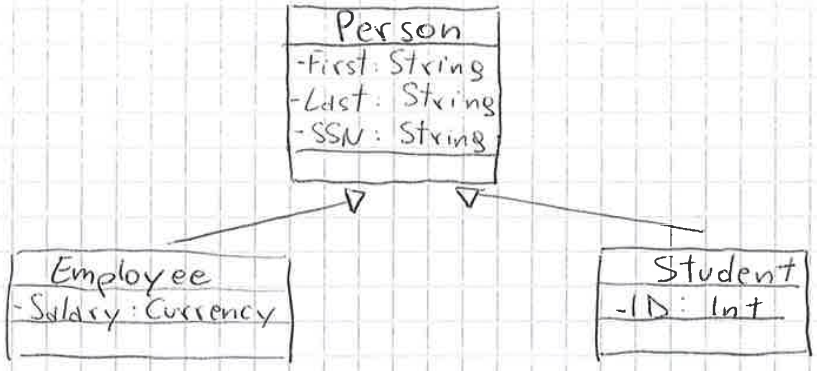
Un attributo che devo dare a Biglietto è contare/memorizzare le corse massime che può fare (corse_tot : int). Non metto le corse rimanenti perché sarebbe ridondante e perché dovrebbe aggiornarsi continuamente.
Un Rivenditore deve sapere quanti biglietti ogni mese ha attivato. Perciò in Biglietto metto un attributo data_attivazione: Date. Nel momento in cui si

linea e che ha un nome (nome: String) e la convalida avviene in una Fermata.
La Convalida può avvenire in una sola Fermata e in una Fermata posso avere 0...* Convalide.
In una Fermata posso avere 1...* Linee e ogni Linea è composta da 1...* Fermate.
Per il pagamento, è forse una cosa semplice, posso creare la classe Pagamento in cui salvo la data, l'importo e se è stato effettuato o meno (data: Date, importo: Currency, pagato: boolean) che lego a Rivenditore.
Il Rivenditore può emettere 0...* richieste di Pagamento e un Pagamento è relativo ad 1 solo Rivenditore.



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 13 di 190

avere inoltre classi più specializzate.
Se B specializza A vuol dire che gli oggetti descritti in B hanno le stesse proprietà di quelli descritti in A. Gli oggetti descritti da A possono avere ulteriori proprietà. Questo si indica



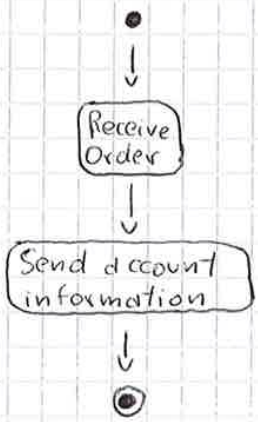
la freccia punta dalla classe speciale verso la classe generale. È una relazione "IS A".
Qualsiasi cosa venga attribuito a Person viene ribaltata anche su Employee e Student. Il contrario non è vero.
Ci sono modi diversi di chiamare uno stesso caso. Una relazione si chiama di specializzazione (generalizzazione il suo inverso) ma si può anche usare il termine ereditarietà.
Le sottoclassi ereditano tutte le relazioni della classe principale. Maggior specificità vuol dire maggiori vincoli, maggiori informazioni che vuol dire minore libertà. Un'attributo ~~sse~~ o una relazione scritte su una classe base vanno "replicati" sulle sottoclassi. Un'attributo ~~de~~ o relazione di una sottoclasse saranno specifici solo di quella.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 15 di 190

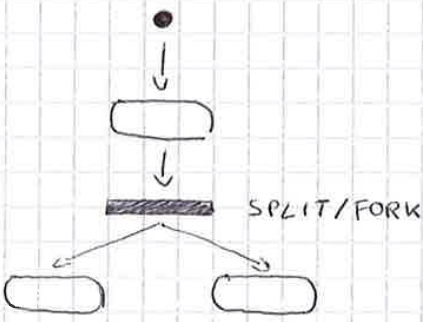
Descrivere un processo vuol dire elencare le regole di viaggio del gettone. Quando il gettone avanza, i suoi "dati" cambiano.

ACTIVITY DIAGRAM

Quali sono le regole di viaggio di questo "token"?
La più semplice è la **sequenza** ovvero una azione non può essere abilitata prima che un'altra non sia completata.

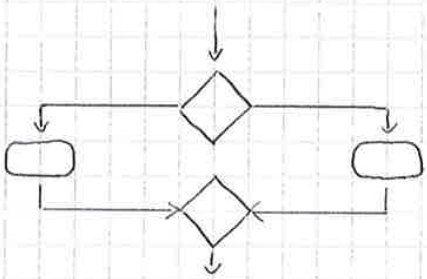


Abbiamo delle alternative, ovvero fare più cose in ~~parallelo~~ **parallelo** in cui posso avere più blocchi attivi che dovranno fare un lavoro. Questo lo si indica con uno split (divisione)

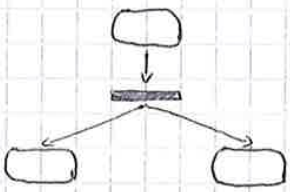


La fork duplica il token tante volte quante sono le frecce uscenti.
L'opposto dello split è la **sincronizzazione**: va bene fare le cose in parallelo, ma quando tutti terminano la bisogno che questi flussi si ricongiungono in uno solo, unire tutti token che è un "riassunto" di tutti.

informazioni sufficienti e fargli prendere la strada giusta. Nel nodo di scelta il token non si ferma. Una volta che ho separato i due flussi, potrei volere che si ricongiungano perciò mi serve una figura analoga al join. Si usa sempre un rombo detto di "convergenza" in cui abbiamo più frecce entranti ed una sola uscente.



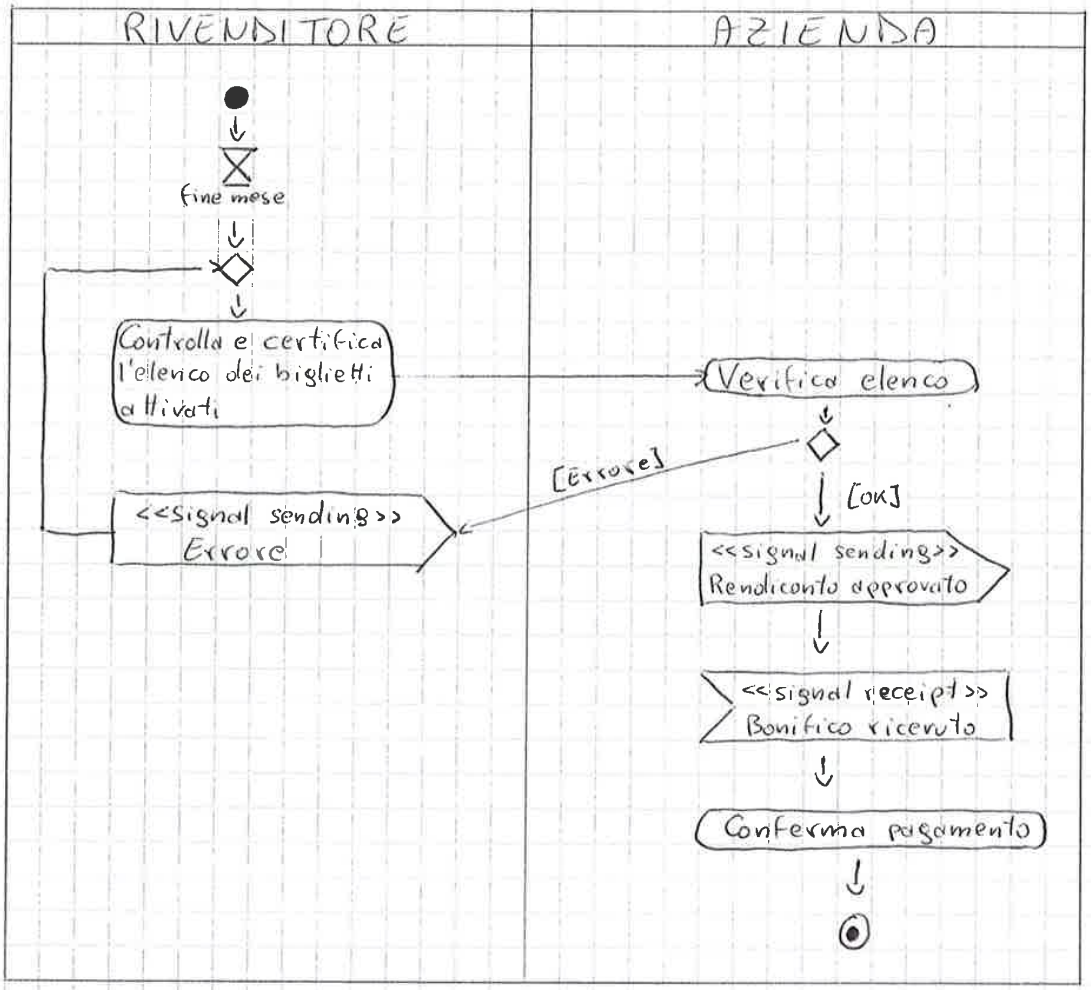
Esistono dei casi in cui l'utente può decidere di fare cose diverse. Si può modellare questa situazione chiedendo all'utente cosa vuole fare e dopo presentargli l'opzione di fare solo ciò che ha scelto; in alternativo posso far vedere all'utente le alternative e far scegliere a lui. Il sistema informativo si deve predisporre affinché ciascuna di queste opzioni possa essere fatta.



Proviamo ad usare questi costrutti per fare un esercizio (tratto dal tema di esame precedente).

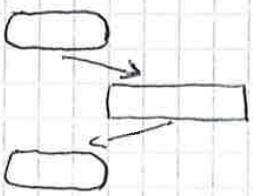
PROCESSO STRUTTURATO è quello per cui ci sia sempre un punto di ingresso e uno di uscita per ciascuna azione o gruppo di azioni, per ogni fork ci sia un join e per ogni scelta ci sia una convergenza.

segnali o scambio messaggi che serve a scambiare l'informazione in due così: tra il sistema informativo e il mondo esterno oppure lo useremo per far parlare tra loro vari processi.
Un'altra condizione che può intervenire nel processo è il tempo, col esempio quando il token deve aspettare un momento esatto per muoversi. Questo si indica con ⧖ in cui viene specificato il tempo di quanto deve stare fermo. (relativo o assoluto). Se devo aspettare l'inizio o la fine di un'azione (aspettare che l'acqua bollisca), si indica ⧖ che definisce un blocco di attesa.
Se devo far uscire dell'informazione, il blocco viene disegnato ▱ ed è un blocco di passaggio dove il token non si ferma.
(... continua) ESEMPIO
Activity diagram - PAGAMENTO
In primis bisogna aspettare fine mese.
Il token ora è il pagamento mensile di un rivenditore.
Il rivenditore controllo l'elenco dei biglietti attivati e poi l'azienda verifica l'elenco. Devo quindi vedere se ci siano errori o no nella revisione. Se c'è un errore metto il blocco che mi faccia avvisare subito il rivenditore per correggere l'errore senza aspettare il mese seguente.
Se tutto è corretto, avviso lo stesso il rivenditore e continuo con il pagamento. Ora devo capire quale sistema informativo usi per il pagamento e se quindi mi lascia dell'informazione o no.



