SERVICE ENGINEERING LIFE CYCLE

- Contract First

SERVICE DEVELOPMENT → SERVICE TESTING → SERVICE DEPLOYMENT

↓

SERVICE CANDIDATE IDENTIFICATION → SERVICE INTERFACE DESIGN → SERVICE DESIGN & IMPLEMENTATION

↓ ↓ ↓

REQUISITI SPECIFICHE dell'INTERFACCIA SERVIZIO IMPLEMENTATO

- Code First

↓

SERVICE CANDIDATE IDENTIFICATION → SERVICE DESIGN & IMPLEMENTATION → SERVICE INTERFACE DESIGN

COMPOSIZIONE di SERVIZI

- Coreografia: coordinamento della conversazione di un peer-service

↓

nulla gestisce l'ordine dei messaggi

- Orchestrazione: Servizio Composto che utilizza le operazioni offerte da altri servizi

↓

gestione dell'ordine dei messaggi

SOSE LIFECYCLE

- Top-Down ANALYSIS → DESIGN → DEVELOPMENT → TESTING → DEPLOYMENT → ADMINISTRATION

- Bottom-Up DEVELOPMENT → TESTING → DEPLOYMENT → ANALYSIS → DESIGN → ADMINISTRATION

INTEGRATION ARCHITECTURE : LEVEL 6

Routing, Protocolli e altri meccanismi di integrazione

- Point-to-Point PRO: Facile da progettare e veloce da implementare

CONS: Strettamente accoppiati. Cambiare qualcosa può spezzare l'integrazione

Il numero di punti di integrazione esplode se i sistemi sono tanti

- ENTERPRISE SERVICE BUS (ESB)

PRO: SCALABILITA': - componenti
architettura condivisa

INTEROPERABILITA': tecnologie standard
funzionalità come servizi

SUPPORTO SOA: orchestrazione
discovery
gestione messaggi

CONS: Maggior latenza

METRICHE di CHIDAMBER & KEMER

- WEIGHTED METHODS per CLASS (WMC)

$$WMC = \sum_{i=1}^{n} c_i \rightarrow \text{complessità ciclomatica del metodo } i$$

se $c_i = 1$ WMC = numero dei metodi della classe

se c_i = cardinalità delle base del grafo associato al metodo i

WMC = V (complessità ciclomatica)

GRAFO di CONTROLLO di FLUSSO

↳ nodi: sequenze atomiche

↳ archi: sequenze subito successive all'altra

$V(G) = e - n + 2p$

archi ↙ ↓ nodi ↘ componenti connesse

se il grafo è fortemente connesso (nodo finale collegato con quello iniziale)

$V(G) = e - n - p$

$\boxed{V(G) \leq 10}$

- RESPONSE for CLASS (RFC)

Numero di metodi della classe + il numero di metodi di altre classi chiamati da questi metodi

$RFC = [0, \infty)$, $RFC = 0 \Rightarrow$ massima sufficienza ed ereditabilità

$\boxed{RFC \leq 50}$

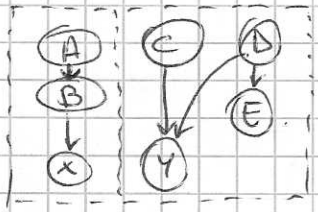
- LACK of COHESION METHODS

$$LCOM1 = \begin{cases} |P| - |Q| & \text{se } |P| > |Q| \\ 0 & \text{altrimenti} \end{cases}$$

P = coppie di metodi pubblici che non condividono attributi

Q = coppie di metodi pubblici che condividono attributi

$LCOM1 \in [0, \infty)$, $LCOM1 = 0$ massima coesione



LCOM4 = numero di partizioni nel grafo

- COUPLING BETWEEN OBJECT (CBO)

Numero di classi di dipendenza in uscita dalla classe (NON di ereditarietà)

