L'analisi si traduce in un modello del sistema che mira a essere corretto, completo, coerente e inequivocabile. Gli sviluppatori formalizzano (un procedimento che identifica le ambiguità) le specifiche dei requisiti prodotti durante la fase Requirements Elicitation ed esaminano in modo più dettagliato condizioni e casi eccezionali. Convalidano, correggono e chiariscono le specifiche se vengono rilevati errori o ambiguità. Il cliente e l'utente sono generalmente coinvolti in questa attività quando le specifiche dei requisiti devono essere modificate e quando devono essere raccolti informazioni aggiuntive. In questa fase, gli sviluppatori costruiscono un modello che descrive il dominio del sistema. Esempio: il modello di analisi di un orologio descrive come l'orologio rappresenta il tempo:

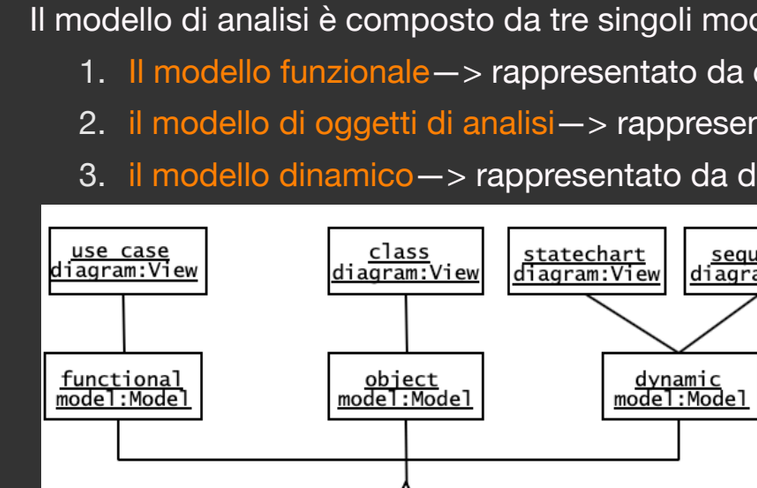
1. Conosce gli anni bisestili?
2. Sa del giorno del sistema?
3. Conosce le fasi lunari?

Procedimenti a manipolare il modello di dominio dell'applicazione tramite delle varie iterazioni che si hanno tra gli attori e il sistema.

1. in che modo il proprietario dell'orologio reimposta l'ora?
2. in che modo il proprietario dell'orologio reimposta il giorno della settimana?

Tutte queste informazioni verrano utilizzati per progettare l'architettura del sistema nella fase "progettazione di alto livello".
L'elaborazione e l'analisi dei requisiti sono attività iterative e incrementali che si verificano contemporaneamente.
Questa fase è diversa dal Requirements Elicitation in quanto gli sviluppatori si concentrano sulla strutturazione e la formalizzazione dei requisiti richiesti dai utenti portando nuove intuizioni e scoperte di errori nei requisiti.
Il modello di analisi è composto da tre singoli modelli:

1. Il modello funzionale—> rappresentato da casi d'uso e scenari;
2. il modello di oggetti di analisi—> rappresentato da diagrammi di classe e diagrammi di oggetti;
3. il modello dinamico—> rappresentato da diagrammi di stato e diagrammi di sequenza;



Concetti di Analisi
Ci sono delle differenze per quando si tratta il modello di oggetti di analisi e il modello dinamico:
Il modello di oggetti di analisi rappresenta il sistema in fase di sviluppo dal punto di vista dell'utente. Si concentra sui singoli concetti manipolati dal sistema, dalle loro proprietà e dalle loro relazioni.
Il modello a oggetti di analisi rappresentato con i diagrammi di classe UML include classi , attributi e operazioni. Si tratta di un dizionario visivo dei concetti principali visibili all'utente.
Il modello dinamico rappresenta il comportamento del sistema quindi rappresentato con diagrammi di sequenza e diagrammi di stato. I diagrammi di sequenza elencano le interazioni tra un insieme di oggetti durante un singolo caso d'uso. Mentre i diagrammi di stato rappresentano il comportamento di un singolo oggetto. Il modello dinamico serve ad assegnare le responsabilità alle singole classi, a identificare nuove classi,ecc..

In questa fase ci dobbiamo soltanto preoccuparci nell'inserire degli oggetti che possano essere rappresentati da un livello più astratti in modo da poter essere comprensibile all'utente. Dopodiché nelle prossime fase ci saranno sicuramente delle aggiunte degli nuovi oggetti, oggetti che non saranno visibili all'utente.
Entità, confine e oggetti di controllo
Il modello di oggetti di analisi è costituito da oggetti di entità, di confine e di controllo.
Gli oggetti entità (Entity) rappresentano le informazioni persistenti tracciate dal sistema.
Gli oggetti di confine (Boundary) rappresentano le interazioni tra gli attori e il sistema.
Gli oggetti di controllo (Control) hanno il compito di realizzare i casi d'uso ovvero prendendo possesso del sistema, cerca di far realizzare la richiesta relativa.
Nell'esempio di 2BWatch:

- Anno, Mese e Giorno sono oggetti entità
- Button e LCDDisplay sono oggetti di confine
- ChangeDateControl è un oggetto di controllo che rappresenta l'attività di modifica della data premendo combinazioni di pulsanti.

