I contratti sono talmente legati ai SSD da un lato e al modello di dominio dall'altro che è pressoché impossibile presentarne una versione "possibile" che abbia senso rispetto ai vostri documenti. Per questa ragione il feedback che presentiamo in questo documento, più che mostrare uno o più esempi completi di soluzione, si concentrerà sui seguenti punti:

- una discussione (con esempi) degli errori più frequenti che abbiamo osservato sui Contratti; prendetela come una check list di cose da controllare... controllare di **non** averle fatte!
- un'analisi articolata di diverse possibilità riguardanti un'unica operazione, quella in cui si assegnano i compiti, prendendo in esame diverse possibili strutture di SSD e di modello di dominio; l'idea è quella di mostrare alcune differenze basilari di impostazione e come queste ricadono sulla stesura dei contratti. I vostri SSD e modello di dominio potrebbero essere diversi da tutti i nostri esempi, o magari contenere un po' dell'uno e un po' dell'altro; ciononostante che i punti sollevati nella discussione possono esservi utili se riuscite ad applicarli sul vostro contesto.
- qualche suggerimento sull'uso di attributi o associazioni derivabili, in abbinamento alle regole di business, per rendere più leggibili i contratti.
- qualche indicazione aggiuntiva sul ruolo dei parametri
- qualche indicazione su come descrivere i casi di fallimento delle operazioni

Errori frequenti

Nell'illustrare gli errori più comuni prenderemo come esempio i contratti dell'UC Gestire menù

1) usare una terminologia da programmatore per indicare i tipi degli attributi nel modello di dominio o dei parametri delle operazioni

   NO boolean, string -> creaMenu(titolo?: string)
   SI si/no, testo -> creaMenu(titolo?: testo)

2) nelle precondizioni dire quello che sta per succedere anziché descrivere il contesto pregresso

   2.definisciSezione(nome: testo)

   Pre-condizioni:
   - **NO** si vuole aggiungere una sezione a un Menù *m*
   - **NO** si vuole aggiungere una sezione
   - **SI** è in corso la definizione di un Menu *m*

3) dimenticare nelle precondizioni eventuali vincoli fra le istanze passate come parametri o fra quelle e altre istanze che pre-esistono all'operazione in sé

   4c.1 eliminaVoce(voce: Voce)

   Pre-condizioni:
   - **NO** nessuna precondizione
   - **NO** è in corso la definizione di un Menu *m*
     questa precondizione è corretta ma da sola non è sufficiente
   - **NO** è in corso la definizione di un Menu *m*, esiste voce

la prima parte questa precondizione è corretta ma da sola non è sufficiente, la seconda è inutile: non dobbiamo preoccuparci dell'esistenza dell'istanza voce passata come parametro

- **NO** è in corso la definizione di un Menu *m*, e voce **appartiene a** *m*
  questa precondizione è corretta ma incompleta perchè sappiamo che le voci possono appartenere al menù in 2 modi e qui ne viene rappresentato uno solo

- **SI** è in corso la definizione di un Menu *m*, e voce **appartiene a** *m* OPPURE esiste una Sezione *sez* tale che *m* **contiene** *sez* e voce **appartiene a** *sez*

4) non scrivere il contratto in modo autocontenuto, ossia menzionare istanze introdotte in altri contratti: i contratti di una operazione non possono parlare di istanze di oggetti concettuali introdotte in un'*altra* operazione. Qualunque istanza concettuale menzionata in un contratto deve essere introdotta o come parametro, o fra le precondizioni del contratto stesso. Per questo punto si rimanda alla discussione sotto nella sezione **I contratti per i SSD "modo 2".**

5) pensare che tutto ciò di cui si parla debba essere passato come parametro

SI 2.definisciSezione(nome: testo)

Pre-condizioni:

- è in corso la definizione di un Menu *m*

NO 2.definisciSezione(menu: Menù, nome: testo)

Per come abbiamo descritto lo scenario principale si presume che questa operazione avvenga subito dopo il passo 1 ovvero dopo la creazione o l'apertura di un menù e che quindi operi su tale menù. I contratti di una operazione però non possono parlare di istanze di oggetti concettuali introdotte in un'*altra* operazione. Dobbiamo usare le pre-condizioni, che servono proprio per caratterizzare il contesto in cui un'operazione può avvenire. Se passassimo un menù come parametro questo non farebbe necessariamente riferimento al menù creato o aperto nel passo precedente.

6) o al contrario non mettere fra i parametri qualcosa di cui si parla e che solo l'attore può comunicare al sistema

SI 5. inserisciVoce(ricetta: Ricetta, sezione?: Sezione, descrizione?: testo)

Pre-condizioni:

- è in corso la definizione di un Menu *m*
- [se è specificata una sezione] *m* **contiene** sezione

Post-condizioni:

- è stata creata un'istanza *v* di Voce
- *v* **fa riferimento** a ricetta
- [se non è specificata una sezione] *v* **appartiene a** Menu *m*
- [se è specificata una sezione] *v* **appartiene a** sezione
- [se è specificata una descrizione] *v.descrizione* = *descrizione*
  [altrimenti] *v.descrizione* = ricetta.nome

NO 5. inserisciVoce(ricetta: Ricetta)

Pre-condizioni:
- è in corso la definizione di un Menu *m*

Post-condizioni:
- è stata creata un'istanza *v* di Voce
- *v* **fa riferimento** a ricetta
  .....

nel caso in cui l'attore voglia mettere la voce in una sezione e/o specificare una descrizione questi oggetti devono essere passati come parametri, non vengono creati dal nulla, né passati magicamente da un'altra operazione

NO 5. inserisciVoce()

Pre-condizioni:
- è in corso la definizione di un Menu *m*

Post-condizioni:
- è stata creata un'istanza *v* di Voce
- *v* **fa riferimento** a **??**
- .....

come sopra. Se l'attore vuole creare una voce a partire da una ricetta questa deve essere passata come parametro dell'operazione.

7) Descrivere nelle post-condizioni le "azioni interne" del sistema durante l'operazione, o comunque raccontarne gli effetti in termini di cose che devono accadere anziché in termini di uno stato osservato

SI 5. inserisciVoce(ricetta: Ricetta, sezione?: Sezione, descrizione?: testo)

Pre-condizioni:
- è in corso la definizione di un Menu *m*
- [se è specificata una sezione] *m* **contiene** sezione

Post-condizioni:
- è stata creata un'istanza *v* di Voce
- *v* **fa riferimento** a ricetta
- [se non è specificata una sezione] *v* **appartiene a** Menu *m*
- [se è specificata una sezione] *v* **appartiene a** sezione
- [se è specificata una descrizione] *v.descrizione* = *descrizione*
  [altrimenti] *v.descrizione* = ricetta.nome

NO 5. inserisciVoce(ricetta: Ricetta, sezione?: Sezione, descrizione?: testo)

Pre-condizioni: (le precondizioni sono corrette!)

- è in corso la definizione di un Menu *m*
- [se è specificata una sezione] *m* **contiene** sezione

Post-condizioni:

Si crea/si deve creare un'istanza di voce.
La voce creata deve riferirsi alla ricetta passata come parametro.
Si inserisce/deve inserire la voce creata nell'array delle voci della sezione specificata.
Se la sezione non è specificata si inserisce la voce nell'array delle voci libere del Menù m.