$CBO \in [0, \infty)$, $CBO = 0$ minima interdipendenza

- DEPTH OF INHERITANCE TREE

DIT = [0, ∞)
DIT = 0 minima ereditarietà

Massima profondità dell'albero di ereditarietà

DIT elevato = maggiore complessità, maggior riutilizzo

DIT basso = codice più semplice, poco riutilizzabile

- NUMBER of CHILDREN

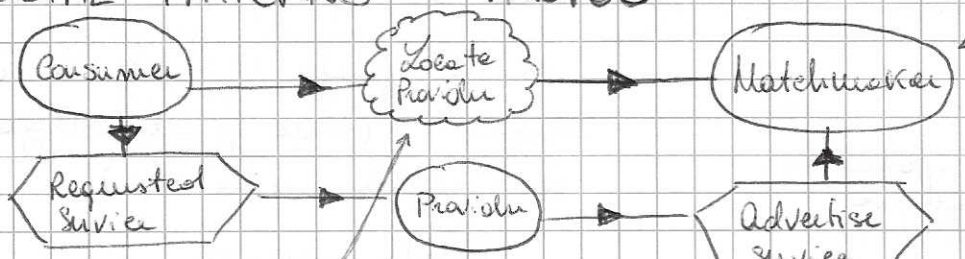
NOC ∈ [0, ∞), NOC = 0 minima ereditarietà

Numero di sottoclassi dirette

NOC elevato = uso esteso astrazione, maggiore riutilizzo

NOC basso = scarso riutilizzo del codice

SOCIAL PATTERNS TROPOS

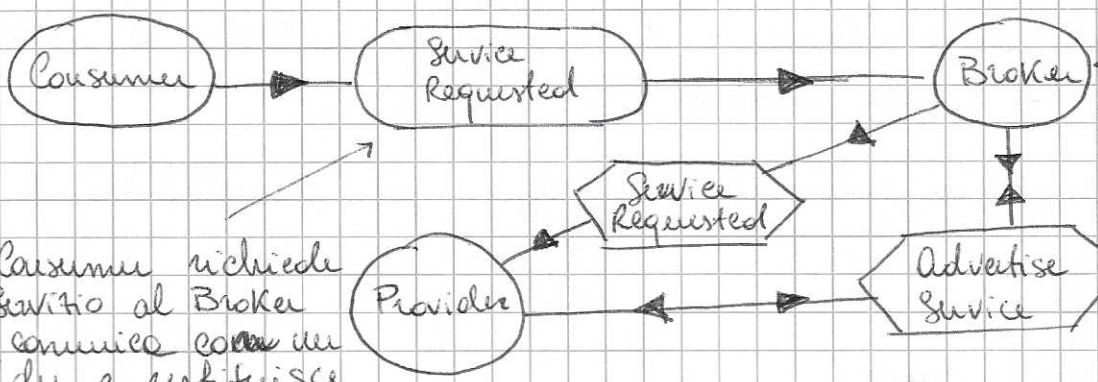


MATCHMAKER gestisce una lista di providers

ogni provider può chiedere di essere aggiunto o rimosso

il Consumer richiede il servizio e riceve l'indirizzo del provider

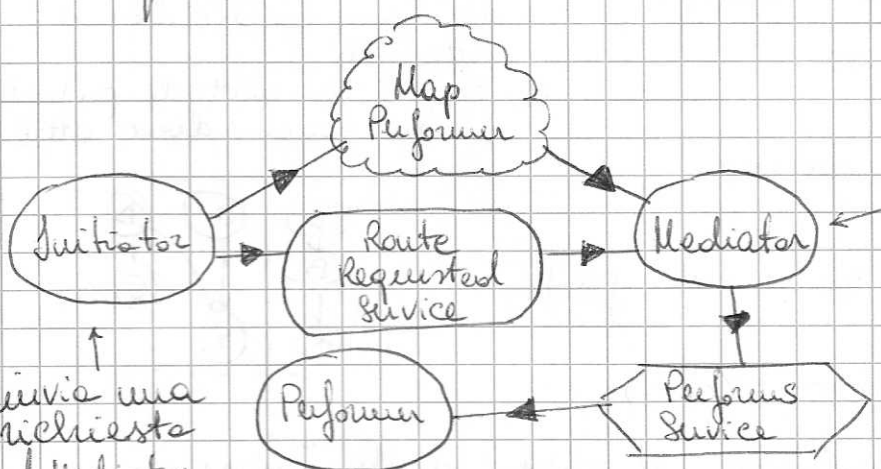
il Consumer comunica con il Provider



BROKER gestisce la lista dei providers

Il Consumer richiede un servizio al Broker che comunica con un provider e restituisce la risposta