20/10/2015

Andiamo a vedere dei costrutti più complessi ed alcuni pattern che si possono verificare realmente.
I dati e le informazioni sul processo sono disponibili nel modello concettuale. In alcuni casi potrebbe essere utile esplicitare quali dati vengono trasferiti.
Questo si può fare "appendendo" ad una transizione un oggetto, chiamato nodo oggetto. Questo è un'istanza di una delle due classi del modello concettuale. È reo così rappresentato



È un'informazione trovata da un'azione e riutilizzata da un'altra.

implicita

che indica la chiusura di quello specifico token e non di tutte l'attività.

22/10/2015

A livello intuitivo abbiamo già usato i nodi di scelta per le strutture di loop ovvero il ciclo è costruito per eseguire una certa serie di azioni ed eventualmente ripeterle.
Un altro modo di costruire un ciclo è far ricadere la scelta all'inizio in cui mi chiedo se è il caso di far ripetere quella parte di processo ancor prima di averla eseguita la prima volta (struttura di loop - while). Questo è indicato con un rombo in cui entrano due frecce e ne escono due, ma il token uscirà sempre dalla parte che soddisfa la guardia.
Cercheremo di limitarci a queste due tipologie di cicli che sono di tipo strutturato (che hanno un punto di inizio e una freccia sola in ingresso e una sola via di uscita). Usando questo tipo è difficile dire dove inizia o finisce un ciclo.
Parlando di attività complesse, bisogna considerare ~~attività implicita~~ la terminazione implicita, ovvero un altro modo di terminare un token. Il nodo si raffigura con un cerchio con la croce dentro:

ed è un nodo che termina il token che lo raggiunge.
Questo modo serve a modellare "concetti" un poco più complessi, come ad esempio i casi in cui ci sono

Prova scritta del 25/06/2015

Tempo a disposizione: 2 ore. Non è permesso consultare testi o appunti.

Parte 1

Si consideri il seguente scenario:

Negli ultimi mesi si è assistito ad un'esplosione di servizi di trasporto persone alternativi ai Taxi, come ad esempio il famoso Uber. Questo scenario descrive un ipotetico servizio simile, semplificato, offerto da una società chiamata Tuber.

La società Tuber si offre come intermediario tra due categorie di utenti: i guidatori ed i passeggeri.

Un guidatore è un privato, registrato nel sistema con tutti i propri dati personali, proprietario di un'auto di cui ha fornito gli estremi (targa, assicurazione, ...). Ciascun guidatore è dotato di un'applicazione sul proprio smartphone, attraverso la quale può dichiarare in ogni istante la propria disponibilità al trasporto. Quando il guidatore è disponibile, può ricevere delle richieste di trasporto da un qualsiasi passeggero, il quale richiede di essere trasportato dalla sua posizione attuale (che sarà diversa, ma auspicabilmente vicina, alla posizione attuale del guidatore), ad un indirizzo specificato. Il guidatore può accettare o rifiutare la richiesta. Se la richiesta viene accettata, il guidatore si reca all'indirizzo ove si trova il passeggero, lo carica (segnalandolo all'applicazione), lo trasporta, e lo lascia all'indirizzo di destinazione (confermando il termine del trasporto tramite l'applicazione).

Dal lato opposto, i passeggeri sono privati che hanno scaricato un'apposita applicazione per smartphone ed hanno fornito alcuni minimi dati personali. Ciascun passeggero può, in ogni momento, richiedere un trasporto, indicando l'indirizzo di destinazione (quello di partenza viene rilevato automaticamente dall'applicazione). Il sistema cercherà il guidatore disponibile più vicino, e segnalerà al passeggero che il servizio è stato preso in carico. Nel caso in cui il primo guidatore non accetti il servizio, si passerà al successivo più vicino, e così via. Al termine di ogni viaggio, il passeggero deve esprimere un giudizio sul guidatore, poiché i guidatori con i giudizi troppo bassi vengono periodicamente espulsi dal sistema.

Il costo del tragitto viene calcolato automaticamente dall'applicazione, la quale compie una stima iniziale sulla base della distanza da percorrere, ed un valore finale sulla base dell'effettivo percorso e del tempo trascorso. L'importo viene addebitato automaticamente dalla carta di credito del passeggero (fornita in fase di registrazione) ed accreditato a fine mese al guidatore (al netto di una fee di servizio).

Nel contesto dello scenario delineato sopra, si definisca:

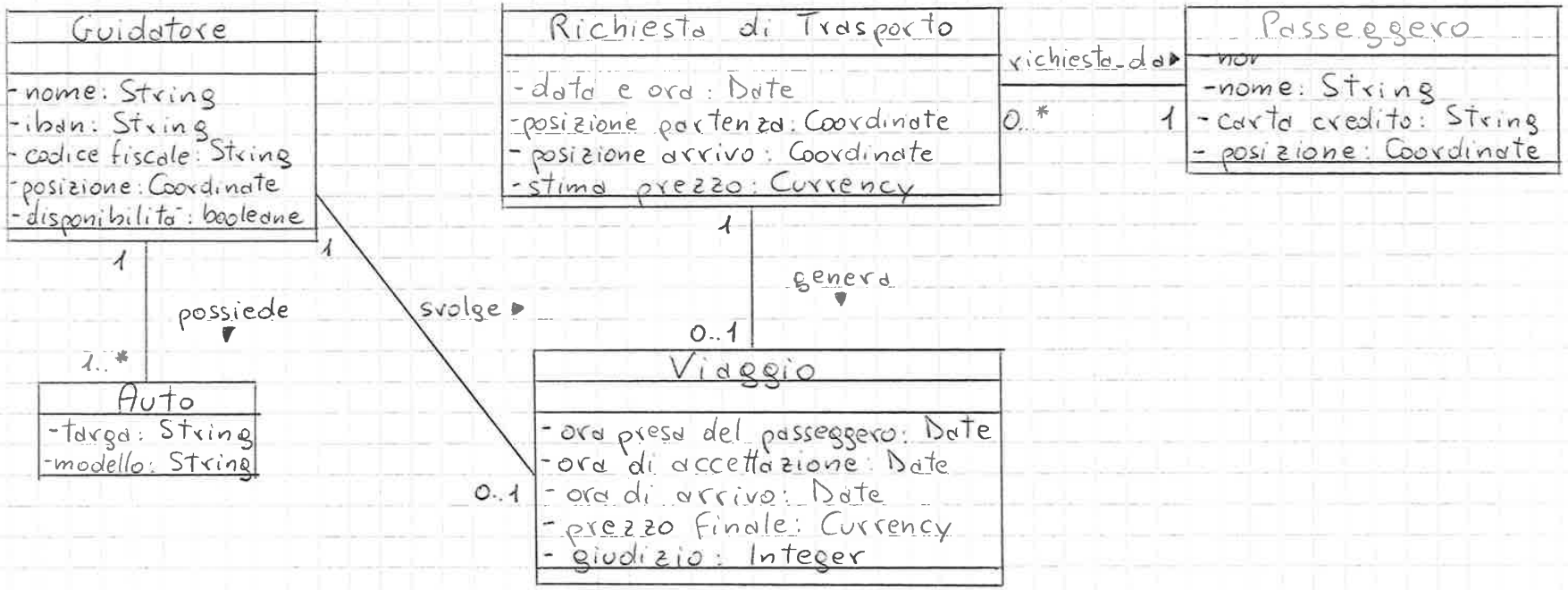
1. Il modello informativo concettuale (diagramma delle classi UML).
2. Il modello del processo (diagramma delle attività UML), comprendente sia le fasi di registrazione, che quelle di trasporto.
3. I tre KPI ritenuti più rilevanti per la valutazione del processo, in particolare facendo riferimento alla *convenienza per il guidatore*.
4. Il modello dei casi d'uso (diagramma dei casi d'uso UML) a livello *user goal*.

NB: è necessario modellare esclusivamente gli aspetti direttamente rilevanti per il sistema informativo.

Abbiamo 2 classi inizialmente: GUIDATORE e PASSEGGERO. Non serve creare una superclasse PRIVATO perché non ci sarebbe nulla da metterci e perché non serve.
Abbiamo una classe AUTO che metto in relazione con GUIDATORE.
Questa parte è statica perché cambia molto lentamente, ad esempio se si iscrivono nuovi utenti o i guidatori cambiano macchine.
GUIDATORE avrà vari attributi personali e due che cambiano (posizione e disponibilità).
Di PASSEGGERO avrò anche attributi personali e la posizione.
Anche di AUTO avrò attributi relativi alla macchina.
Iniziamo a modellare la vera parte di trasporto.
abbiamo due classi RICHIESTA DI TRASPORTO e VIAGGIO.
La RICHIESTA DI TRASPORTO nasce perché un passeggero fa qualcosa ed ho bisogno dell'informazione su chi ha fatto richiesto.

- La RICHIESTA ~~DI~~ TRASPORTO ha le informazioni di chi ha fatto richiesta. È diverso avere chi ha fatto la richiesta e chi devo trasportare.

Anche RICHIESTA sarà definito da alcuni attributi come data, posizione e prezzo.
Un PASSEGGERO potrà fare un numero arbitrario di RICHIESTA (0,... *) e ogni RICHIESTA è relativa ad 1 PASSEGGERO.
In RICHIESTA ~~di~~ ho come attributo la posizione di partenza perché il passeggero può spostarsi e a me serve sapere il punto di inizio e di arrivo.

