Indice dei contenuti

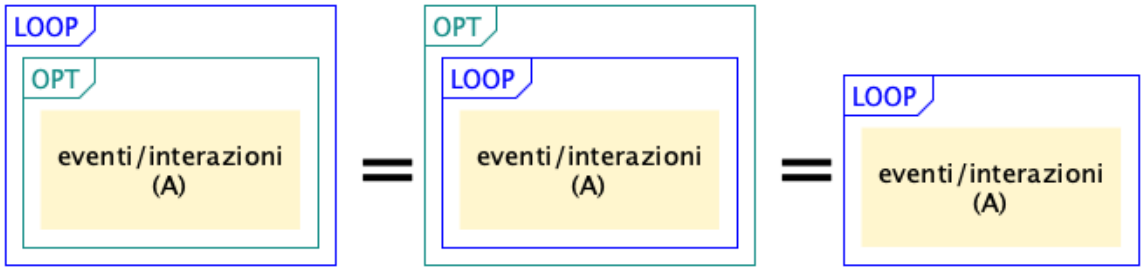
Introduzione

Vediamo adesso alcuni suggerimenti per creare i SSD di "Gestire i compiti della cucina" a partire dai principali scenari di successo descritti nel documento di discussione sullo UC Dettagliato.

In generale se lo UC dettagliato è descritto in modo sensato realizzare i SSD è abbastanza "automatico"; viceversa, possono sorgere delle difficoltà se lo UC contiene dei passi che sono troppo generici. Infatti l'ideale sarebbe poter tradurre ciascun passo dello UC in un "evento di sistema", ossia in un messaggio dall'Attore al Sistema contenente una richiesta, al quale il Sistema risponde prendendosi in carico le responsabilità descritte nello UC. Tuttavia talvolta si scopre in questa fase che abbiamo scritto nello UC un passo che può essere portato a termine solo attraverso più interazioni fra Attore e Sistema, e allora avremo che a un singolo passo corrispondono più eventi nel SSD.

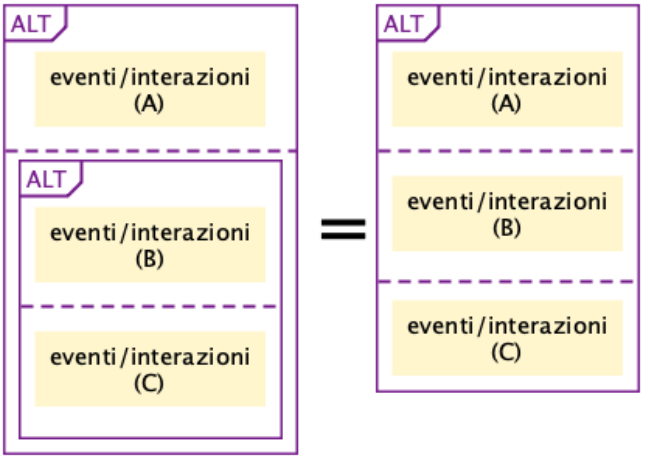
Un'altra difficoltà che tipicamente si incontra è quella di organizzare il "flusso del processo": gli UC dettagliati e i SSD usano infatti due modi molto diversi di esprimere questo aspetto. Gli UC dettagliati usano principalmente due costrutti: l'indicazione di ripetizione e l'indicazione di salto (in avanti, per evitare di svolgere una parte del processo, e all'indietro, per ripeterne una). Invece i SSD usano dei blocchi molto più simili ai costrutti che si usano in programmazione: il LOOP somiglia a un ciclo while, l'OPT ad un if, e l'ALT ad un if...else. In un certo senso dunque il flusso negli SSD è espresso in modo meno narrativo, e più strutturato – l'obiettivo infatti è avvicinarsi sempre di più a una rappresentazione formale dei requisiti, che li renda facilmente traducibili in indicazioni di progettazione e quindi in codice.

Prima di approfondire alcuni esempi relativi all'esercitazione su "Gestire i compiti della cucina", alcune indicazioni pratiche che possono tornare utili per non creare dei SSD troppo complicati.
Innanzitutto, un blocco loop generico (in cui cioè non si specifica un minimo o un massimo di ripetizioni, cosa che si potrebbe fare scrivendo ad esempio LOOP(2,100) nell'intestazione) si intende che può essere ripetuto da 0 a n volte, quindi di fatto "racchiude" in sé l'opzionalità. Quindi un eventuale blocco OPT, che si trovi immediatamente all'esterno, o immediatamente all'interno, di un blocco LOOP, può essere rimosso.



