Ingegneria del Software

Analisi dei requisiti (II)

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

Mapping casi d'uso – oggetti con Diagrammi di sequenze

- Un diagramma di sequenze lega i casi d'uso con gli oggetti
  - Mostra come il comportamento di un caso d'uso (o scenario) sia distribuito tra gli oggetti partecipanti
  - Non sono molto adeguati come mezzo di comunicazione con i clienti
    - Richiedono un certo background sulla notazione
  - Possono però essere, per alcuni clienti, intuitivi e più precisi dei casi d'uso
  - In ogni caso, rappresentano una ulteriore prospettiva che consente agli sviluppatori di individuare oggetti mancanti o punti oscuri nella specifica dei requisiti

Diagrammi di sequenze per *ReportEmergency*

- Vediamo i diagrammi di sequenze associati con il caso d'uso *ReportEmergency*
  - Le colonne rappresentano gli oggetti che partecipano al caso d'uso
  - La colonna più a sinistra rappresenta l'attore che inizia il caso d'uso
  - Le frecce orizzontali attraverso le colonne rappresentano messaggi o stimoli inviati da un oggetto all'altro
  - Il tempo trascorre verticalmente dall'alto in basso

Diagramma delle Sequenze per il caso d'uso *ReportEmergency*

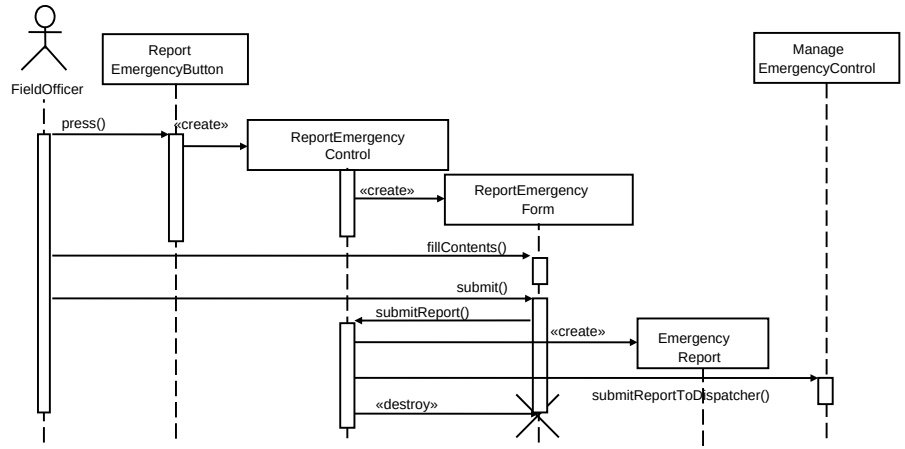


Diagramma delle Sequenze per il caso d'uso *ReportEmergency*

- Un'operazione può essere pensata come un servizio che un oggetto fornisce ad altri oggetti
- I diagrammi di sequenza illustrano anche il tempo di vita degli oggetti
  - Gli oggetti che esistono già prima dell'occorrenza del primo stimolo nel diagramma sono disegnati in cima
  - Gli oggetti creati durante l'interazione sono disegnati con il messaggio *create* che punta all'oggetto
  - Le istanze che sono distrutte durante l'interazione sono disegnate con una croce che indica quando l'oggetto cessa di esistere
  - Tra il rettangolo che rappresenta l'oggetto e la croce (o il fondo del diagramma, se l'oggetto esiste dopo l'interazione), una linea tratteggiata rappresenta l'arco di tempo in cui l'oggetto può ricevere messaggi
  - L'oggetto non può ricevere messaggi al di sotto della croce

Diagramma delle Sequenze per il caso d'uso *ReportEmergency*

- In generale, la seconda colonna di un diagramma di sequenze rappresenta l'oggetto *boundary* con cui l'attore interagisce per iniziare il caso d'uso (es., *ReportEmergencyButton*)
- La terza colonna è un oggetto control che gestisce il resto del caso d'uso (es., *ReportEmergencyControl*)
  - Da quel momento in poi, l'oggetto *control* crea altri oggetti *boundary* e può interagire anche con altri oggetti *control* (*ManageEmergencyControl*)

Diagramma delle Sequenze per il caso d'uso *ReportEmergency* (II)