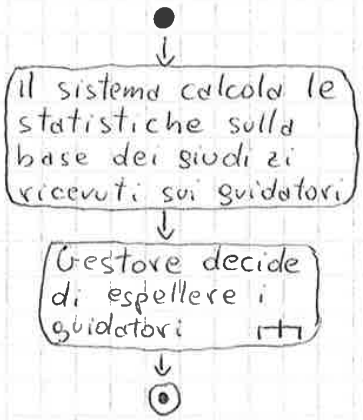


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 31 di 190

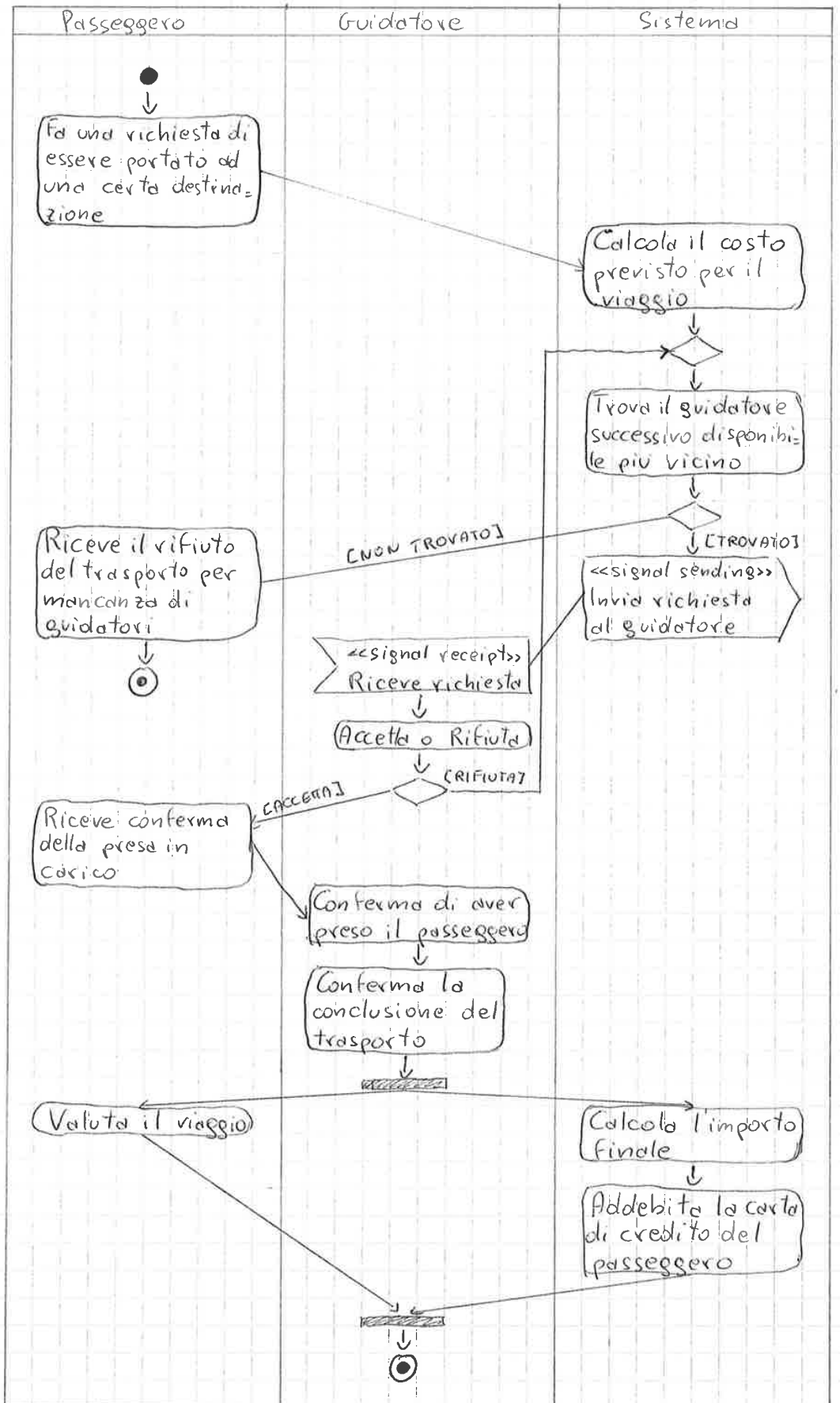
act Pagamento guidatori

Per la valutazione dei guidatori, periodicamente qualcuno deve controllare la media dei voti. Il token è la valutazione da parte del gestore del sistema. Non è specificata la modalità di espulsione dei guidatori, posso comunque creare un'attività complessa che me lo indichi, ma che non vado a dettagliare.

act Valutazione guidatori



Queste 3 attività appena viste sono asincrone fra di loro e non devono comunicare.
L'attività di TRASPORTO la analizzeremo tramite le corsie e cercheremo di esprimere cosa succede.
Il tutto inizia dai passeggeri, perché se non ci fossero il sistema starebbe fermo.
Il token è l'intero ciclo di vita di un viaggio: dalla richiesta alla conclusione. Avremo generato tanti



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 35 di 190

Definitions and specifications

User requirement definition

The software must provide a means of representing and accessing external files edited by other tools

System requirements specification

1.1 The user should be provided with facilities to define the type of external files
1.2 Each external file type may have an associated tool which may be applied to the file
1.3 Each external file type may be represented as a specific icon on the user's display
1.4 Facilities should be provided for the icon representing an external file type to be defined by the user
1.5 When a user selects an icon representing an external file the effect of that selection is to apply the tool associated with the external file type to the file represented by the selected icon

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 37 di 190

Cosa è un requisito? È un evento di varia natura che "parla" del sistema. Può essere di alto livello (descrizione o frase) oppure può essere formalizzato maggiormente (formulazione specifica di tipo matematico o di foglio). A qualunque livello, la caratteristica principale del requisito è che sia VERIFICABILE.

REQUISITI

Quali sono le fasi che guidano il processo di ingegneria dei requisiti?

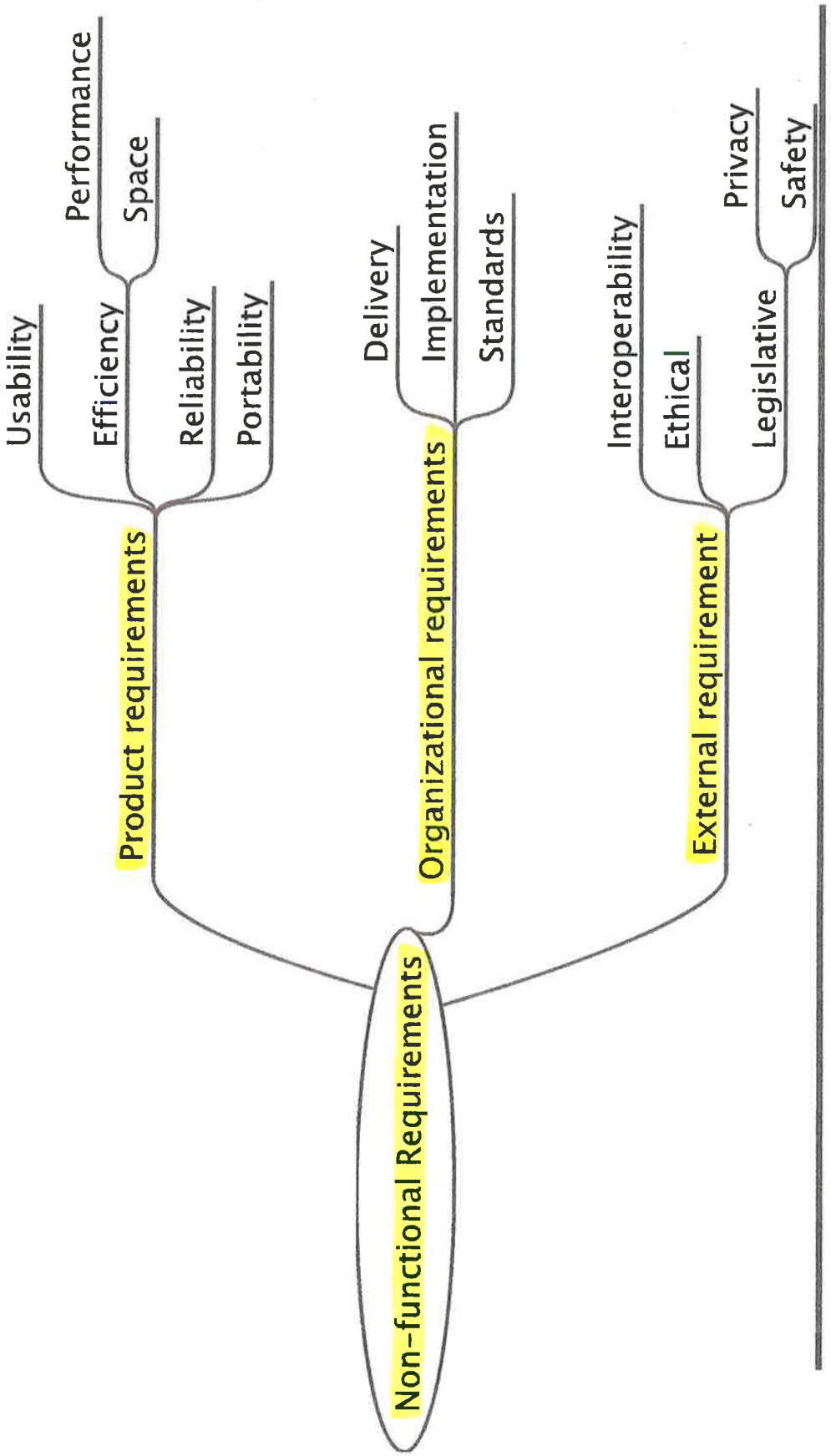
- **Sollecitazione**: cioè bisogna riuscire a capire dal committente quali sono i requisiti;
- **Analisi**: sulle base delle informazioni ricavate, devo metterle in ordine e analizzo per scrivere i requisiti;
- **Formalizzazione**: scrittura dei requisiti (diagrammi, tabelle, ...);
- **Verifica e convalida**

I requisiti vengono scritti dal progettista/analista e ce ne sono vari che vengono scritti per persone diverse. Quelli più importanti sono i requisiti utente che descrivono, dal punto di vista dell'utente, cosa fa e come è fatto il sistema. Poi ci sono i requisiti di sistema che contengono le informazioni che l'utente non vede. I primi sono i più importanti perché su quelli avviene l'accettazione del prodotto finale.
I requisiti sono degli sportiacque: da un lato c'è l'utente che conferma se quello scritto fosse effettivamente quello che voleva, dall'altra c'è il mondo più "tecnico" che capisce quali sono questi requisiti.
I diversi tipi di requisiti sono pensati per essere leggibili

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 39 di 190

specificato, quando qualunque aspetto del sistema informativo è descritto da qualche parte. Il sistema include tutte le funzionalità e relativi casi particolari
Un altro requisito quasi "scontato" è che siano tra loro COERENTI ALL'INTERNO ovvero non sia possibile trovare pagine di requisiti con due frasi in conflitto tra loro.
Correttezza e non ambiguità si valutano sul singolo requisito, completezza e coerenza su un insieme.
La completezza e la coerenza sono due cose che spesso sono in conflitto tra loro: più cerco di essere completo e aggiungo informazioni e più è alto il rischio di non essere coerente.
Questo per quanto riguarda la parte statica.
Il sistema informativo si costruisce col tempo, non è immediato: si pianifica e, lotto per lotto, lo si realizza.
Ogni requisito dovrà avere un RANKING o PRIORITÀ diverso dagli altri.
Altra caratteristica è la VERIFICABILITÀ cioè deve essere noto come fare a verificare, requisito per requisito, se sia soddisfatto dal sistema finito.
Deve esistere una procedura di tempo finito e costo ragionevole attraverso la quale una persona o un sistema automatico possa verificare che il sistema rispetti quel requisito. Un requisito non verificabile è inutile.
I requisiti dovrebbero essere MODIFICABILI per via di errori, incongruenze, nuove esigenze.
Abbiamo un documento di requisiti che evolve nel

Non-functional requirements



3/11/2015

Iniziamo a parlare dei CASI D'USO che sono una delle possibili rappresentazioni per i requisiti funzionali.
Il caso d'uso è uno dei modi in cui il sistema può essere usato dai suoi utenti. Il sistema informativo promette ai propri utenti che potranno fare una determinata operazione.
L'utente viene chiamato ATTORE PRIMARIO.
Un caso d'uso sono le funzioni che il sistema mi offre e che per me hanno un significato.
Ci stiamo dando un MODO per descrivere che cosa il sistema fa. Il fatto che sia dal punto di vista dell'utente, mi obbliga a non descrivere aspetti tecnologici.
Lo scambio di dati tra sistemi informativi diversi non può essere descritto con i casi d'uso, come neanche il processo.

> Un caso d'uso descrive un insieme di interazioni, orientato verso l'obiettivo, tra attori esterni ed il sistema considerato

Per descrivere un caso d'uso dobbiamo descrivere chi è l'attore/attori di cui uno sarà l'attore primario.
Il caso d'uso inizia quando mi predispongo mentalmente a fare un'azione.
Il caso d'uso è definito da quali sono le interazioni che l'utente può fare.
Dobbiamo, quindi, definire l'attore e l'obiettivo del perché sta usando il sistema.
Avremo due livelli: uno grafico e uno descrittivo.
Nel descrivere la sequenza dobbiamo tener conto

delle varianti.

