UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

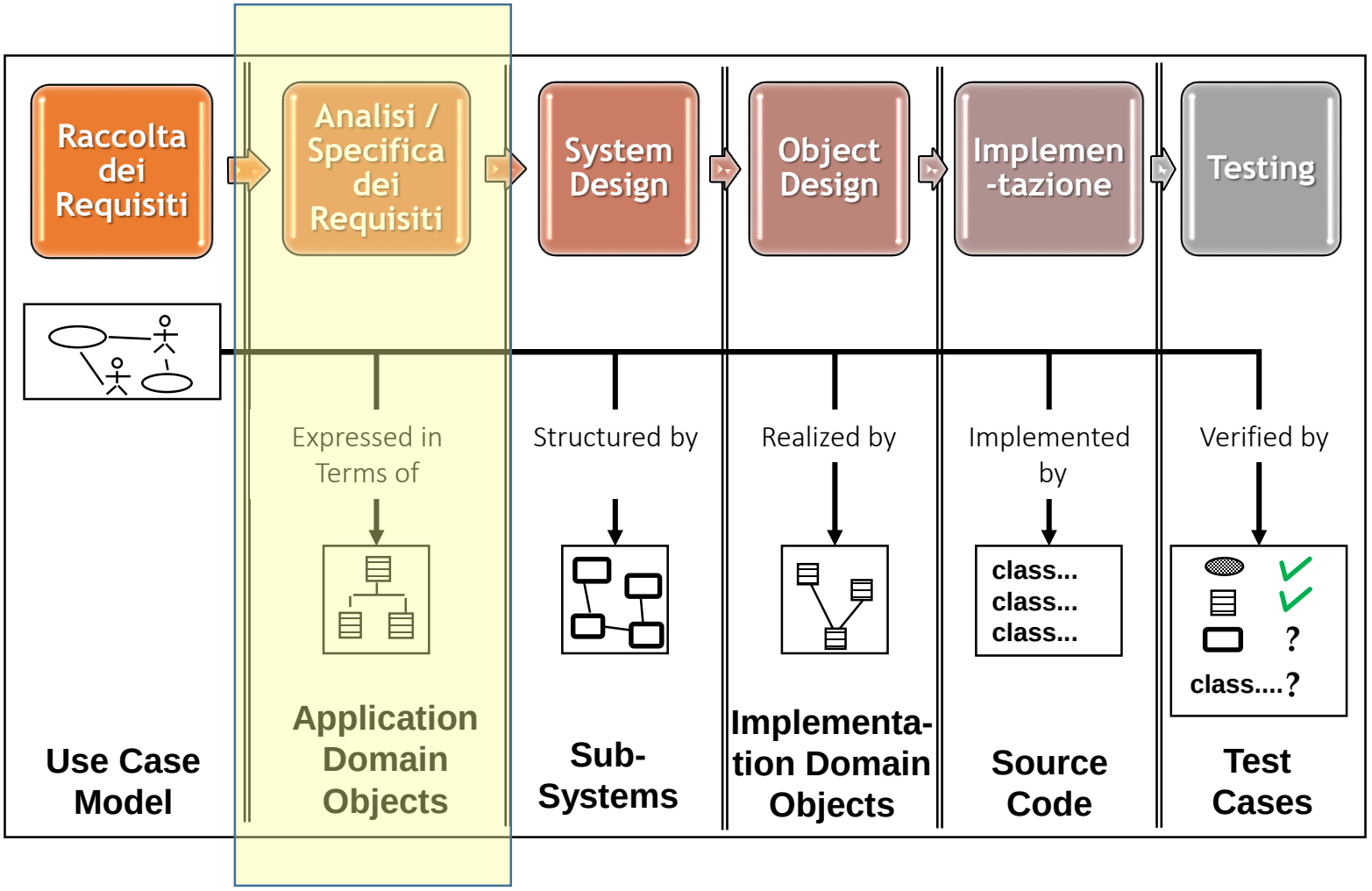
Ingegneria del Software – I Modelli di Dominio

Prof. Sergio Di Martino

Verso la progettazione…

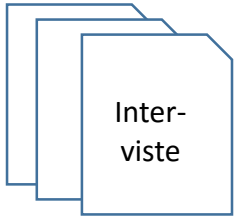
- Una delle maggiori difficoltà è “tradurre” i requisiti software in un progetto di sistema implementabile
- Obiettivo della fase successiva all’analisi dei requisiti è di iniziare ad individuare le classi, il loro comportamento dinamico, e le relazioni che vi intercorrono.
- Obiettivo della lezione: Definire i Modelli di Dominio
  - Identificazione degli oggetti
  - Definizione del comportamento degli oggetti
  - Definizione delle relazioni tra gli oggetti
  - Classificazione degli oggetti
  - Organizzazione degli oggetti

