Ingegneria del Software

Processi di Ingegneria dei requisiti (II)

Antonino Staiano
e-mail: **antonino.staiano@uniparthenope.it**

Rifinire i casi d'uso

- Consideriamo il caso d'uso ReportEmergency,
  - esso è sufficientemente illustrativo per descrivere come FRIEND supporta la creazione dei rapporti di emergenza e per ottenere un feedback generale dagli utenti
  - Tuttavia, esso non fornisce dettagli sufficienti per la specifica dei requisiti
  - ReportEmergency si può ampliare per includere dettagli sul tipo di incidenti noti a FRIEND e interazioni dettagliate che indicano come il Dispatcher notifica il FieldOfficer

Nome	ReportEmergency
Attori Partecipanti	Iniziato dal FieldOfficer Comunica con il Dispatcher
Flusso eventi:	1. Il FieldOfficer attiva la funzione "Report Emergency" del terminale 2. FRIEND risponde presentando una form al FieldOfficer 3. Il FieldOfficer riempie la form selezionando il livello di emergenza, tipo, località, breve descrizione della situazione. Il FieldOfficer descrive anche possibili risposte alla situazione di emergenza. Appena finito Il FieldOfficer invia la form 4. FRIEND riceve la form e notifica al Dispatcher 5. Il Dispatcher rivede le informazioni sottomesse e crea un Incidente nel DB invocando il caso d'uso OpenIncident. Il Dispatcher seleziona una risposta e accetta il rapporto. 6. FRIEND visualizza l'accettazione e la risposta selezionata al FieldOfficer
Condizioni di entrata	Il FieldOfficer è loggato in FRIEND
Condizioni di uscita	Il FieldOfficer ha ricevuto un'accettazione e una risposta selezionata dal Dispatcher, OR Il FieldOfficer ha ricevuto una spiegazione indicante perché la transazione non è stata eseguita
Requisiti di qualità	Il rapporto del FieldOfficer è accettato entro 30 secondi La risposta selezionata arriva non più tardi di 30 secondi dopo che è stata inviata dal Dispatcher

Nome	ReportEmergency
Attori Partecipanti	Iniziato dal FieldOfficer Comunica con il Dispatcher
Flusso eventi:	1. Il FieldOfficer attiva la funzione "Report Emergency" del terminale 2. FRIEND risponde presentando una form al FieldOfficer. *La form include un menu del tipo di emergenza (emergenza generale, incendio, trasporto) e la località, la descrizione dell'incidente, la richiesta di risorse, i campi dei materiali nocivi.* 3. Il FieldOfficer riempie la form *specificando al minimo il tipo di emergenza e i campi* descrizione. Il FieldOfficer descrive anche possibili risposte alla situazione di emergenza *e può richiedere risorse specifiche.* Appena finito Il FieldOfficer invia la form 4. FRIEND riceve la form e notifica al Dispatcher *con un finestra pop-up.* 5. Il Dispatcher rivede le informazioni sottomesse e crea un Incidente nel DB invocando il caso d'uso OpenIncident. *Tutte le informazioni contenute nella form del FieldOfficer sono incluse automaticamente nell'Incident. Il Dispatcher seleziona una risposta allocando le risorse all'Incidente (con il caso d'uso AllocateResources) e notifica il rapporto di emergenza inviando un breve messaggio al FieldOfficer.* 6. FRIEND visualizza l'accettazione e la risposta selezionata al FieldOfficer
Condizioni di entrata	…

Rifinire i casi d'uso

- L'uso di scenari e casi d'uso per definire le funzionalità del sistema mira a creare requisiti che saranno validati dall'utente prima dello sviluppo
- Appena inizia la progettazione e l'implementazione del sistema, aumenta il costo delle variazioni nella specifica dei requisiti e dell'aggiunta di nuove funzionalità
- Sebbene i requisiti cambiano tardi nel corso dello sviluppo, gli sviluppatori e gli utenti dovrebbero sforzarsi di affrontare i problemi con i requisiti quanto prima