Questo approccio con tre tipi di oggetti produce oggetti più piccoli e più specializzati. Inoltre porta a modelli che sono più resistenti al cambiamento: l'interfaccia al sistema (tramite oggetti di confine) ha maggior probabilità di cambiare rispetto alla sua funzionalità di base (tramite oggetti di entità e di controllo). Grazie alla separazione di questi oggetti, siamo in grado di mantenere intatta la maggior parte di un modello quando l'interfaccia utente cambia ma gli oggetti di entità no.
Per distinguere tra diversi tipi di oggetti, UML fornisce il meccanismo stereotipato per consentire allo sviluppatore di collegare tali meta-informazioni agli elementi di modulazione. Ad esempio si allega lo stereotipo <<control>> all'oggetto ChangeDateControl. Oltre agli stereotipi, per agevolare il problema della mancanza di UML, è possibile utilizzare delle convezioni per delimitare questi tre tipi di oggetti. Gli oggetti control possono aver il suffisso "Control" mentre gli oggetti boundary dovrebbero avere nomi che ricordano aspetti dell'interfaccia come suffisso Form, Button, ecc..
Attività di Analisi: dagli use case agli oggetti
Le attività che consentono di trasformare gli use case e gli scenari della raccolta dei requisiti in un modello di analisi sono:

- Identificare gli Oggetti Entity
- Identificare gli Oggetti Boundary
- Identificare gli Oggetti Control
- Mappare gli Use Case in Oggetti con Sequence Diagram
- Identificare le Associazioni con diagrammi di classe
- Identificare gli Aggregati
- Identificare gli Attributi
- Modellare il Comportamento dipendente dallo stato degli Oggetti individuali con diagrammi di stato
- Modellare le Relazioni di Ereditarietà
- Rivedere il Modello di Analisi

Esempio UseCase che verrà nominato nei prossimi esempi

- Nome dello use case: ReportEmergency
- Attori Partecipanti:
  - FieldOfficer(Bob e Alice nello scenario)
  - Dispatcher(John nello scenario)
- Eccezioni:
  - Il FieldOfficer viene subito notificato se la connessione tra il suo terminale e quello della centrale cade.
  - Il Dispatcher viene subito notificato se la connessione tra ciascun FieldOfficer che ha avuto accesso e la centrale cade.
- Vincoli:
  - Il report del FieldOfficer deve essere comunicato entro 30 secondi.
  - La risposta selezionata arriva non più tardi di 30 secondi dopo che è stato mandato dal Dispatcher.

Nome Use Case	ReportEmergency
Entry Condition	1. Il field attiva la funzione "ReportEmergency" dal suo terminale
Flow of event	2. FRIEND risponde presentando un form al FieldOfficer. Il form include un menu con i tipi di emergenza (emergenza generale, incendio), locazione, descrizione dell'incidente e richiesta di risorse. 3. Il FieldOfficer completa il form specificando il tipo di emergenza e i campi di descrizione. Il FieldOfficer descrive anche possibili risposte alla situazione di emergenza e richiede risorse specifiche. Quando il forma è completato, il FieldOfficer sottomette il forma premendo il bottone "SendReport" a quel punto il Dispatcher è notificato. 4. Il Dispatcher rivede l'informazione sottomessa da FieldOfficer e crea un Incident nel database invocando lo use case "OpenIncident". Tutta l'informazione contenuta nella form del FieldOfficer è inclusa automaticamente nell'Incident. Il Dispatcher seleziona una risposta allocando risorse all'Incidente (con lo use case AllocateResource) e informa della ricezione del report dell'incidente inviando un messaggio al FieldOfficer.
Exit Condition	5. Il FieldOfficer riceve l'avviso di ricezione e le risposte selezionate.

Identificare gli oggetti Entity
Gli oggetti partecipanti vengono trovati esaminando ogni caso d'uso e identificando gli oggetti candidati. Tramite l'analisi del linguaggio naturale (formulato da Abbott) ci consente di identificare oggetti, attributi, associazioni dalla specifica dei requisiti. Esso mappa parti del discorso ai componenti del modello. Tuttavia, come ogni modello, ha i suoi pro e contro:

- Contro:
  - La qualità del modello a oggetti dipende dallo stile di scrittura dell'analista quindi essendo che il linguaggio naturale è uno strumento impreciso, si rischia che l'oggetto creato sia impreciso.
  - Ci possono molti più sostantivi delle classi rilevanti. L'ordinamento tra tutti nomi per una specifica di requisiti è un'attività che richiede tempo.
- Pro:
  - Ci si focalizza sui termini dell'utente.

Euristiche per identificare gli oggetti Entity
L'euristica di Abbot funziona bene per generare una lista iniziale di candidati da brevi descrizioni, come il flusso di eventi di uno scenario o caso d'uso.

Parti del parlato	Componente del modello	esempio
Nome proprio	Istanza	Alice
Nome comune	Class	Funzionario(FieldOfficer)
Verbo fare/azione	Operazione	Crea, Sottoponi, Seleziona
Verbo essere	gerarchia	È un tipo di, è uno di
Verbo avere	aggregazione	Ha, consiste di, include
aggettivo	attributo	Descrizione dell'incidente

E' possibile combinare l'euristica di Abbott con la seguente euristica:

1. Termini che gli sviluppatori e gli utenti hanno bisogno di chiarire per comprendere gli use case.
2. Sostativi ricorrenti negli use Case(Esempio: Incidente)
3. Entità del mondo reale che il sistema deve considerare.
4. Attività del mondo reale che il sistema deve considerare.
5. Sorgenti o destinazioni di dati.

Per ogni oggetto identificato, si assegna un nome(univoco) e una breve descrizione. Per gli oggetti Entity è preferibile utilizzare gli stessi nomi utilizzati dagli utenti e dagli specialisti del dominio applicativo. Descrivere gli oggetti consente agli sviluppatori di chiarire i concetti onde ad evitare delle incomprensioni. Si dovrebbero documentare attributi e responsabilità se non sono ovvi; altrimenti è sufficiente un nome provvisorio e una breve descrizione per ciascun oggetto. Essendo che in tale fase ci sono molte iterazioni, gli sviluppatori non devono passarci molto tempo a dettagliare oggetti o attributi poiché verrano raffinati nelle prossime iterazioni.
Dall'esempio dello Use Case ReportEmergency, possiamo identificare i seguenti oggetti:

Oggetti Entità per il caso d'uso "The ReportEmergency"

Dispatcher	Ufficio di polizia che gestisce Incidenti. Un dispatcher apre, documenta e chiude Incident in risposta a Report di emergenza e altra comunicazione con FieldOfficers. I Dispatcher sono identificati dal numero del badge.
EmergencyReport	Report iniziale su un Incident da un FieldOfficer a un Dispatcher. Un EmergencyReport solitamente determina la creazione di un Incident da parte di un Dispatcher. Un EmergencyReport è composto da un livello di emergenza, un tipo(fuoco, stradale), un luogo e una descrizione.
FieldOfficer	Un funzionario di un ufficio di polizia o dei vigli del fuoco in servizio. Un FieldOfficer può essere allocato al più ad un Incident alla volta. I FieldOfficer sono identificati da badge.
Incident	Situazione che richiede l'attenzione di un FieldOfficer. Un "Incident" può essere riportato nel sistema da un FieldOfficer o da qualcuno anche esterno al sistema. "Un'Incident" è composto da una descrizione, una risposta, uno status(aperto, chiuso, documentato), una locazione e un numero di FieldOfficer.