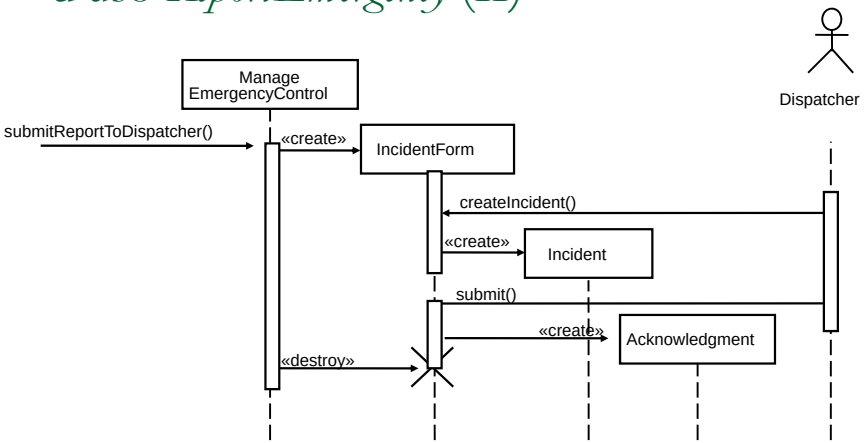
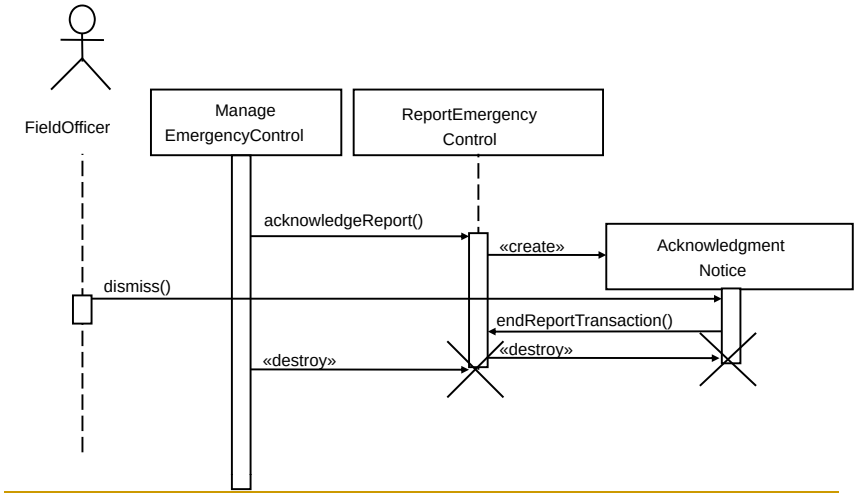


Diagramma delle Sequenze per il caso d'uso *ReportEmergency* (III)



Oggetto *Acknowledgment* per il caso d'uso *ReportEmergency*

- Quando si descrive l'oggetto Acknowledgment ci si rende conto che il caso d'uso originale ReportEmergency è incompleto
- Esso evidenzia solo l'esistenza di un Acknowledgment e non descrive le informazioni ad esso associate
  - Gli sviluppatori hanno bisogno di un chiarimento dal cliente per definire quale informazione è necessaria in Acknowledgment
  - Ottenuto il chiarimento viene aggiunto l'oggetto Acknowledgment al modello di analisi e il caso d'uso ReportEmergency è aggiornato di conseguenza

Oggetto *Acknowledgment* per il caso d'uso *ReportEmergency*

Acknowledgment	Risposta di un *Dispatcher* ad un *EmergencyReport* di un *FieldOfficer*. Inviando un *Acknowledgment*, il *Dispatcher* comunica con il *FieldOfficer* che ha ricevuto un *EmergencyReport*, ha creato un *Incident* e vi ha assegnato delle *Resource*. L'*Acknowledgment* contiene le risorse associate e il loro tempo di arrivo stimato

Nome	**ReportEmergency**
Condizioni di entrata	1. Il FieldOfficer attiva la funzione "Report Emergency" dal proprio terminale.
Flusso eventi:	3. 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. 4. 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 premendo il pulsante "Send Report", e il Dispatcher viene notificato. 4. Il Dispatcher rivede le informazioni sottomesse dal FieldOfficer 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. **L'Acknowledgment indica al FieldOfficer che l'EmergencyReport è stato ricevuto, un Incident creato e le Resource allocate all'Incident. L'Acknowledgment include le risorse ed il loro tempo di arrivo stimato.**
Condizioni di uscita	Il FieldOfficer riceve l'accettazione e la risposta selezionata

Diagrammi di sequenze: distribuzione del comportamento

- Con i diagrammi di sequenze non solo si modella l'ordine delle interazioni fra gli oggetti ma si distribuisce anche il comportamento del caso d'uso
  - Si attribuiscono le responsabilità ad ogni oggetto sotto forma di insieme di operazioni
  - Queste operazioni possono essere condivise da qualsiasi caso d'uso in cui un dato oggetto partecipa
  - E' da osservare che la definizione di un oggetto condiviso attraverso due o più casi d'uso dovrebbe essere identica
    - Se un'operazione appare in più di un diagramma di sequenze, il suo comportamento dovrebbe essere lo stesso