Euristiche per scrivere scenari e casi d'uso

- Usare gli scenari per comunicare con gli utenti e per validare le funzionalità
- Primo, rifinire un singolo scenario per capire le assunzioni sul sistema dell'utente. L'utente può avere familiarità con sistemi simili, nel cui caso, adottare convenzioni specifiche sull'interfaccia utente renderebbe il sistema più fruibile
- Poi, definire scenari non molto dettagliati per definire lo scopo del sistema. Validare con l'utente.
- Usare mock-up solo come supporto visuale; il progetto dell'interfaccia utente dovrebbe avvenire come task separato dopo che la funzionalità è sufficientemente stabile
- Presentare all'utente alternative multiple e differenziate. Valutare diverse alternative allarga l'orizzonte dell'utente. Generare alternative differenti forza gli sviluppatori a "pensare oltre la scatola"
- Dettagliare una parte più ampia quando lo scopo del sistema e le preferenze dell'utente sono ben capiti. Validare con l'utente.

Rifinire i casi d'uso

- L'enfasi di questa attività è la completezza e la correttezza
- Gli sviluppatori identificano le funzionalità non coperte dagli scenari e le documentano rifinendo i casi d'uso o scrivendone nuovi
- Gli sviluppatori descrivono raramente casi e gestione delle eccezioni così come visti dagli attori
- Mentre l'identificazione iniziale dei casi d'uso e degli attori mira a stabilire il confine del sistema, la rifinitura dei casi d'uso porta di volta in volta più dettagli sulle caratteristiche fornite dal sistema e i vincoli ad essi associati

Rifinire i casi d'uso

- Gli aspetti dei casi d'uso che sono inizialmente ignorati e dettagliati durante la rifinitura sono:
  - Gli elementi manipolati dal sistema (in *ReportEmergency* sono stati aggiunti dettagli sugli attributi della form del rapporto di emergenza ed il tipo di incidente)
  - La sequenza a basso livello delle interazioni tra l'attore ed il sistema (sono state aggiunte informazioni su come il *Dispatcher* genera una notifica selezionando le risorse)
  - Permessi di accesso (quale attore può invocare quale caso d'uso)
  - Eccezioni mancanti e la loro gestione
  - Funzionalità comuni tra i casi d'uso

Associazioni nei casi d'uso

- Un modello dei casi d'uso consiste di casi d'uso ed associazioni di casi d'uso
- Un'associazione è una relazione tra casi d'uso
- Le associazioni più importanti sono: Communication, Include, Extend, Generalization

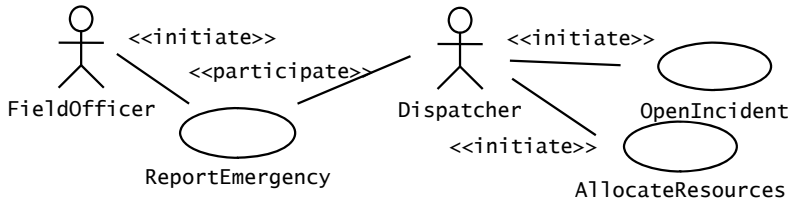
Identificazione delle relazioni tra attori e casi d'uso

- Anche sistemi di medie dimensioni hanno numerosi casi d'uso
- Le relazioni tra gli attori ed i casi d'uso danno modo a sviluppatori ed utenti di ridurre la complessità del modello ed aumentarne la comprensione
  - Si usano relazioni di comunicazione tra attori e casi d'uso per descrivere il sistema in livelli di funzionalità
  - Le relazioni di estensione sono usate per separare flussi di eventi eccezionali da flussi di eventi comuni
  - Le relazioni di inclusione sono usate per ridurre la ridondanza tra casi d'uso
  - Le generalizzazioni specializzano casi d'uso astratti