ogni provider può chiedere di essere aggiunto o rimosso dalla lista

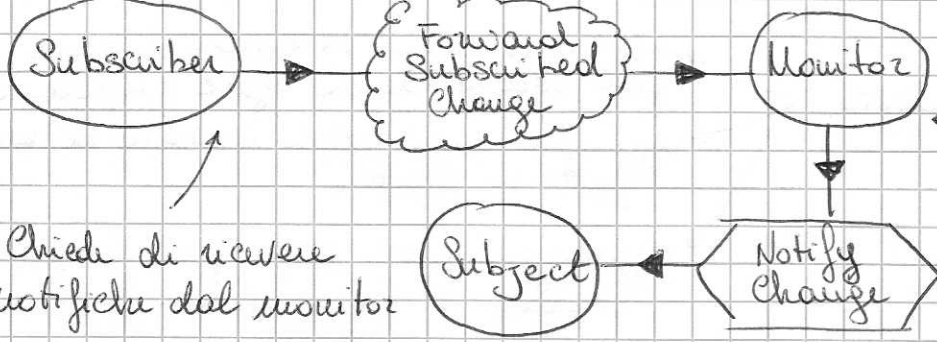


MEDIATOR

utilizza una mappa per scomporre la richiesta e invia le parti ai Performers

invia una richiesta al Mediator

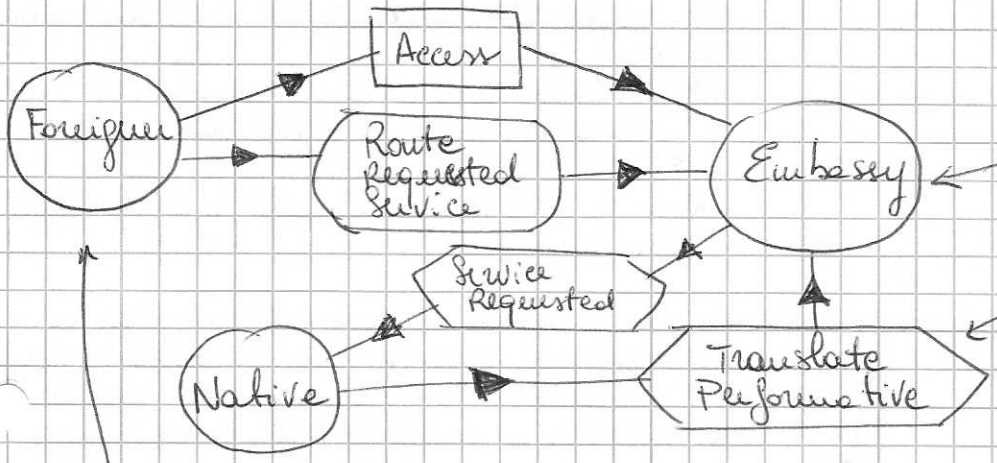
ogni performer risponde al mediator che le ricompone e risponde all'Initiator



MONITOR (PUSH)

