Sviluppo delle Applicazioni Software

Laboratorio

a.a 2023/24

Lorenzo Bonincontro, Alberto Aiello

Gestione dei Turni

Informazioni generali

Nome caso d'uso: Gestione dei Turni
Portata: Sistema
Livello: Obiettivo utente
Attore primario: Organizzatore
Parti Interessate: Personale di servizio, Cuochi
Pre-condizioni: L'attore deve essere autenticato come organizzatore
Garanzie di successo o post-condizioni: I turni sono correttamente gestiti e aggiornati nel sistema.

Scenario principale di successo

#	Attore	Sistema
1.	Crea turno scegliendo tra Turno di Servizio e di cucina (assegna data, OrarioInzio, OrarioFine e **opzionalmente** luogo,oraInizioAgg e oraFineAgg).	Predispone turno (Cucina/servizio)
2.	Verifica le disponibilità.	Fornisce tabella disponibilità.
	Ripete dal passo 1 finchè non è soddisfatto *oppure* *Continua.*	
3.	Per i turni in cucina **opzionalmente** aggiunge dei raggruppamenti turni	Aggiorna specifiche turno.
	Ripete dal passo 1 finchè non è soddisfatto *oppure* *Ripete dal passo 3 finchè non è soddisfatto* *oppure* *Termina il caso d'uso.*	

Estensione 1a

#	Attore	Sistema
1a.1	Modifica turno esistente	Registra aggiornamento
	Continua dal passo 2	

Eccezione 1a

#	Attore	Sistema
1a.1	Crea **turno di cucina**, con conseguenza sovrapposizione con altri turni esistenti	L'attore tenta di creare turno di cucina, con conseguenza sovrapposizione con altri turni esistenti
	Termina il caso d'uso	

Eccezione 1a.1a

#	Attore	Sistema
1a.1a.1	Modifica dei turno esistente	L'attore tenta di modificare i turni con disponibilità assegnate.
	Termina il caso d'uso	

Eccezione 1a.1b

#	Attore	Sistema
1a.1b.1	Modifica dei turno esistente	L'attore tenta di modificare turni con date antecedenti alla data corrente
	Termina il caso d'uso	

Eccezione 1a.1c

#	Attore	Sistema
1a.1c.1	Modifica dei **turni di cucina** esistente,	L'attore tenta di modificare turno di cucina

	con conseguenza sovrapposizione con altri turni esistenti	esistente, con conseguenza sovrapposizione con altri turni esistenti
	Termina il caso d'uso	

Estensione 1b

#	Attore	Sistema
1b.1	Cancella turno esistente	Registra aggiornamento
	Continua dal passo 2	

Eccezione 1b.1a

#	Attore	Sistema
1b.1a.1	Cancella turno esistente	L'attore tenta di cancellare turni con date antecedenti alla data corrente
	Termina il caso d'uso	

Estensione 1c

#	Attore	Sistema
1c.1	creaRipetizione	Registra aggiornamento
	Continua dal passo 2	

Estensione 1d

#	Attore	Sistema
1d.1	ModificaRipetizione	Registra aggiornamento
	Continua dal passo 2	

Eccezione 1d.1a

#	Attore	Sistema
1d.1a.1	ModificaRipetizione	L'attore tenta di modificare i turni con disponibilità assegnate.
	Termina il caso d'uso	

Eccezione 1d.1b

#	Attore	Sistema
1d.1b.1	ModificaRipetizione	L'attore tenta di modificare turno (di cucina) esistente, con conseguenza sovrapposizione con altri turni esistenti
	Termina il caso d'uso	

Estensione 1e

#	Attore	Sistema
1e.1	cancellaRipetizione	Registra aggiornamento
	Continua dal passo 2	

Eccezione 1e

#	Attore	Sistema
1e.1a.1	cancellaRipetizione	L'attore tenta di cancellare turni con date antecedenti alla data corrente
	Termina il caso d'uso	

Estensione 3a

#	Attore	Sistema
3a.1	Modifica raggruppamenti	Registra aggiornamento
	Continua dal passo 1 oppure *Continua dal passo 3 oppure* *Termina il caso d'uso.*	