In secondo luogo, due blocchi ALT direttamente innestati uno dentro l'altro sono equivalenti ad un unico blocco ALT con tre alternative. Per analogia si può pensare alla differenza fra la scrittura:

```
if (cond1) {
      istruzioni A
} else {
      if (cond2) {
            istruzioni B
      } else {
            istruzioni C
      }
}
```



e la scrittura:

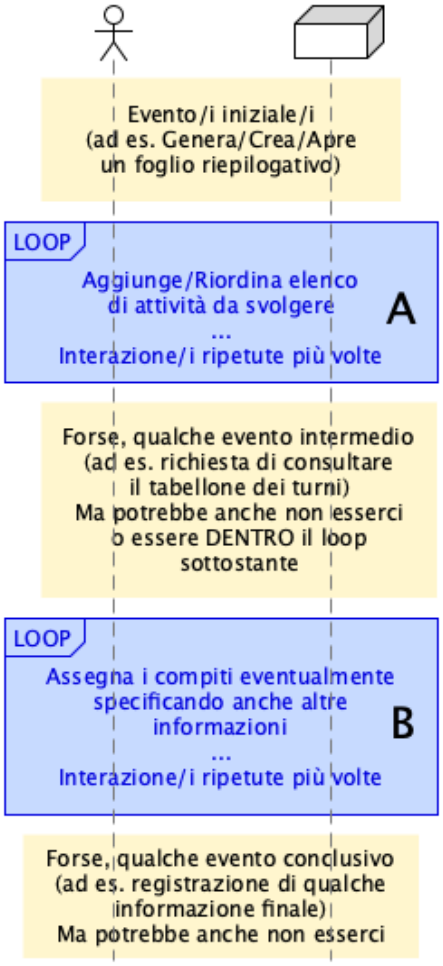
```
if (cond1) {
      istruzioni A
} else if (cond2) {
      istruzioni B
} else {
      istruzioni C
}
```

Passiamo ora ad analizzare alcuni esempi di come potrebbe presentarsi la struttura del vostro SSD per lo scenario principale di "Gestire compiti della cucina". Per ciascun esempio riportiamo eventuali criticità da tenere in conto e possibili migliorie da apportare. Attenzione, il vostro SSD potrebbe essere un misto degli esempi presentati, non è detto che abbiamo coperto tutti i casi possibili!

Il flusso del processo nei SSD per lo scenario principale di Gestire compiti

Vediamo alcune ipotesi di struttura che potrebbe avere il vostro SSD dello scenario principale, sulla base di quanto discusso relativamente a questo UC e soprattutto sulla base dei documenti da voi stilati che ci avete mostrato. **Attenzione: le figure che seguono non mostrano dei SSD ma degli "schemi di struttura logica" contenenti commenti e note, dove gli scambi di eventi fra attore e sistema sono sostituiti da astrazioni visto che non possiamo sapere cosa voi avete scritto precisamente!**