Le post condizioni sopra sono scorrette perché, oltre a raccontare gli effetti in termini di cose che devono accadere anziché in termini di uno stato osservato, fanno riferimento a dettagli implementativi come l'inserimento della voce creata in un array. Questi dettagli non devono assolutamente comparire in questa fase.

8) Descrivere le postcondizioni discorsivamente anziché attenendosi al modello concettuale

NO 5. inserisciVoce(ricetta: Ricetta, sezione?: Sezione, descrizione?: testo)

Pre-condizioni: (le precondizioni sono corrette!)

- è in corso la definizione di un Menu *m*
- [se è specificata una sezione] *m* **contiene** sezione

Post-condizioni:

Si crea una voce che fa riferimento alla ricetta passata come parametro.
Se non è specificata una sezione la voce viene messa nel menù come voce libera altrimenti viene messa in sezione.
Infine se è specificata una descrizione viene associata alla voce al posto del nome della ricetta.

9) Non essere consistenti con la terminologia del modello di dominio, soprattutto di solito succede sulle associazioni

NO 5. inserisciVoce(ricetta: Ricetta, sezione?: Sezione, descrizione?: testo)

Pre-condizioni: (le precondizioni sono corrette!)

- è in corso la definizione di un Menu *m*
- [se è specificata una sezione] *m* **contiene** sezione

Post-condizioni:

è stata creata un'istanza *v* di Voce

- *v* si riferisce alla ricetta
- [se non è specificata una sezione] *v* viene messa nel Menu *m*
- [se è specificata una sezione] *v* viene messa nella sezione
- [se è specificata una descrizione] *v.descrizione* = *descrizione*
  [altrimenti] *v.descrizione* = ricetta.nome

Discussione su un esempio: il contratto dell'operazione con cui si assegna il compito

Per prima cosa bisogna considerare il modello di dominio, e in particolare la parte in cui viene descritto cos'è un "compito". Abbiamo già discusso in precedenza che assegnare un compito richiede di formare un terzetto: (i) una preparazione o ricetta, (ii) un cuoco, (iii) un turno. Abbiamo anche discusso come per formare una "associazione a tre" sia necessario introdurre un concetto apposito, che potremmo chiamare Compito o Attività o anche altro. Nel seguito assumeremo che si chiami Compito. Un'altra ipotesi che faremo è che si sia scelto di rappresentare le Ricette e le Preparazioni come specializzazioni di un concetto più generale di Mansione di Cucina; come abbiamo visto nella discussione sul modello di dominio questa non è l'unica soluzione possibile pertanto adattate quanto diremo al tipo di rappresentazione che avete scelto voi.

Ricordiamo che i contratti descrivono in sostanza il ciclo di vita delle [1]–istanze di classi concettuali, ossia quando "nascono", quando cambiano attributi/stato, quando si "associano" ad altri, e quando scompaiono: dal momento che il concetto di Compito è centrale a questo caso d'uso, può aver senso innanzitutto chiedersi quand'è che i Compiti iniziano ad esistere.
In particolare: **un compito (A) inizia ad esistere prima di essere assegnato, o (B) "nasce" quando viene assegnato?**
Sono possibili entrambe le opzioni.
Nel **caso A** probabilmente quando il Sistema precompila il Foglio Riepilogativo dell'Evento prescelto, e successivamente lo Chef opzionalmente aggiunge altre preparazioni, queste vengano già associate ciascuna a un'istanza di Compito, che per il momento risulta non assegnata né a un Cuoco né a un Turno.
Nel **caso B** invece il Foglio Riepilogativo probabilmente avrà soltanto all'inizio delle associazioni con Ricette e/o Preparazioni e/o loro generalizzazioni, e solo nel momento in cui una di queste viene scelta per essere assegnata "nasce" l'oggetto Compito che sarà assegnata ad un Turno ed opzionalmente ad un Cuoco.
Vale la pena di osservare che il modo in cui un Compito "nasce" influenza anche il modo in cui poi "scompare". Nel **caso A** eliminare un Compito significa eliminare anche il fatto che quella particolare Preparazione (o Ricetta, o generalizzazione di tali concetti) sia da fare. Potrà dunque essere utile distinguere l'eliminazione del Compito in quanto cosa da fare, e invece l'eliminazione del suo assegnamento, che significa soltanto "scollegarlo" dal Cuoco e dal Turno, ma non certo eliminarlo completamente. Nel **caso B** invece eliminare un Compito non fa perdere l'associazione fra il Foglio Riepilogativo e la Preparazione, dunque la sua eliminazione "toglie" l'assegnamento al Cuoco e al Turno, ma l'associazione con il Foglio Riepilogativo "ricorda" allo schef che c'è quella Preparazione (o Ricetta, o generalizzazione di tali concetti) e che va prima o poi assegnata.

Dal punto di vista dell'assegnamento dei compiti, ossia dell'operazione di cui ci vogliamo occupare, questo è rilevante perché nel **caso A** ci aspettiamo che le istanze di Compito esistano già prima

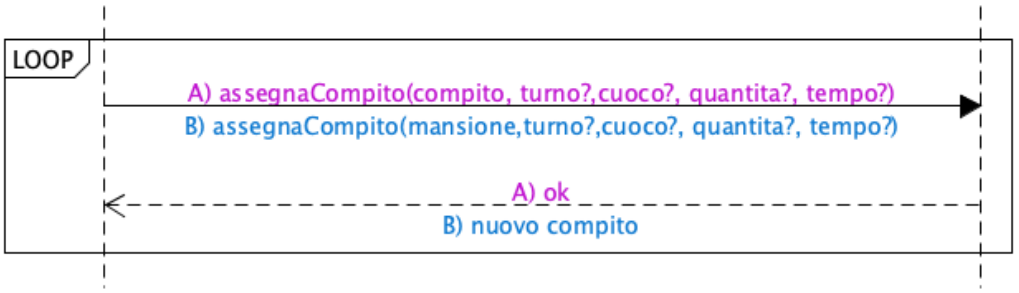
[1] Alcuni di voi hanno ipotizzato che le istanze di Compito – e forse anche il Foglio Riepilogativo – esistano addirittura da quando lo Chef sceglie il Menu per un dato Servizio, cosa che accade addirittura in un altro UC. Sebbene sia una soluzione un po' strana – ha degli aspetti da "scaricabarile" su un altro UC, mentre di solito si cerca di affrontare il più possibile le questioni rilevanti nei primi UC che sia analizzano – non è sbagliata, ma deve essere portata avanti coerentemente. Richiede infatti di avere precondizioni globali più estese, e di appuntare sull'UC a cui si demanda la creazione dei compiti che appunto questa responsabilità viene trasferita lì. Suggeriamo comunque, come esercizio, di provare a scrivere il contratto dell'operazione in cui si apre per la prima volta (o si genera) il foglio riepilogativo, anche per il caso in cui i compiti risultino creati proprio come effetto di questa operazione.

dell'operazione stessa, seppur non assegnate, e siano associate all'istanza di Foglio Riepilogativo (o qualche altro concetto analogo) su cui l'attore sta lavorando, mentre nel **caso B** ci aspettiamo che le istanze di Compito "nascano" e vengano associate al Foglio Riepilogativo proprio in seguito a questa operazione.