È collegato ad un insieme di Subject

Quando un Subject modifica il suo stato invia una notifica al monitor

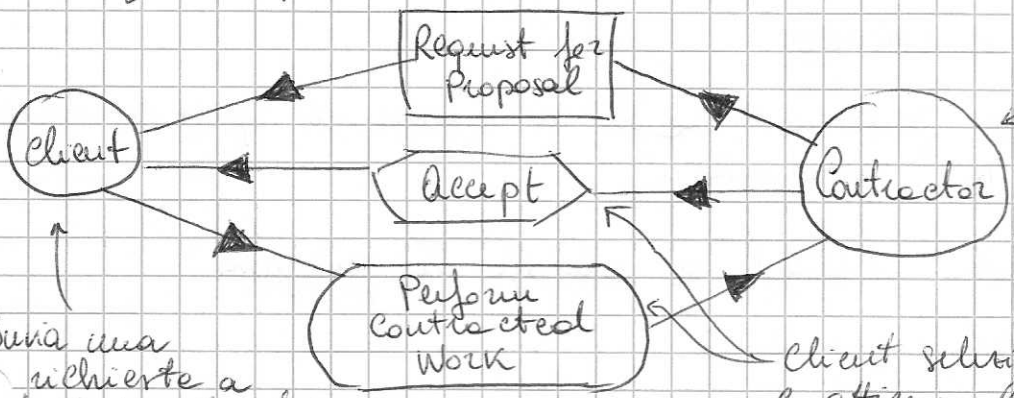


EMBASSY

Traduce le richieste da inviare a Native

Native risponde ad Embassy che traduce la risposta e la invia a Foreigner

Invia una richiesta ad Embassy (Wrapper)



CONTRACT-NET

Ogni contractor invia la sua offerta a client

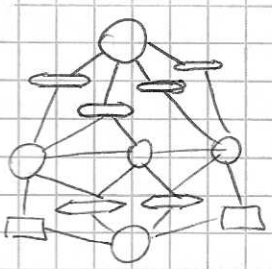
Invia una richiesta a tutti i contractor

Client seleziona l'offerta migliore e ottiene la risposta dal Contractor selezionato

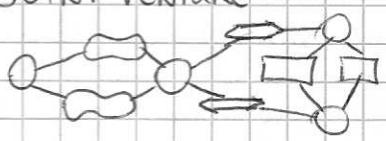
ORGANIZATIONAL STYLES

- STRUTTURA a 5

Ad alto livello si definiscono gli obiettivi, a basso livello i task



- JOINT VENTURE



Tutti i partner collaborano per obiettivi più elevati

ANALISI dei REQUISITI TROPOS:

1. Early Requirements Analysis: definire come è organizzato l'ente da risolvere
2. Late Requirements Analysis: introduzione del Sistema come attore e vedere tra i vari elementi nello scenario quali sono quelli da far analizzare al software.

PROGETTAZIONE TROPOS

1. ANALISI e RAFFINAMENTO
   - aggiungere attori ai quali delegare i goal ottenuti dalla fase precedente
   - aggiungere nuovi attori in base al social pattern scelto
   - aggiungere nuovi attori per il soddisfacimento di requisiti non funzionali
2. IDENTIFICAZIONE delle CAPACITÀ
   - Per ogni attore individuare le sue capacità dall'analisi delle dipendenze
   - Per ogni attore individuare le sue capacità dall'analisi dei task
   - Riportare le capacità nelle tabelle delle capacità
3. DAGLI ATTORI AI COMPONENTI
   - Ristrutturare gli attori in base alle capacità
     - ↳ MASSIMA COESIONE: capacità simili stesso programma
     - ↳ MINIMA INTERDIPENDENZA: le capacità devono essere suddivise in modo bilanciato tra i programmi
   - Stesse operazioni dei package
   - Ogni attore diventa un componente

TEORIA di GOODENOUGH e GERHART

Un programma P è una funzione che da un insieme di input D restituisce un insieme di output R $P: D \rightarrow R$.

Dato $d \in D$, $P(d)$ è corretto $OK(P,d)$ se soddisfa le specifiche

Un test T per P è sottoinsieme di D

P è corretto in T $OK(P,T)$ se e solo se $\forall t \in T \rightarrow OK(P,t)$

P è corretto $OK(P)$ se P è corretto in D

Un test T è ideale se e solo se $OK(P,T) \rightarrow OK(P)$

Criterio di selezione C per P: insieme di sottoinsiemi di D. Un test T è selezionato da un criterio C se e solo se: $\forall t \in T \; \exists c \in C / t \in c$ e $\forall c \in C \; \exists t \in T / t \in c$. Un criterio C è affidabile se comunque prese T_1 e T_2 selezionati da C $\Rightarrow$ $OK(P,T_1) \leftrightarrow OK(P,T_2)$. Un criterio C è valido se $\neg OK(P) \rightarrow \exists t (\text{selezionato}(t,C) \wedge \neg OK(P,t))$

AFFIDABILE: Per tutti i test dà la stessa risposta

VALIDO: Il criterio mostra che il programma non va bene con almeno un test.