Ciclo di Vita del Software

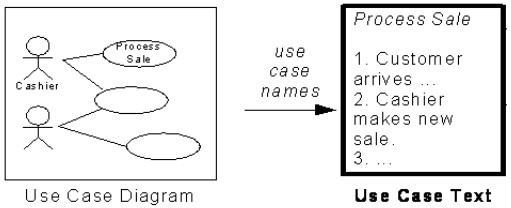


Requirement Engineering

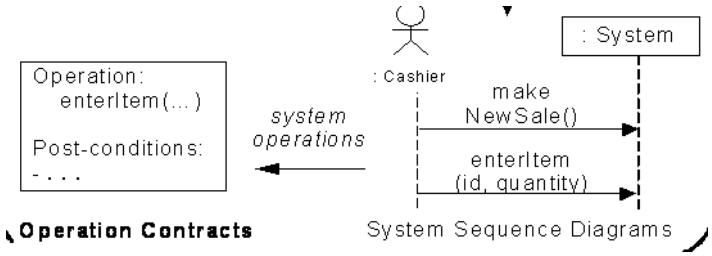
1. **Requirement Elicitation**

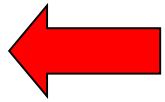


2. **Requirement Analysis**



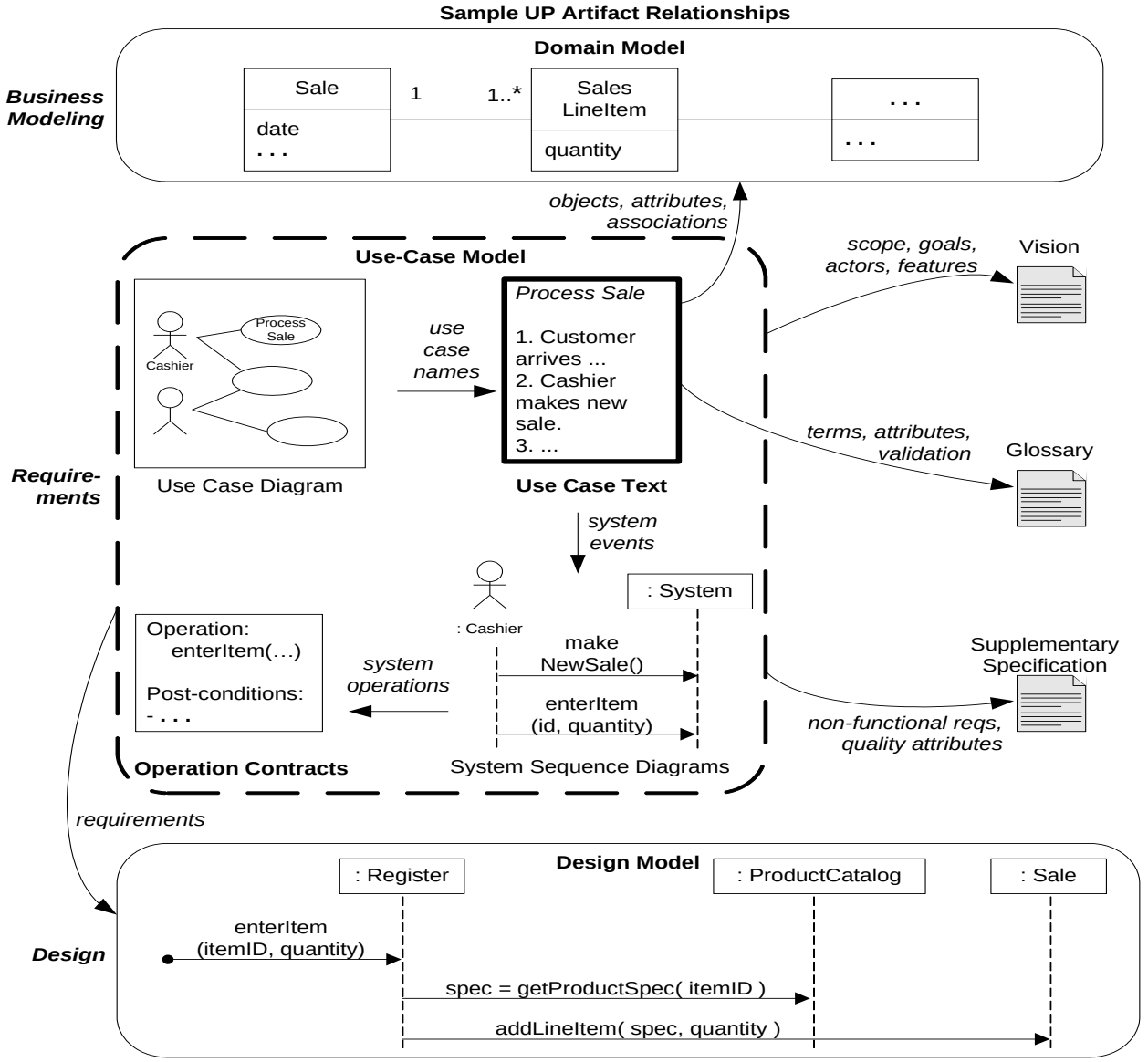
3. **Requirement Specification**





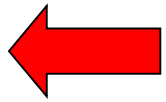
4. **Requirement Validation**

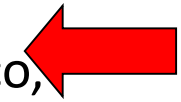
Artefatti coinvolti



Il Documento di Specifica dei Requisiti

- E’ composto da tre modelli:
  - **Modello Funzionale**, rappresentato da use cases
    - Possibile contratto col cliente, è scritto per i “profani”
  - **Modello di Dominio**, rappresentato dal diagramma delle classi
  - **Modello Dinamico**, rappresentato da statechart e sequence diagram
- Gli use case prodotti nella fase di raccolta dei requisiti vengono raffinati per derivare il Modello di Dominio e il Modello Dinamico, primo passo per la successiva Progettazione del sistema





Il modello di Dominio

- E’ il più importante modello della fase di Specifica dei Requisiti
- E’ una rappresentazione visuale di classi concettuali, relative al dominio del problema
  - Si esprime attraverso un class diagram, con classi, attributi e operazioni
- Si focalizza sui concetti che sono manipolati dal sistema, le loro proprietà e le relazioni

Il modello di Dominio (2)

- E’ un dizionario visuale dei concetti principali relativi al dominio
- E’ detto anche Modello a Oggetti di Analisi
- E’ una rappresentazione di oggetti del mondo reale in uno specifico dominio, NON di oggetti software
  - NB: sia il modello dinamico che il modello ad oggetti rappresentano concetti a livello utente, non a livello di componenti e classi software
  - Le classi di analisi rappresentano astrazioni che saranno dettagliate e/o raffinate successivamente

Il modello dinamico

- Si focalizza sul comportamento del sistema
  - I sequence diagram rappresentano le interazioni tra un insieme di oggetti durante un singolo use case
  - Gli statechart rappresentano il comportamento di un singolo oggetto o di alcuni oggetti strettamente accoppiati
- Consente di assegnare le responsabilità alle classi e quindi individuare nuove classi che sono aggiunte al modello ad oggetti di analisi, in un procedimento iterativo

Identificazione degli oggetti

Identificare gli Oggetti

- La fase fondamentale dello sviluppo del software OO e' quella dell'analisi (OOA Object Oriented Analysis) e della progettazione (OOD Object Oriented Design), perche' sono queste fasi che garantiscono il successo e il raggiungimento degli obiettivi dell'OOP
- Per individuare gli oggetti partecipanti si esamina la descrizione di ogni use case e si individuano le classi candidate
- E’ un procedimento altamente creativo e soggettivo
- Può essere guidato da 2 Euristiche:
  - Three-Object-Type → Classificazione Oggetti in Entity, Boundary e Controlo Categorie
    - Abbott → Identificazione oggetti Entity