Questa riflessione su "vita morte e miracoli" delle istanze di Compito vuole un po' essere un esempio dei ragionamenti che bisogna fare quando si trattano i contratti. Bisogna capire, per quanto riguarda le classi concettuali più rilevanti, se le istanze "arrivano" da altri UC, ossia si può supporre che pre-esistano nel Sistema, o se iniziano ad esistere nel nostro di UC, e in tal caso quando. In questo caso ad esempio possiamo presumere che le istanze di Turno o di Cuoco pre-esistano: nel documento "Attori e titoli UC" possiamo vedere che ci sono dei casi d'uso specifici (che noi non tratteremo) sia per l'inserimento dei dati del personale sia per la creazione del calendario dei turni di utilizzo della cucina.
Un'altra riflessione utile è quella sulla "fine dell'esistenza" delle istanze di una certa classe concettuale. Ricordiamo che dire che un'istanza è eliminata implica il fatto che il Sistema la dimentichi completamente, e questo accade molto di rado. Molto più spesso quando l'utente chiede di "eliminare" qualcosa, chiede in realtà di eliminare un'associazione. Ad esempio nello UC che stiamo esaminando l'utente potrebbe voler eliminare una Preparazione dal Foglio Riepilogativo; questo implicherebbe eventualmente la rottura di una relazione e non certo una eliminazione "fisica" della Preparazione, che continua ad esistere nel Ricettario. Invece, come discusso prima, l'eliminazione di un Compito potrebbe avere senso, sia nel **caso A** che nel **caso B**, anche se con due significati un po' diversi.

Torniamo all'operazione specifica dell'assegnamento dei compiti. Prenderemo in esame tre diversi modi in cui l'assegnamento dei compiti potrebbe essere stato rappresentato nei vostri SSD. Attenzione, i diagrammi che seguono sono stralci di ipotetici SSD relativi allo scenario principale, e non riportano eventi precedenti o successivi all'assegnamento dei compiti. In tutti e tre i diagrammi abbiamo tenuto conto di possibili varianti a seconda che si sia ipotizzato il **caso A** o il **caso B**: ciò che si riferisce esclusivamente al caso A è rappresentato in viola, ciò che si riferisce esclusivamente al caso B è rappresentato in blu, ciò che è comune è rappresentato in nero.

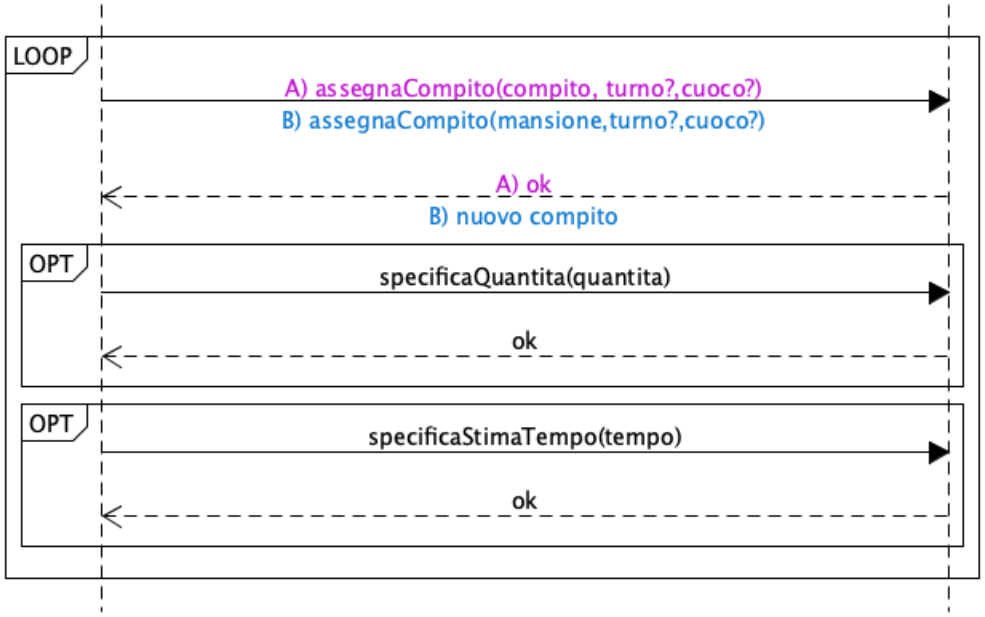
Modo 1



Nel "modo 1" l'assegnamento del compito avviene tramite un unico evento (e dunque un'unica operazione). Nel **caso A** si parte da un'istanza di Compito già esistente (e dunque già precedentemente associata ad una istanza di Mansione di Cucina, che potrà essere nello specifico una Ricetta oppure una Preparazione) mentre nel **caso B** l'istanza di Compito non esiste ancora quindi si dovrà specificare a quale istanza di Mansione di Cucina associarla nel momento in cui inizia ad esistere. Nel **caso A** al termine dell'operazione l'attore non riceve nulla di nuovo; semplicemente l'istanza di Compito che egli già aveva risulterà modificata. Invece nel **caso B** l'attore riceve una nuova istanza di Compito appena creata.

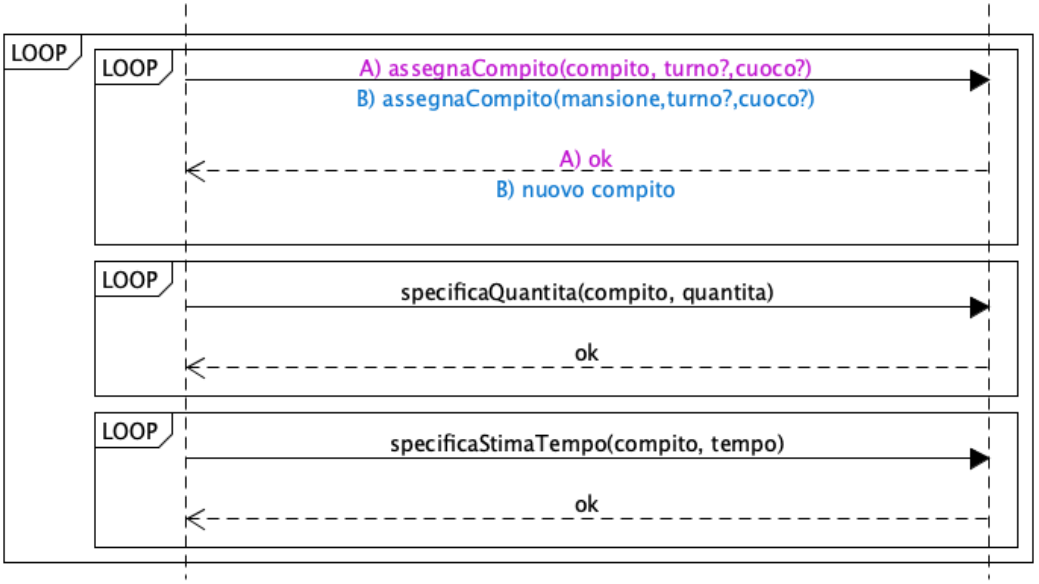
Quasi tutti i parametri di questa operazione sono opzionali: ci aspettiamo in generale che almeno uno sia valorizzato dall'utente, ma nessuno di essi è di per sé obbligatorio, tranne l'indicazione del Compito di cui si sta parlando nel caso A, o della mansione di cucina che si vuole assegnare nel caso B.