L'oggetto EmergencyReport non è stato nominato nello use case ma grazie al passaggio 4 del caso d'uso ReportEmergency notiamo l'esistenza delle informazioni inviate dal FieldOfficer. Tali informazioni, grazie alla revisione con il cliente, vengono chiamate "rapporto di emergenza" così mettiamo l'oggetto entità EmergencyReport.

Identificazione degli oggetti Boundary
Gli oggetti Boundary rappresentano l'interfaccia di sistema con gli attori. In ogni caso d'uso, ogni attore interagisce con almeno un oggetto Boundary. Quest'ultimo raccoglie le informazioni dell'attore e le traduce in una forma che può essere utilizzata da entrambi gli oggetti entità e controllo. In tutto ciò, gli oggetti Boundary rappresentano soltanto ad un livello approssimativo quindi non ha senso parlare degli oggetti come "voce di menu" o "barra di scorrimento" essendo dettagliati. Infatti lo sviluppo dell'interfaccia è solitamente di tipo prototipale e iterativo; tramite i testi di usabilità fanno evolvere continuamente l'interfaccia, di conseguenza non sarebbe né pratico né utile modificare il modello di analisi ad ogni modifica dell'interfaccia.

Euristiche per identificare gli oggetti Boundary

- Identificare i controlli dell'interfaccia utente necessari all'utente per avviare il caso d'uso(ad esempio, ReportEmergencyButton).
- Identificare i moduli necessari agli utenti per inserire i dati nel sistema (ad esempio, EmergencyReportForm).
- Identificare avvisi e messaggi che il sistema utilizza per rispondere all'utente (ad esempio, AcknowledgementNotice).
- Quando più attori sono coinvolti in un caso d'uso, identificare i terminali dell'attore (DispatcherStation) per fare riferimento all'interfaccia utente in esame.
- Non modellare gli aspetti visivi dell'interfaccia utente con oggetti Boundary (meglio mock-up)
- Utilizzare sempre i termini dell'utente finale per descrivere le interfacce; non utilizzare i termini della soluzione o domini di implementazione.

AcknowledgementNotice	Avviso usato per mostrare acknowledgement del Dispatcher al FieldOfficer
DispatcherStation	Computer usato dal Dispatcher
ReportEmergencyButton	Bottone usato dal FieldOfficer per iniziare lo use case ReportEmergency
EmergencyReportForm	Form usata per l'input del *ReportEmergency*. Questa form è presentata al *FieldOfficer* sul *FieldOfficerStation* quando la funzione "Report Emergency" è selezionata. *EmergencyReportForm* contiene campi per specificare tutti gli attributi di un report di emergenza e un bottone (o altro controllo) per sottomettere la form completata.
FieldOfficerStation	Computer usato dal FieldOfficer
IncidentForm	Form usata per la creazione di *Incident*. Questa form è presentata al *Dispatcher* sul *DispatcherStation* quando è ricevuto l'*EmergencyReport*. Il Dispatcher usa anche questa form per allocare le risorse e notificare il report del *FieldOfficer*

L'oggetto "IncidentForm", poiché non è stato menzionato nello use case, è stato inserito in quanto il Dispatcher necessita di un interfaccia per visualizzare il rapporto di emergenza inviato dal FieldOfficer e inviare un riconoscimento.
Identificare gli oggetti Control
Gli oggetti Control sono responsabili del coordinamento degli oggetti di confine e entità. Si preoccupano di raccogliere delle informazioni dagli oggetti di confine e inviare agli oggetti Entity. Spesso esiste una stretta relazione tra un caso d'uso e un oggetto Control ; di solito un oggetto Control viene creato all'inizio di un caso d'uso e cessa di esistere alla fine.
Il flusso di controllo dello use case ReportEmergency viene modellato con due oggetti Control, uno per ogni attore:

- ReportEmergencyControl per FieldOfficer
- ManageEmergencyControl per Dispatcher

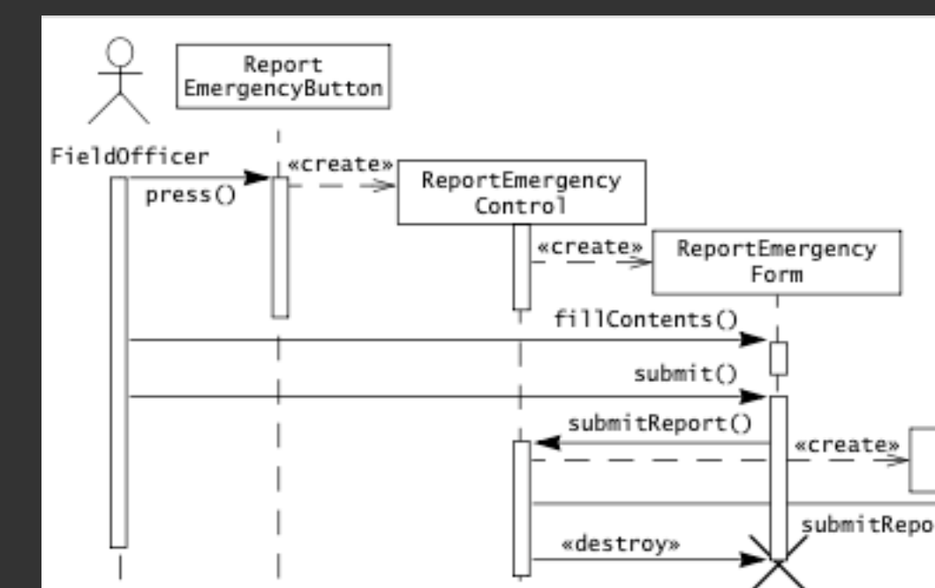
Tale decisione deriva dalla consapevolezza che FieldOfficerStation e DispatcherStation sono due sottosistemi che comunicano su un collegamento asicrono. Renderla visibile tale modellazione ci consente di concentrarci su comportamenti di eccezione come la perdita di comunicazione tra le due stazioni.
Euristica per l'identificazione degli oggetti Control