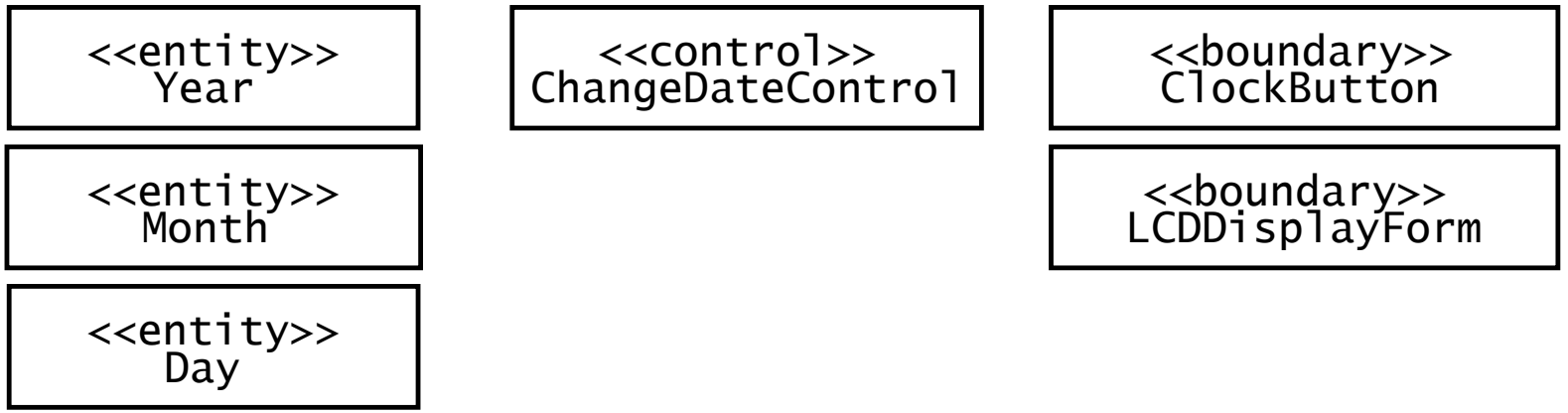
Euristica three-object-type

- Secondo tale euristica, il modello ad oggetti di analisi classifica gli oggetti in Entity, Boundary e Control:
  - Gli oggetti **Entity** modellano l'informazione persistente
  - Gli oggetti **Boundary** modellano le interazioni tra gli attori e il sistema
  - Gli oggetti **Control** modellano la logica che si occupa di realizzare gli use case

- L'approccio three-object-type porta a modelli che sono più flessibili e facili da modificare:
  - L'interfaccia al sistema (rappresentata da oggetti boundary) è più soggetta a cambiamenti rispetto alle funzionalità (rappresentate da oggetti entity e control)

Regole di Naming

- UML fornisce il meccanismo degli stereotipi per consentire di aggiungere tale meta-informazione agli elementi di modellazione
- E' opportuno usare convenzioni sui nomi:
  - Gli oggetti control possono avere il suffisso Control
  - Gli oggetti boundary dovrebbero avere nomi che ricordano aspetti dell'interfaccia (es. suffisso Form, Button, ecc)



Identificare gli Oggetti Entity e l'euristica di Abbott

- Gli oggetti Entity rappresentano i concetti del dominio.
- La loro individuazione può essere facilitata dall'uso dell'Euristica di Abbott
- L'euristica di Abbott si basa sull'analisi linguistica per identificare oggetti, attributi, associazioni dai requisiti di sistema
  - mappano parti delle parole (nomi, verbo avere, verso essere, aggettivi) per modellare componenti (oggetti, operazioni, relazioni di ereditarietà, classi)

Attività di Analisi: Euristica di Abbott

Parti del parlato	Componente del modello	esempio
Nome proprio	Istanza	Alice
Nome comune	Classe	Agente di Polizia (FieldOfficer)
Verbo fare	Operazione	Crea, Submit, Select
Verbo essere	gerarchia	È un tipo di, è uno di
Verbo avere	aggregazione	Ha, consiste di, include
Verbo modale	vincoli	Deve essere
Aggettivo	attributo	Descrizione dell'incidente

Euristica di Abbott

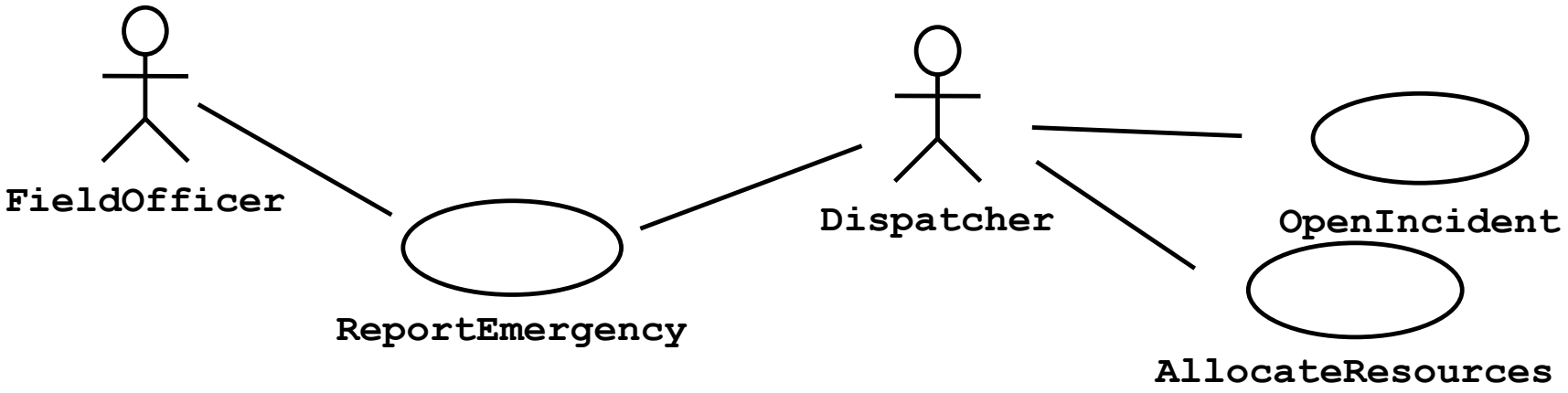
- Vantaggi:
  - Ci si focalizza sui termini dell'utente
- Svantaggi:
  - Il linguaggio naturale è impreciso, anche il modello ad oggetti derivato rischia di essere impreciso
  - La qualità del modello dipende fortemente dallo stile di scrittura dell'analista
  - Ci possono essere molti più sostantivi delle classi rilevanti, corrispondenti a sinonimi o attributi

Identificare gli Oggetti Boundary

- Gli oggetti Boundary rappresentano l'interfaccia del sistema con gli attori
  - In ogni use case, ogni attore (sia esso umano o sw già esistente) interagisce almeno con un oggetto Boundary
  - L'oggetto Boundary colleziona informazioni dall'attore e le traduce in una forma che può essere usata sia dagli oggetti Control che dagli oggetti Entity
  - Uno o più oggetti Boundary effettuano anche il processo inverso.

- Gli oggetti Boundary modellano l'interfaccia senza descriverne gli aspetti visuali!
  - Non ha senso parlare di "item di menu" o "scroll bar"
  - Lo sviluppo dell'interfaccia è solitamente di tipo prototipale, e iterativo

Modello di Caso d'Uso per la gestione di Incidenti



Esempio: ReportEmergency

Nome Use Case	ReportEmergency
Partecipanti	Inizializzato dal *FieldOfficer* Comunica con il *Dispatcher*

Flusso degli eventi

Attori	Sistema
Il FieldOfficer attiva la funzione "ReportEmergency" dal suo terminale	
	Il sistema risponde presentando un form al FieldOfficer
Il FieldOfficer completa il form selezionando il livello di emergenza, il tipo, la località, e una breve descrizione della situazione. Il FieldOfficer descrive anche possibili risposte alla situazione di emergenza. Quando il form è completo, il FieldOfficer sottomette il form	
	Il sistema riceve il form e notifica il Dispatcher
Il Dispatcher rivede le informazioni sottomesse e crea un Incident invocando lo use case OpenIncident. Il Dispatcher seleziona una risposta e comunica il report	
	Il sistema visualizza il report e la risposta selezionata per il FieldOfficer