Un attore è normalmente un essere ~~es~~ umano, esterno al sistema, che vi DEVE interagire. Non interessa sapere gli utenti singoli quali sono, ma serve sapere i gruppi, i ruoli che svolgono.

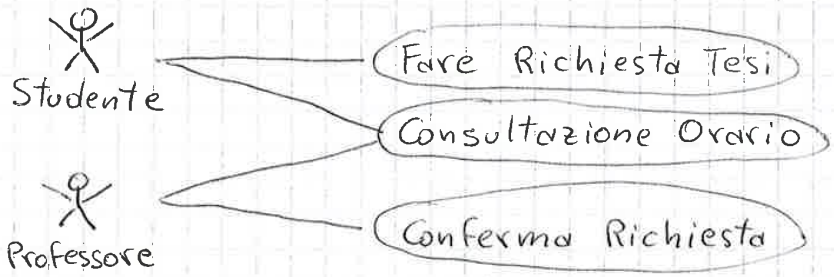
C'è sempre l'attore primario che è colui che ha l'obiettivo. In alcuni casi è necessaria la presenza di attori secondari che aiutano la conclusione di un caso d'uso.

L'obiettivo deve essere qualcosa che abbia un certo valore per l'attore. Deve rispondere a delle domande: voglio fare queste azioni, per quale scopo? Cosa voglio raggiungere? Accetto di fare dei passaggi per poter raggiungere l'obiettivo.

Oltre agli utenti ci sono anche gli stakeholders. Dobbiamo tenere conto di quale sia il loro beneficio quantificabile.

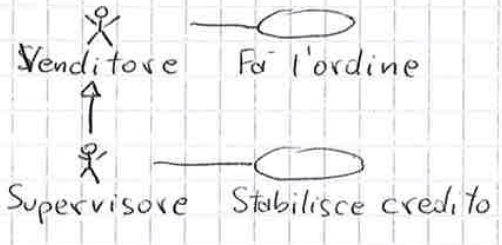
Devo capire lo scopo e capire quali sono i casi d'uso di responsabilità di questo pezzo di sistema informativo che sto descrivendo. Devo sempre capire l'ambito nel quale viene sviluppato.

Nella RAPPRESENTAZIONE GRAFICA ho l'omino e sotto la descrizione della sua ruolo e poi indico ai casi d'uso a cui può accedere.



d'uso principale al secondario. Può essere utile se A serve anche in altri contesti o, se il caso di uso A è molto articolato, fa comodo separarlo e mettere un richiamo. Una cose che hanno in comune i casi d'uso INCLUSI è che, normalmente, non corrispondono ad obiettivi dell'utente.

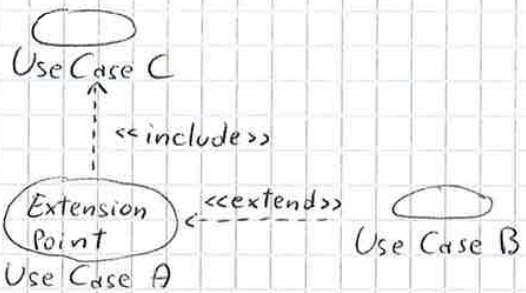
Abbiamo poi la generalizzazione che posso usare tra attori e anche tra casi d'uso.



È possibile anche avere ereditarietà tra casi d'uso.

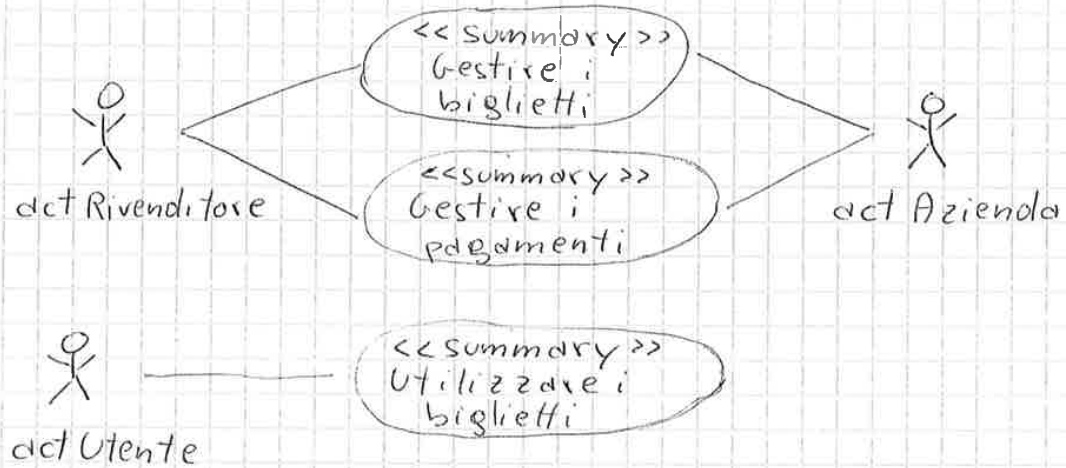
L'altra relazione è l'estensione.

"Una relazione di estensione dal caso d'uso B al caso d'uso A indica che un'istanza del caso d'uso A può essere aumentata attraverso il comportamento specificato da B" => Ho un caso d'uso A che fa il suo lavoro. Mentre faccio A posso fare anche B, indipendentemente. (ESEMPIO: entro in un sito per fare una ricerca e posso fare un acquisto) Nel caso estensione, graficamente, si va controcorrente rispetto la freccia.



Posso indicare nel diagramma dei casi d'uso una sezione in cui indicare i punti estensione

deve offrire e ciascuno di questi attori. Lo possiamo fare attraverso un diagramma di casi d'uso che chiameremo "summary".
Un attore è colui che interagisce in qualche momento col sistema e sicuramente uno sarà il Rivenditore (non può avere questo nome perché così si chiama già un concetto, lo denomineremo act Rivenditore).
In quali macroaree interagisce col sistema? Sicuramente nella fase di convalida dei biglietti (li attiva, li vende, ...) e poi la parte a fine mese in cui gestisce le fatturazioni.
A livello summary troverei lo "gestire biglietti" e le "gestire pagamenti".
Possiamo gestire i casi d'uso con degli stereotipi che permette di estendere il concetto che già esiste e che viene indicato con le doppie frecce (<< >>)
Abbiamo un altro attore che è l'utente (act Utente) che può "utilizzare il biglietto" e anche l'azienda di trasporti (act Azienda) che "gestisce i biglietti" e pagamenti.
Nè Azienda nè utente compaiono nel modello delle classi la prima perché è l'ambiente in cui operiamo, il secondo perché non sappiamo nulla di lui.



sistema e, per ciascun punto, si valutano le possibili deviazioni che possono avvenire.
Il nucleo principale è l'elenco puntato, ma il caso d'uso ha bisogno di altre informazioni, alcune obbligatorie altre facoltative. Due importanti sono il nome e il livello riprese dal diagramma. Poi c'è bisogno dell'ambito in cui si sta lavorando. Le altre due informazioni primarie sono l'attore primario e l'intenzione che l'attore primario ha nell'iniziare il caso d'uso.
Abbiamo poi dei casi opzionali come stakeholders' interests ovvero gli interessi degli stakeholder che vogliono che il caso d'uso vada a buon fine.
Ci sono 4 campi che possono essere descritti insieme: precondition, minimum guarantees, success guarantees e trigger. Il primo elenca delle condizioni che devono essere vere prima di poter cominciare il caso d'uso. Ci dice che se le condizioni non sono verificate, "non si entra", e ci dice che, una volta entrato, si è sicuri che le condizioni erano vere.
[Noi intenderemo sempre sottointesi il login dell'utente e l'azione precedente del processo]
le due voci successive sono condizioni che verifichiamo ex-post. Quali sono i fatti che posso dire certamente quando si è concluso il caso d'uso. Queste sono le garanzie minime e quelle di successo perché il caso d'uso inizia e non è detto finisca bene. Quali sarebbero le condizioni nuove in caso di successo del caso d'uso?

3. il rivenditore seleziona il numero di corse e conferma
4. il sistema restituisce il biglietto caricato e stampe lo scontrino

Extensions:

2.d. il sistema segnala che il biglietto risulta illeggibile o danneggiato e lo espelle

2.d.1. Torno al punto 1.

3.d. il rivenditore annulla l'operazione

3.d.1. Il sistema espelle il biglietto non programmato

TERMINA CON FALLIMENTO

4.d. Il sistema segnala l'impossibilità di restituire il biglietto

TERMINA CON FALLIMENTO

4.b. il sistema segnala l'impossibilità di programmare il biglietto (es: errore di comunicazione col chip) ed espelle il biglietto non programmato

4.b.1. Torno al punto 1

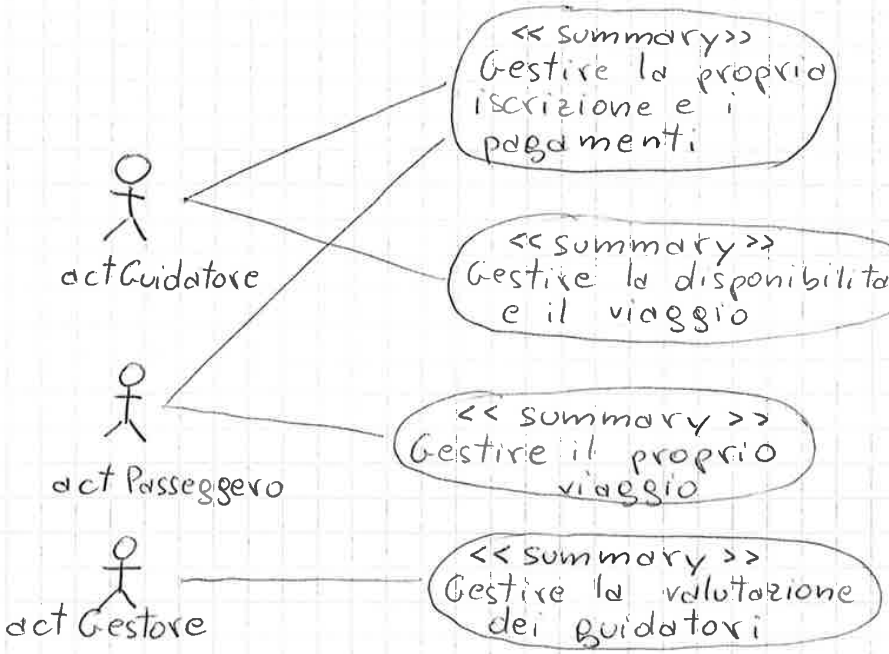
4.c. il sistema segnala l'impossibilità di stampare lo scontrino (es: fine carta) ed espelle il biglietto caricato

TERMINA CON SUCCESSO

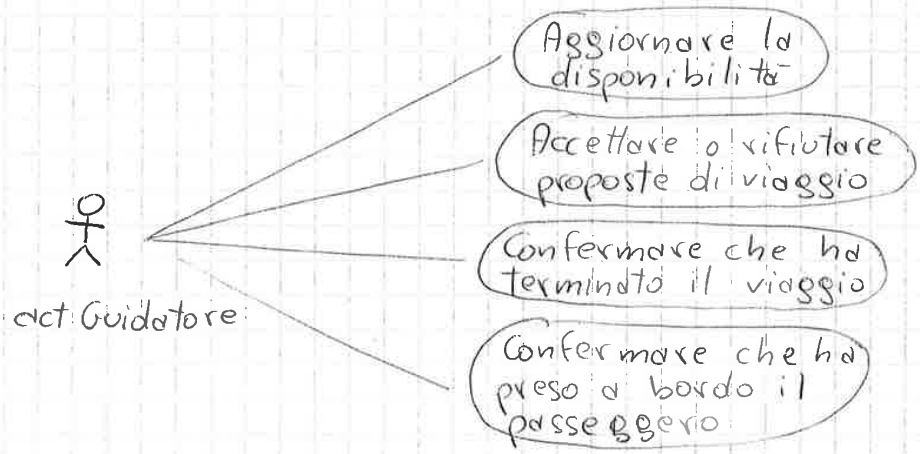
10/11/2015

Considerando l'~~esempio~~ esame del 25/6 in cui abbiamo visto quali sono le operazioni da compiere nelle varie fasi del processo, chi le deve compiere e quali sono le dipendenze logiche tra di esse (le frecce).
Analizzando il "Trasporto" e applicando i casi d'uso, gli utenti cosa vedono di questi processi? Cosa possono fare? Quali obiettivi possono raggiungere?
Vediamo quali sono gli attori di questo sistema e le macroaree che offre.