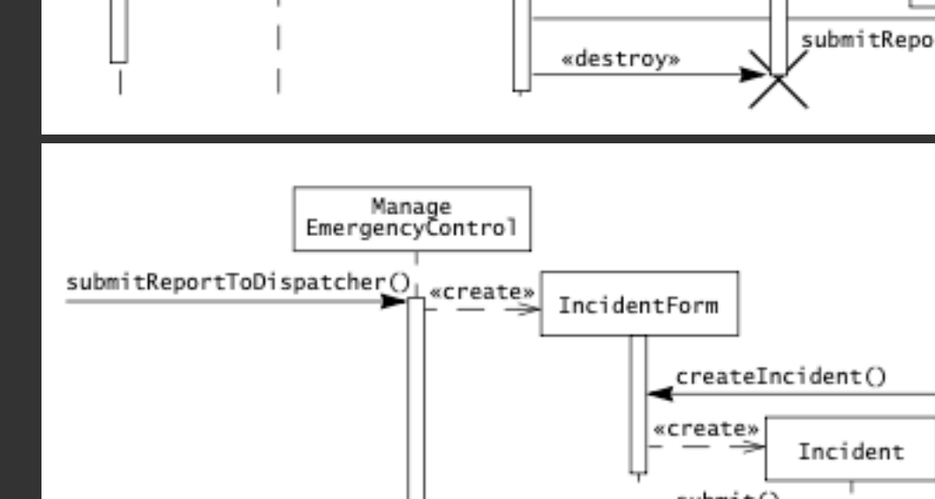
- Identificare un oggetto Control per caso d'uso
- Identificare un oggetto Control per attore nel caso d'uso.
- La durata di un oggetto Control dovrebbe coprire l'estensione del caso d'uso o l'estensione di una sessione utente. Se è difficile identificare l'inizio e la fine dell'attivazione di un oggetto Control, il corrispondente use case probabilmente non ha delle entry e exit condition ben definite.

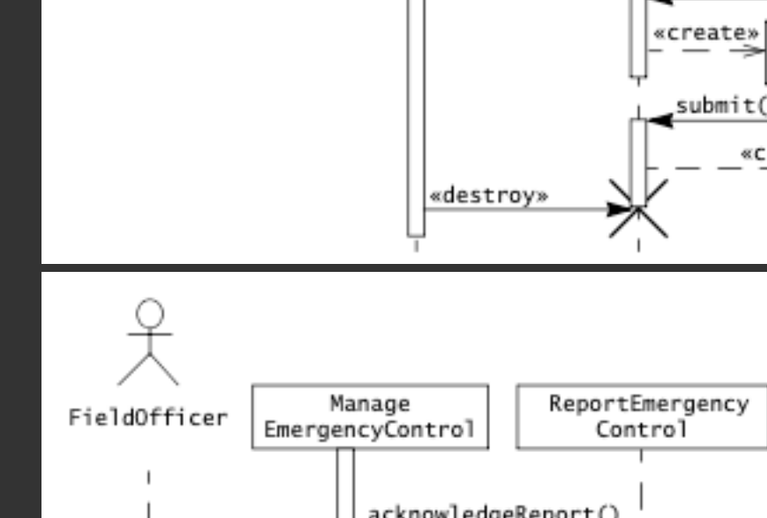
Nel modellare il caso d'uso ReportEmergency, abbiamo modellato la stessa funzionalità utilizzando oggetti Entity, Boundary e Control. Passando dalla prospettiva del flusso di eventi a una prospettiva strutturale, abbiamo aumentato il livello di dettaglio della descrizione e selezionato termini standard per fare riferimento alle entità principali del dominio dell'applicazione e del sistema.

ReportEmergencyControl	Gestisce la funzione *ReportEmergency* sulla *FieldOfficerStation*. Questo oggetto è creato quando il *FieldOfficer* seleziona il bottone "Report Emergency". Crea un *EmergencyReportForm* e lo presenta a *FieldOfficer*. Dopo la sottomissione della form, questo oggetto colleziona l'informazione dalla form, crea un *EmergencyReport*, e lo inoltra al *Dispacher*. L'oggetto Control quindi aspetta una notifica dal *DispatcherStation*. Quando riceve la notifica, l'oggetto *ReportEmergencyControl* crea un *AcknowledgmentNotice* e lo mostra al *FieldOfficer*
ManageEmergencyControl	Gestisce la funzione *ReportEmergency* sulla *DisptacherStation*. Questo oggetto è creato quando viene ricevuto un *EmergencyReport*. Quindi crea un *IncidentForm* e lo presenta al *Dispatcher*. Quando il *Disptacher* ha creato un *Incident*, allocato *Resources*, e sottomesso una notifica, *ManageEmergencyControl* inoltra la notifica a *FieldOfficerStation*

Mappare Use Case in oggetti con Sequence Diagram
Un Sequence Diagram definisce il comportamento di uno use case tramite degli oggetti partecipanti che vi partecipano in tale use case. Infatti vengono assegnati responsabilità a ogni oggetto in termini di un insieme di operazioni. Quindi illustra la sequenza di interazioni tra gli oggetti necessari per realizzare uno use case senza preoccuparci delle implementazioni.
Ma non rappresenta un mezzo comunicativo efficiente per il cliente in quanto vi sono contenuti più informazioni del solito. Di solito è adatti per i client esperti in tale modello.
La Sequence Diagram fornisce una prospettiva diversa che consente di individuare oggetti mancanti e aree non chiare nelle specifiche.