Diagrammi di sequenze: condivisione operazioni

- Condividere le operazioni attraverso i casi d'uso consente agli sviluppatori di eliminare le ridondanze nella specifica dei requisiti e di migliorarne la consistenza
- Osserviamo che alla chiarezza bisognerebbe sempre dare la precedenza rispetto all'eliminazione della ridondanza
- Frammentare il comportamento attraverso molte operazioni complica inutilmente la specifica dei requisiti

Diagrammi di sequenze: uso nell'analisi

- Nell'analisi, i diagrammi delle sequenze sono usati per aiutare ad identificare nuovi oggetti partecipanti e comportamenti mancanti
- Questioni legate all'implementazione come le prestazioni non dovrebbero essere affrontate in questo punto
- I diagrammi delle sequenze richiedono molto tempo per cui gli sviluppatori dovrebbero focalizzarsi prima su funzionalità problematiche o non specificate
  - Disegnare diagrammi delle sequenze per parti semplici o ben definite del sistema potrebbe non sembrare un buon investimento

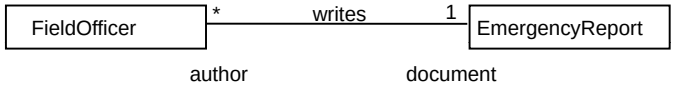
Euristiche per disegnare diagrammi delle sequenze

- *La prima colonna dovrebbe corrispondere all'attore che inizia il caso d'uso*
- *La seconda colonna dovrebbe essere un oggetto boundary (che l'attore ha usato per iniziare il caso d'uso)*
- *La terza colonna dovrebbe essere un oggetto control che gestisce il caso d'uso restante*
- *Gli oggetti control sono creati dagli oggetti boundary che iniziano i casi d'uso*
- *Gli oggetti boundary sono creati dagli oggetti control*
- *Gli oggetti entity sono acceduti dagli oggetti control e boundary*
- *Gli oggetti entity non accedono mai oggetti boundary o control; ciò rende più semplice condividere oggetti entity attraverso i casi d'uso*

Identificare le associazioni

- Un'associazione mostra una relazione fra due o più classi
  - Ad esempio, un *FieldOfficer* scrive un *EmergencyReport*
- Identificare le associazioni comporta due vantaggi
  - Chiarisce il modello di analisi rendendo esplicite le relazioni tra gli oggetti (es., *EmergencyReport* può essere creato da un *FieldOfficer* o un *Dispatcher*)
  - Consente agli sviluppatori di scoprire casi limite associati con i link
    - I casi limite sono eccezioni che devono essere chiarite nel modello
  - Ad esempio, è intuitivo assumere che più di un *EmergencyReport* è scritto da un *FieldOfficer*. Tuttavia, è lecito chiedersi se il sistema dovrebbe supportare *EmergencyReport* scritti da più di una persona o consentire *EmergencyReport* anonimi
  - Le questioni dovrebbero essere discusse in fase di analisi con clienti ed utenti

Esempio di associazione tra le classi *EmergencyReport* e *FieldOfficer*



Identificare le associazioni

- Le associazioni hanno diverse proprietà
  - Un **nome** per descrivere le associazioni tra due classi (*write* nella figura precedente)
    - I nomi delle associazioni sono opzionali e non necessariamente devono essere globalmente unici
  - Un **ruolo** in ciascuna estremità, identificando la funzione di ogni classe rispetto alle associazioni (ad esempio, *author* è il ruolo del *FieldOfficer* nell'associazione *writes*)
  - Una **molteplicità** in ciascuna estremità, identificando il numero di possibili istanze (es., * indica che un *FieldOfficer* può scrivere zero o più *EmergencyReport*, mentre 1 indica che ogni *EmergencyReport* ha esattamente un *FieldOfficer* come autore)

Identificazione delle associazioni

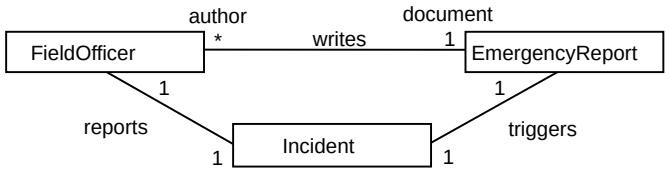
- Le associazioni fra gli oggetti sono le più importanti
  - Rivelano molte informazioni sul dominio applicativo
- In accordo alle euristiche di Abbott, le associazioni possono essere identificate esaminando verbi e predicati verbali che denotano uno stato (es., *ha*, *è parte di*, *gestisce*, *si rapporta a*, *è innescato da*, *è contenuto in*, *parla a*, *include*)