Identificazione degli Oggetti Entity

- Dall'esame dello use case ReportEmergency, dalla conoscenza del dominio e dalle interviste al cliente è possibile identificare i seguenti oggetti
  - Dispatcher
  - FieldOfficer
  - Incident
  - EmergencyReport

- Si noti che EmergencyReport non è nominato esplicitamente nello use case:
  - nel passo 4 si nomina "informazione sottomessa dal FieldOfficer "
  - e dal colloquio con il cliente si deduce che è proprio ciò che solitamente è detto "emergency report"

Euristiche per Identificare gli Oggetti Boundary

- Identificare i controlli della UI di cui l'utente ha bisogno per iniziare lo use case (ReportEmergencyButton)
- Identificare form di cui l'utente ha bisogno per inserire dati nel sistema (ReportEmergencyForm)
- Identificare avvisi e messaggi che il sistema usa per rispondere all'utente (AcknowlegmentNotice)
- Non modellare aspetti visuali della UI con oggetti Boundary (meglio mock-up)
- Usare sempre i termini dell'utente finale per descrivere l'interfaccia, non usare termini del dominio di implementazione

Identificazione degli Oggetti Entity dallo use case ReportEmergency

Dispatcher	Agente di polizia che gestisce *Incidenti*. Un *Dispatcher* apre, documenta e chiude *Incident* in risposta al Report di Emergenza e ad altre comunicazioni con *FieldOfficers*. I *Dispatcher* sono identificati dal numero del badge
EmergencyReport	Report iniziale su un *Incident* inviato 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 agente di polizia o dei vigili 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*

Esempio: ReportEmergency

Nome Use Case ReportEmergency

Partecipanti
Inizializzato dal *FieldOfficer*
Comunica con il *Dispatcher*

Flusso degli eventi

Attore	Sistema
Il FieldOfficer attiva la funzione "ReportEmergency" dal suo terminale	
	Il sistema risponde presentando un form al FieldOfficer
Il FieldOfficer completa il form selezionando il livello di emergenza, il tipo, la località, e una breve descrizione della situazione. Il FieldOfficer descrive anche possibili risposte alla situazione di emergenza. Quando il form è completo, il FieldOfficer sottomette il form	
	Il sistema riceve il form e notifica il Dispatcher
Il Dispatcher rivede le informazioni sottomesse e crea un Incident invocando lo use case OpenIncident. Il Dispatcher seleziona una risposta e comunica il report	
	Il sistema visualizza il report e la risposta selezionata per il FieldOfficer

Identificazione degli Oggetti Boundary dallo use case ReportEmergency

ReportEmergencyButton	Bottone usato dal *FieldOfficer* per iniziare lo use case *ReportEmergency*
EmergencyReportForm	Form usato per l'input del *ReportEmergency*. Questa form è presentata sul *FieldOfficerStation* quando viene selezionata "Report Emergency". *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 sul *DispatcherStation* quando è ricevuto *l'EmergencyReport*. Il Dispatcher usa anche questa form per allocare le risorse e notificare il report del *FieldOfficer*
AcknowledgmentNotice	Avviso usato per mostrare l'acknowlegment del *Dispatcher* al *FieldOfficer*

Identificare gli Oggetti Control

- Gli oggetti Control sono responsabili del coordinamento degli oggetti Boundary e Entity
  - Si preoccupano di collezionare informazioni dagli oggetti Boundary e inviarle agli oggetti Entity
- Di solito non hanno una controparte nel mondo reale: modellano la logica di funzionamento di un caso d'uso.
- Esiste una stretta relazione tra oggetti Control e use case:
  - Un oggetto Control è creato all'inizio dello use case e cessa di esistere alla fine
- Linee Guida
  - Identificare almeno un oggetto Control per ogni use case
  - La vita di un oggetto Control dovrebbe corrispondere alla durata di uno use case o 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

Identificare gli Oggetti Control dallo use case ReportEmergency

- Il flusso di controllo dello use case ReportEmergency viene modellato con due oggetti Control
  - ReportEmergencyControl per FieldOfficer
  - ManageEmergencyControl per Dispatcher
- Tale decisione deriva dalla consapevolezza che FieldOfficerStation e DispatcherStation sono due sottosistemi che comunicano su un link asincrono
  - Questa decisione potrebbe essere rimandata all'attività di design, comunque renderla visibile in fase di analisi consente di focalizzare l'attenzione su comportamenti eccezionali, come la perdita di comunicazione tra due stazioni
- Nel modellare lo use case ReportEmergency sono state modellate le stesse funzionalità usando oggetti Boundary, Entity e Control
  - Si è passati da una prospettica "flusso di eventi" ad una "strutturale"
  - È aumentato il livello di dettaglio della descrizione
  - Sono stati selezionati termini standard per riferirci alle entità principali del dominio di applicazione e del sistema

Oggetti Control dallo use case ReportEmergency

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

l Sequence Diagrams

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, Boundary e Control
  - Mappare gli Use Case in Oggetti con Sequence Diagrams
  - Identificare le Associazioni
  - Identificare gli Attributi
  - Modellare il Comportamento dipendente dallo stato degli Oggetti individuali
  - Modellare le Relazioni di Ereditarietà
  - Rivedere il Modello di Analisi