Estensione 3b

#	Attore	Sistema
3b.1	Modifica raggruppamenti	Registra aggiornamento
	Continua dal passo 1 oppure *Continua dal passo 3 oppure* *Termina il caso d'uso.*	

Eccezione 3a.1a

#	Attore	Sistema
3a.1a.1	Modifica un raggruppamento	L'attore tenta di modificare raggruppamento turni con disponibilità assegnate.
	Termina il caso d'uso	

Estensione 3c

#	Attore	Sistema
3c.1	creaRipetizioneRaggruppamento	Registra aggiornamento
	Continua dal passo 1 oppure *Continua dal passo 3 oppure* *Termina il caso d'uso.*	

Estensione 3d

#	Attore	Sistema
3d.1	modificaRipetizioneRaggruppamento	Registra aggiornamento
	Continua dal passo 1 oppure *Continua dal passo 3 oppure* *Termina il caso d'uso.*	

Eccezione 3d.1a

#	Attore	Sistema
3c.1a.1	modificaRipetizioneRaggruppamento	L'attore tenta di modificare i turni con disponibilità assegnate.
	Termina il caso d'uso	

Estensione 3e

#	Attore	Sistema
3d.1	EliminaRipetizioneRaggruppamento	Registra aggiornamento
	Continua dal passo 1 oppure *Continua dal passo 3 oppure* *Termina il caso d'uso.*	

Sezione
nome: testo

contiene
{ordered}
0..n
1

Una voce puo' comparire in un menu' all'interno di una sezione oppure come voce "libera" quindi verrà valorizzata una e una sola delle due relazioni appartiene

0..1
appartiene
{ordered}
0..n

Menù
titolo: testo
pubblicato: sì/no
richiede cuoco:si/no
piatti caldi: si/no
buffet: si/no
finger food: si/no

{ordered}
appartiene
0..1
0..n

Voce
descrizione: testo

0..1
0..n
0..n
0..n
fa riferimento a
1

Un menu é in uso in un evento se e solo se é in uso in uno dei servizi previsti dall'evento stesso

in uso in
0..n
in uso in
0..n

Evento
informazioni

Proprietario di

1
0..n
assegnato a
prevede
0..1

Chef

1

Ricetta

Preparazione

0..n
usa

Procedura

0..n
si trova in
0..n
1

Ricettario

istruzioni da seguire

1..n
0..n
1
contiene

Servizio

1
riferito a
1
è effettuato in

Foglio Riepilogativo

0..1
contiene
1
{ordered}

viene svolto
1..n
0..n

0..n
0..n
0..n

Turno di Servizio
oraInizioAgg:numero
oraFineAgg:numero

Turno di Cucina
al completo: sì/no

appartiene
2..n
0..1

Raggrupamento

Un Raggruppamento può essere:
singolo o ricorrento(che si ripete)

Personale di Servizio

Cuoco

assegnato
0..1
0..n

Compito
Stima del tempo richiesto: lasso di tempo
Porzioni: Numero

Compito:
questo concetto identifica una “tripletta”:
cosa (ricetta o preparazione),
chi (cuoco) e
quando (turno).