Modo 2



Nel "modo 2" l'assegnamento del compito avviene con tre eventi successivi, di cui il secondo e il terzo sono opzionali. Concettualmente l'interazione qui descritta è del tutto equivalente alla precedente: l'attore può assegnare tutti i compiti che vuole, e specificare opzionalmente turno, cuoco, quantità e tempo. Il messaggio che si manda scrivendo così il SSD è però che l'assegnamento debba avvenire in tre step successivi; sottolineando questo aspetto chi progetta la UI riceve il suggerimento di prevedere un'interazione in più fasi: l'utente sceglie un compito e opzionalmente un cuoco e un turno, **poi** eventualmente specifica una quantità, **poi** eventualmente specifica una stima del tempo. E' come prevedere tre moduli diversi per le tre attività, che l'utente riempie esclusivamente in sequenza, anche se può riempirli solo parzialmente o per nulla. Una UI fatta così è diversa da una in cui invece c'è un unico modulo di assegnamento in cui l'utente specifica quel che vuole e poi dà l'ok per l'assegnamento. Quindi possiamo vedere che il "modo 2" è un pochino meno generale perché dà un suggerimento specifico su una ben precisa modalità di interazione.
Con il "modo 1" chi progetta la UI è più libero di scegliere come raccogliere i dati (tutti in un colpo o step by step) per poi far scattare l'evento che dà il via all'assegnamento. Questo non vuol dire che il "modo 2" non sia accettabile, semplicemente è un po' più specifico sull'interazione fra Attore e Sistema, quindi va bene adottarlo se si vuole dare un'indicazione più precisa su come tale interazione debba avvenire. Se però non abbiamo idea di come vogliamo realizzarla, meglio il "modo 1" che è meno vincolante.
Notiamo che negli eventi successivi al primo in questo caso si dà per scontato che si stia ancora parlando del Compito assegnato con la prima operazione: c'è un processo sequenziale ben preciso e il Sistema ricorda quanto accaduto precedentemente.

Modo 3



Nel "modo 3" abbiamo ancora tre eventi separati, ma in questo caso sono completamente indipendenti. L'attore può assegnare un nuovo compito specificando opzionalmente cuoco e turno, oppure specificare la quantità di un compito esistente (non necessariamente quello che ha appena assegnato), oppure specificare la stima del tempo di un compito esistente. Anche qui c'è una descrizione un po' più specifica dell'interazione: ci immaginiamo una UI con tre moduli separati, ognuno con un suo evento, ma visibili contemporaneamente all'attore, che può decidere quale dei tre utilizzare in ciascun momento.
Anche in questo caso, questo modo di descrivere le cose va bene se si vuole entrare in maggior dettaglio sull'interazione; altrimenti sarà da preferire il "modo 1" che resta più generico. Attenzione: non stiamo dicendo che con il modo 2 o il modo 3 la UI dovrà **obbligatoriamente** essere fatta in un certo modo; anche se abbiamo descritto l'interazione nel modo 2 o 3 possiamo ugualmente decidere in seguito di far una UI con un modulo unico. Tuttavia se passiamo la nostra analisi dei requisiti ad un altro team per progettare la UI, il modo in cui raccontiamo il SSD per loro diventa un input per prendere delle decisioni. Non sono decisioni irreversibili, ma è conveniente per tutti se comunichiamo correttamente, senza involontariamente spingere i designer in una direzione che non era nostra intenzione suggerire e su cui il nostro cliente non si è espresso. In generale la UI è la parte del software su cui i clienti hanno maggiormente da ridire, perché è ciò che vedono: più riusciamo a progettare la parte "logica" del SW senza vincolare eccessivamente l'interazione, più sarà facile successivamente adattare la UI alle esigenze del cliente senza scomodare troppo il resto dell'applicazione.

Vediamo adesso come potrebbero essere i contratti delle operazioni nei modi 1, 2 e 3, e soprattutto quali sono le differenze fra uno e l'altro.

I contratti per i SSD "modo 1"

Nel modo 1 c'è una sola operazione (nelle due varianti caso A e caso B). Riportiamo il contratto, inframmezzato dai commenti evidenziati in azzurro.

```
A) assegnaCompito(compito: Compito, turno?: Turno, cuoco?: Cuoco,
                                                  quantita?: testo, tempo?: numero)
```

B) assegnaCompito(mansione: Mansione di Cucina, turno?: Turno, cuoco?: Cuoco, quantita?: testo, tempo?: numero)

Potrà forse sembrare strano che il parametro quantita sia di tipo testo; tuttavia il testo permette al nostro attore di scrivere "30 porzioni", "50 pezzi" o "10 kg" in tutta libertà.

Precondizioni:

- è in corso la gestione dei compiti relativi a un Foglio Riepilogativo *foglio*
- A) compito è **richiesto per** *foglio*
  B) mansione è **richiesta per** *foglio*

Quello che vogliamo dire è che per poter fare questa operazione l'attore deve aver scelto un'istanza di Foglio Riepilogativo su cui lavorare (ossia: deve avere iniziato il caso d'uso), e anche che ciò che vuole assegnare deve essere associato a questa stessa istanza di Foglio Riepilogativo. Poiché abbiamo bisogno di parlarne, l'istanza di Foglio Riepilogativo deve avere un nome (in questo caso, con molta fantasia, si chiama *foglio*). Se non fosse stata necessaria la seconda precisazione avremmo anche potuto scrivere più genericamente "è in corso la gestione dei compiti relativi a un Foglio Riepilogativo" senza necessariamente battezzarlo. Abbiamo qui ipotizzato che l'associazione fra Foglio Riepilogativo e Compito/Mansione di Cucina si chiami "**richiede**", quindi il Compito/Mansione di Cucina è **richiesto da** un Foglio Riepilogativo.

Postcondizioni:

Qui si vede bene la differenza fra caso A e caso B: nel caso A ci aspettiamo che al termine dell'operazione l'istanza compito su cui l'attore ha scelto di operare abbia per così dire "le associazioni e gli attributi al posto giusto"; nel caso B ci aspettiamo che sia stata creata una istanza di Compito che prima non c'era, e che sia questa nuova istanza ad avere "associazioni e attributi al posto giusto".

A) [se turno è specificato] compito si **svolge in** turno
   [se cuoco è specificato] compito è **eseguito da** cuoco
   [se quantita è specificato] compito.quantita = quantita
   [se tempo è specificato] compito.tempo stimato = tempo
B) E' stata creata un'istanza *comp* di Compito
   comp **è richiesto da** *foglio*
   comp **riguarda** mansione
   [se turno è specificato] *comp* **si svolge in** turno
   [se cuoco è specificato] *comp* è **eseguito da** cuoco
   [se quantita è specificato] *comp.quantita* = quantita
   [se tempo è specificato] *comp.tempo stimato* = tempo

Come si nota l'unica differenza sostanziale fra il caso A e il caso B è la presenza, nel caso B, delle prime due postcondizioni: "E' stata creata un'istanza *comp* di Compito" e naturalmente che *comp* **è richiesto da** *foglio*, o comunque si chiami la relazione fra un Compito e il Foglio Riepilogativo a cui si riferisce. Questa relazione (o una equivalente) deve esserci se vogliamo che sia possibile attribuire un Compito al Foglio Riepilogativo, e dunque al Servizio, corretto. Il resto delle postcondizioni è assolutamente simile fra i due casi, solo nel caso A si parla del compito ricevuto come parametro dall'attore, mentre nel caso B si parla dell'istanza *comp* che ci sia aspetta sia stata creata come effetto di questa operazione.