Mappare Use case in Oggetti con Sequence Diagram

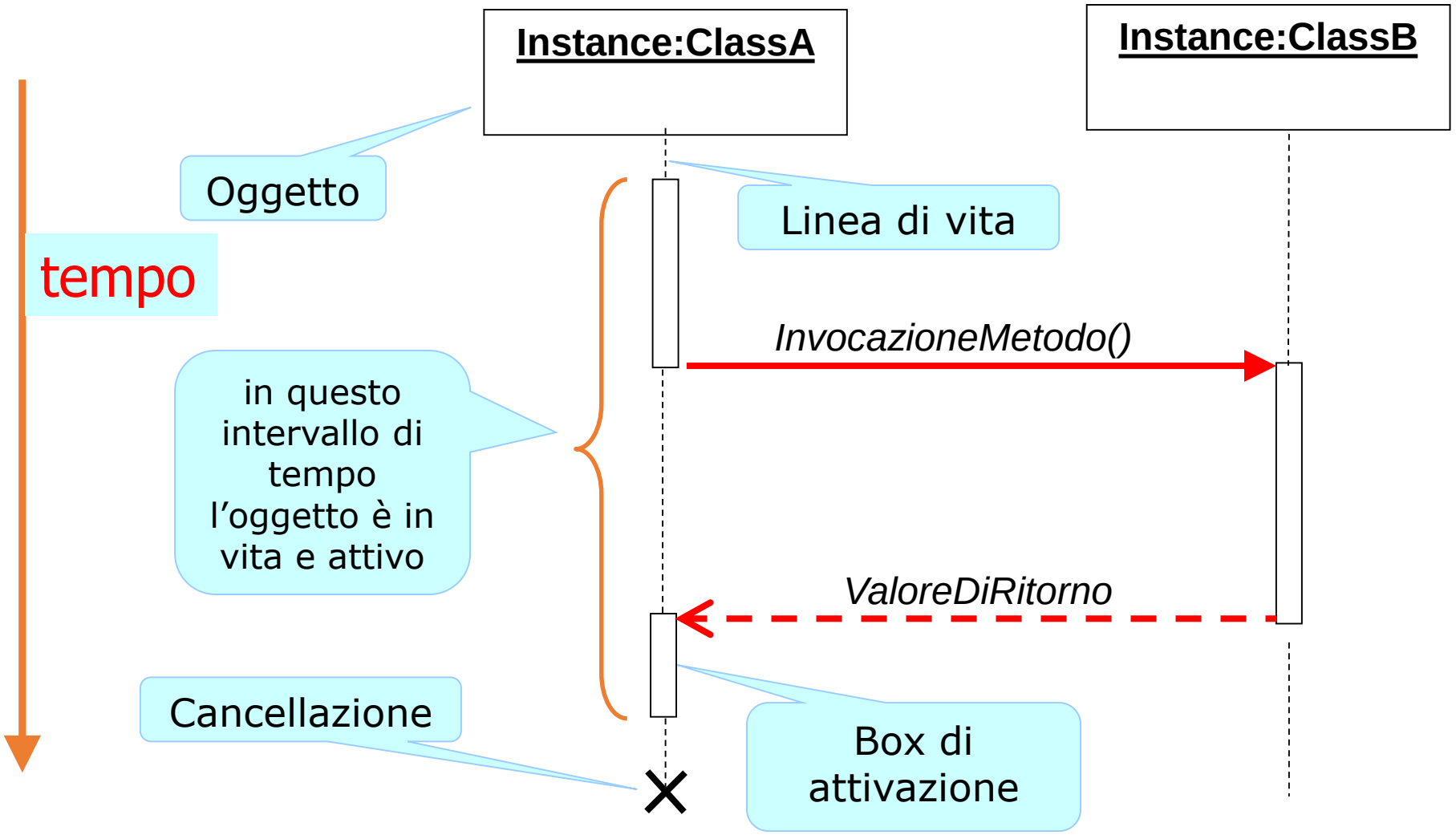
- Un Sequence Diagram lega use case con oggetti.
  - mostra come il comportamento di uno use case (o scenario) è distribuito tra i suoi oggetti partecipanti
    - Vengono assegnate responsabilità a ogni oggetto in termini di un insieme di operazioni
  - Illustra la sequenza di interazioni tra gli oggetti necessaria per realizzare uno use case
    - non ci occupiamo di questioni di implementazioni, come l'efficienza!
- Non è adatto alla comunicazione con il cliente
  - Solo per i clienti esperti è più intuitivo e preciso degli use case
- Fornisce una prospettiva diversa che consente di individuare classi mancanti e aree non chiare nelle specifiche

Richiami sui Sequence Diagrams

Sequence Diagram

- Mostra la sequenza temporale dei messaggi che gli oggetti si scambiano per portare a termine una funzionalità.
- E’ un diagramma di interazione: evidenzia come una funzionalità è realizzata tramite la collaborazione di un insieme di oggetti
- E’ uno dei principali input per l’implementazione dello scenario
  - E’ utilizzato in analisi e poi ad un maggior livello di dettaglio in design

Sequence Diagram



Sequence Diagram

- La ricezione di un messaggio determina l'attivazione di un metodo
  - L'attivazione è rappresentata da un rettangolo sulla linea della vita, da cui altri messaggi possono prendere origine
  - La lunghezza del rettangolo rappresenta il tempo durante il quale l'operazione è attiva
- La vita degli oggetti
  - Il tempo procede verticalmente dal top al bottom
  - Al top del diagramma si trovano gli oggetti che esistono prima del 1° messaggio inviato
  - Oggetti creati durante l'interazione sono illustrati con il messaggio <<create>>
  - Oggetti distrutti durante l'interazione sono evidenziati con una croce
  - La linea tratteggiata indica il tempo in cui l'oggetto può ricevere messaggi

Messaggi nei Sequence

- In generale un messaggio rappresenta il trasferimento del controllo da un oggetto ad un altro

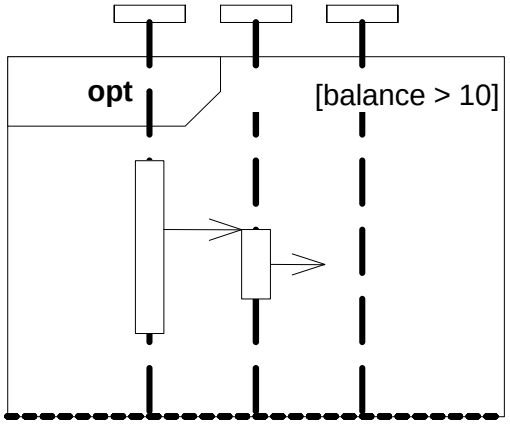
- Se l'oggetto che invia il messaggio rimane in attesa che l'oggetto ricevente ritorni, si ha un messaggio **sincrono**

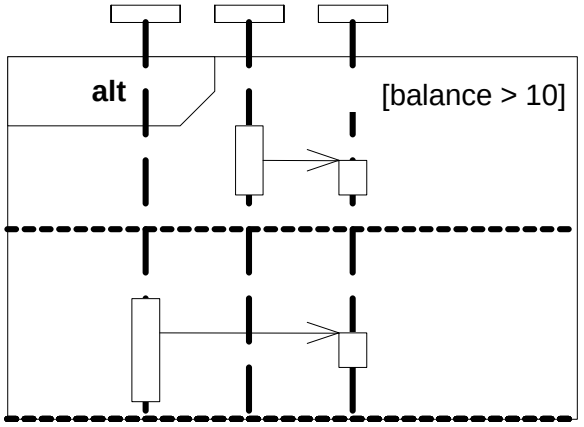
- Se l'oggetto che invia il messaggio prosegue la propria elaborazione in parallelo all'oggetto chiamato, siamo in presenza di un messaggio **asincrono**.