Esempio I



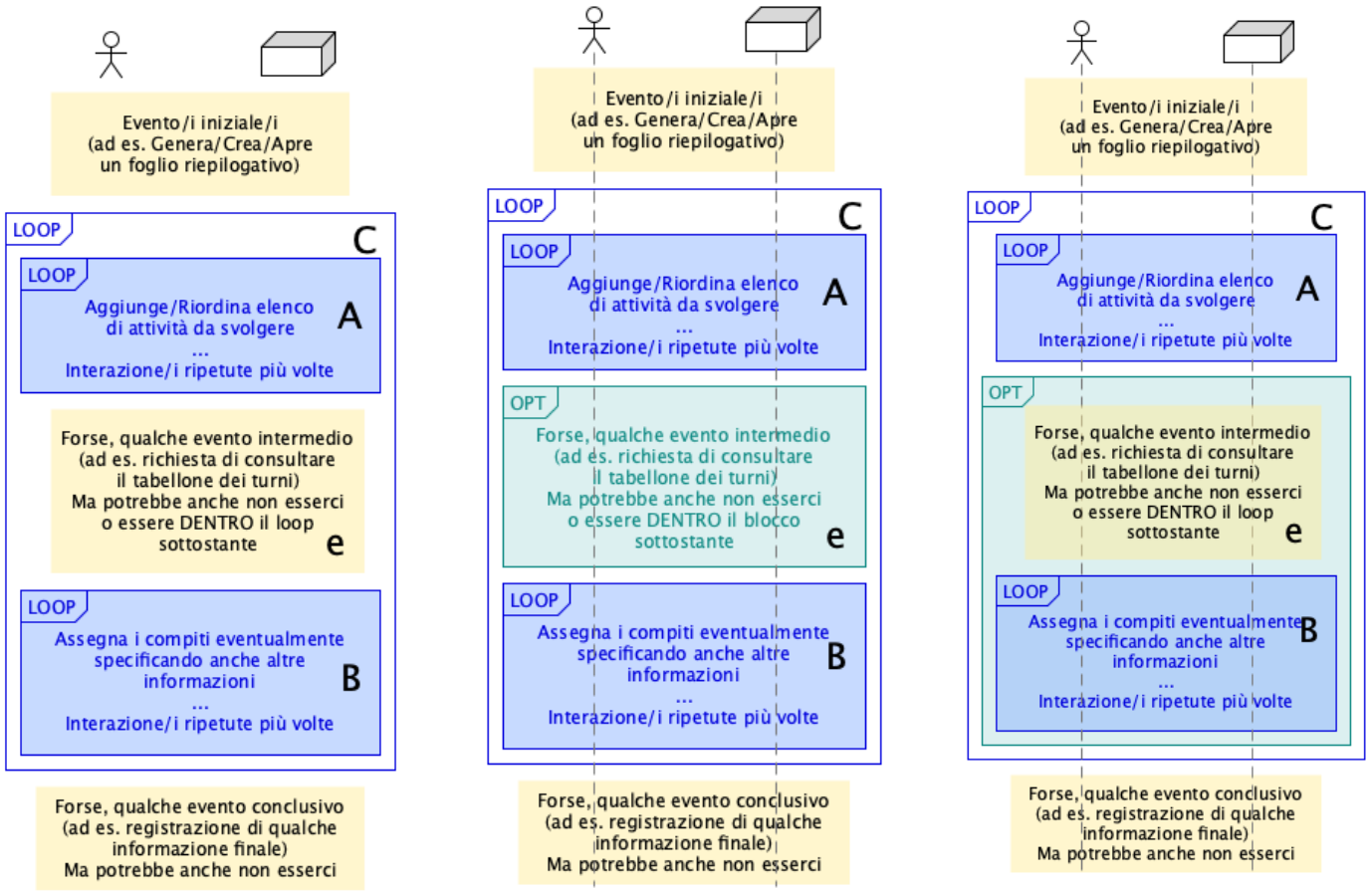
L'esempio qui è sinistra denota uno schema base in cui l'attore dopo aver "iniziato in qualche modo il suo lavoro" procede ad aggiornare l'elenco delle attività da svolgere (sappiamo che può aggiungere delle attività – e, tramite le estensioni, toglierne – e anche riordinare questo elenco). Le azioni che riguardano l'aggiornamento possono essere ripetute più volte e dunque si trovano in un loop **(A)**. Dopodiché l'attore prosegue con l'assegnamento dei compiti, che potrebbe essere preceduto da qualche attività preliminare. L'assegnamento dei compiti può consistere in una o più interazioni col sistema, anch'esso potrà essere ripetuto più volte e quindi è inserito in un loop **(B)**. Infine è possibile – dipende da come avete scritto il vostro UC – che ci sia qualche interazione conclusiva.

Questo schema è un pochino rigido, e se vi è venuto così forse il processo era già un po' rigido a livello di UC. Infatti non è possibile tornare ad aggiornare l'elenco di attività dopo aver assegnato qualche compito. Potrebbe essere un elemento da rivedere, o potete decidere che vi va bene così. Ricordate solo di questo vincolo quando andrete a progettare e soprattutto implementare!

Esempio II

L'esempio che presentiamo qui di seguito è più flessibile del precedente ma richiede alcune riflessioni in più. Guardiamo l'immagine (II-1): in questo caso si è inserito il loop aggiuntivo **(C)** che permette all'utente di tornare ad aggiornare l'elenco delle attività dopo aver

assegnato dei compiti. Notiamo però che in qualunque ripetizione del loop **(C)** la fase centrale **(e)**, se presente, deve *necessariamente* essere effettuata. Se **(e)** contiene interazioni che devono obbligatoriamente svolgersi dopo aver riorganizzato le attività, allora questo è corretto. Se invece **(e)** contiene interazioni che sono preliminari al loop **(B)**, ma che qualora esso non venga effettuato possono anche essere omesse, allora potrebbe essere necessario renderla opzionale. Nell'immagine (II-2) l'interazione **(e)** è opzionale, può però in questo modo accadere che il loop **(B)** venga effettuato *senza* che sia stato fatto (e). Di nuovo, in alcuni casi questo potrebbe andare bene, in altri meno. Ad esempio, se (e) descrive la consultazione del tabellone dei turni, potremmo ritenere che non si possano assegnare i compiti senza prima aver richiesto il tabellone al Sistema. Allora sarà necessario rendere opzionale l'intero blocco **(e)+(B)** (si veda l'immagine (II-3)).