Euristiche per identificare le associazioni

- *Esaminare i predicati verbali*
- *Assegnare nomi e ruoli alle associazioni in modo preciso*
- *Usare i qualificatori quanto più spesso per identificare gli attributi chiave*
- *Eliminare qualsiasi associazione che deriva da altre associazioni*
- *Non preoccuparsi delle molteplicità fino a che l'insieme delle associazioni è stabile*
- *Troppe associazioni rendono un modello illeggibile*

Eliminare le associazioni ridondanti

- Il modello ad oggetti include, inizialmente, troppe associazioni se gli sviluppatori includono tutte le associazioni identificate dopo aver esaminato i predicati verbali



Identificazione delle associazioni

- Molti oggetti entity hanno una caratteristica che li identifica
  - usata dagli attori per accederli
    - *FieldOfficer* e *Dispatcher* hanno un numero di badge
    - *Incident* e *Report* hanno associati dei numeri e sono archiviati per data
- Una volta che il modello di analisi include molte classi e associazioni, gli sviluppatori dovrebbero andare attraverso ogni classe e controllare come esse sono identificate dagli attori e in che contesto

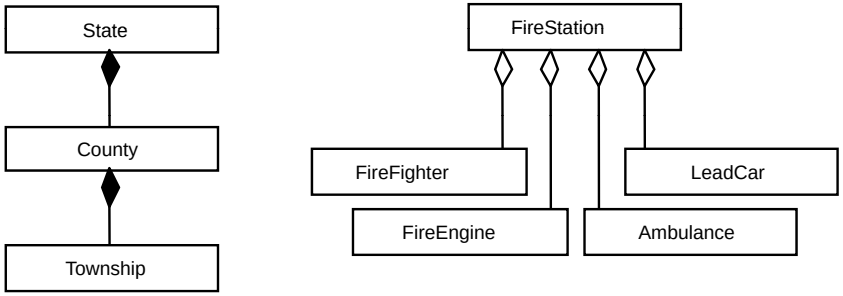
Esempio

- I numeri di badge dei FieldOfficer sono unici in tutto l'universo dell'applicazione? In una città? Una stazione di polizia?
- Se sono unici nell'ambito di una città, il sistema FRIEND può conoscere i FieldOfficer di più città?
- Questo approccio può essere formalizzato esaminando ogni classe ed identificando la sequenza di associazioni che devono essere attraversate per accedere a specifiche istanze di quella classe

Identificare le aggregazioni

- Le aggregazioni sono speciali tipi di associazioni che denotano una relazione *tutto-parte*
  - Una *FireStation* consiste di un numero di *FireFighter*, *FireEngine*, *Ambulance* e una *LeadCar*
  - Uno stato è composto da un numero di regioni che, a loro volta, sono composte da un numero di città
- Un'aggregazione è denotata con un diamante sull'estremità della parte tutto

Esempi di aggregazioni e composizioni

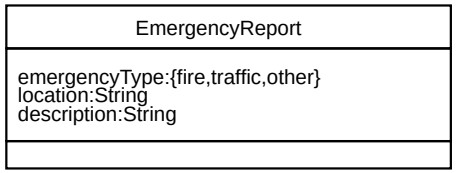


Aggregazioni:composizione e condivisione

- Esistono due tipi di aggregazioni
  - Composizioni, denotate da un diamante pieno
  - Condivise, denotate da un diamante vuoto
- Un'aggregazione di composizione indica che l'esistenza delle parti dipende dal tutto
  - Una regione è parte esattamente di uno stato
  - Una città è parte esattamente di una regione
- Un'aggregazione condivisa indica che il tutto e la parte possono esistere in modo indipendente
  - Sebbene una *FireEngine* sia parte di al più una *FireStation* alla volta, essa può essere riassegnata a differenti *FireStation* durante il suo tempo di vita

Identificare gli attributi

- Gli attributi sono proprietà degli oggetti
  - *EmergencyReport* ha le proprietà *tipo*, *località* e *descrizione*



  - Immesse *dal FieldOfficer* quando rapporta un'emergenza e mantenute dal sistema
- Solo gli attributi rilevanti per il sistema dovrebbero essere considerati quando si identificano le proprietà degli oggetti
  - *FieldOfficer* ha un codice fiscale che non è rilevante per il sistema di gestione emergenze al contrario del *badgeNumber* che è usato dal sistema per identificare i *FieldOfficer*

Attributi

- Gli attributi hanno
  - Un nome che li identifica in un oggetto
    - EmergencyReport può avere un attributo reportType e un attributo emergencyType
  - Una breve descrizione
  - Un tipo che descrive i valori leciti che può assumere
    - String, integer ecc.