0..n
appartiene
1

Ripetizione

appartiene
0..1
1..n

0..n
contiene
1

appartiene
0..n
1

Personale

0..n
si rende disponibile in
0..n

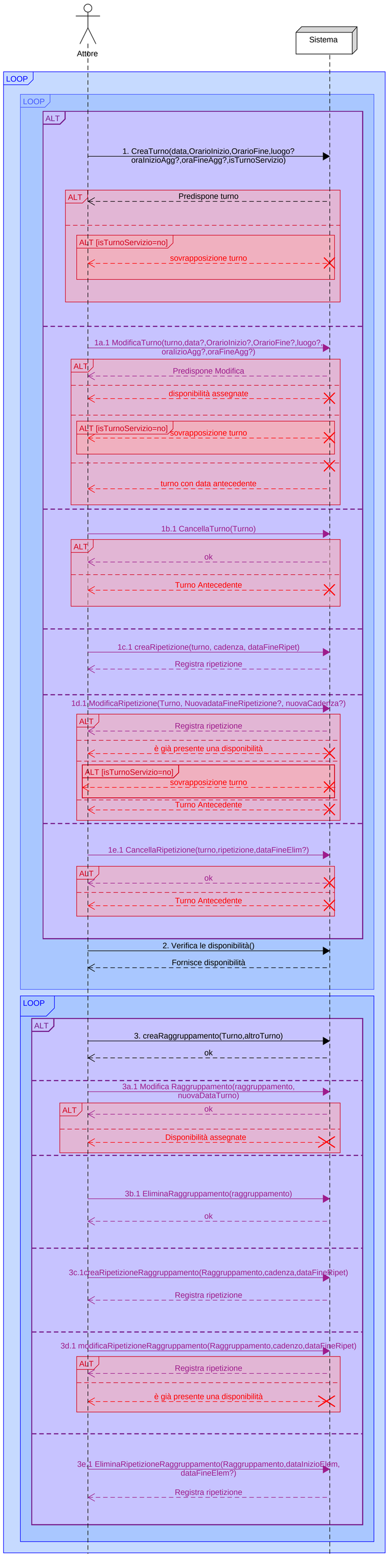
Turno
orarioInizio : ora
orarioFine : ora
data:data
luogo
isServiceTurn:si/no

fa parte
0..n
1

tabellone dei Turni

0..n
crea
1

Organizzatore


Attore
Sistema
LOOP
LOOP
ALT
1. CreaTurno(data,OrarioInizio,OrarioFine,luogo?
oraInizioAgg?,oraFineAgg?,isTurnoServizio)
ALT
Predispone turno
ALT [isTurnoServizio=no]
sovrapposizione turno
1a.1 ModificaTurno(turno,data?,OrarioInizio?,OrarioFine?,luogo?,
oralizioAgg?,oraFineAgg?)
ALT
Predispone Modifica
disponibilità assegnate
ALT [isTurnoServizio=no]
sovrapposizione turno
turno con data antecedente
1b.1 CancellaTurno(Turno)
ALT
ok
Turno Antecedente
1c.1 creaRipetizione(turno, cadenza, dataFineRipet)
Registra ripetizione
1d.1 ModificaRipetizione(Turno, NuovadataFineRipetizione?, nuovaCaderza?)
ALT
Registra ripetizione
è già presente una disponibilità
ALT [isTurnoServizio=no]
sovrapposizione turno
Turno Antecedente
1e.1 CancellaRipetizione(turno,ripetizione,dataFineElim?)
ALT
ok
Turno Antecedente
2. Verifica le disponibilità()
Fornisce disponibilità
LOOP
ALT
3. creaRaggruppamento(Turno,altroTurno)
ok
3a.1 Modifica Raggruppamento(raggruppamento,
nuovaDataTurno)
ALT
ok
Disponibilità assegnate
3b.1 EliminaRaggruppamento(raggruppamento)
ok
3c.1creaRipetizioneRaggruppamento(Raggruppamento,cadenza,dataFineRipet)
Registra ripetizione
3d.1 modificaRipetizioneRaggruppamento(Raggruppamento,cadenzo,dataFineRipet)
ALT
Registra ripetizione
è già presente una disponibilità
3e.1 EliminaRipetizioneRaggruppamento(Raggruppamento,dataInizioElim,
dataFineElem?)
Registra ripetizione

Contratti per lo UC “Gestire I Turni”

Pre-condizione generale:
L’attore è identificato con un’istanza *Org di Organizzatore*
Esiste un Tabellone *tab*

1. creaTurno(data: Data, oraInizio: Numero, oraFine: Numero, luogo?:Testo,oraInizioAgg?:numero,oraFineAgg?:numero, isTurnoServizio: si/no)

Pre-Condizioni:
- -
Post-Condizioni:
[se isTurnodiServizio= no]
per ogni turno *tr* in *tab*:
tr.data , tr.oraInizio , tr.oraFine e tr.luogo è stato controllato che non si sovrappongono con data, oraInizio, oraFine, luogo
allora:
è stata creata un’istanza *tr* di Turno:
tr.data = data.
tr.oraInizio = oraInizio.
tr.oraFine = oraFine .
[se specificato un luogo] tr.luogo=luogo.
[se non è specificato un luogo] tr.luogo = cucina.
tr è inserito in *tab*.
altrimenti:
[se isTurnodiServizio= si]
è stata creata un’instanza di *tr* di Turno
tr.data = data.
tr.oraInizio = oraInizio.
tr.oraFine = oraFine .
[se specificato un luogo] tr.luogo=luogo.
[se non è specificato un luogo] tr.luogo = cucina.
[se è stato specificato oraInizioAgg]tr.oraInzioAgg=oraInzioAgg.
[se è stato specificato oraFineAgg] tr.oraFineAgg=oraFineAgg.
tr è inserito in *tab*.