Relazioni di comunicazione

- Rappresentano il flusso di informazioni durante il caso d'uso
  - L'attore che inizia il caso d'uso dovrebbe essere distinto da altri attori con cui il caso d'uso comunica
  - Specificando quale attore può invocare uno specifico caso d'uso, specifichiamo implicitamente quale attore non può invocare il caso d'uso
  - Similmente, specificando quali attori comunicano con uno specifico caso d'uso, specifichiamo quali attori accedono informazioni specifiche e quali non possono
  - In definitiva, documentando l'iniziazione e le relazioni di comunicazioni tra attori e casi d'uso, specifichiamo grosso modo il controllo di accesso per il sistema
- Le relazioni tra attori e casi d'uso sono identificate quando sono identificati i casi d'uso

Esempio di relazioni di comunicazione tra attori e casi d'uso in FRIEND



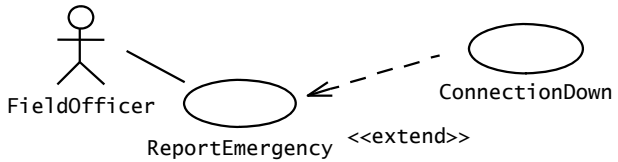
Associazione extend

- Problema
  - La funzionalità nel problema originale ha la necessità di essere estesa
- Soluzione
  - Un'associazione *extend* dal caso d'uso A al caso d'uso B indica che il caso d'uso B è un'estensione del caso d'uso A

Relazioni *extend* tra casi d'uso

- Un caso d'uso estende un altro caso d'uso se il caso d'uso esteso può inglobare il comportamento delle'estensione sotto certe condizioni
- In FRIEND, assumiamo che la connessione tra le stazioni di *FieldOfficer* e *Dispatcher* sia interrotta mentre il *FieldOfficer* sta compilando la form (ad esempio, l'auto del *FieldOfficer* entra in un tunnel)
  - La stazione del *FieldOfficer* deve notificare il *FieldOfficer* che la sua form non è stata consegnata e quali misure dovrebbe intraprendere
  - Il caso d'uso *ConnectionDown* viene modellato come una estensione di *ReportEmergency*
    - Le condizioni sotto cui il caso d'uso *ConnectionDown* è iniziato sono descritte in *ConnectionDown* anziché in *ReportEmergency*

Utilizzo della relazione *extend*



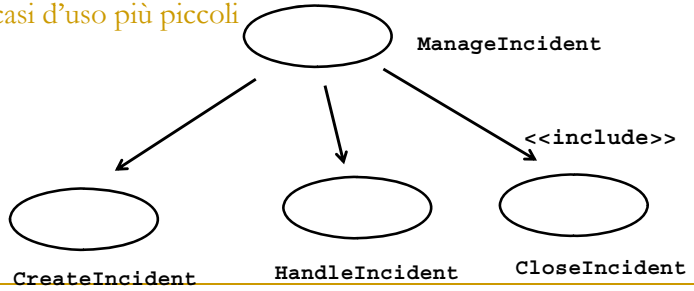
E' da osservare che il caso d'uso base può essere eseguito senza la sua estensione nelle associazioni *extend*

Relazioni *extend* tra casi d'uso

- Separare flussi di eventi eccezionali ed opzionali dal caso d'uso base comporta due vantaggi
  - Il caso d'uso base è reso più breve e più semplice da capire
  - Il caso d'uso comune è distinto dal caso eccezionale, ciò consente agli sviluppatori di trattare ogni tipo di funzionalità in modo differente
    - Ottimizzare il caso d'uso comune rispetto al tempo di risposta
    - Ottimizzare il caso eccezionale rispetto alla robustezza
  - Ambo i casi d'uso esteso ed estensione sono casi d'uso completi
    - Devono avere condizioni di entrata ed uscita ed essere comprensibili all'utente in modo indipendente

Associazione *include*: decomposizione funzionale

- Problema:
  - Una funzione nella definizione del problema originale è troppo complessa per risolverla immediatamente
- Soluzione
  - Descrivere la funzione come un'aggregazione di un insieme di funzioni più semplici. Il caso d'uso associato è decomposto in casi d'uso più piccoli