Ancora sugli attributi

- Gli attributi rappresentano la parte meno stabile del modello ad oggetti
- Sono spesso scoperti o aggiunti tardi nello sviluppo quando il sistema è valutato dagli utenti
- Gli attributi aggiunti (se non relativi a nuove funzionalità) non comportano cambiamenti radicali nella struttura dell'oggetto (e del sistema)
  - Gli sviluppatori non necessitano di impiegare troppe risorse nel'identificare e dettagliare gli attributi che rappresentano aspetti meno importanti del sistema

Euristiche per identificare gli attributi

- *Esaminare frasi possessive*
- *Rappresentare stati memorizzati come attributi di oggetti entity*
- *Descrivere ogni attributo*
- *Non rappresentare un attributo come un oggetto; usare invece un'associazione*
- *Non sprecare tempo nel descrivere dettagli prima che la struttura dell'oggetto sia stabile*

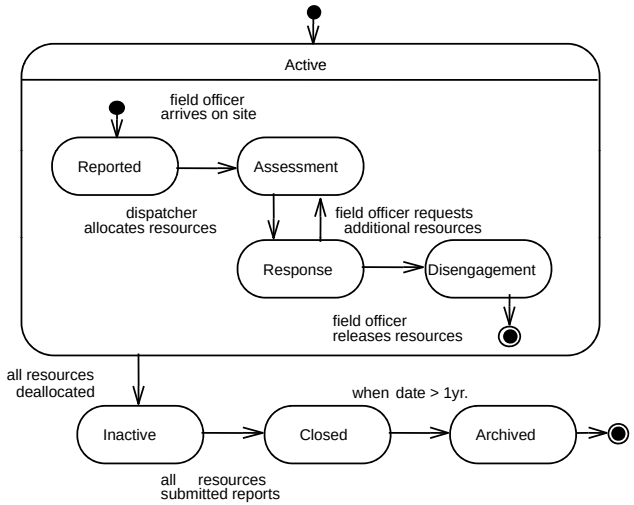
Modellare il comportamento dipendente dallo stato degli oggetti

- I diagrammi delle sequenze sono usati per distribuire il comportamento tra gli oggetti e per identificare le operazioni
- I diagrammi delle sequenze rappresentano il comportamento del sistema dalla prospettiva di un caso d'uso singolo
- I **diagrammi degli stati** rappresentano il comportamento dal punto di vista di un singolo oggetto
- Vedere il comportamento dalla prospettiva di ogni oggetto consente allo sviluppatore di costruire una descrizione più formale del comportamento dell'oggetto, e conseguentemente, di identificare casi d'uso mancanti

Modellare il comportamento dipendente dallo stato degli oggetti

- Focalizzandosi sugli stati, gli sviluppatori possono identificare nuovi comportamenti
- E' da osservare che non è necessario costruire diagrammi degli stati per ogni classe del sistema
  - E' importante considerare solo gli oggetti con un arco di vita esteso e con un comportamento che dipende dallo stato
  - Questo è quasi sempre la regola per gli oggetti control, meno spesso per gli oggetti entity e quasi mai il caso per gli oggetti boundary

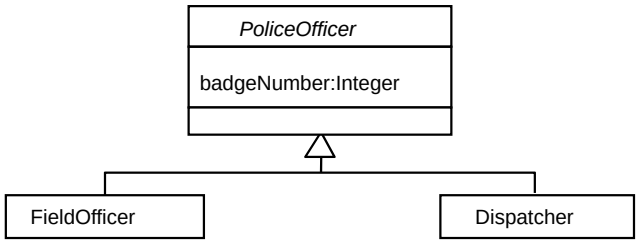
Diagramma degli stati per *Incident*



Modellare relazioni di ereditarietà tra oggetti

- La generalizzazione è usata per eliminare la ridondanza dal modello di analisi
- Se due o più classi condividono attributi o un comportamento, le similitudini sono consolidate in una superclasse
  - *Dispatcher* e *FieldOfficer* hanno entrambi un attributo *badgeNumber* che serve per identificarli nel contesto di una città
  - *FieldOfficer* e *Dispatcher* sono entrambi *PoliceOfficer* a cui sono assegnate funzioni diverse
  - Per modellare esplicitamente questa similitudine si introduce una classe astratta *PoliceOfficer* da cui le classi *FieldOfficer* e *Dispatcher* ereditano