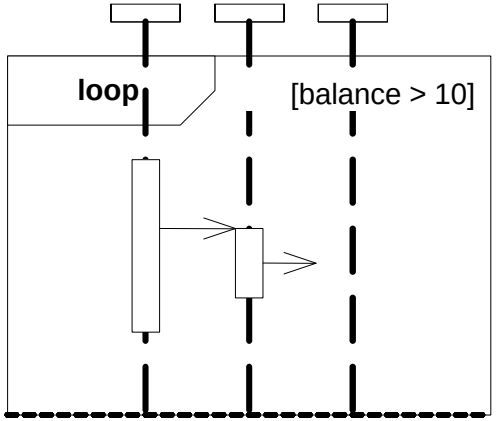
- Il valore restituito all'oggetto chiamante si indica con un **messaggio di ritorno**

Opzioni e Cicli

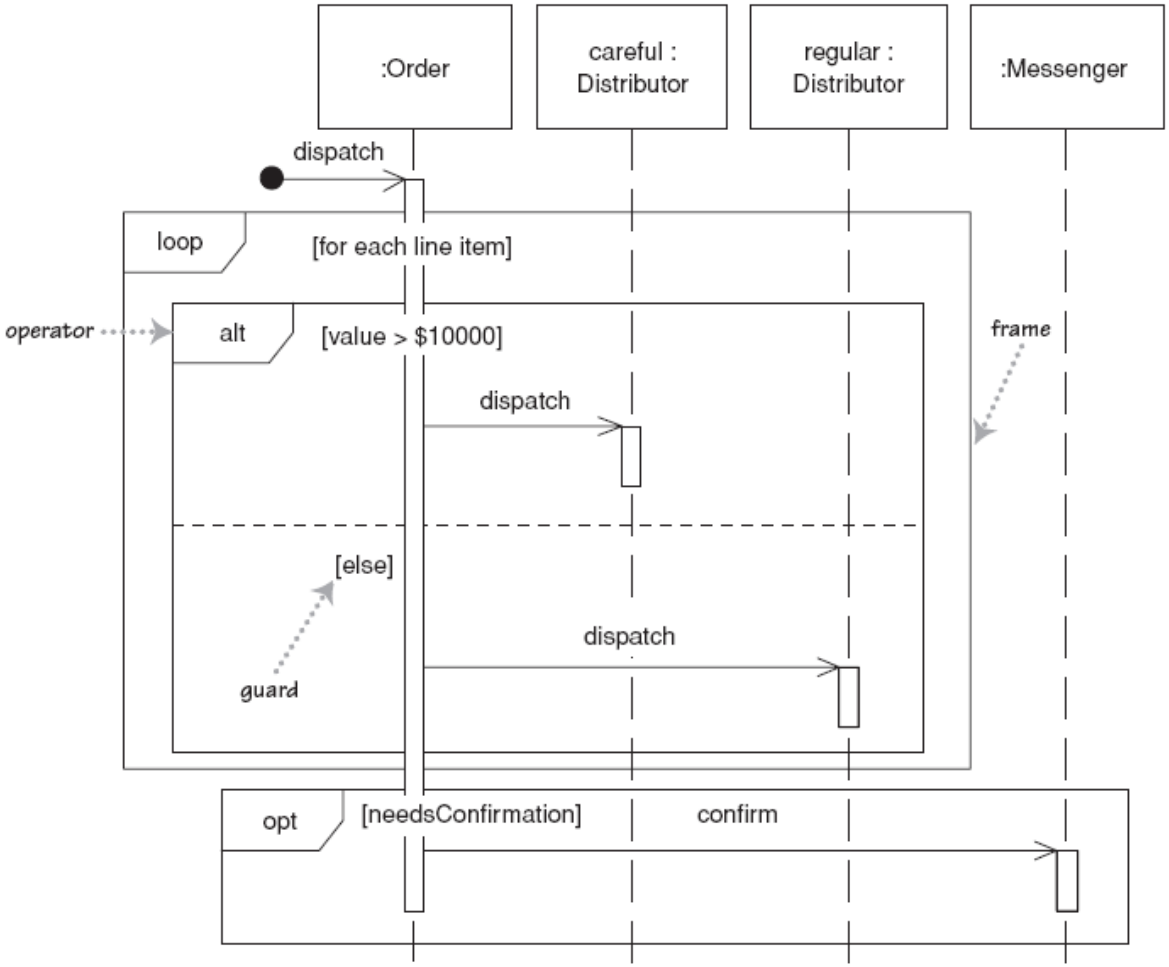
- Si indicano con una cornice intorno ad una parte del Sequence, per indicare che quella sezione è opzionale o viene ripetuta.
- Rappresentazioni:
  - if -> OPT [condition]
  - if/else -> ALT [condition], separati da linea orizzontale tratteggiata
  - cicli -> LOOP [condition o items su cui ciclare]





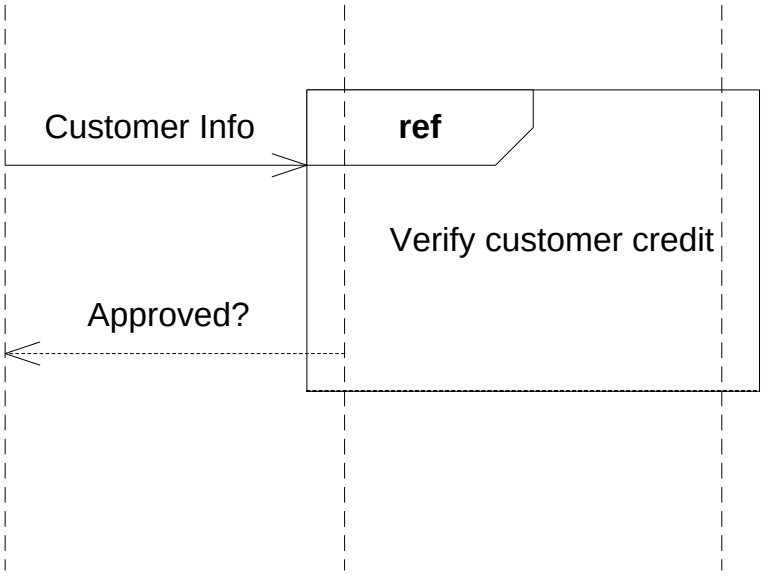


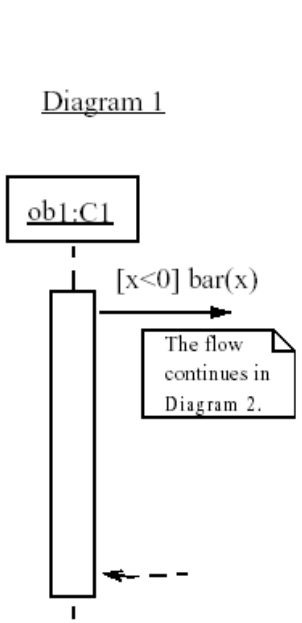
Esempio

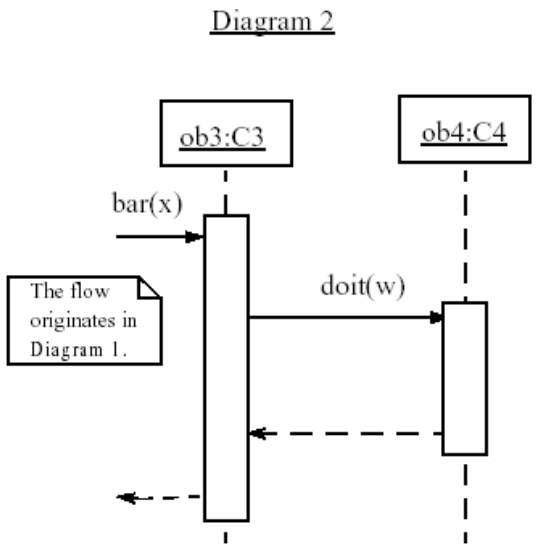

:Order
careful : Distributor
regular : Distributor
:Messenger
dispatch
loop
[for each line item]
operator
alt
[value > $10000]
frame
dispatch
[else]
guard
dispatch
opt
[needsConfirmation]
confirm