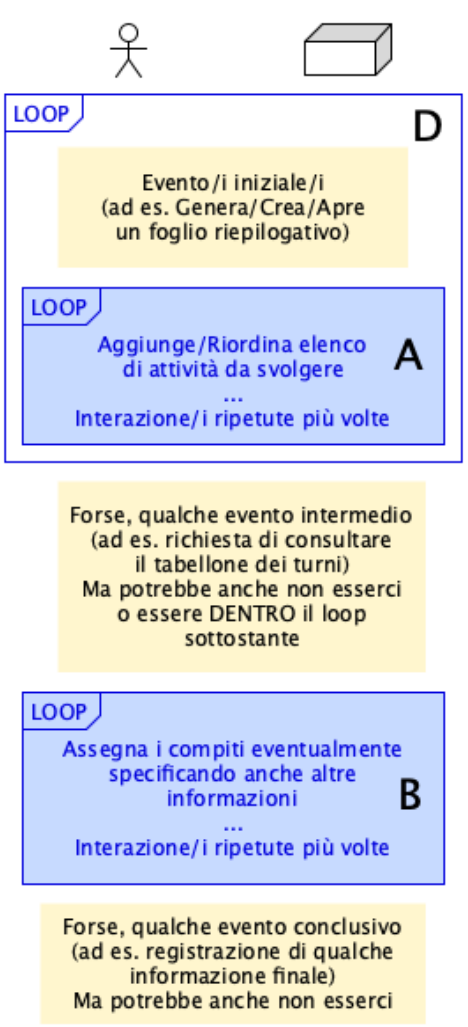


(II-1) (II-2) (II-3)

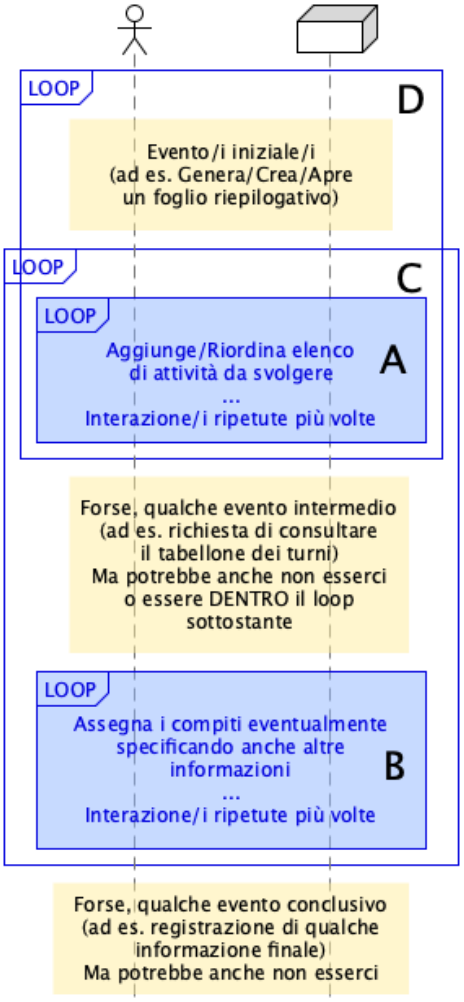
Esempio III

La figura (III-1) mostra una variante dell'esempio I. In questo caso si è introdotto un loop **(D)** per permettere all'attore di aprire più fogli riepilogativi prima di passare all'assegnamento dei compiti. Questa variante può andar bene, pur presentando le stesse rigidità di processo discusse per l'esempio I: il processo si divide infatti rigidamente in due parti, e dopo aver iniziato la seconda non è più possibile tornare alla prima. Per rendere

meno rigido questo flusso dobbiamo introdurre anche il loop **(C)** visto nell'esempio 2. La figura (III-2) mostra questo caso. Come visto nell'esempio II, potrebbe essere necessario in quest'ultima versione rendere opzionale la parte **(e)** o l'intero blocco **(e)+(B)**. Omettiamo le figure corrispondenti, che possono essere facilmente ottenute combinando la (III-2) con la (II-2) o la (II-3).



(III-1)



(III-2)

Attenzione: Nella figura (III-2) i due loop **(C)** e **(D)** si intersecano parzialmente. Ci si può chiedere se un SSD così costruito sia "ben formato" o meno. Sebbene in generale non sia una buona prassi intersecare parzialmente i blocchi (che siano essi loop, alt o opt), abbiamo stabilito che **limitatamente ai sequence di sistema** e **al caso di due blocchi loop** permettiamo di farlo, perché non ci sono ambiguità sul significato che si vuole esprimere, e d'altra parte cercare di eliminare questa sovrapposizione porterebbe a realizzare un SSD più complicato e confuso. Se però stessimo usando i Diagrammi di Sequenza per esprimere un algoritmo, anziché il processo di un attore umano, e dunque il blocco loop rappresentasse un ciclo while, una simile intersezione parziale sarebbe vietata, perché sarebbe come scrivere:

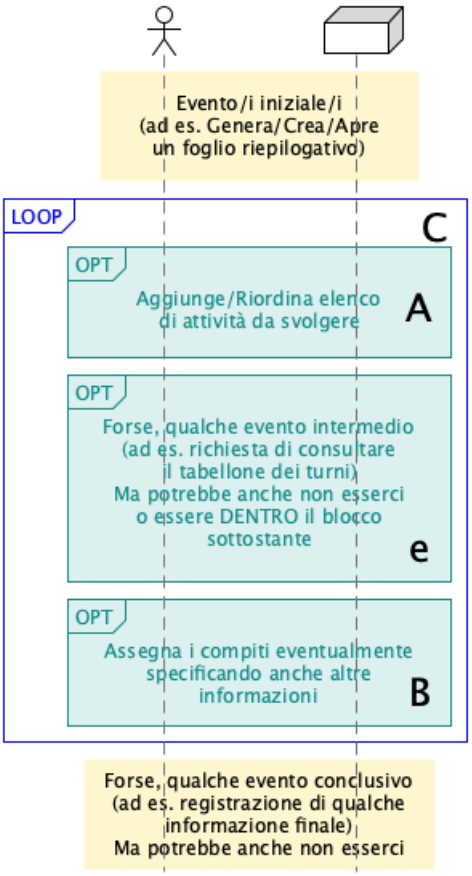
```
while (condizione_D) {
    istruzioni_iniziali;
```

```
    while (condizione_C) {
          istruzioni_A;
} // fine del loop con condizione_D
          istruzioni_e;
          istruzioni B;
    } // fine del loop con condizione_C
istruzioni_finali;
```

e vedete bene che una cosa del genere non avrebbe senso!

Distinguiamo quindi bene il caso in cui i Diagrammi di Sequenza sono usati per esprimere un algoritmo (allora in quel caso dobbiamo essere formali e rispettare la semantica dei blocchi) dal caso in cui rappresentiamo il processo di un attore umano: in questo caso privilegiamo la leggibilità.

Esempio IV



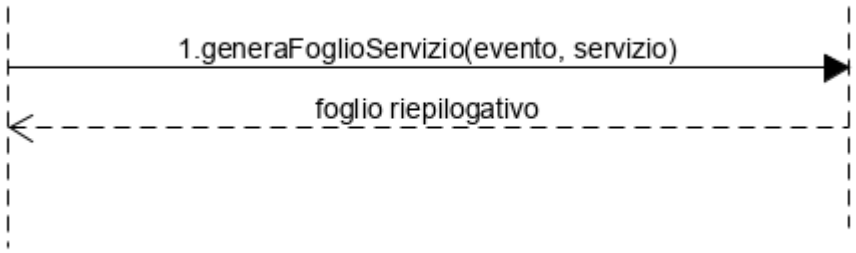
Questo tipo di struttura prevede, se escludiamo le interazioni iniziali e finali, un unico grande loop **(C)** all'interno del quale tutto è opzionale.
Dal punto di vista formale questo esempio è equivalente a quello riportato in figura II-2, quindi valgono le stesse accortezze là descritte. In generale, quando si mette in atto una struttura di questo tipo bisogna essere certi che tutti i blocchi indicati come opzionali siano realmente indipendenti, ossia ad esempio che **(B)** possa essere fatto senza **(e)**, ma anche che abbia senso fare **(e)** senza poi fare **(B)**.

I messaggi fra Attore e Sistema

Nello UC le interazioni fra Attore e Sistema sono rappresentate con una narrazione. Ogni riga della tabella dell'UC rappresenta una interazione. Prendiamo per esempio:

#	Attore	Sistema
1	Genera il foglio riepilogativo per un evento (di cui ha ricevuto l'incarico)	Precompila il foglio riepilogativo per l'evento specificato.

Nell'SSD questa interazione viene rappresentata con un messaggio dall'Attore al Sistema e una risposta dal Sistema.



Prima di descrivere il messaggio ricordiamo che c'è un foglio riepilogativo per ogni servizio all'interno di evento (ricordiamo dal glossario che un evento può comprendere più servizi, ad esempio sia il pranzo che la cena, o più cene su giorni diversi, ecc…).

Nel descrivere il messaggio dall'Attore al Sistema dovremo avere cura di indicare:

- il **tipo** di messaggio, che specifica che cosa si sta chiedendo di fare al Sistema (in questo caso (**generaFoglioServizio**). Scriviamo il nome del messaggio in camelCase.
- fra parentesi, le informazioni che l'Attore deve dare al Sistema perché esso possa fare ciò che deve. In questo caso il Sistema non può generare il foglio riepilogativo senza sapere di quale servizio (e in quale evento) si sta parlando!

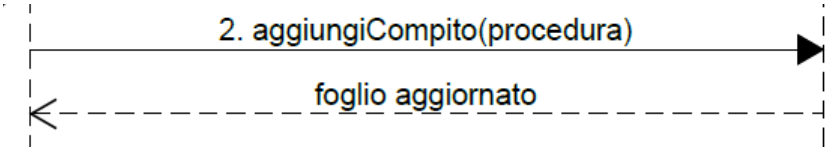
Nel descrivere la risposta del Sistema dovremo indicare quali informazioni il Sistema manda all'attore; se c'è un risultato dell'elaborazione, manderà il risultato. Altrimenti potrà anche solo rispondere con "ok" se tutto è andato bene, o "errore" se qualcosa è andato storto.

Le informazioni che Attore e Sistema si scambiano dovrebbero essere o informazioni "semplici" (numeri, testo, si/no…) o istanze di concetti descritti nel modello di dominio. Se nel modello di dominio abbiamo un concetto di "foglio riepilogativo" allora il Sistema potrà rispondere con il foglio riepilogativo che ha appena generato.