Un esempio di relazione di ereditarietà



Rivedere il modello di analisi

- Il modello di analisi è costruito incrementalmente ed iterativamente
  - Raramente si ottiene un modello corretto o anche completo al primo passo
  - Numerose iterazioni con clienti ed utenti sono necessarie prima che il modello di analisi converga verso una specifica corretta usabile dagli sviluppatori per il progetto e l'implementazione
- Una volta che il modello di analisi diventa stabile (cioè, quando il numero di modifiche al modello sono minimali e il raggio delle modifiche è localizzato), viene prima rivisto dagli sviluppatori (revisione interna) e poi anche con i clienti
- L'obiettivo della revisione è rendere la specifica dei requisiti sia corretta, completa, consistente e non ambigua

Rivedere il modello di analisi

- Sviluppatori e clienti effettuano la revisione anche se i requisiti sono realistici e verificabili
- Gli sviluppatori dovrebbero essere preparati a scoprire errori a valle ed effettuare le modifiche alle specifiche
- Tuttavia, sarebbe importante individuare quanti più errori possibili nei requisiti a monte
- La revisione può essere facilitata da una lista di controllo o una lista di domande

Domande da porsi perché il modello sia *corretto*

- *Il glossario degli oggetti entità è comprensibile all'utente?*
- *Le classi astratte corrispondono ai concetti a livello utente?*
- *Le descrizioni sono tutte in accordo con le definizioni dell'utente?*
- *Tutti gli oggetti entity e boundary hanno predicati nominali significativi come nomi?*
- *I casi d'uso e gli oggetti control hanno tutti i predicati verbali significativi come nomi?*
- *I casi d'errore sono tutti descritti e gestiti?*

Domande da porsi perché il modello sia *completo*

- *Per ogni oggetto: è necessario per un caso d'uso? In quale caso d'uso è creato? Modificato? Distrutto? Può essere acceduto da un oggetto boundary?*
- *Per ogni attributo: Quando è impostato? Quale è il suo tipo? Dovrebbe essere un qualificatore?*
- *Per ogni associazione: quando è attraversata? Perché è stata scelta una specifica molteplicità?Le associazioni con molteplicità uno-a-molti e molti-a-molti possono essere qualificate?*
- *Per ogni oggetto control: Ha le necessarie associazioni per accedere gli oggetti che partecipano nel caso d'uso corrispondente?*

Domande da porsi perché il modello sia *consistente*

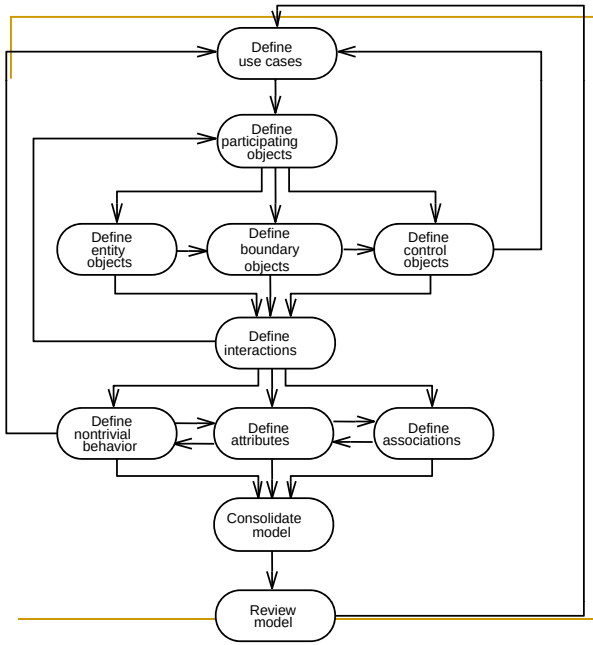
- *Ci sono classi o casi d'uso multipli con lo stesso nome?*
- *Entità (casi d'uso, classi, attributi) con lo stesso nome denotano concetti simili?*
- *Ci sono oggetti con attributi e associazioni simili che non sono nella stessa gerarchia di generalizzazione?*

Domande da porsi perché il modello sia *realistico*

- *Ci sono caratteristiche nuove nel sistema? Sono stati costruiti prototipi o eseguiti studi per assicurarne la fattibilità?*
- *I requisiti delle prestazioni e dell'affidabilità possono essere soddisfatti? Questi requisiti sono stati verificati da qualche prototipo eseguito su hardware selezionato?*

Sommario dell'analisi

- L'attività di scoperta dei requisiti è altamente iterativa e incrementale
  - Pezzi di funzionalità sono delineate e proposte a utenti e clienti
  - Il cliente aggiunge requisiti, critica le funzionalità esistenti e modifica i requisiti esistenti
  - Gli sviluppatori investigano i requisiti non funzionali attraverso i prototipi e studi della tecnologia e valutano ogni requisito proposto
- Inizialmente, la scoperta dei requisiti somiglia ad una'attività di brainstorming
  - Non appena la descrizione del sistema cresce ed i requisiti diventano più concreti gli sviluppatori devono estendere e modificare il modello di analisi in maniera più ordinata per gestire la complessità delle informazioni