Linking sequence diagrams

- Se un Sequence è troppo complesso o fa riferimento ad un altro Sequence, si può utilizzare il REF, con:
  - Un rettangolo con label REF, col nome dell'altro diagramma
  - Una freccia che punta a tale rettangolo
  - Una eventuale condizione per specificare quando si fa il riferimento



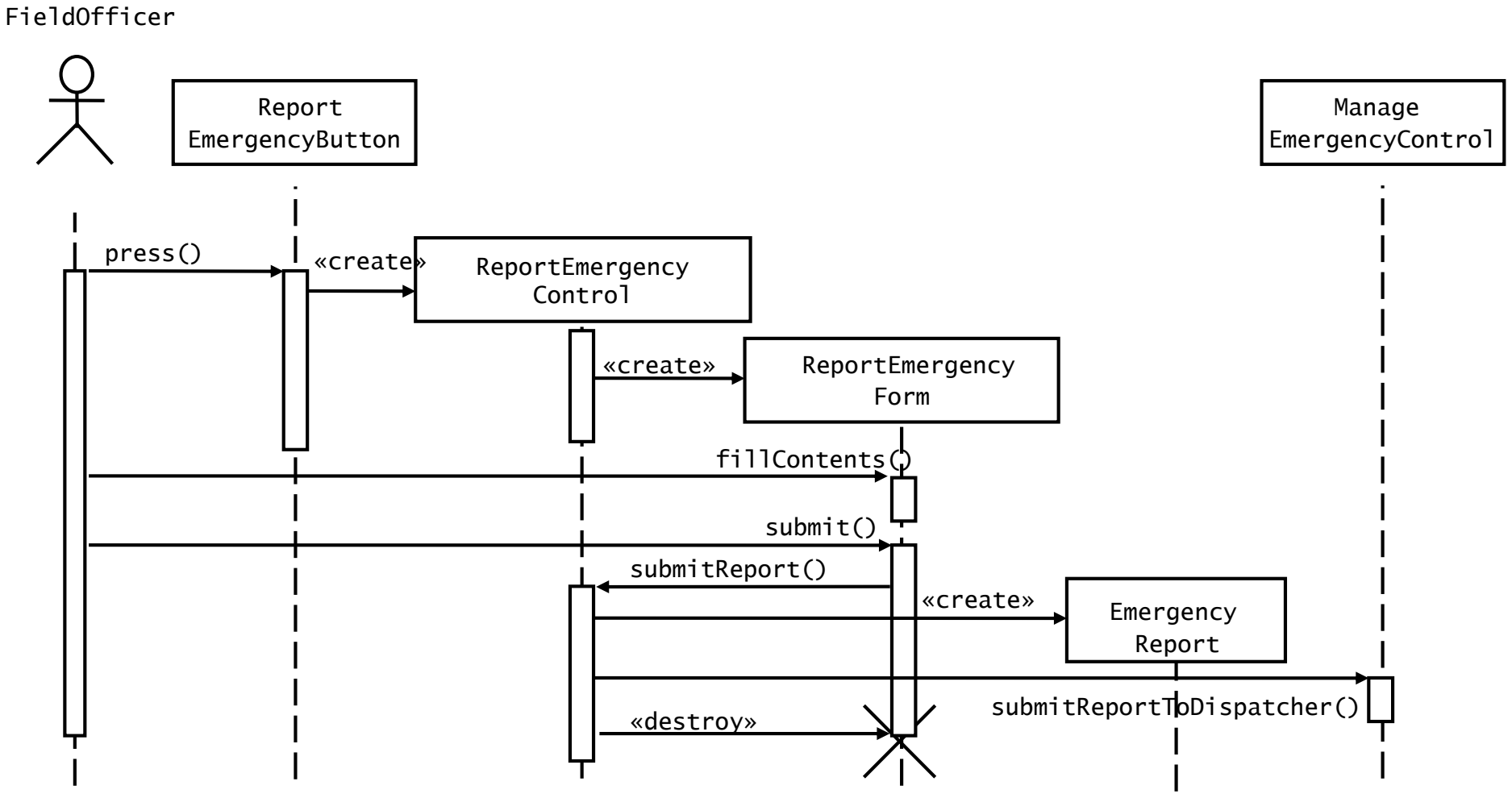




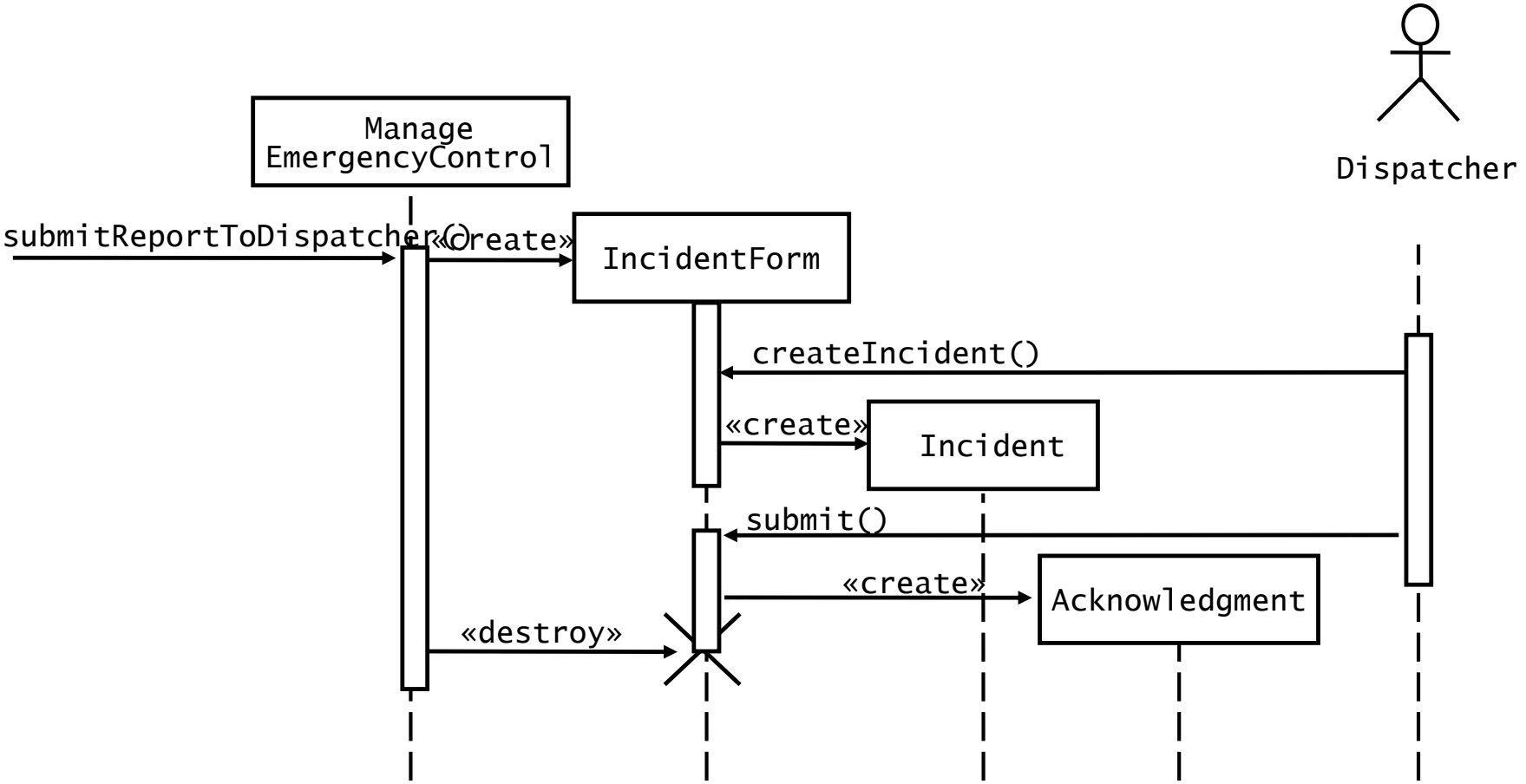
Sequence Diagram in fase di analisi

- Convenzioni per utilizzare i Sequence Diagrams con l'euristica Three-Object-Type:
  - La colonna più a sinistra rappresenta l'attore che inizia lo use case
  - La seconda colonna -> oggetto Boundary con cui l'attore interagisce per iniziare lo use case
  - La terza colonna -> oggetto Control che gestisce il resto dello use case
  - Le altre colonne possono rappresentare qualunque oggetto che interviene nel caso d'uso.
    - Gli oggetti Control creano altri oggetti Boundary/Entity e possono interagire con altri oggetti
  - Oggetti Control accedono ad altri oggetti Entity e Boundary
  - Gli oggetti Entity non accedono mai agli oggetti Control e Boundary: ciò rende più facile condividere oggetti Entity tra più use case