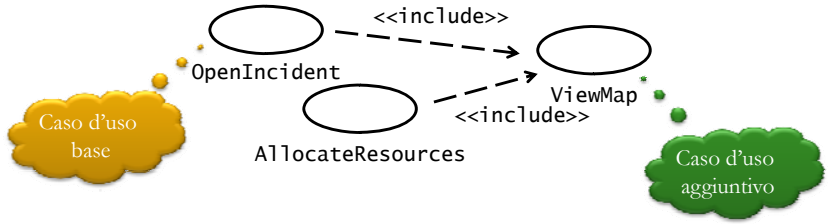
inlude: riuso di funzionalità esistenti

- Problema
  - Ci sono già funzioni esistenti. Come possiamo riusarle?
- Soluzione
  - L'associazione *include* da un caso d'uso A ad un caso d'uso B indica che un'istanza del caso d'uso A esegue tutti i comportamenti descritti nel caso d'uso B ("A delega a B")

Relazione *include* tra casi d'uso

- In questo modo la ridondanza tra casi d'uso può essere esplicitata usando la relazione di inclusione
  - Il *Dispatcher* deve consultare la mappa della città quando inizia la gestione di un incidente (ad esempio, per valutare quali zone sono a rischio durante un incendio) e quando deve allocare le risorse sul luogo (per verificare quali sono le risorse più vicine)
  - Il caso d'uso *ViewMap* descrive il flusso degli eventi richiesto quando si consulta la mappa
    - Usato si da *OpenIncident* che da *AllocateResources*

Relazione *include* tra casi d'uso



E' da osservare che il caso base non può esistere da solo. E' sempre invocato insieme al caso d'uso aggiuntivo

Relazione *include* tra casi d'uso

- Esplicitare comportamenti condivisi dai casi d'uso ha molti benefici
  - Descrizioni più brevi e meno ridondanze
- Il comportamento dovrebbe essere esplicitato in un caso d'uso separato solo se è condiviso tra due o più casi d'uso
- Un'eccessiva frammentazione della specifica dei requisiti attraverso un elevato numero di casi d'uso rende confusa la specifica ai clienti ed agli utenti

Relazioni *extend* vs relazioni *include*

- Entrambe le relazioni sono simili
  - Inizialmente potrebbe non essere chiaro allo sviluppatore quando usarle
- La principale distinzione tra le due relazioni sta nella direzione della relazione
  - Per le relazioni *include*, l'evento che innesca il caso d'uso target (quello incluso) è descritto nel flusso di eventi del caso d'uso sorgente
  - Per le relazioni *extend*, l'evento che innesca il caso d'uso sorgente (quello che estende) è descritto nel caso d'uso sorgente come precondizione
  - In altre parole, per le relazioni *include* , ogni caso d'uso che include deve specificare dove il caso d'uso incluso dovrebbe essere invocato
  - Per le relazioni *extend*, solo il caso d'uso che estende specifica quali casi d'uso sono estesi

ReportEmergency (relazione include)	**ReportEmergency (relazione extend)**
1. …	1. …
2. …	2. …
3. Il *FieldOfficer* completa la form selezionando il livello, il tipo, l'ubicazione dell'emergenza e una breve descrizione della situazione. Il *FieldOfficer* descrive anche le risposte possibili alla situazione di emergenza. Una volta che la form è completa, il *FieldOfficer* sottomette la form, al cui punto, è notificato il *Dispatcher*. *Se la connessione con il Dispatcher è interrotta, è usato il caso d'uso ConnectionDown.*	3. Il *FieldOfficer* completa la form selezionando il livello, il tipo, l'ubicazione dell'emergenza e una breve descrizione della situazione. Il *FieldOfficer* descrive anche le risposte possibili alla situazione di emergenza. Una volta che la form è completa, il *FieldOfficer* sottomette la form, al cui punto, è notificato il *Dispatcher*.
4. Se la connessione è ancora valida, il *Dispatcher* rivede le informazioni sottomesse e crea un *Incident* nel db invocando il caso d'uso *OpenIncident*. Il *Dispatcher* seleziona una risposta e accetta il rapporto di emergenza. *Se la connessione è interrotta, è usato il caso d'uso ConnectionDown.*	4. Il *Dispatcher* rivede le informazioni sottomesse e crea un *Incident* nel db invocando il caso d'uso *OpenIncident*. Il *Dispatcher* seleziona una risposta e accetta il rapporto di emergenza.
5. …	5. …