I contratti per i SSD “modo 2”

Nel modo 2 la prima operazione è una versione semplificata di quella vista nel modo 1, quindi il contratto sarà sostanzialmente analogo:

A) assegnaCompito(compito: Compito, turno?: Turno, cuoco?: Cuoco)

B) assegnaCompito(mansione: Mansione di Cucina, turno?: Turno, cuoco?: Cuoco)

Precondizioni:

- è in corso la gestione dei compiti relativi a un Foglio Riepilogativo *foglio*
- A) compito è **richiesto per** *foglio*
  B) mansione è **richiesta per** *foglio*

Postcondizioni:

A) [se turno è specificato] compito si svolge in turno
   [se cuoco è specificato] compito è **eseguito da** cuoco

B) E’ stata creata un’istanza *comp* di Compito
   comp **è richiesto da** *foglio*
   comp **riguarda** mansione
   [se turno è specificato] *comp* **si svolge in** turno
   [se cuoco è specificato] *comp* è **eseguito da** cuoco

Occupiamoci ora dell’operazione successiva, **specificaQuantita**. Abbiamo detto prima che nel modo 2 si presume che questa operazione avvenga subito dopo **assegnaCompito** e che operi sul compito che è stato individuato nell’operazione precedente. I contratti di una operazione però non possono parlare di istanze di oggetti concettuali introdotte in un’*altra* operazione.
Qualunque istanza concettuale menzionata in un contratto deve essere introdotta o come parametro, o fra le precondizioni del contratto stesso. Fanno eccezione solo le istanze concettuali introdotte nelle precondizioni globali. Attenzione: questo non è un invito a menzionare nelle precondizioni globali tutto ciò di cui volete parlare. Le precondizioni globali devono corrispondere alle precondizioni dell’intero UC descritte nella scheda di quest’ultimo.
Come possiamo dunque indicare che **specificaQuantita** lavora proprio su quel Compito particolare? Dobbiamo usare le pre-condizioni, che servono proprio per caratterizzare il contesto in cui un’operazione può avvenire.
Abbiamo detto che nel modo 2 le tre operazioni formano un processo unico, che potremmo chiamare “l’assegnamento di un compito”, che viene “iniziato” proprio dall’operazione **assegnaCompito**. Quindi il modo corretto di specificare che, perché **specificaQuantita** possa aver luogo, deve essere prima avvenuta **assegnaCompito**, è di richiedere nelle precondizioni che sia “in corso l’assegnamento di un Compito”. Se poi, come probabile, dovremo anche parlare di questo Compito specifico, dovremo battezzarlo, e richiederemo dunque che sia “in corso l’assegnamento di un Compito *comp*” (o *c*, o *compito*, l’importante quando battezziamo un’istanza è che il seguito sia leggibile, quindi consigliamo nomi un po’ esplicativi!).

---

specificaQuantita(quantita: testo)

Precondizioni:

- E’ in corso l’assegnamento di un Compito *comp*

Postcondizioni:

- *comp.quantita* = quantita

Potrebbe stupire che in questo caso non abbiamo scritto nelle precondizioni che "E' in corso la gestione dei compiti relativi a un Foglio Riepilogativo", come nell'operazione precedente. Come mai? In primo luogo, perché la precondizione che abbiamo scritto **implica necessariamente** che sia in corso la gestione dei compiti relativi a un Foglio Riepilogativo. Pensiamoci: se è in corso l'assegnamento di un Compito *comp*, allora deve essere avvenuta l'operazione **assegnaCompito**; ma allora le precondizioni di **assegnaCompito** devono essere vere e quindi deve anche essere in corso la gestione dei compiti di un Foglio Riepilogativo. E se avessimo avuto bisogno di parlare nel contratto del Foglio Riepilogativo, che in questo modo non viene menzionato? Potremmo parlare del "Foglio Riepilogativo *frep* che **richiede** *comp*". In altre parole possiamo "trovare" il Foglio Riepilogativo tramite l'associazione fra esso e il Compito di cui stiamo parlando, che nel nostro esempio si chiama **richiede/è richiesto da**.

Passiamo ora alla terza operazione, **specificaStimaTempo**. La situazione è simile a quella di **specificaQuantita**: infatti date le opzionalità **specificaStimaTempo** potrebbe aver luogo sia dopo **specificaQuantita** sia immediatamente dopo **assegnaCompito**. La verità è che ci interessa poco ai fini del contratto: quello che ci interessa è sapere di quale istanza di Compito si sta parlando, quindi, come nel caso precedente, che sia "in corso l'assegnamento di un Compito *comp*":

---

specificaStimaTempo(tempo: numero)

Precondizioni:

- E' in corso l'assegnamento di un Compito *comp*

Postcondizioni:

- *comp.tempo stimato* = tempo

I contratti per i SSD "modo 3"

Nel modo 3, questa parte del SSD inizia esattamente come nel modo 2, quindi per l'operazione **assegnaCompito** possiamo usare lo stesso contratto:

A) assegnaCompito(compito: Compito, turno?: Turno, cuoco?: Cuoco)

B) assegnaCompito(mansione: Mansione di Cucina, turno?: Turno, cuoco?: Cuoco)

Precondizioni:

- è in corso la gestione dei compiti relativi a un Foglio Riepilogativo *foglio*
- A) compito è **richiesto per** *foglio*
  B) mansione è **richiesta per** *foglio*

Postcondizioni:

A) [se turno è specificato] compito si svolge in turno
[se cuoco è specificato] compito è **eseguito da** cuoco

B) E' stata creata un'istanza *comp* di Compito
comp **è richiesto da** *foglio*
comp **riguarda** mansione
[se turno è specificato] *comp* **si svolge in** turno
[se cuoco è specificato] *comp* è **eseguito da** cuoco

Sono invece diversi i contratti per le operazioni **specificaQuantita** e **specificaStimaTempo**. Nel modo 3 abbiamo infatti visto che queste operazioni sono indipendenti da **assegnaCompito**: possono essere usate per specificare la quantità e la stima del tempo di un qualsiasi istanza di Compito indicata dall'attore (e infatti c'è un Compito come parametro), non necessariamente quella appena assegnata. Di conseguenza in questo caso non esiste alcun "processo di assegnamento dei compiti" fatto da tre operazioni; ci sono tre operazioni separate, una che assegna un Compito, e altre due che aggiungono informazioni su un Compito qualunque (nel caso A, dove le istanze di Compito esistono già da prima dell'assegnamento, questo potrebbe persino voler dire dare la stima del tempo o la quantità prima di aver assegnato il Compito a un Turno e a un Cuoco: ha senso? o vogliamo impedirlo con delle precondizioni ad-hoc? Ai nostri committenti sembra che abbia senso, ma sotto troverete anche le precondizioni che è necessario aggiungere se si vuole impedirlo)

---

specificaQuantita(compito: Compito, quantita: testo)

Precondizioni:

- E' in corso la gestione dei compiti relativi a un Foglio Riepilogativo *foglio*
- compito è **richiesto da** *foglio*

Ribadiamo che in questo caso sarebbe sbagliato scrivere "E' in corso l'assegnamento di un Compito compito": il SSD non descrive un "processo di assegnamento dei compiti"; l'assegnamento è fatto da una singola operazione e non è neanche detto, quando eseguiamo **specificaQuantita**, che tale operazione sia avvenuta e l'assegnamento dunque è bell'e che terminato. Se vogliamo richiedere che, per specificare la quantità, il Compito in questione abbia già un cuoco e/o un turno, possiamo aggiungere una o entrambe le precondizioni: "compito **si svolge in** un Turno" / "compito è **eseguito da** un Cuoco". Notiamo l'uso dell'articolo indeterminativo "un": non ci interessa quale sia la specifica istanza di Turno o Cuoco, ci interessa che ce ne sia una!