1a.1 modificaTurno(tr:Turno,data?: Data, oraInizio?: Numero, oraFine:Numero,luogo?:Testo,oraInizioAgg?:numero, oraFineAgg?:numero)

Pre-Condizioni:

Esiste turno *tr*
è in corso la modifica di un turno *tr*
Non è presente **disponibilità** in *tr*
Post-Condizioni:
[se tr.isTurnoServizio=no]
per ogni turno *tr* in *tab*:
tr.data , tr.oraInizio,tr.oraFine e tr.luogo è stato controllato che diverso con data, oraInizio, oraFine, luogo
allora:
[se è specificato una data] tr.data = data.
[se è specificato un oraInizio] tr.oraInizio = OraInizio.
[se è specificato un oraFine] tr.oraFine = OraFine.
[se è specificato un luogo] tr.luogo = luogo.
tr aggiornato in tab

[se tr.isTurnoServizio=si]
[se specificato oraInzioAgg] tr.oraInzioAgg=oraInzioAgg.
[se specificato oraFineAgg] tr.oraFineAgg=oraFineAgg.
[se è specificato una data] tr.data = data.
[se è specificato un oraInizio] tr.oraInizio = OraInizio.
[se è specificato un oraFine] tr.oraFine = OraFine.
[se è specificato un luogo] tr..luogo = luogo.
tr aggiornato in *tab*.

1b.1 cancellaTurno(tr:Turno)

Pre-Condizioni:
Esiste Turno *tr*
Post-Condizioni:
tr è stato rimosso da *tab*
l'istanza *tr* è eliminata

1c.1 creaRipetizioneTurno(tr:Turno, cadenza: testo, dataFineRipetizione: numero)

Pre-Condizioni:
Esiste Turno tr
Post-Condizioni:
[se tr.isTurnoServizio=si]
è stata creata un'istanza *ripet* di Ripetizione.
per ogni cadenza fino a dataFineRipetizione:
viene creata un'istanza trNuovo di Turno
trNuovo.orarioInzio=tr.oraInizio
trNuovo.oraFine=tr.oraFin

trNuovo.luogo = tr.luogo
trNuovo.data viene modificata in accordo a cadenza
[se specificato tr.oraInizioAgg]trNuovo.oraInizioAgg= tr.oraInizioAgg
[se specificato tr.oraInizioAgg]trNuovo.oraInizioAgg= tr.oraInizioAgg
trNuovo **appartiene** a *ripet*

ripet è stato inserito in *tab*

[se IsTurnoServizio=no]

per ogni *tr* **appartenente** a *tab*:

è stato controllato che per ogni *trNuovo* siano diversi data , oraInizio, oraFine , luogo
(viene verificato che per ogni trNuovo non si sovrappone con altri turni in tab)

allora:

è creata un'istanza *ripet* di Ripetizione.

per ogni cadenza fino a dataFineRipetizione:

viene creata un'istanza *trNuovo* di Turno
trNuovo.orarioInzio=tr.oraInizio
trNuovo.oraFine=tr.oraFine
trNuovo.luogo = tr.luogo
trNuovo.data viene modificata in accordo a cadenza
trNuovo **appartiene** a ripet

ripet è stato inserito in *tab*

1d.1modificaRipetizioneTurno(tr:Turno, NuovadataFineRipetizione?: Data, nuovaCadenza?: Testo)

Pre-Condizioni:

esiste turno *tr* in *tab*
esistono *tr* che **appartiene** a un'istanza di *ripet*
Non è presente **disponibilità** in *tr*

Post-Condizioni:

[se tr.isTurnoServizio=si]

[se è stata specificata una nuovaCadenza:]

[se è stata specificata una NuovaDataFineripetizione]

per ogni turno ripetuto *trp* **appartenente** a *ripet* con trp.data > tr.data
è stato aggiornato trp.data in base alla nuovaCadenza fino alla NuovaDataFineRipetizione
sono state eliminate tutte le istanze di *trp* con trp.data > NuovaDataFineripetizione