Teoria Sequence Diagram
Le colonne di un diagramma di sequenza rappresentano gli oggetti che partecipano al caso d'uso. La colonna più a sinistra è l'attore che avvia il caso d'uso. Le frecce orizzontali tra le colonne rappresentano messaggi o stimoli che vengono inviati da un oggetto all'altro. La ricezione di un messaggio attiva l'attivazione di un'operazione. L'attivazione è rappresentato da un rettangolo verticale da cui possono provenire altri messaggi. La lunghezza del rettangolo rappresenta il tempo in cui l'operazione è attiva. Un'operazione può essere considerata come un servizio che l'oggetto fornisce ad altri oggetti. Gli oggetti già esistenti prima dei primi stimoli nel diagramma di sequenza sono rappresentati nella parte superiore del diagramma. Gli oggetti creati durante l'interazione sono rappresentati con il messaggio <<crea>> che punta all'oggetto. Le istanze che vengono distrutte durante l'interazione hanno una croce che indica quando l'oggetto cessa di esistere. Tra il rettangolo orizzontale che rappresenta l'oggetto e la croce, una linea tratteggiata indica l'intervallo di tempo in cui l'oggetto può ricevere messaggi, oltre alla croce non può ricevere.
In generale, la prima colonna rappresenta l'attore che da avvio all'use case, la seconda colonna di un sequence diagramma rappresenta l'oggetto Boundary con cui l'attore interagisce per avviare il caso d'uso (ad esempio, ReportEmergencyButton). La terza colonna è un oggetto Control che gestisce il resto del caso d'uso (ad esempio, ReportEmergencyControl).
Nella seconda figura possiamo notare che è stato creato un oggetto Acknowledgement, diverso da AcknowledgementNotice. Infatti nello use Case ReportEmergency vi era una menzione dell'esistenza di un messaggio di notifica ma non descriveva l'informazione ad esso associata. Così gli sviluppatori sono andati a chiedere tali dubbi al committente ottenendo così il chiarimento,Viene creato così l'oggetto "Acknowledgement". Viene aggiunto quindi al modello di analisi e il caso d'uso ReportEmergency viene raffinato.

Acknowledgement	Risposta di un Dispatcher a un EmergencyReport di un FieldOfficer. Inviano un Acknowledgement, il Dispatcher comunica al FieldOfficer che ha ricevuto l'EmergencyReport, crea un Incident, e assegna risorse. L'Acknowledgement contiene le risorse assegnate e il tempo stimato del loro arrivo.

Use Case ReportEmergency Raffinato

Nome Use Case	ReportEmergency
Entry Condition	1. Il field attiva la funzione "ReportEmergency" dal suo terminale
Flow of event	2. FRIEND risponde presentando un form al FieldOfficer. Il form include un menu con i tipi di emergenza (emergenza generale, incendio), locazione, descrizione dell'incidente e richiesta di risorse. 3. Il FieldOfficer completa il form specificando il tipo di emergenza e i campi di descrizione. Il FieldOfficer descrive anche possibili risposte alla situazione di emergenza e richiede risorse specifiche. Quando il forma è completato, il FieldOfficer sottomette il forma premendo il bottone "SendReport" a quel punto il Dispatcher è notificato. 4. Il Dispatcher rivede l'informazione sottomessa da FieldOfficer e crea un Incident nel database invocando lo use case "OpenIncident". Tutta l'informazione contenuta nella form del FieldOfficer è inclusa automaticamente nell'Incident. Il Dispatcher seleziona una risposta allocando risorse all'Incidente (con lo use case AllocateResource) e informa della ricezione del report dell'incidente inviando un messaggio al FieldOfficer. **L'acknowlegment indica al FieldOfficer che l'EmergencyReport è stato ricevuto, creato un Incident, e allocate risorse a Incident. L'L'acknowlegment include le risorse e il tempo stimato del loro arrivo.**
Exit Condition	5. Il FieldOfficer riceve l'avviso di ricezione e le risposte selezionate.

Domandi che aiutano alla costruzione di un sequence diagram

1. Quali casi d'uso creano questo oggetto? quali attori possono accedere a queste informazioni?
2. Quali casi d'uso modificano e distruggono questo oggetto? Quali attori avviano questi casi uso?
3. Questo oggetto è necessario?

Euristiche per disegnare un Sequence Diagram

1. La 1° colonna dovrebbe corrispondete all'attore che inizia lo use case.
2. La 2° colonna dovrebbe essere un oggetto Boundary (che l'attore usa per iniziare lo use case)
3. La 3° colonna dovrebbe essere l'oggetto Control che gestisce il resto dello use case
4. Gli oggetti Control sono creati dagli oggetti Boundary che iniziano gli use case
5. Oggetti Control e Boundary accedono a oggetti Entity
6. Gli oggetti Boundary sono creati da oggetti Control
7. Gli oggetti Entity non accedono mai agli oggetti Control e Boundary: ciò rende più facile condividere oggetti Entity tra più use case

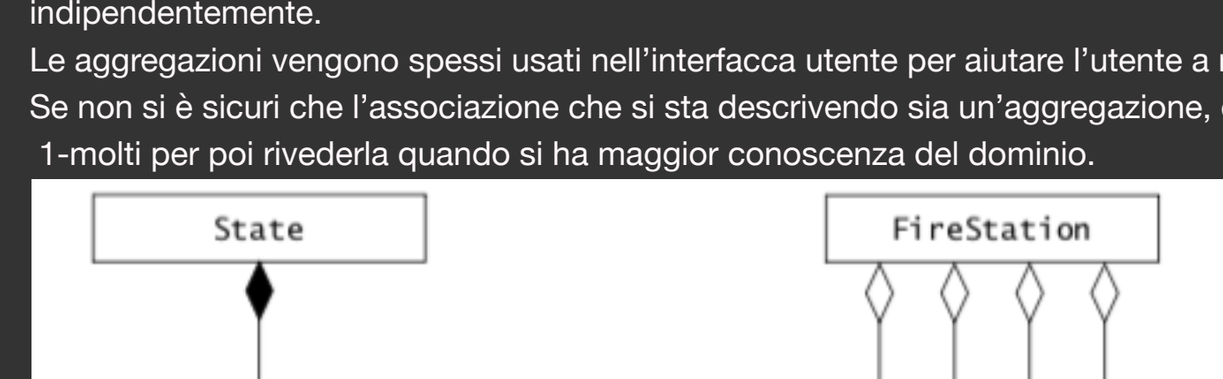
---

Identificare Associazioni con diagrammi di classe
Mentre i diagrammi di sequenza consentono agli sviluppatori di rappresentare le interazioni tra gli oggetti nel tempo, i diagrammi di classe consentono agli sviluppatori di descrivere le interdipendenze degli oggetti.
Un'associazione mostra una relazione tra due o più classi.
L'identificazione delle associazioni presente due vantaggi:

1. Chiarisce il modello di analisi rendendo esplicite le relazioni tra gli oggetti
2. Consente allo sviluppatore di scoprire casi limite associati ai collegamenti. Per casi limite si intendono le eccezioni che devono essere chiarite nel modello.