Postcondizioni:

- **compito.quantita = quantita**

Notiamo che in questo caso si parla del compito passato come parametro.

Per **specificaStimaTempo** valgono esattamente le stesse considerazioni di **specificaQuantita**:

---

specificaStimaTempo(compito: Compito, tempo: numero)

Precondizioni:

- E' in corso la gestione dei compiti relativi a un Foglio Riepilogativo *foglio*
- compito è **richiesto da** *foglio*

Postcondizioni:

- compito.tempo stimato = tempo

Quando l'operazione richiede un "elenco" di oggetti su cui lavorare

Può capitare che l'attore, per svolgere un'operazione, debba comunicare al sistema un "elenco" di oggetti. Ad esempio alcuni di voi hanno previsto che un compito possa svolgersi su più turni. Se prendiamo a titolo di esempio il **caso A** (non è difficile adattare l'esempio al **caso B**, se volete provarci), l'operazione di assegnamento del compito potrà diventare allora qualcosa del tipo:

`assegnaCompito(compito: Compito, insieme T di Turni, cuoco?: Cuoco)`
Come si può vedere, si dice che l'attore comunica al Sistema un "insieme" T di Turni (usiamo qui la maiuscola per il nome del parametro, perché è una collezione di oggetti e non un oggetto singolo). Avremmo potuto scrivere "lista" o "elenco", ma "insieme" è un termine preferibile in quanto più generico, non presuppone infatti un ordine particolare fra gli elementi in questione, e potrebbe anche essere vuoto. Quindi il consiglio è di usare "insieme" o anche "collezione" se non è rilevante l'ordine in cui l'attore indica questi oggetti, se invece l'ordine è rilevante, potete anche usare "lista" o "elenco", e potete anche scegliere di esplicitare gli elementi che compongono la liste/elenco in questo modo:
`assegnaCompito(compito: Compito, lista? t1,...,tN di Turni, cuoco?: Cuoco)`

Nella versione con un solo Turno (peraltro opzionale) le post-condizioni recitavano:

> `[se turno è specificato] compito si svolge in turno`

Come dobbiamo modificare questa indicazione nel caso dei turni multipli? Ci sembra che la soluzione più efficace sia la seguente:

> `compito si svolge in tutti e soli i Turni t appartenenti a I`

Stiamo così dicendo che la relazione "si svolge" esiste per tutti i turni nell'insieme specificato, e nessun altro. Perché è importante quest'ultima precisazione? Qui stiamo impostando i turni per la prima volta, e non ci aspettiamo che ci siano delle associazioni "si svolge" pre-esistenti. Ma se stessimo modificando un compito pre-esistente esso potrebbe essere già in relazione con alcuni Turni, e noi vogliamo dire che se un Turno non compare nell'insieme specificato dall'attore, l'associazione con quel Turno non ci sarà più.

Leggibilità dei contratti: associazioni e attributi derivabili + regole di business

Può capitare nello scrivere i contratti di pensare che potrebbe essere utile avere un'associazione o un'attributo che nel modello di dominio non c'è, e per quanto detto sinora non è il caso di aggiungere, in quanto sarebbe **derivabile** da associazioni o attributi già presenti, tramite qualche tipo di regola logica o di calcolo, e pertanto **ridondante**.

Talvolta però esistono associazioni (o attributi) derivabili che esprimono qualcosa che è sì calcolabile da altri elementi, ma tramite una regola o un calcolo abbastanza specifici, o complessi da descrivere, che, se espressi esplicitamente nei contratti, non fanno altro che minarne la leggibilità. In altre parole, escludere un'associazione o un attributo evidentemente "noto" ai nostri potenziali utenti, e per di più utile alla stesura dei contratti, solo perché è **derivabile**, sarebbe un po' autolesionista.

Facciamo un esempio concreto.
Abbiamo insistito – il committente ha insistito – perché quando l'attore apre per la prima volta il Foglio Riepilogativo di un Servizio in un dato Evento, tale Foglio Riepilogativo sia "pre-compilato" con i Compiti o le Mansioni di Cucina (a seconda che siamo nel caso A o B sopra descritti) che servono per preparare un dato Menu. Ora, noi **non** abbiamo nel nostro modello di dominio un'associazione:
Mansione di Cucina → **serve per** → Menu
In effetti, tale associazione sarebbe ridondante. Quali sono le Mansioni di Cucina che servono per un Menu?

- le Ricette a cui fanno riferimento le Voci "libere" del Menu
- le Ricette a cui fanno riferimento le Voci nelle Sezioni del Menu
- le Preparazioni che servono per Ricette che a loro volta servono per il Menu

Quindi possiamo esprimere il fatto che una Mansione serva per un dato Menu a partire da associazioni già presenti.

Le postcondizioni di un'ipotetica operazione **generaFoglioRiepilogativo(evento: Evento, servizio: Servizio)** [NOTA: la vostra potrebbe essere diversa e avere senso comunque!] verrebbero dunque più o meno così:

- è stato creato un Foglio Riepilogativo *fr*
- *fr* **si riferisce a** servizio
- dato il Menu *m* **usato in** servizio:
  - per ogni Ricetta *ric* tale che esiste una Voce *v* **appartenente a** *m* e che **fa riferimento a** *ric;*
    per ogni Ricetta *ric* tale che esiste una Voce *v* **appartenente** a una Sezione *sez* **contenuta in** *m*, e *v* **fa riferimento a** *ric*
    - (A) è stata creata un'istanza *c* di Compito, *c* **riguarda** *ric* e *c* **è richiesto da** *fr*
    - (B) *ric* **è richiesta da** *fr*
  - (A) per ogni Preparazione *prep* tale che esiste un Compito *comp* **richiesto da** *fr* che riguarda una Ricetta *ric* che **usa** *prep*
    - è stata creata un'istanza *c* di Compito, *c* **riguarda** *prep* e *c* **è richiesto da** *fr*

    (B) per ogni Preparazione *prep* tale che esiste una Ricetta *ric* **richiesta da** *fr* che **usa** *prep*
    - *prep* **è richiesta da** *fr*

Sebbene si possa anche scrivere una postcondizione così, si vede bene che è assai complicata da leggere – per capirla bisogna *studiarla*! Questo risulta un po' strano giacché l'idea che una Mansione di Cucina "serva" per un certo Menu è in realtà piuttosto intuitiva.