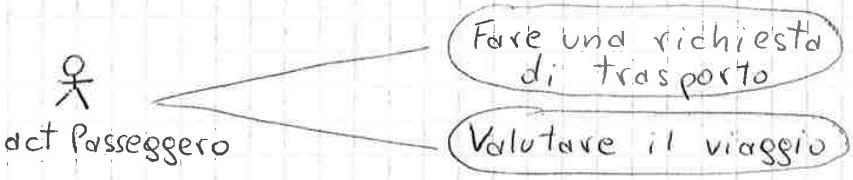
questo può essere incluso in "Gestire il viaggio".



Vediamo ora il caso d'uso riguardante la Gestione della disponibilità e il viaggio. Per quanto riguarda le conferme da dare, quanti casi d'uso faccio? Bisogna farne 2 separati perché l'obiettivo a breve termine di entrambi è diverso.



Vedo cosa l'utente può fare, ma non quando può farlo. Analizziamo dal lato viaggiatore il caso d'uso "Gestire il proprio viaggio"



Success Guarantees: La richiesta è confermata e un guidatore si sta recando dall'utente

Trigger: --

Main Success Scenario:

1. Il Passeggero specifica l'indirizzo di destinazione (nella App) e richiede il viaggio
2. Il sistema mostra l'indirizzo di destinazione da lui compreso ed il costo stimato del viaggio
3. Il passeggero conferma
4. Il sistema mostra un messaggio di attesa
5. Il sistema conferma la presa in carica del servizio

Extensions:

2.a. Il sistema indica che l'indirizzo non è stato riconosciuto. Torno al punto 1

3.a Il passeggero non conferma la destinazione. Torno al punto 1

3.b Il passeggero annulla l'operazione

TERMINE CON FALLIMENTO

5.a. Il sistema rifiuta il servizio

TERMINE CON FALLIMENTO

Vediamo ora la descrizione dell user case "Accettare o rifiutare proposte di viaggio"

Use Case: Accettare o rifiutare proposte di viaggio

Scope: Sistema Informativo Tuber, App mobile Guidatore

Level: User Goal

Intention in Context: Accettare la proposta di viaggio proveniente dal sistema

Primary Actor: Guidatore

Stakeholder's Interests: Guidatore = svolgere il viaggio
Passeggero = arrivare a destinazione

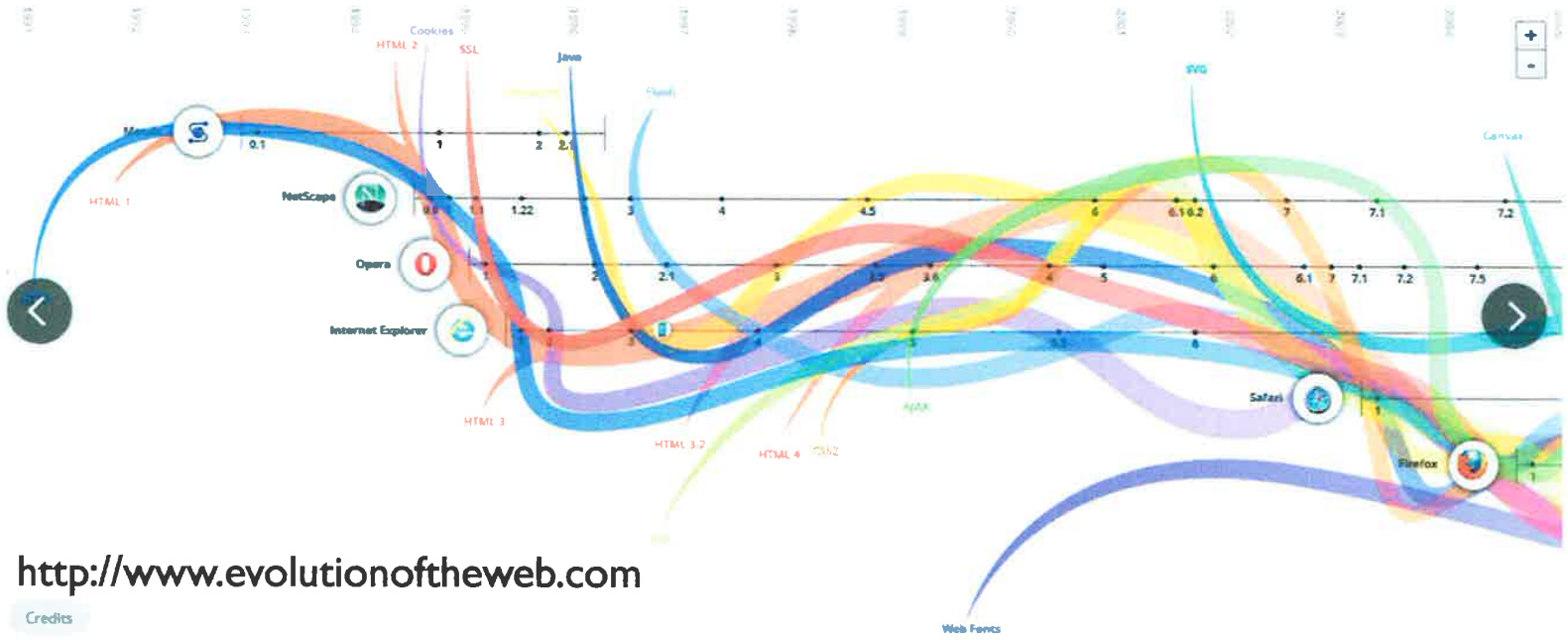
Precondition: Il guidatore deve avere il G-PS attivo
Il guidatore deve essere disponibile

Minimum Guarantees:

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 59 di 190

unicamente ~~al processo~~ alle informazioni che il sistema deve gestire e al processo con cui devono essere gestite.
Cosa è un CLOUD? Si tratta di un modello di calcolo in cui, invece che avere un server o un gruppo di server dedicati al mio sistema, ho una quantità variabile di server e pago l'utilizzo a consumo. Il difetto di questo sistema è che se ne dimensiono la grandezza in modo sbagliato, non sono efficiente nell'uso delle risorse. Il futuro della mia azienda potrebbe dipendere dal fornitore di cloud. In "pericolo" potrebbero essere anche la privacy perché i dati su server che non so dove sono e come sono gestiti.
Dal punto di vista tecnologico, cosa mi distingue un'architettura da un'altra? Posso misurarla in base al tempo di risposta, la scalabilità ovvero quanto il sistema può gestire un numero crescente di domande, la disponibilità del sistema, ...
Il dimensionamento del sistema va fatto tenendo conto delle reti locali (LAN) o geografiche (MAN o WAN).
Il modello funzionale lo abbiamo diviso in 3 categorie (dati, casi d'uso e processi). Un modello alternativo per la realizzazione di processi è CRASO (Customer Request Activity ~~&~~ organiSation Output) che li descrive da un punto di vista diverso. Diciamo chi è il customer del processo, quali sono gli input, gli output, le unità organizzative coinvolte nel processo e le attività presenti nel processo.
Posso osservare vari tipi di processo: mono-funzionali, inter-funzionali ed inter-organizzativi.

Evolution of web architectures



http://www.evolutionoftheweb.com

Credits

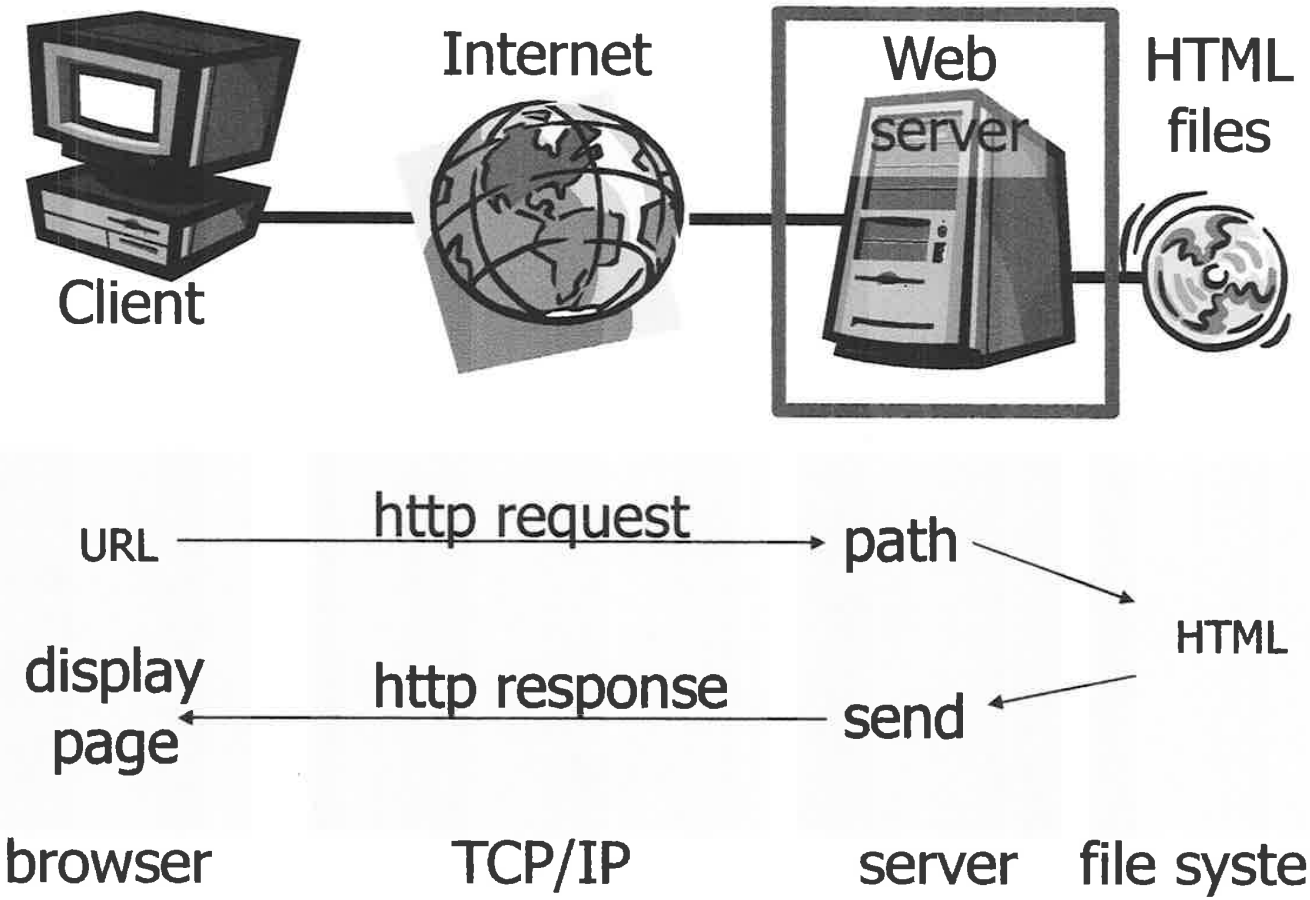
© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 63 di 190

più recenti.
Chiunque lavori nell'ambito web deve saper gestire tanti linguaggi e tante tecnologie diverse.
Vedremo ora alcune tecnologie di cui dovremo capire il funzionamento base.