Attività di Analisi

Gestione dell'analisi

- La sfida principale nella gestione dei requisiti di un progetto consiste nell'assicurare la consistenza nonostante l'utilizzo di numerose risorse
- Il documento di analisi finale dovrebbe descrivere un singolo sistema coerente comprensibile ad una singola persona
- I temi che tratteremo in tale ambito
  - Template del documento di analisi
  - Assegnamento dei ruoli comunicazione durante l'analisi
  - Gestione legata alla natura iterativa ed incrementale dei requisiti

Documento di analisi dei requisiti

- E' la descrizione del risultato delle attività di scoperta dei requisiti e di analisi
  - Descrive il sistema in modo completo in termini di requisiti funzionali e non funzionali e funge da base contrattuale tra clienti e sviluppatori
  - I destinatari del documento sono il cliente, gli utenti, la gestione del progetto, gli analisti del sistema (gli sviluppatori che partecipano nei requisiti) e i progettisti del sistema (gli sviluppatori che partecipano nel progetto del sistema)
  - La prima parte del documento (casi d'uso, requisiti non funzionali,…) è scritta durante la scoperta dei requisiti
  - La formalizzazione della specifica in termini di modelli degli oggetti è scritta durante l'analisi

Documento di analisi dei requisiti

- La prima sezione è l'*Introduzione*
  - Fornisce una breve panoramica della funzione del sistema e le ragioni del suo sviluppo, l'ambito, ed i riferimenti al contesto di sviluppo (descrizione del problema scritta dal cliente, riferimento a sistemi esistenti, studi di fattibilità)
- La seconda sezione è il *Sistema corrente*
  - Descrive lo stato della situazione attuale
    - Sostituzione di un sistema esistente
      - si descrivono le funzionalità ed i problemi del sistema corrente
    - Nuovo sistema
      - si descrive come i compiti supportati dal nuovo sistema sono gestiti correntemente

Documento di analisi dei requisiti

- La terza sezione è il *Sistema Proposto*
  - Documenta la scoperta dei requisiti ed il modello di analisi del nuovo sistema. E' diviso in quattro sottosezioni:
    - **Panoramica** (visone funzionale del sistema)
    - **Requisiti funzionali** (le funzionalità ad alto livello del sistema)
    - **Requisiti non funzionali**
    - **Modelli del sistema**: descrive, scenari, casi d'uso, modello ad oggetti, e modelli dinamici. Questa sezione contiene anche delle schermate che mostrano l'interfaccia utente del sistema e i cammini di navigazione. Le sottosezioni Modello ad oggetti e Modello dinamico sono scritte durante l'analisi.

Documentare l'analisi

- Nella sottosezione relativa ai modelli del sistema:
  - tutti gli oggetti identificati devono essere documentati in dettaglio, i loro attributi e, quando sono stati usati i diagrammi delle sequenze, le loro operazioni (descrizione testuale degli oggetti e relazioni tra di essi mediante diagrammi delle classi)
  - Si documenta il comportamento del modello ad oggetti in termini di diagrammi degli stati e diagrammi delle sequenze
- Una sezione relativa alla storia delle revisioni fornirà una cronistoria delle modifiche che include l'autore di ciascuna modifica, la data ed una breve descrizione

Documento di analisi dei requisiti

1. ***Introduzione***
   - Scopo del sistema
   - Ambito del sistema
   - Obiettivi e criteri di successo del progetto
   - Definizioni, Acronimi e abbreviazioni
   - Riferimenti
   - Panoramica
2. ***Sistema attuale***
3. ***Sistema proposto***
   - Panoramica
   - Requisiti funzionali
   - Requisiti non funzionali
     - Usabilità
     - Affidabilità
     - Prestazioni
     - Supportabilità
     - Implementazione
     - Interfaccia
     - Packaging
     - Note legali
   - Modelli del sistema
     - Scenari
     - Modello dei casi d'uso
     - Modello ad oggetti
     - Modello dinamico
     - Interfaccia utente – schermate mock-up e percorsi di navigazione
4. ***Glossario***