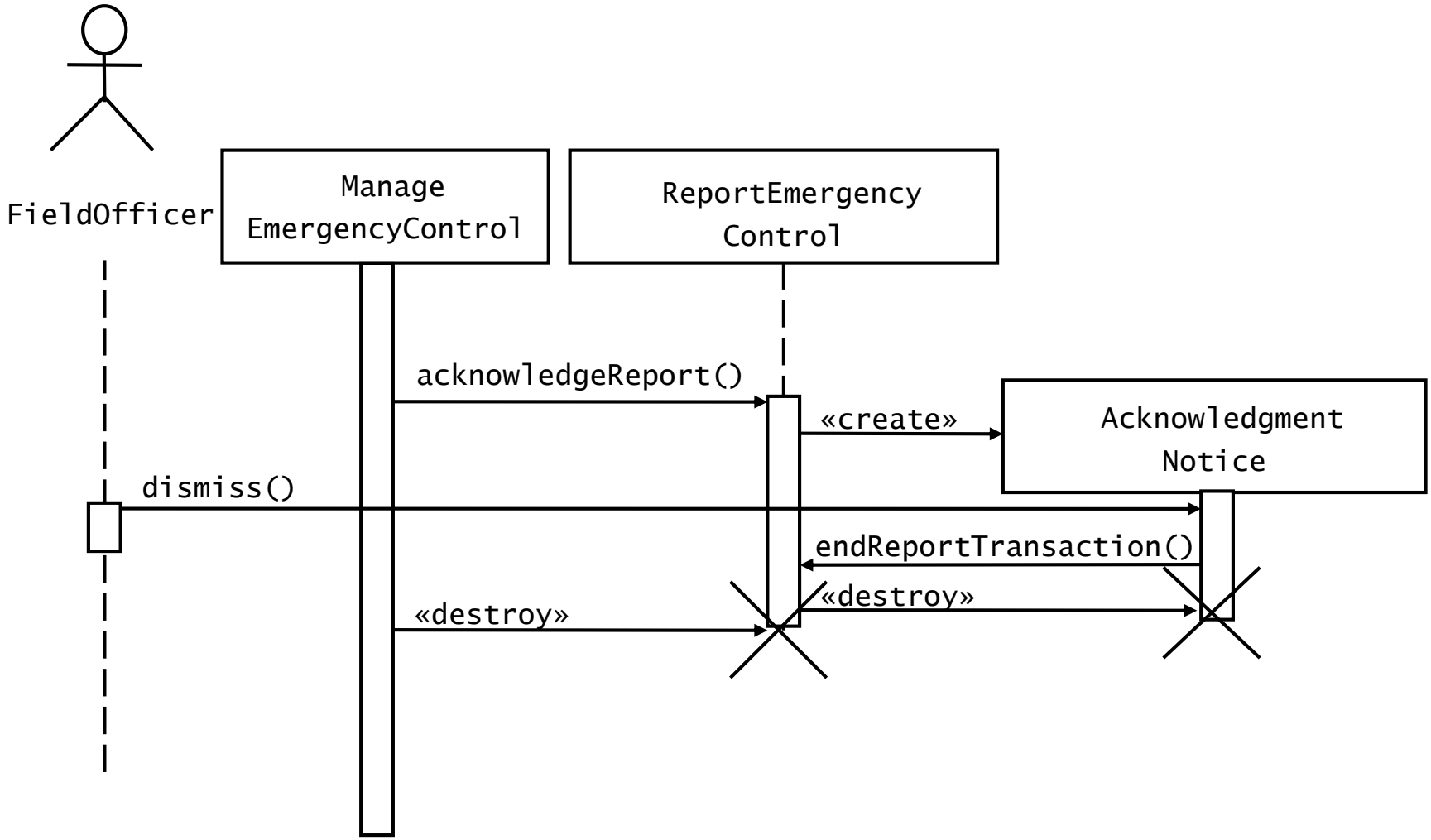
Sequence diagram for the ReportEmergency use case.



Sequence diagram for the ReportEmergency use case (continued).



Sequence diagram for the ReportEmergency use case (continued).



Identificazione di nuovo oggetto

- Lo use case ReportEmergency è incompleto: menziona l'esistenza di una notifica ma non descrive l'informazione ad essa associata
- C'è necessità di chiarire con il cliente→ l'oggetto Acknowledgment è aggiunto al modello di analisi e lo use case è raffinato
- L'oggetto Acknowlegment è creato prima dell'oggetto Boundary AcknowldegmentNotice

Identificazione di nuovo oggetto

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

Analisi e Sequence Diagram

- Durante l'analisi i Sequence Diagram sono usati per individuare
  - nuovi oggetti
  - comportamenti mancanti

- Disegnare Sequence Diagram è un'attività laboriosa, quindi:
  - Occorre dare priorità a quelle funzionalità problematiche o non ben specificate
  - Per le parti ben definite può essere utile solo per evitare di posticipare alcune decisioni chiavi