E' importante ricordare che un SSD è una narrazione unica, nel corso della quale possiamo presumere che il Sistema "ricordi" ciò che gli è già stato comunicato, e a maggior ragione ciò che ha elaborato lui stesso.
Consideriamo ad esempio il passo successivo:

2	Opzionalmente aggiunge preparazioni e ricette all'elenco delle cose da fare	Aggiorna il foglio riepilogativo.

Possiamo rappresentarlo col seguente scambio di messaggi:



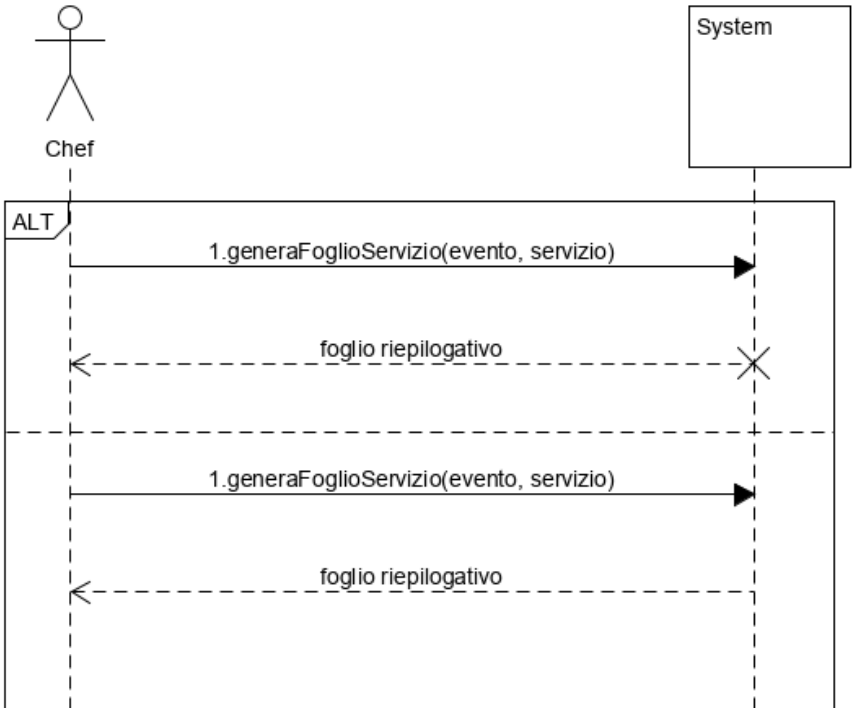
Come si vede, al Sistema viene comunicata solo la "procedura" (un concetto che generalizza i concetti di ricetta e preparazione), perché il Sistema conosce già il foglio riepilogativo a cui aggiungerla (l'ha generato lui al passo precedente!) e l'evento a cui ci si riferisce (l'attore glielo ha detto al passo precedente).

Struttura del SSD

Vediamo ora alcune possibilità di strutturazione del SSD basate sulle diverse varianti di scenari principali proposte nel documento di discussione sullo UC Dettagliato. Come già sottolineato in precedenza, la vostra potrebbe essere ancora diversa, quindi prendete questi ragionamenti come spunti.
Iniziamo prendendo in esame la versione II dello scenario principale di successo:

#	Attore
1	Genera il foglio riepilogativo per un evento (di cui ha ricevuto l'incarico)
	Se desidera prosegue con il passo 2, altrimenti termina il caso d'uso

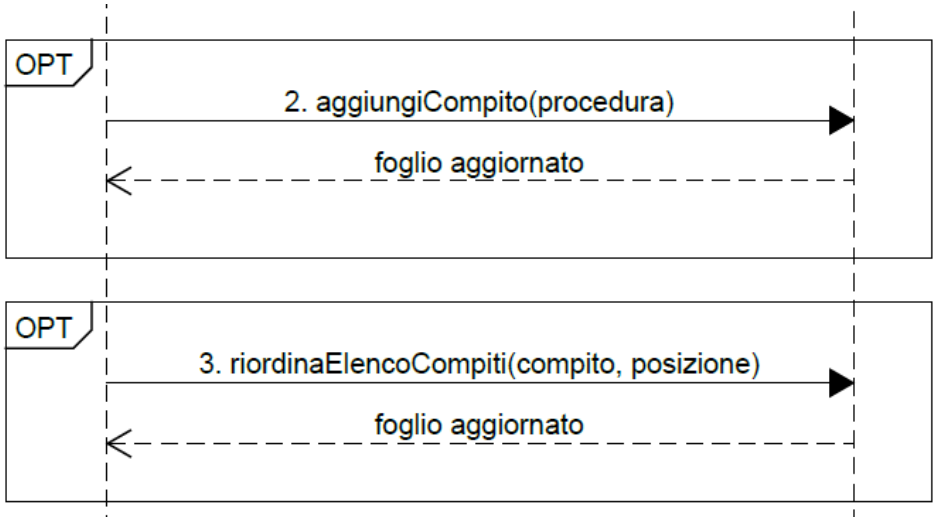
La frase successiva al primo indica la possibilità di terminare il caso d'uso in caso non si debba aggiungere nulla al foglio riepilogativo. Visto che tutti i passi successivi sono opzionali si può semplicemente assumere che in caso di terminazione non si eseguano i passi dal 2 in poi.
In caso si volesse rappresentare l'opzione di "uscita" dal caso d'uso potremmo utilizzare un'alternativa tra le risposte del passo 1:



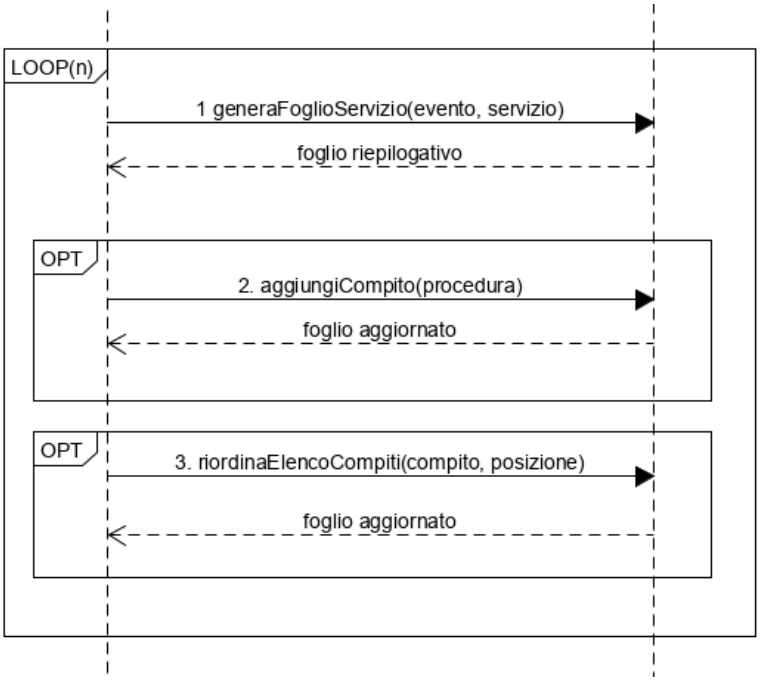
La prima alternativa rappresenta l'uscita dal caso d'uso, la seconda il proseguimento con i passi dal 2 in poi.

I passi 2,3 sono opzionali pertanto li rappresentiamo dentro frame OPT

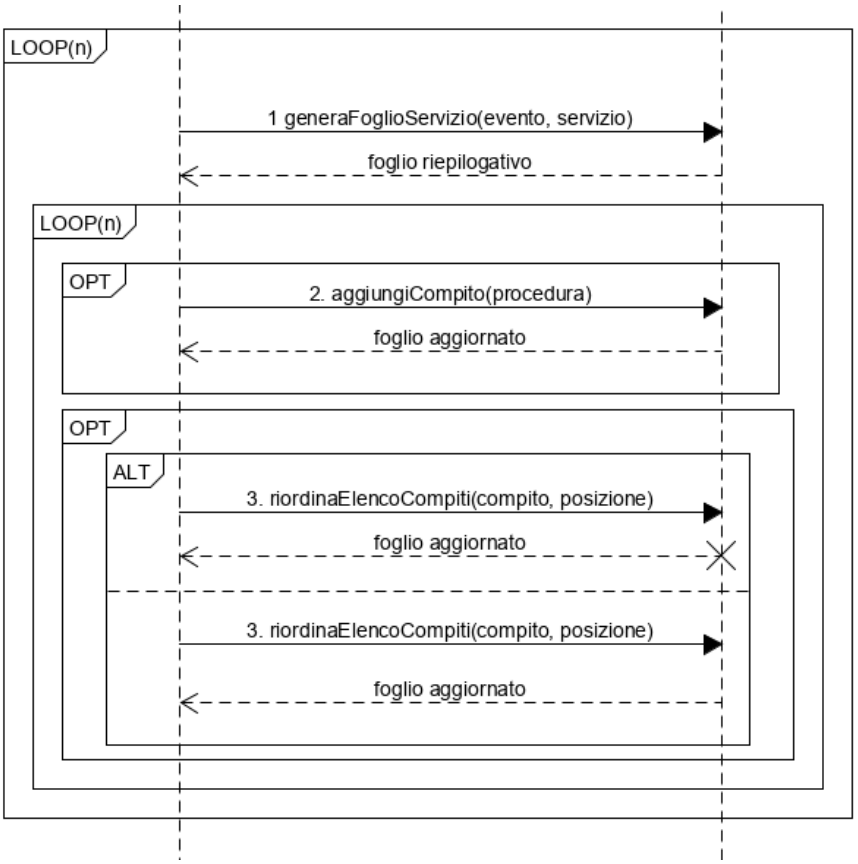
2	Opzionalmente aggiunge preparazioni e ricette all'elenco delle cose da fare
3	Opzionalmente, ordina l'elenco