ConnectionDown (relazione include)	**ConnectionDown (relazione extend)**
	Il caso d'uso ConnectionDown estende qualsiasi caso d'uso in cui la comunicazione tra il FieldOfficer e il Dispatcher può essere persa.
1. Il FieldOfficer e il Dispatcher sono notificati dell'interruzione della connessione. Sono avvisati delle possibili ragioni per cui si è verificato l'evento (ad esempio, "la stazione del FieldOfficer è in un tunnel?")	1. Il FieldOfficer e il Dispatcher sono notificati dell'interruzione della connessione. Sono avvisati delle possibili ragioni per cui si è verificato l'evento (ad esempio, "la stazione del FieldOfficer è in un tunnel?")
2. La situazione vien trascritta dal sistema e recuperata quando la connessione è ristabilita.	2. La situazione vien trascritta dal sistema e recuperata quando la connessione è ristabilita.
3. Il FieldOfficer e il Dispatcher entrano in contatto in altri modi ed il Dispatcher inizia ReportEmergency dalla stazione del Dispatcher.	3. Il FieldOfficer e il Dispatcher entrano in contatto in altri modi ed il Dispatcher inizia ReportEmergency dalla stazione del Dispatcher.

Relazioni *extend* vs relazioni *include*

- Nella colonna di sinistra (relazione include), dobbiamo inserire il testo in due punti nel flusso di eventi dove il caso d'uso *ConnectionDown* può essere invocato
  - Inoltre, se sono descritte situazioni eccezionali aggiuntive (un funzione help sulla stazione del *FieldOfficer*), il caso d'uso *ReportEmergency* dovrà essere modificato è sarà ingombrato dalle condizioni

Relazioni *extend* vs relazioni *include*

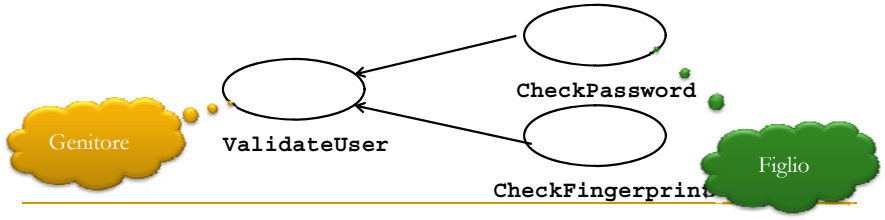
- Nella colonna di destra, dobbiamo descrivere solo le condizioni sotto cui il caso d'uso eccezionale è invocato, il che può includere numerosi casi d'uso
  - Inoltre, situazioni eccezionali aggiuntive possono essere aggiunte senza modificare il caso d'uso base (*EmergencyReport*)
- L'abilità di estendere il sistema senza modificare parti esistenti è critica, poiché consente di lasciare inalterato il comportamento originale
- La distinzione tra *include* e *extend* è una questione di documentazione
  - Usare il tipo giusto di relazione riduce le dipendenze tra i casi d'uso, la ridondanza, e abbassa la probabilità di introdurre errori quando cambiano i requisiti

Euristiche per le relazioni *extend* e *include*

- *Usare le relazioni **extend** per comportamenti eccezionali, opzionali o che si verificano di rado*
  - Un esempio di comportamento raro è il guasto di una risorsa
  - Un esempio di comportamento opzionale è la notifica di risorse nelle vicinanze che rispondono ad un incidente che non le riguarda
- *Usare le relazioni **include** per un comportamento condiviso tra due o più casi d'uso*
- *Bisogna comunque usare con discrezione le due euristiche precedenti e non sovra strutturare il modello del caso d'uso. Casi d'uso un po' più lunghi (due pagine) sono più semplici da capire e rivedere rispetto a molti casi d'uso brevi (lunghi dieci righe, ad esempio)*