Quello che proponiamo in questo caso è di "spostare" un po' di questa complessità di lettura fra le cosiddette "regole di business" (di solito sono descritte in un documento a parte, ma noi per ora le abbiamo messe fra le note modello "post-it" nel modello di dominio).
Quindi:

- aggiungiamo nel modello di dominio un'associazione **derivabile** "serve per" fra Mansione di Cucina e Menu. I nomi di associazioni e attributi derivabili da altri elementi già presenti sono premessi dal simbolo /
  Quindi: Mansione di Cucina → **/serve per** → Menu
- specifichiamo con una regola di business come si deriva questa associazione (la complessità di cui sopra si sposta qui, ma dovendo scrivere una regola anziché una postcondizione il grado di complessità è leggermente inferiore):
  Una Mansione di Cucina *man* **/serve per** un Menu *m* se e solo se vale una delle seguenti condizioni:
  - Esiste una Voce *v* appartenente a *m* tale che *v* **fa riferimento a** *man*
  - Esistono una Sezione *s* e una Voce *v* tali che *m* **contiene** *s*, *v* **appartiene a** *s* e *v* **fa riferimento a** *man*
  - Esiste un'altra mansione *man2* che **/serve per** *m* tale che *man2* **usa** *man*

  Notiamo fra le altre cose che questa regola è definita in modo ricorsivo, cosa che in una postcondizione non si può fare (e in effetti questo fa sì che la postcondizione esemplificata sopra sia un po' imprecisa!)

Le post-condizioni dell'operazione ora potrebbero apparire così:

- è stato creato un Foglio Riepilogativo *fr*
- *fr* **si riferisce a** servizio
- dato il Menu *m* **usato in** servizio:
    - per ogni Mansione di Cucina *man* che **/serve per** *m*
        (A) è stata creata un'istanza *c* di Compito, *c* **riguarda** *man* e *c* **è richiesto da** *fr*
        (B) *man* **è richiesta da** *fr*

In questo modo il contratto risulta molto più leggibile.
Attenzione: ha senso operare in questo modo solo quando l'associazione (o attributo) esprime un concetto chiaro all'utente, oltre che "essere utile" per scrivere i contratti.

Vediamo velocemente un esempio di attributo – anziché associazione – derivabile, che potrebbe essere utile per i vostri contratti. Molti di voi hanno avuto bisogno di esprimere l'idea che un dato Turno sia passato, futuro, o in corso. Per poter parlare di queste caratteristiche di un turno è necessario rappresentarle tramite il modello di dominio. Naturalmente possiamo metterci a fare affermazioni sui rapporti tra data/ora del turno e la ora/data odierna, ma potrebbe essere più semplice inserire nel concetto di Turno un attributo derivabile **/stato temporale**: {passato, in corso, futuro}[2] e dare una regola di business che determina il valore di questo attributo in base ad altri attributi (data, ora di inizio, ora di fine) e alla data e ora corrente. Nei contratti potremo poi dire: "se *turno.stato temporale* = in corso…" etc etc.

Il ruolo dei parametri

Ricordiamo che i parametri rappresentano ciò che l'attore deve comunicare al sistema perché la sua richiesta possa essere soddisfatta.
Se indichiamo che un parametro è un testo stiamo dicendo che l'attore scriverà un testo (ad esempio in una TextBox). Se indichiamo che un parametro è si\no stiamo dicendo che l'attore avrà la possibilità di esprimere una scelta (ad es. tramite una check box o un menù a scelta multipla, o una finestra di dialogo "Ok/Annulla"…). Se indichiamo che un parametro corrisponde a un oggetto concettuale, stiamo dicendo che la UI rappresenterà in qualche modo gli oggetti concettuali (con delle icone, con delle voci di menù, con dei pulsanti) e l'attore avrà modo di indicare su quali di essi vuole operare (cliccandoci sopra, selezionandoli, ecc).

Se nei contratti di una operazione un parametro non viene mai menzionato, significa che non è realmente utile a quella operazione (perché magari esprime una informazione che dovrebbe essere già in possesso del Sistema, o che è derivabile da quelle che gli sono pervenute). Ricordiamo infatti che in questa modalità di analisi le operazioni si collocano all'interno di un processo e che dunque il sistema ha un suo stato interno che stabilisce in che punto del processo ci si trova.

Quando scriviamo nelle pre-condizioni qualcosa come:

- è in corso la gestione dei compiti relativi a un Foglio Riepilogativo *foglio*

stiamo in effetti dicendo che siamo in un punto del processo **successivo** all'apertura/creazione del *foglio*, e che il sistema è in uno stato per cui **sa qual è** il Foglio Riepilogativo su cui si sta lavorando (per l'appunto, *foglio*).

---

[2] E' possibile avere attributi i cui valori possibili sono elencati esplicitamente. E' un po' più semplice che avere una sequenza di attributi sì/no (anche se di fatto equivalente).

Potrebbero a questo punto sorgere delle domande sull'esempio presentato nella sezione precedente relativo alle regole di business, dove abbiamo introdotto un'ipotetica operazione iniziale, per la gestione dei compiti, descritta così:

generaFoglioRiepilogativo(evento: Evento, servizio: Servizio)

Nelle pre- e post-condizioni che abbiamo descritto l'oggetto evento non viene mai citato. E' a tutti gli effetti un parametro inutile. Peraltro, anche se servisse, possiamo risalire a quale sia l'evento partendo dal servizio.

C'è solo un motivo per cui potrebbe (se l'avete inserito) valere la pena lasciarlo: sottolineare che vogliamo che l'attore scelga prima un evento e poi, nel contesto di quell'evento, un servizio. E' in pratica un indizio che lanciamo a chi si occuperà della UI: non mostrare semplicemente all'attore un elenco di servizi, bensì un elenco di eventi ciascuno con i suoi servizi.

Se decidiamo di lasciarla così, dovremo curarci di esprimere, fra le pre-condizioni, che:

- `evento` **`prevede`** `servizio`

(o qualcosa di equivalente, a seconda di come avete chiamato le associazioni nel vostro modello di dominio.)

Questo per chiarire ulteriormente che non ci stiamo aspettando un Evento e un Servizio potenzialmente scollegati, ma che ci aspettiamo l'utente scelga un servizio nell'ambito di quelli previsti dall'evento.

Quando un'operazione fallisce

Le operazioni che descriviamo possono, in certe circostanze, fallire. Come lo esprimiamo nei contratti? Dipende un po' dal "tipo" di fallimento che vogliamo descrivere.

Fallimento come mancanza di premesse logiche

Se il fallimento è dovuto al fatto che mancano le premesse logiche di base perché l'operazione possa "partire", allora possiamo inserire una pre-condizione che richieda quelle premesse. Alcuni tipi di premesse logiche di base (facendo riferimento allo UC di Gestire i compiti) possono essere:

- un servizio non ha ancora associato un menu, non ha senso generare il foglio riepilogativo (né iniziare ad assegnare i compiti) per quel servizio
- l'attore ha aperto un foglio riepilogativo, non ha senso che cerchi di modificare un compito che non fa parte di quel foglio riepilogativo. Se considerate gli esempi sopra riportati di **assegnaCompito**, vedete che nelle pre-condizioni abbiamo richiesto che il compito sia **richiesto** dal *foglio* in corso di lavorazione.