ARCHITETTURE WEB

Le architetture che vediamo sono fatte a livelli, sono architetture strutturate su dei livelli. Alcuni livelli sono fissi: il browser. Se parliamo di architetture web diciamo che sono applicazioni che l'utente usa aprendo un browser. Questo è all'apice della struttura ed è quello che viene usato per collegarsi al resto. Questo attraverso internet si collega al server e si scambiano informazioni tramite l'infrastruttura di internet e i suoi protocolli.
Il server web è un oggetto abbastanza standardizzato. I livelli sotto sono più specifici: l'application server è quello più applicativo dove c'è la logica applicativa ed è il livello dove si distingue cosa sta facendo quel sito. Questo richiede che ci sia qualcuno che lo programmi.
Al livello più basso ci sono i dati, database server su cui lavora la logica applicativa.
Negli ultimi anni sono nati modelli ancora più complessi per costruire un sistema informativo.
Dall'"alto" posso avere tanti tipi di dispositivi che richiedono servizi e/o svolgono funzioni sul sistema informativo. È sempre più frequente il caso per cui, quando devo interagire col mio sistema informativo, mi servono informazioni che questo non possiede, non ha

Example



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 67 di 190

odotto. Alla fine il browser ci mostra sullo schermo la pagina finita. Le due richieste (file HTML e immagine) sono due richieste completamente separate.

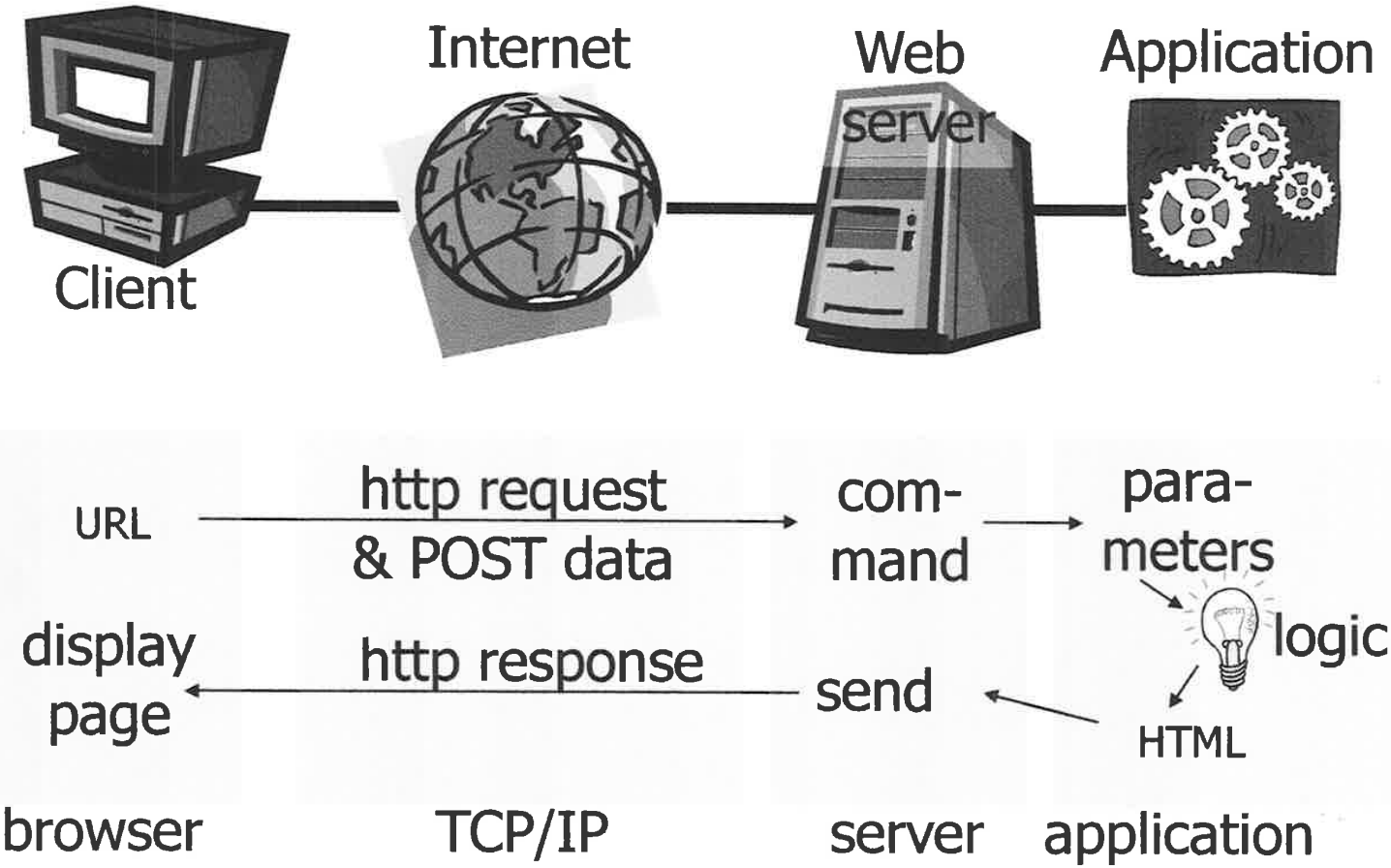
19/11/2015

Né il server web né il browser dipendono dall'applicazione: sul pc dell'utente non deve installare nulla, basta che ci sia un browser. Tutti i browser servono per navigare in internet, ma tra loro sono diversi e, a volte, si comportano in modo più o meno diverso. Un'applicazione web fatta bene deve essere abbastanza indipendente dal browser che la usa.

Apache è un prodotto open source molto utilizzato per svolgere il ruolo di web server. Lui, come gli altri, deve solo rispondere alle [redacted] richieste che gli vengono fatte. Un server in crescita è nginx, anche questo open source. Conoscendo 2/3 prodotti riusciamo a coprire il 80% del mercato del browser web, fortemente dominato da prodotti open source (gratuiti).

Un server web riceve la richiesta http, ma il modo in cui risponde deve dipendere dalle funzionalità del mio sistema operativo. Chi fa le cose diverse? Il livello applicativo, che fa sì che le pagine web siano generate DINAMICAMENTE. Un limite è che questa tecnologia è composta di file html, per cui le informazioni sono sempre le stesse: le pagine html devono essere generate dinamicamente in base all'utente che si collega, per fare in modo che ci siano le impostazioni proprie e possa vedere solo la "sua parte". Questo è quello che fa l'APPLICATION SERVER che ha un livello dove gira la

Dynamic web transaction



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 71 di 190

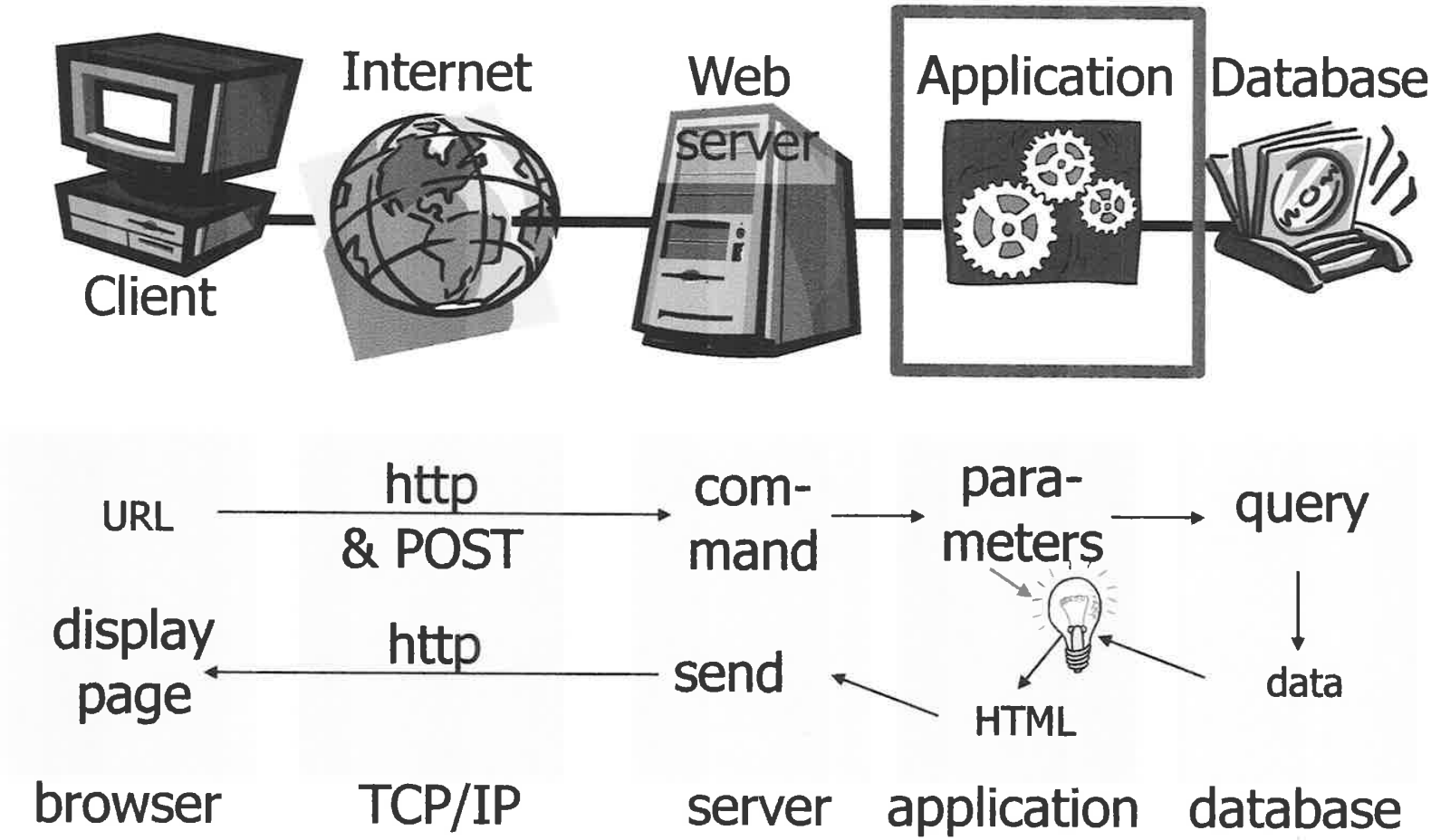
la richiesta ed assemblarla. Una volta assemblato la richiesta http, viene spedita (in t_1) al server web che la riceve un po' dopo. La distanza tra t_1 e t_2 dipende dalla velocità del collegamento, soprattutto, dalla latenza del collegamento internet. (latenza = tempo da quando inizio la connessione a quando trasferisco il primo byte). In t_2 il pacchetto dati arriva al server web, che ci mette un poco di tempo ad esaminare la richiesta, capisce di chi è e capisce essere una richiesta dinamica. In t_3 la passa all'application server. Il Web server funziona da Frontend: analizza tutte le richieste e quelle dinamiche le passa all'application server. Il lavoro finisce in t_5, ~~t_6~~ manda la risposta ~~dell'~~ del web server a cui arriva in t_6. Questo prende il file, ci mette l'intestazione in formato http e in t_7 la spedisce al browser. Fra t_7 e t_8 il contenuto viaggia di nuovo su internet dal server web al browser.