Le associazioni hanno delle proprietà:

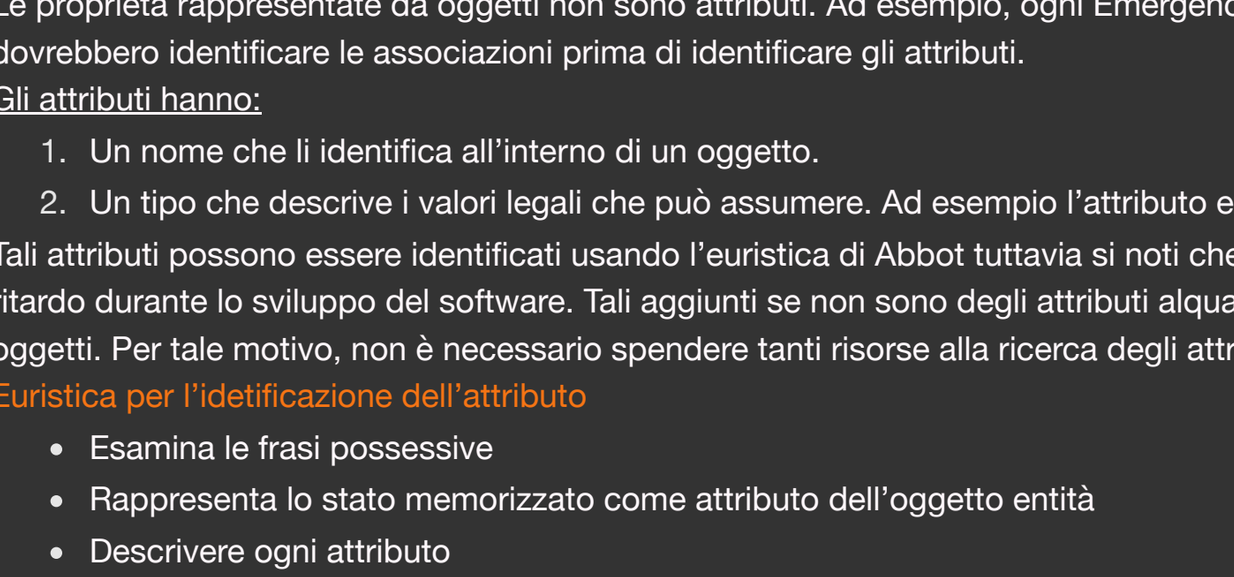
1. Un nome per descrivere un associazione tra due classi. Il nome di un'associazione è opzionale.
2. Un ruolo a ciascuna estremità, identificando la funzione di ogni classe rispetto all'associazione. (l'autore è il ruolo svolto da FieldOfficer nell'associazione Writes)
3. Una molteplicità a ciascun estremità, indicando il possibile il numero di istanze. (* indica a FieldOfficer di scrivere zero o più EmergencyReport mentre 1 indicia che ogni EmergencyReport ha esattamente un FieldOfficer come autore)



Euristiche per indetificare delle associazioni

- Esami le frasi verbali
- Nomina associazioni e ruoli con precisione
- Utilizzare i qualificatori il più spesso possibile per identificare spazi dei nomi e attributi chiave
- Eliminare qualsiasi associazione che può essere derivata da altre associazioni.
- Troppe associazioni rendono il modello illeggibile

Identificare aggregazioni
Le aggregazioni sono tipi speciali di associazioni che indicano una relazione parte intera. Esistono due tipi aggregazioni, composizione e condivisione.
Il diamante pieno indica la composizione. Un'aggregazione di composizione indica l'esistenza di un insieme che dipendeno da un'oggetto, ovvero uno stato è composto da tanti comuni quindi se muore l'oggetto "stato", allo stesso tempo muoiono gli oggetti "comune" mentre il diamante vuoto indica una relazione di aggregazione condivisa indicando che gli oggetti possono esistere indipendentemente.
Le aggregazioni vengono spessi usati nell'interfaccia utente per aiutare l'utente a navigare attraverso molte istanze.
Se non si è sicuri che l'associazione che si sta descrivendo sia un'aggregazione, è meglio modellarla come associazione 1-molti per poi rivederla quando si ha maggior conoscenza del dominio.

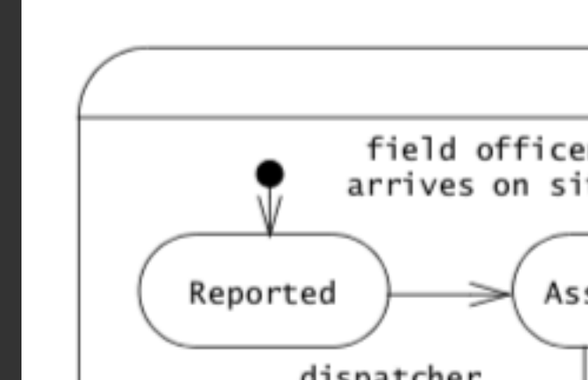


Identificare attributi
Gli attributi sono proprietà dei singoli oggetti. Quando si identificano le proprietà degli oggetti, devono essere considerati solo gli attributi rilevanti per il sistema. Ad esempio, FieldOfficer ha un numero di previdenza sociale che non è rilevante per il sistema informativo di emergenza. Invece FieldOfficers sono identificati dal numero di badge, che è rappresentato dalla proprietà badgeNumber.
Le proprietà rappresentate da oggetti non sono attributi. Ad esempio, ogni EmergencyReport ha un autore rappresentato da un'associazione alla classe FieldOfficer. Di conseguenza gli sviluppatori dovrebbero identificare le associazioni prima di identificare gli attributi.
Gli attributi hanno:

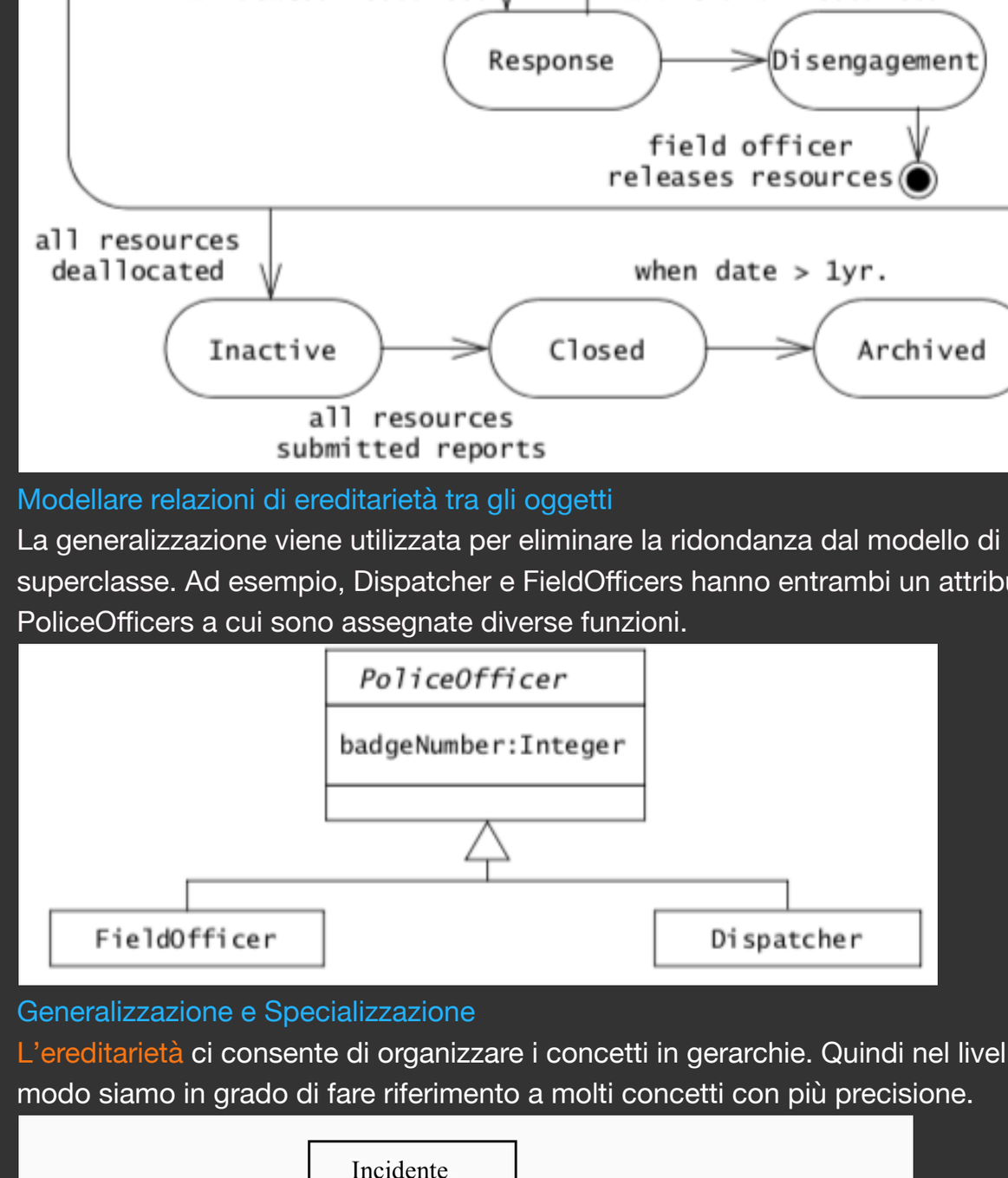
1. Un nome che li identifica all'interno di un oggetto.
2. Un tipo che descrive i valori legali che può assumere. Ad esempio l'attributo emergencyType è un enumerazione che può assumete uno di tre valori: incendio, traffico, altro.

Tali attributi possono essere identificati usando l'euristica di Abbot tuttavia si tratta di oggetti che rappresentano la parte meno stabile del modello a oggetti. Spesso, gli attributi vengono scoperti o aggiunti in ritardo durante lo sviluppo del software. Tali aggiunti se non sono degli attributi alquanto importanti associati a funzionalità aggiuntive, non comportano dei grandi cambiamenti nella struttura ad oggetti. Per tale motivo, non è necessario spendere tanti risorse alla ricerca degli attributi dato che possono essere aggiunti in seguito.
Euristica per l'idetificazione dell'attributo

- Esamina le frasi possessive
- Rappresenta lo stato memorizzato come attributo dell'oggetto entità
- Descrivere ogni attributo
- Non rappresentare un attributo come oggetto; utilizzare invece un'associazione
- Non perdere tempo a descrivere dettagli precisi prima che la struttura dell'oggetto sia stabile.