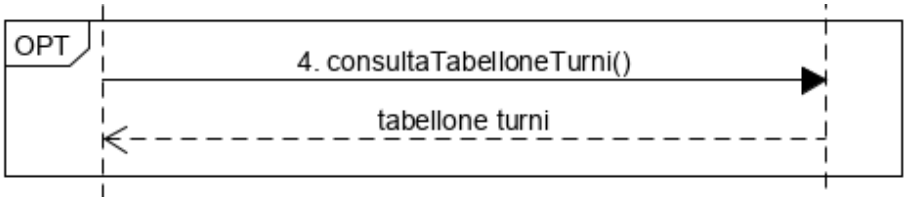
Si può decidere di aprire più fogli riepilogativi ripartendo dal passo 1. Questa possibilità viene rappresentata mettendo i passo dall'1 al 3 dentro un frame LOOP:



Questo loop che comprende i passi 1-3 sarà assente nel caso del SSD derivato dallo **scenario principale di successo III** che non prevede di lavorare su più fogli riepilogativi. Successivamente se non vuole assegnare dei compiti: i) torna al passo 2 o ii) termina il caso d'uso. Rappresentiamo queste opzioni con: i) un frame LOOP che include i passi 2 e 3 e ii) un'alternativa al passo 3 che rappresenta l'uscita dal caso d'uso.

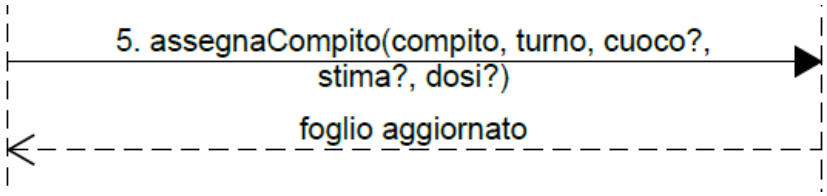


Il passo 4 è opzionale pertanto lo rappresentiamo dentro frame OPT

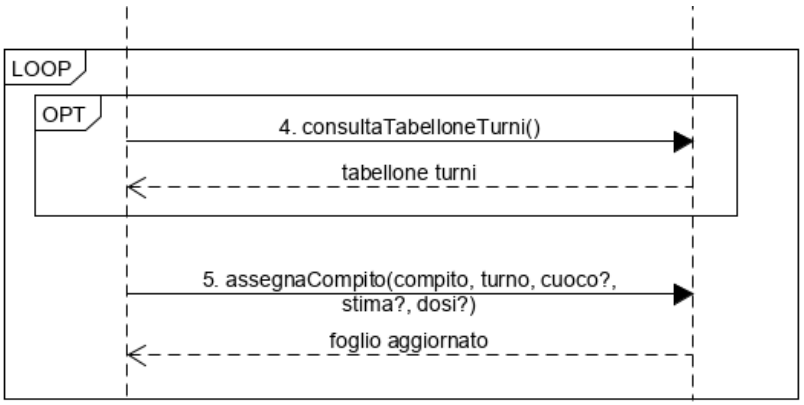


I passi 5 e 6 possono essere fusi in un unico evento con parametri opzionali

5	Assegna un compito specificando cosa (ricetta/preparazione), quando (turno) e opzionalmente chi (cuoco)
6	Opzionalmente, indica una stima del tempo richiesto per lo svolgimento del compito appena assegnato, e la quantità/porzioni preparate in un dato assegnamento



Visto che i passi 4-6 possono essere ripetuti (*Ripete dal passo 4 sinché non è soddisfatto*) mettiamo gli eventi corrispondenti in un frame LOOP



Gli scenari di successo III e VI generano due SSD che si differenziano solo per l'inclusione dell'evento di consultazione del tabellone turni (presente nel IV ma non nel III). In entrambi manca la possibilità di aprire più fogli riepilogativi (come già commentato precedentemente

questo si traduce nell'assenza del loop che include gli eventi 1-4). Infine il passo 4 dello scenario III e 5 dello scenario IV devono essere interpretati entrambi come una ripetizione dell'evento *assegnaCompito*

4 (III)	Assegna un compito specificando cosa (ricetta/preparazione), quando (turno) e opzionalmente chi (cuoco); opzionalmente specifica anche il tempo richiesto per lo svolgimento del compito appena assegnato, e la quantità/porzioni preparate in un dato assegnamento

5 (IV)	Assegna **i compiti** specificando cosa (ricetta/preparazione), quando (turno) e opzionalmente chi (cuoco); opzionalmente specifica anche il tempo richiesto per lo svolgimento del compito appena assegnato, e la quantità/porzioni preparate in un dato assegnamento