Solitamente il tempo che passa tra t_2 e t_7 è molto poco: $t_2 - t_3$ e $t_6 - t_7$ sono dell'ordine dei microsecondi, $t_4 - t_5$ dipende da quanto è complicato il lavoro che deve fare il programma.

Il tempo tra t_7 e t_8 (tempo impiegato per far arrivare la risposta al browser) è più lungo di quello tra t_1 e t_2 perché la quantità di dati da trasferire è maggiore, a parità di velocità di trasmissione di rete. Qui non è importante la latenza quanto la larghezza di banda ovvero il numero di byte al secondo che riesco a trasferire dopo il primo.

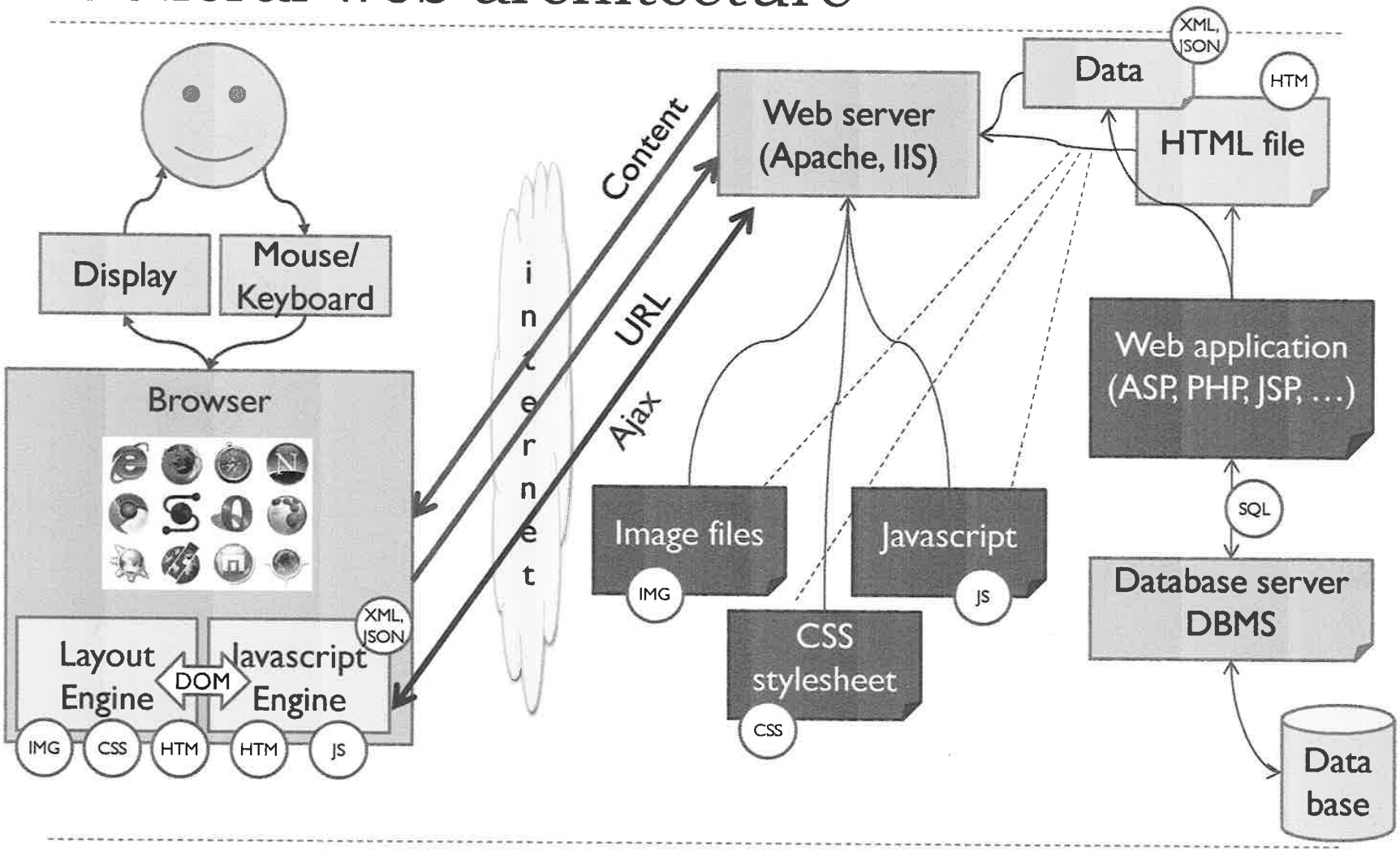
Il tempo introdotto ogni volta che devo trasferire dei dati in rete è somma della LATENZA, che è il

Example



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 75 di 190

General web architecture



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 77 di 190

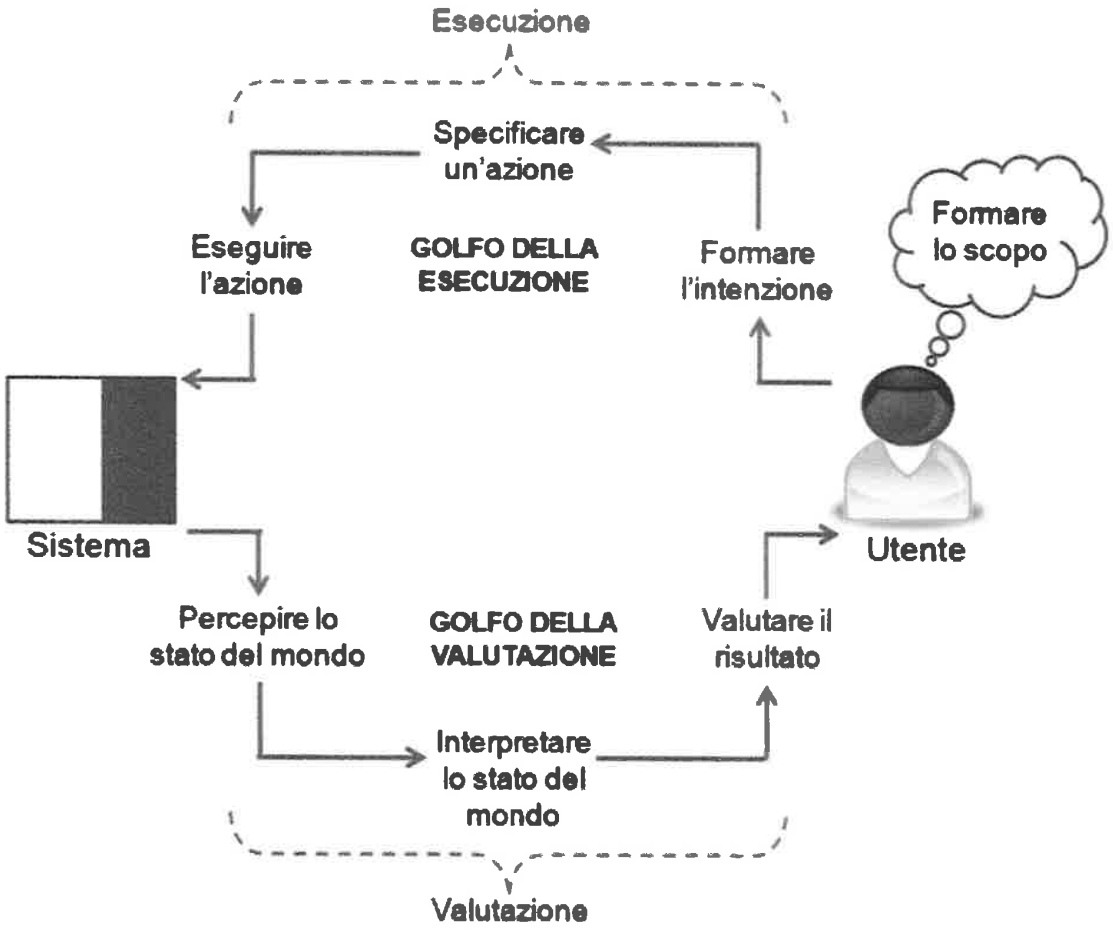
e poi ha un "foglio di stile" che definisce come quell' html deve essere visualizzato.
Il salto di complessità maggiore c'è stato per fare evolvere i contenuti in tempo reale: per far sì che il browser possa offrire un'interazione dell'utente con la pagina, deve sapere come questo reagisce ma non può saperlo perché è standard. Il browser è in grado di gestire i contenuti statici, ma al suo interno nasce un linguaggio di programmazione (Java script).
Si è riusciti così ad avere delle applicazioni web più facilmente usabili. Il limite è però che questo migliora solamente il contenuto che già ha.
È stata aggiunta la possibilità, per il codice javascript, di fare, all'interno della stessa pagina, delle richieste http in più (Ajax): da ora il controllo della logica applicativo lo ha javascript, che garantisce ciò che l'utente sta facendo e se ha bisogno di informazioni le chiede al server.
Abbiamo due momenti nella vita di una pagina web: il momento della richiesta della pagina e poi quando pagina ha mi arriva un nuovo html e da qui il controllo passa a javascript, che può chiedere altre informazioni.

24/11/2015

Vediamo come possano dei casi d'uso all'applicazione delle interfacce, cioè quali sono le modalità concrete con cui l'utente può interagire col sistema informativo ed effettivamente scambiare dati o informazioni.
Chiediamoci, dal punto di vista della progettazione, come affrontare il problema. Devo fornire all'utente un

modello mentale di ciò che sto facendo, che corrisponda col modello concettuale di dati: l'utente non sa nulla e deve capire ciò che può o deve fare e che informazioni può o deve inserire o avere. L'importante è che il sistema informativo sia USABILE il più possibile.
Noi abbiamo utente e sistema e nei casi d'uso abbiamo definito i "dialoghi" tra loro, quali informazioni si scambiano. Il sistema contiene al suo interno una parte dedicata a gestire l'interfaccia con l'utente. Questa parte intercetta e gestisce TUTTE le interazioni che l'utente può fare col sistema: raccoglie tutti gli input dell'utente, ma anche le azioni che forniscono informazioni al sistema. Viceversa, l'interfaccia dovrà generare tutti gli output che il sistema fornisce all'utente.
Due cose sovente associate, ma tra loro indipendenti, sono la complessità strutturale e funzionale. La complessità funzionale è il numero di diverse funzioni che può svolgere un sistema. La complessità strutturale è, invece, di quante parti è composto il sistema.
Un altro tipo di complessità è la complessità d'uso ovvero quanto è facile da usare o difficile le funzioni del sistema. Non è vero che se una cosa ha più funzionalità, abbia anche una complessità d'uso maggiore. Quanto è difficile usare qualcosa rispetto a quante funzioni offre sono scale diverse.
Dovremo cercare di inserire dentro la complessità funzionale una elevata, una funzionalità d'uso di modellizzazione dell'interfaccia che renda la complessità d'uso la più bassa possibile.