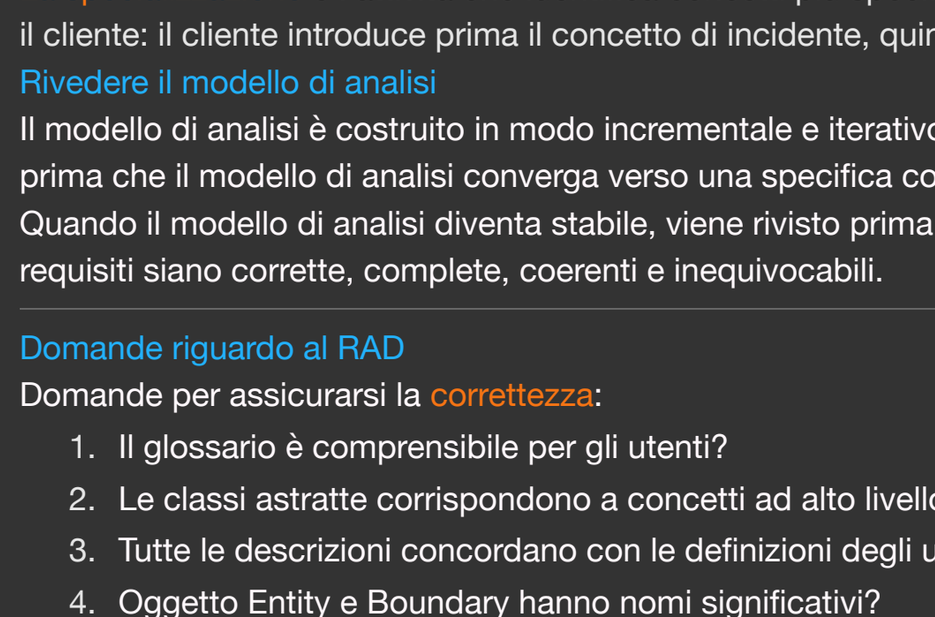


Modellare il comportamento dipendente dallo stato di singoli oggetti con diagrammi di stato
Mentre i diagrammi di sequenza vengono utilizzati per distribuire il comportamento tra gli oggetti e per identificare le operazioni, i diagrammi delle macchine a stato rappresentano il comportamento dalla prospettiva di un singolo oggetto. Tramite la prospettiva di ciascun oggetto, ci consente di creare una descrizione più formale del comportamento dell'oggetto e identificare i casi d'uso mancanti. Concentrandosi sui singoli stati, gli sviluppatori possono identificare nuovi comportamenti. Ad esempio, esaminando ogni transizione nel diagramma della macchina a stati che viene attivata da un'azione dell'utente, lo sviluppatore dovrebbe essere in grado di identificare una fase del flusso in un caso d'uso che descrive tale azione.



Modellare relazioni di ereditarietà tra gli oggetti
La generalizzazione viene utilizzata per eliminare la ridondanza dal modello di analisi. Se uno o più classi condividono attributi o comportamenti, le somiglianze vengono consolidate in una superclasse. Ad esempio, Dispatcher e FieldOfficers hanno entrambi un attributo badgeNumber che serve a identificarli all'interno di una città. FieldOfficers e Disptacher sono entrambi PoliceOfficers a cui sono assegnate diverse funzioni.

Generalizzazione e Specializzazione
L'ereditarietà ci consente di organizzare i concetti in gerarchie. Quindi nel livello più alto, rappresenta un concetto generale mentre nei livelli bassi rappresentano concetti più specifici. In questo modo siamo in grado di fare riferimento a molti concetti con più precisione.



Quando intendiamo il termine incidente, intendiamo tutte le istanze di tutti i tipi di incidenti. Quando utilizziamo il termine Emergenza, ci riferiamo solo a un Incidente che richiede una risposta immediata.
La generalizzazione è l'attività di moderazione che identifica concetti astratti da quelli di livello inferiore. Ad esempio, supponiamo che stiamo retroingegnerizzando un sistema di gestione delle emergenze e scopriamo schermate per la gestione di incidenti stradali e incendi. Notando le caratteristiche comuni tra questi tre concetti, creiamo un concetto astratto chiamato "Emergenza" per descrivere le caratteristiche comuni di IncAutomobilistico e PalazzoInFiamme.
La specializzazione è l'attività che identifica concetti più specifici da uno di alto livello. Esempio: Stiamo costruendo un sistema di gestione delle emergenze e stiamo discutendo le funzionalità con il cliente: il cliente introduce prima il concetto di incidente, quindi descrive tre tipi di incidenti: disastri, emergenze, incidenti a bassa priorità.
Rivedere il modello di analisi
Il modello di analisi è costruito in modo incrementale e iterativo. E' raramente corretto o addirittura completo al primo passaggio quindi sono necessarie diverse iterazioni con il client e l'utente prima che il modello di analisi converga verso una specifica corretta utilizzabile dagli sviluppatori per la progettazione e l'implementazione.
Quando il modello di analisi diventa stabile, viene rivisto prima dagli sviluppatori, poi congiuntamente dagli sviluppatori e dal cliente. L'obiettivo della revisione è assicurarsi che le specifiche dei requisiti siano corrette, complete, coerente e inequivocabili.

Domande riguardo al RAD
Domande per assicurarsi la correttezza:

1. Il glossario è comprensibile per gli utenti?
2. Le classi astratte corrispondono a concetti ad alto livello?
3. Tutte le descrizioni concordano con le definizioni degli utenti?
4. Oggetto Entity e Boundary hanno nomi significativi?
5. Oggetti Control e use case sono nominati con verbi significativi?
6. Tutti gli errori sono descritti e trattati?

Domande per assicurarsi la completezza:

1. Per ogni oggetto: E' necessario per uno use case? In quale use case è creato? Modificato? Distrutto?
2. Per ogni attributo: Quando è dettato? Quale è il tipo?
3. Per ogni associazione: Quando è attraversata? Perchè ha una data molteplicità?
4. Per ogni oggetto control: ha le associazioni necessarie per accendere agli oggetti che partecipano nel corrispondete use case?

Domande per assicurarsi la consistenza:

1. Ci sono classi o use case con lo stesso nome?
2. Ci sono entità con nomi simili e che denotano concetti simili?
3. Ci sono oggetti con attributi e associazioni simili che non sono nella stessa gerarchia di ereditarietà?

Domande per assicurarsi realismo:

1. Ci sono caratteristiche innovative nel sistema? Sono stati fatti studi o costruiti prototipi per valutarne la fattibilità?
2. Le richieste di performance e affidabilità possono essere assicurate? E' stato costruito un prototipo per assicurarsi della fattibilità? Tale prototipo è in esecuzione su qualche hardware particolare?