Associazione di generalizzazione tra casi d'uso

- Problema
  - C'è un comportamento comune ma più specifico tra casi d'uso e vogliamo esplicitarlo
- Soluzione
  - L'associazione di generalizzazione tra casi d'uso esplicita comportamenti comuni. I casi d'uso figli ereditano il comportamento ed il significato del genitore ed aggiungono altri comportamenti
- Esempio
  - Consideriamo il caso d'uso "ValidateUser", responsabile della verifica dell'identità di un utente. Il cliente potrebbe richiedere due realizzazioni: "CheckPassword" e "CheckFingerprint"



Identificazione degli oggetti di analisi iniziali

- Uno dei primi ostacoli che sviluppatori ed utenti incontrano quando cominciano a collaborare è la terminologia differente
  - Anche se gli sviluppatori imparano la terminologia degli utenti, il problema si pone ancora quando si aggiungono nuovi sviluppatori al progetto
  - Le incomprensioni si verificano per l'uso di stessi termini usati in contesti differenti con significati diversi
- Per stabilire una terminologia chiara, gli sviluppatori identificano gli oggetti partecipanti per ogni caso d'uso
  - Devono identificare il nome, descriverli in modo non ambiguo e raccoglierli in un glossario
    - La costruzione del glossario costituisce il primo passo verso l'analisi

Glossario

- Il glossario è incluso nella specifica dei requisiti e successivamente nei manuali utente
- Gli sviluppatori mantengono il glossario aggiornato durante l'evoluzione della specifica dei requisiti
- I benefici del glossario sono diversi
  - I nuovi sviluppatori hanno a disposizione un insieme consistente di definizioni
  - Un termine singolo è usato per ogni concetto
  - Ogni termine ha un preciso e chiaro significato ufficiale

Identificazione degli oggetti di analisi iniziali

- L'identificazione degli oggetti partecipanti da luogo al modello ad oggetti di analisi iniziale
  - Questo passo, durante la scoperta dei requisiti, costituisce solo un primo passo verso il modello degli oggetti di analisi completo
  - Il modello di analisi completo solitamente non è usato come mezzo di comunicazione per sviluppatori ed utenti
    - Gli utenti il più delle volte non hanno familiarità con concetti orientati agli oggetti
    - Tuttavia, la descrizione degli oggetti ed i loro attributi sono visibili agli utenti e revisionati

Euristiche per identificare gli oggetti di analisi iniziale

- *Termini che gli sviluppatori o gli utenti devono chiarire per capire i casi d'uso*
- *Sostantivi ricorrenti nei casi d'suo (ad esempio, Incident)*
- *Entità del mondo reale di cui il sistema deve tenere traccia (ad esempio, FieldOfficer e Resource)*
- *Processi del mondo reale di cui il sistema deve tenere traccia (EmergencyOperationPlan)*
- *Casi d'uso (ReportEmergency)*
- *Sorgenti di dati (Printer)*
- *Artefatti con cui l'utente interagisce (Station)*
- *Usare sempre i termini del dominio dell'applicazione*

Identificazione degli oggetti

- Durante la scoperta dei requisiti, sono generati gli oggetti partecipanti per ciascun caso d'uso
  - Se due casi d'uso si riferiscono allo stesso concetto, l'oggetto corrispondente dovrebbe essere lo stesso
  - Se due oggetti condividono lo stesso nome e non corrispondono allo stesso concetto, uno o entrambi i concetti sono rinominati per enfatizzarne la differenza
    - Si eliminano così le ambiguità nella terminologia impiegata