Il modello di Norman



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 83 di 190

Definire i requisiti insieme all'utente

Tecnica	Serve per	Vantaggi	Svantaggi
Questionari	Rispondere a domande specifiche.	Si possono raggiungere molte persone con poco sforzo.	Vanno progettati con grande accuratezza, in caso contrario le risposte potrebbero risultare poco informative. Il tasso di risposta può essere basso.
Interviste individuali	Esplorare determinati aspetti del problema e determinati punti di vista.	L'intervistatore può controllare il corso dell'intervista, orientandola verso quei temi sui quali l'intervistato è in grado di fornire i contributi più utili.	Richiedono molto tempo. Gli intervistati potrebbero evitare di esprimersi con franchezza su alcuni aspetti delicati.
Focus group	Mettere a fuoco un determinato argomento, sul quale possono esserci diversi punti di vista.	Fanno emergere le aree di consenso e di conflitto. Possono far emergere soluzioni condivise dal gruppo.	La loro conduzione richiede esperienza. Possono emergere figure dominanti che monopolizzano la discussione.
Osservazioni sul campo	Comprendere il contesto delle attività dell'utente.	Permettono di ottenere una consapevolezza sull'uso reale del prodotto che le altre tecniche non danno.	Possono essere difficili da effettuare e richiedere molto tempo e risorse.
Suggerimenti spontanei degli utenti	Individuare specifiche necessità di miglioramento di un prodotto.	Hanno bassi costi di raccolta. Possono essere molto specifici.	Hanno normalmente carattere episodico.
Analisi della concorrenza e delle best practices	Individuare le soluzioni migliori adottate nel settore di interesse.	Evitare di "reinventare la ruota" e ottenere vantaggio competitivo.	L'analisi di solito è costosa (tempo e risorse)

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 85 di 190

mano che il progetto evolve, mi aiutino ad ottenere le conferme da parte degli utenti che collaborano col progetto. Ci sono vari livelli: • schizzo • wireframes • immagine statica • functional mockups. Questi 4 sono livelli sempre più vicini al risultato finale.
Ogni volta, inizialmente, dobbiamo decidere in che ambito lavorare (web, desktop, mobile) perché poi le convenzioni cambiano.
Ora, per ogni passo del caso d'uso (per ogni elemento di interfaccia) dobbiamo capire che elementi ci sono sullo schermo, che connessione hanno tra loro, e devo dare delle priorità (quali sono le azioni immediate e quali richiedono più passaggi). In questo processo mi aiuto costruendo dei prototipi ovvero dei modelli incompleti del sistema, che permettono di vedere la sua interfaccia utente.
[Riprendiamo i livelli di sopra! Uno SCHIZZO è fatto a mano su un foglio di carta; WIREFRAMES, "cornici", schizzi fatti dal pc che richiamano delle interfacce; IMMAGINE STATICA dove ci sono figure o grafici per migliorare l'interfaccia utente; FUNCTIONAL MOCKUPS ovvero immagini parzialmente funzionali]

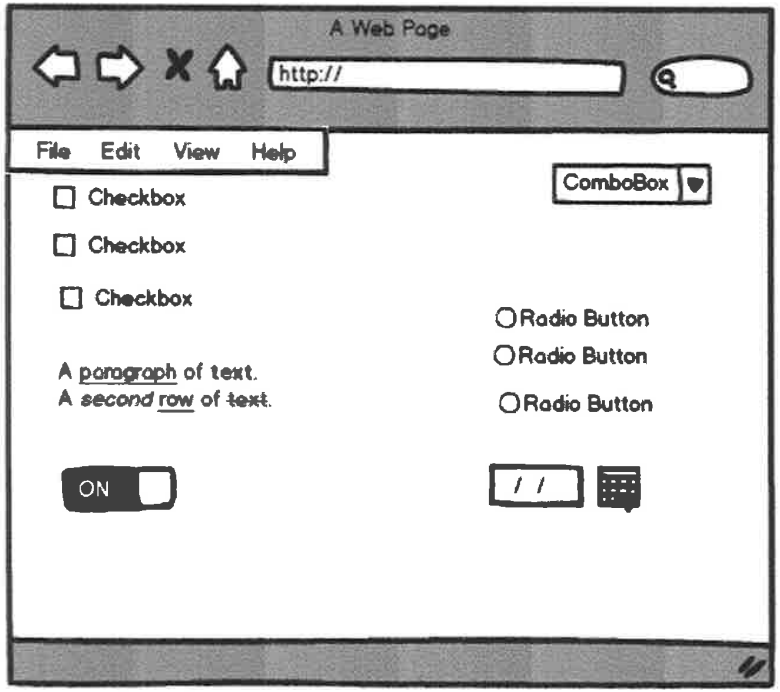
26/11/2015

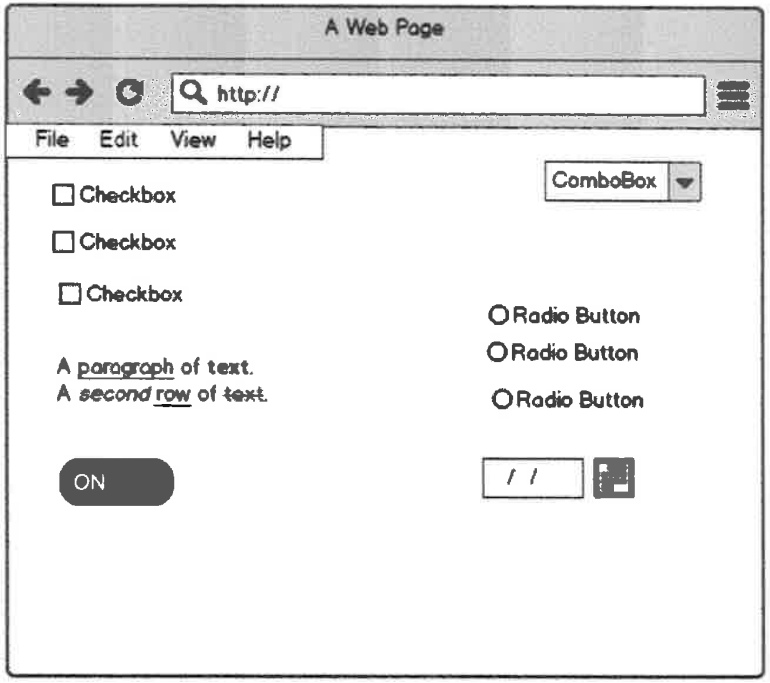
La fase del prototipo va dal documento dei requisiti fino al sistema finito. Dai documenti, in cui non c'è praticamente nulla, si può modificare il prototipo rendendolo dinamico finché non assume sempre maggiore completezza e arriva quasi a trasformarsi nel sistema quasi ultimato. Serve ad avere un feedback

Schizzo: step 0

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 89 di 190

Mockup vs Wireframe





© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 91 di 190

uno ~~stesso~~ ORIZZONTALE fa riferimento ad un prototipo di modello molte caratteristiche con poco dettaglio. Dal punto di vista VERTICALE abbiamo dei prototipi che modellano meno funzionalità del sistema, ma vengano definite con maggiore dettaglio.
Il prototipo può essere ~~usato~~ USATO in 3 modi diversi: possiamo avere prototipi STATICI (qualcosa, ad esempio, disegnato su fogli di carta), DINAMICI (mentre l'utente fa delle cose, può cambiare, come ad esempio Power Point. Il contenuto non cambia in modo dipendente dalle azioni che fa l'utente), INTERATTIVO (dove il prototipo può reagire in maniera anche diversa alle azioni che l'utente fa).
A seconda di come i prototipi vengono usati, possono DURARE a lungo o essere scartati subito. È possibile fare della base del sistema anche con power point, ma già nell'ottica di farlo per il sistema completo; altrimenti posso avere prototipi "use e getta" di tipo operativo.
Ciascuno su queste variabili ci aspettiamo diversi tipi, più o meno ricchi o dettagliati o fedeli, di realizzazione di un prototipo.
Con uno schizzo si vede con un livello di dettaglio minimo, un punto di partenza per l'interfaccia e andando avanti si riesce a migliorare sempre più l'idea di base con la quale si era partiti.
Uno degli elementi costituenti le interfacce: il mio obiettivo finale è quello di costruire un'interfaccia che permetta lo scambio di informazioni che devo sviluppare su 3

Le problematiche principali sono capire come costruire la navigazione, quale ordine di lettura dare e che elementi visualizzare. In queste fase devo puntare ad avere un risultato che sia facile da apprendere, efficiente (deve permettere all'utente di fare il minor numero di errori possibili), facilmente memorizzabile, soddisfacente.

1) SE ci hanno delle alternative mutuamente escludenteri, dati discreti, ~~non~~ se i dati sono rappresentati in modo verbale e ci sono poche opzioni
E non è possibile inserire un valore nuovo, il contenuto non cambia mai, ho abbastanza spazio, uso i Radio Buttons;
O non è possibile inserire un valore nuovo, il contenuto non cambia mai, non ho abbastanza spazio, uso la Tendina a Scomparsa;
O è necessario scrivere qualcosa, il contenuto può cambiare, ho abbastanza spazio sullo schermo, uso la Combo box;
O è necessario scrivere qualcosa, il contenuto può cambiare, non ho abbastanza spazio sullo schermo, uso la Tendina Chiusa;

2) SE ci sono alternative mutuamente escludenteri, dati discreti e rappresentati in modo verbale e sono in numero elevato
E ~~ho~~ non è necessario inserire un valore nuovo, il contenuto non cambia, ho abbastanza spazio, uso la Single-Selection List Box;
O non è possibile inserire un valore nuovo, il contenuto non può cambiare, non ho abbastanza spazio, uso la Drop-Down/Pop-Up List Box;
O si può inserire un valore, il contenuto può cambiare, ho abbastanza spazio, uso la Combo box;

mockup fatti di più videate che rappresentano il caso d'uso, facendo attenzione che in ogni videata ci sia lo scambio di informazioni previste dal caso d'uso e che nell'interfaccia grafica ci siano gli elementi che permettono l'inserimento e la visualizzazione dei dati.
!!! All'esame è importante che le informazioni nel mockup siano giuste, che siano coerenti con quelle descritte nel caso d'uso, che l'utilizzo degli elementi nell'interfaccia massimizzi l'usabilità. !!!
Vediamo come fare un mockup relativo ~~ao~~ al caso d'uso di Tuber, portando dal passeggero che ha una richiesta di trasporto dove avevamo 5 passaggi. Dobbiamo accorparli a 2 o 2perché, ad esempio, i punti 2 e 3 [vd. casi d'uso] sono la stessa videata. Praticamente il rapporto tra casi d'uso e mockup è 2 a 1.
Avrò una videata iniziale dove l'utente deve specificare l'indirizzo. Poi una videata dove ~~la~~ è mostrato l'indirizzo trovato e se il passeggero conferma. Una terza videata di attesa e una quarta dove il sistema conferma. Ci saranno poi ~~&~~ i mockup relativi alle estensioni.
SOLITAMENTE, se l'eccezione è dovuta al comportamento diverso da parte dell'utente, vuol dire che quel tipo di comportamento era già stato previsto nella pagina precedente e NON SERVE un mockup nuovo.

1. Il Passeggero specifica l'indirizzo di destinazione (nella App) e richiede il viaggio [FIGURA 1]
Siamo a livello Mobile e quindi ci mettiamo la "base" dell'iPhone come contenitore.