Assegnazione delle responsabilità

- L'analisi richiede la partecipazione di un ampio numero di persone
  - L'utente fornisce la conoscenza del dominio applicativo
  - Il cliente finanzia il progetto e coordina il lato utente dello sforzo
  - L'analista scopre e formalizza la conoscenza del dominio applicativo
  - Gli sviluppatori forniscono feedback sulla fattibilità ed i costi
  - Il project manager coordina lo sforzo dal lato sviluppo
- Per sistemi di grosse dimensioni possono essere coinvolti molti utenti, analisti e sviluppatori
  - E' importante assegnare ruoli ben definiti e ambiti di lavoro a ciascun individuo

Ruoli (I)

- Esistono tre tipi principali di ruoli: generazione di informazione, integrazione e revisione
  - **Utente finale**: è l'esperto del dominio dell'applicazione che genera informazioni sul sistema attuale, l'ambiente del sistema futuro ed i compiti che esso dovrebbe supportare
  - **Cliente**: è un ruolo di integrazione e definisce l'ambito del sistema sulla base dei requisiti utente. Poiché differenti utenti possono avere punti di vista diversi del sistema, il cliente funge da integratore delle informazioni del dominio applicativo e risolve le inconsistenze nelle attese degli utenti
  - **Analista**: è colui che modella il sistema attuale e genera le informazioni sul sistema futuro. Ogni analista ha la responsabilità di dettagliare uno o più casi d'uso. Per un insieme di casi d'uso, l'analista identificherà un certo numero di oggetti, le loro associazioni e i loro attributi

Ruoli (II)

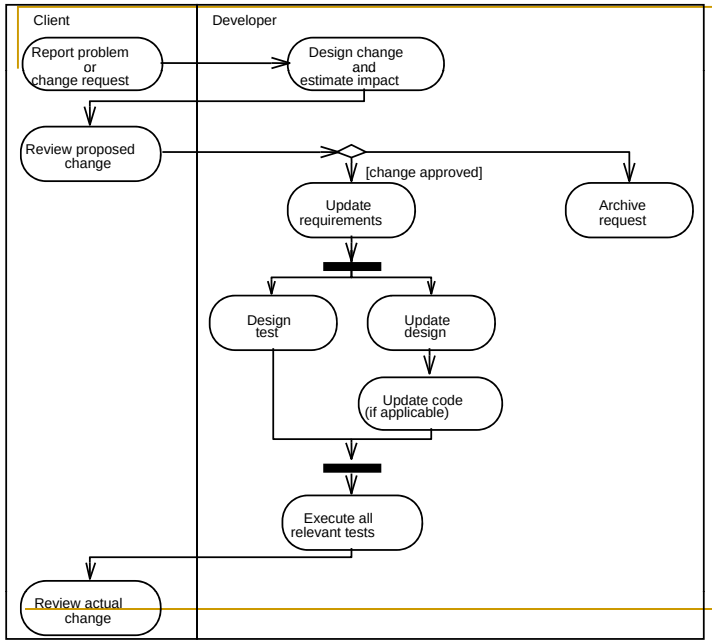
- **Architetto**: ruolo di integrazione, unifica i modelli dei casi d'uso e degli oggetti dal punto di vista del sistema. Analisti diversi possono avere differenti stili di modellazione, per cui la figura dell'architetto è necessaria per fornire la filosofia del sistema e identificare le omissioni nei requisiti
- **Editor del documento**: è responsabile dell'integrazione a basso livello del documento e del formato complessivo del documento e del suo indice
- **Manager della configurazione**: è responsabile di mantenere la storia delle revisioni del documento e della tracciabilità delle informazioni
- **Revisore**: valida il documento dei requisiti rispetto alla correttezza, completezza, consistenza e chiarezza. Utenti, clienti, sviluppatori o altre persone possono essere revisori durante la validazione dei requisiti. Le persone che non sono state ancora coinvolte nello sviluppo rappresentano revisori eccellenti

Iterazioni nel modello di analisi

- L'analisi si verifica in modo iterativo ed incrementale, spesso in parallelo con altre attività di sviluppo (progettazione ed implementazione)
- E' necessario gestire con attenzione iterazioni ed incrementi e tener traccia delle richieste di modifica una volta che i requisiti sono revisionati e approvati formalmente
- Le attività dei requisiti possono essere viste come un insieme di passi che convergono verso un modello stabile
  - Brainstorming
  - Solidificazione
  - Maturità

Sign-off del cliente

- Rappresenta l'accettazione da parte del cliente del modello di analisi
  - Cliente e sviluppatori convergono su un'idea singola e trovano l'accordo sulle funzioni e le caratteristiche che avrà il sistema
- Dopo il sign-off, i requisiti sono approvati e usati per rifinire la stima del costo del progetto
  - I requisiti continuano a cambiare dopo il sign-off , ma le modifiche sono sottoposte ad un processo di revisione più formale



Processo di revisione