TEOREMA di GOODENOUGH e GERHART

affidabile $(C,P) \wedge$ C è affidabile per P

valido $(C,P) \wedge$ C è valido per P $\longrightarrow$ OK(P)

selezionato $(C,T) \wedge$ T è selezionato da C — P è valido in tutti i casi

OK (P,T) P va bene per T

TEOREMA HOWDEN: Dato un programma arbitrario P, non esiste algoritmo che generi un test ideale finito e cioè un test finito definito da un criterio affidabile e valido

CRITERI di SELEZIONE dei CRITERI di SELEZIONE

WHITE BOX TESTING (Test Strutturali): basato sul programma indipendentemente dalle specifiche, si basa sulla nozione di copertura (esecuzione di elementi del grafo di controllo). Bisogna conoscere la struttura del programma.

BLACK BOX TESTING (Test funzionali): basato sulle specifiche indipendentemente dal programma. E' basato su cosa il programma prende in input e cosa restituisce in output. Basta sapere come si interagisce con il software.

I criteri sono complementari

GRADO di COPERTURA $G = \frac{\text{numero di elementi coperti da } T}{\text{numero totale di elementi coperibili}}$

CRITERI di COPERTURA $\longrightarrow$ Control-Flow
Data-Flow
Behaviour-based

Control-Flow: copertura delle strutture di controllo
- comandi (nodi)
- decisioni (archi)
- condizioni
- condizioni e decisioni

- cammini $\rightarrow$ decisioni $\rightarrow$ comandi
- cammini indipendenti (base del grafo) = complessità ciclomatica
- n-loop

Data-Flow: i dati condizionano i cammini. Scelta dei dati per ottenere i percorsi più significativi

- copertura delle definizioni
- copertura di tutti gli use
- copertura dei cammini du

BEHAVIOUR-BASED: ~~OOP~~ OOP
- copertura dei cammini in un diagramma delle sequenze e collaborazioni
- cammini in un diagramma di stato di una classe.

(Diagrammi di stato invece che grafi di controllo)

BLACKBOX TESTING → EQUIVALENCE PARTITIONING
BLACKBOX TESTING → BOUNDARY VALUE ANALYSIS

STRATEGIE di COLLAUDO

1. UNIT TESTING: isolare un'unità da tutto il resto
   - metodi invocati sostituiti con stub
   - metodi invocatori sostituiti con driver
1. CLASS TESTING
2. TEST di INTEGRAZIONE: una volta compiuto lo unit test si testano le unità con le interazioni reali
   - big bang test: tutto insieme in una volta
   - incremental test: moduli integrati e testati via via che sono pronti

   COME AGGANCIARE I METODI TRA LORO
   - TOP DOWN INTEGRATION: test unità di alto livello collegate a stub. Sostituiamo degli stub con unità reali e test collegati ad altri stub e così via.
   - BOTTOM UP: speculare alla top down, si parte dal basso
   - SANDWICH TESTING: intermedio fra le precedenti.
2. OBJECT-ORIENTED INTEGRATION
   - THREAD-BASED: classi coinvolte in uno stesso evento
   - CLUSTER ~~BASED~~: classi coinvolte in una collaborazione
   - USE-BASED: classi associate ad un caso d'uso
3. TEST di SISTEMA
   - stress: sistema in sovraccarico
   - performance: prestazioni del sistema
   - robustezza: proprietà del sistema fino a che non cede
   - sicurezza: proprietà di sicurezza del sistema
4. TEST di REGRESSIONE: compatibilità di una nuova versione con la vecchia
5. TEST di ACCETTAZIONE: accettazione del software da parte degli stakeholder
   - alpha testing (ambiente controllato)
   - beta testing (insieme di utenti)