Oggetti partecipanti al caso d'uso *ReportEmergency*

Dispatcher	Ufficiale di Polizia che gestisce gli Incident. Un Dispatcher apre, documenta, e chiude gli incidenti in risposta ad un EmergencyReport e altre comunicazioni con i FieldOfficer. I Dispatcher sono identificati con numeri di badge.
EmergencyReport	Rapporto iniziale su un Incident da un FieldOfficer ad un Dispatcher. Un EmergencyReport solitamente innesca la creazione di un Incident dal Dispatcher. Un EmergencyReport è composto da un livello di emergenza, un tipo (fuoco, incidente stradale, altro), una località e una descrizione.
FieldOfficer	Ufficiale di Polizia o dei Vigili del fuoco in servizio. Un FieldOfficer può essere allocato al più ad un incidente alla volta. I FieldOfficer sono identificati dai numeri di badge.
Incident	Situazione che richiede l'attenzione di un FieldOfficer. Il rapporto su un Incident può essere inserito nel sistema da un FieldOfficer o chiunque altro esterno al sistema. Un Incident è composto da una descrizione, una risposta, uno stato (aperto, chiuso, documentato), una località, e un numero di FieldOfficer.

Euristiche per il controllo incrociato di casi d'uso ed oggetti partecipanti

- *Quali casi d'uso creano questo oggetto (cioè, durante quali casi d'uso i valori degli attributi di un oggetto sono immessi nel sistema)?*
- *Quali attori possono accedere a tali informazioni?*
- *Quali casi d'uso modificano e distruggono questo oggetto (cioè, quali casi d'uso editano o rimuovono questa informazione dal sistema)?*
- *Quale attore può iniziare questi casi d'uso?*
- *E' necessario questo oggetto (cioè, c'è almeno un caso d'uso che dipende da questa informazione)?*

Identificazione dei requisiti non funzionali

- I requisiti non funzionali possono avere un impatto importante sul lavoro dell'utente in modo inaspettato
- Per scoprire in modo accurato tutti i requisiti non funzionali essenziali, ambo i clienti e sviluppatori devono collaborare in modo da identificare quali attributi del sistema (minimali) difficili da realizzare sono critici per il lavoro dell'utente
- L'insieme risultante di requisiti non funzionali tipicamente include requisiti in conflitto tra loro

Euristiche per i requisiti non funzionali

- Ci sono pochi metodi sistematici per scoprire i requisiti non funzionali
- Gli analisti usano una tassonomia (schema FURPS+) dei requisiti non funzionali per generare una lista di controllo di domande per aiutare clienti e sviluppatori a focalizzarsi sugli aspetti non funzionali del sistema
- Poiché gli attori in questa fase sono già stati identificati, la lista di controllo può essere organizzata per ruolo e distribuita agli utenti rappresentativi

Euristiche per i requisiti non funzionali

- Il vantaggio di una tale lista è che può essere riutilizzata ed espansa per ogni nuovo sistema in un dato dominio applicativo, riducendo così il numero di omissioni
- Una volta che clienti e sviluppatori identificano un insieme di requisiti non funzionali questi possono essere organizzati in grafici di rifinitura e di dipendenza per identificare ulteriori requisiti non funzionali ed identificare i conflitti

Sommario: come specificare un caso d'uso

- Nome del caso d'uso
- Attori
  - Descrizione attori coinvolti nel caso d'uso
- Condizione di ingresso
  - "Questo caso d'uso inizia quando …"
- Flusso di eventi
  - Forma libera, linguaggio naturale informale
- Condizione uscita
  - "Questo caso d'uso termina quando …"
- Eccezioni
  - Descrivere cosa accade quando qualcosa va storto
- Requisiti speciali
  - Requisiti non funzionali, vincoli

Sommario

- Il processo dei requisiti consiste nella scoperta e nell'analisi dei requisiti
- L'attività di scoperta dei requisiti differisce per
  - Greenfield Engineering, Reengineering, Interface Engineering
- Scenari
  - Modo efficace per comunicare con i clienti
  - Differenti tipi di scenari: As-Is, visionari, valutazione e addestramento
  - Casi d'uso: Astrazione di scenari
- La chiave per una buona analisi
  - Iniziare con i casi d'uso e trovare gli oggetti partecipanti
  - Se qualcuno chiede "Cosa è questo?", non rispondere immediatamente. Rispondere più appropriatamente: "Per cosa è usato questo?"