L'indicazione che diamo in questo modo (sia al committente che dovrà verificare se i requisiti gli vanno bene, sia ai progettisti che dovranno poi realizzare l'applicazione) è che di fatto se **non** si verificano le precondizioni l'attore non deve neanche sapere che "avrebbe potuto fare quell'operazione, ma non può perché…". In mancanza delle precondizioni, l'operazione sarà come non esistesse.

Per chi dovrà progettare l'interfaccia grafica questo potrà significare, ad esempio:

- se è prevista un'area dell'applicazione dedicata esclusivamente ai fogli riepilogativi, in essa non mostrare nemmeno i servizi che non hanno un menù (alternativamente, se l'attore/Chef vede un elenco dei servizi a lui assegnati e può farci delle operazioni sopra, quando seleziona un servizio che non ha un menu l'operazione "Genera foglio riepilogativo" non è disponibile)
- quando l'attore apre un foglio riepilogativo, vede solo i compiti che ne fanno parte, quindi è impossibile che cerchi di modificare un compito "esterno"

Per quanto riguarda i progettisti della business logic (quelli che dovranno architettare e infine codificare le classi e i metodi che concorrono alla realizzazione dell'operazione descritta nei contratti), le pre-condizioni sono qualcosa da verificare prima di iniziare a svolgere le attività che compongono l'operazione, e che, se violate, daranno luogo ad un errore o a un'eccezione a livello globale di business logic, che segnali a chi ha dato il via all'operazione (tipicamente al modulo di UI) che quell'operazione, date le circostanze, neanche avrebbe dovuto essere chiamata.

Fallimento come richiesta non attuabile su certi oggetti (o certe combinazioni di oggetti) concettuali

Se il fallimento è dovuto al fatto che l'attore ha effettuato una richiesta che non è effettuabile perché i parametri specificati violano qualche regola di business, allora potrebbe essere più opportuno far sì che l'attore venga messo a conoscenza (a) dell'esistenza dell'operazione e (b) delle ragioni per cui su certi oggetti o per certi parametri inseriti quell'operazione non è disponibile. In altre parole vogliamo che l'attore venga a conoscenza del fatto che c'è una regola di business e che non gli è permesso di violarla.

Esempi di questo tipo possono essere:

- lo chef cerca di aprire il foglio riepilogativo di un servizio che non è di sua competenza
- lo chef cerca di assegnare un compito ad un cuoco che non ha dato disponibilità per il turno in cui si svolge il compito

(Se pensate che non sia così ovvio distinguere fra questi esempi e quelli del "Fallimento come mancanza di premesse logiche", pazientate ancora un poco e leggete il seguito…)

In questo caso possiamo descrivere la regola che non deve essere violata come una "guardia" sulle post-condizioni. Quindi ad esempio per **assegnaCompito** potremmo iniziare le post-condizioni con questa guardia:
[se cuoco **è disponibile in** turno]
e poi procedere con le post-condizioni.
Questo a livello di contratti dice che se la condizione non vale non si realizzeranno le post-condizioni. Nella maggior parte dei casi quando l'operazione fallisce dal punto di vista degli oggetti concettuali non succede proprio niente (succederà magari qualcosa a livello di UI). Naturalmente se per caso succedesse qualcosa (ad esempio, se in conseguenza del tentativo erroneo di assegnazione per punizione lo chef dovesse svolgersi il compito da solo) dovremmo descriverlo in un secondo caso, ossia dovremmo aggiungere alle post condizioni:
[se cuoco non **è disponibile in** turno]
compito **è eseguito da** *ch*

L'indicazione che diamo in questo modo è che se l'attore cerca di compiere l'operazione in violazione di questa regola, l'operazione avrà delle conseguenze diverse (o, nella maggior parte dei casi non ne avrà), pertanto l'attore dovrà essere avvisato della violazione effettuata affinché possa regolarsi diversamente.
Il modo più semplice a livello di UI è mostrargli un messaggio di errore in risposta alla sua richiesta; molti progettisti di UI tuttavia adottano dei segnali visivi che scoraggino o evitino del tutto che la richiesta "proibita" venga effettuata, senza però nasconderla come si farebbe nel primo caso.
Una scelta potrebbe essere ad esempio quella di mostrare, durante l'assegnamento dei compiti, i nomi di tutti i cuochi, mettendo un'indicazione grafica o testuale di "indisponibilità" accanto ai nomi di quelli che non sono appunto disponibili nel turno selezionato.

A livello di progettazione di business logic questo tipo di fallimenti possono essere rappresentati tramite eccezioni ma anche tramite specifici valori di ritorno dei metodo/funzioni. Infatti non è detto che ciò che

costituisce un “fallimento” dal punto di vista dell’attore lo sia anche dal punto di vista degli oggetti software. Se si utilizzano eccezioni, di solito sono eccezioni specifiche di un certo oggetto concettuale, che segnala quale regola di business è stata violata. Vedremo meglio questo discorso quando affronteremo la parte di progettazione.

Ma non è sempre chiaro con quale delle due tipologie di fallimento abbiamo a che fare…

In effetti è così, non è possibile tracciare una linea di demarcazione netta (anche se alcuni casi sono un po’ più chiari di altri). Una cosa che ci si può chiedere è: se lo tratto come un fallimento del primo tipo, e quindi impedisco agli utenti di accedere all’operazione, questo farà sorgere in loro delle domande? Prendiamo i due esempi visti qui sopra.
Se lo chef non vede nemmeno i fogli riepilogativi dei servizi altrui non cercherà di aprirli; non saprà nemmeno che esistono dunque difficilmente si farà domande.
Viceversa, se lo chef sa che Gigi è un cuoco dell’azienda di Catering, ma non trova il suo nome fra i cuochi che può scegliere, potrebbe pensare ad un errore… meglio segnalargli che non può sceglierlo per delle ragioni ben precise.

Un altro criterio utile è chiedersi se il sistema sia in possesso delle informazioni che servono per verificare che l’operazione sia attuabile senza cercare di attuarla. Nel nostro esempio di Cat&Ring questo è probabilmente sempre vero, ma pensate ad operazioni in cui il Sistema effettua una richiesta a un servizio esterno. Se solo il servizio esterno possiede le informazioni necessarie a effettuare la verifica delle condizioni, non possiamo verificarle a priori. Bisognerà aspettare di aver effettuato la richiesta e demandare la verifica al servizio invocato.
Solitamente anche se in fase iniziale di analisi dei requisiti non lavoriamo sul “come”, sappiamo già se il nostro sistema dovrà interagire con “fornitori” esterni. Ad esempio, pensate al sistema di carte di credito per i pagamenti. Finché non cerchiamo di effettuare il pagamento non possiamo sapere se la carta inserita è valida o meno, quindi “carta di credito in corso di validità” non potrebbe essere descritta come pre-condizione.

Ad ogni modo, purché siate coerenti nella descrizione del requisito e la portiate avanti correttamente quando passate alla progettazione, a livello didattico vanno bene entrambe le soluzioni, quindi non preoccupatevi di avere scelto “quella giusta”. Si tratta di sfumature che spesso hanno una componente soggettiva (che però va condivisa da tutto il team di sviluppo!).