[se non è stata specificata una NuovaDataFineripetizione]

per ogni turno ripetuto *trp* **appartenente** a *ripet* con trp.data > tr.data
è stato aggiornato trp.data in base alla nuovaCadenza

[se non è stata specificata una NuovaCadenza]

[se è stata specificata una NuovaDataFineRipetizione]

[se NuovaDataFineRipetizione < tr.DataFineRipetizione]
sono state eliminate tutte le istanze di *trp* con trp.data >
NuovaDataFineripetizione
[se NuovaDataFineRipetizione > tr.DataFineRipetizione]
sono state create istanze di *trNuovo* fino all NuovaDataFineRipetizione
rispettando la cadenza
i turni nuovi **appartengono** a *ripet*

[se tr.isTurnoServizio=no]
[se è stata specificata una nuovaCadenza:]
per ogni turno *tr* in *tab* per ogni turno ripetuto *trp* trp.data calcolato
con nuovaCadenza deve essere diverso da tr.data
[se è stata specificata una NuovaDataFineripetizione]
per ogni turno ripetuto *trp* **appartenente** a ripet con trp.data > tr.data
è stato aggiornato trp.data in base alla nuovaCadenza fino alla
NuovaDataFineRipetizione
sono state eliminate tutte le istanze di *trp* da *ripet* con trp.data >
NuovaDataFineripetizione

[se non è stata specificata una NuovaDataFineripetizione]
per ogni turno ripetuto *trp* **appartenente** a *ripet* con trp.data > tr.data
è stato aggiornato trp.data in base alla nuovaCadenza
[se non è stata specificata una NuovaCadenza]
[se è stata specificata una NuovaDataFineRipetizione]
[se NuovaDataFineRipetizione < tr.DataFineRipetizione]
sono state eliminate tutte le istanze di *trp* con trp.data >
NuovaDataFineripetizione
[se NuovaDataFineRipetizione > tr.DataFineRipetizione]
sono state create istanze di *trNuovo* fino all NuovaDataFineRipetizione
rispettando la cadenza
i turni nuovi **appartengono** a *ripet*

1e.1 cancellaRipetizioneTurno(tr:Turno, dataFineElim?: numero)

Pre-Condizioni:
tr **appartiene** ad una ripetizione *ripet*
Post-Condizioni:
[se è stata specificata dataFineElim]
per ogni istanza turno ripetuto *trp* in ripet, con tr.data<=trp.data
per ogni trp **appartenente** a ripet trp.data<=dataFineElim
trp non **appartiene** più a *ripet*.
tr è stato eliminato .
[se non è stata specificata dataFineElim]
per ogni istanza turno ripetuto *trp* in *ripet*, con tr.data<=trp.data
tr è stato eliminato
trp non **appartiene** più a *ripet*.

2. verificaDisponibilità()

Pre-Condizioni: --
Post-Condizioni: --

3. creaRaggruppamento(tr:Turno, altroTr:Turno)

Pre-condizioni:
Esiste turno *tr* e altro turno *altroTr*
tr diverso da *altroTr*
tr.isTurnoServizio=no e altroTr.isTurnoServizio=no
Post-condizioni:
[se tr.isTurnoServizio=no e altroTr.isTurnoservizio=no]
è stata creata un'istanza *rag* di Raggruppamento
tr **appartiene** a *rag*
altroTr **appartiene** a *rag*

tr e *altroTr* vengono rimossi da *tab*
ragg è stato inserito in *tab*

3a.1modificaRaggruppamento(ragg:Raggruppamento,nuovaDataTurno:data)

Pre-condizioni:
esiste *ragg* **appartenente** a *tab*
Post-condizioni:
per ogni Turno *t* **appartenente** a *ragg*
per ogni turno *tr* in *tab* tr.data è stato controllato non sia uguale a nuovaDataTurno
t.data= nuovaDataTurno

3b.1 EliminaRaggruppamento(rag:Raggruppamento)

Pre-condizioni
esiste *ragg* **appartenente** a *tab*

Post-condizioni:
per ogni *tr* **appartenente** a *rag*:
tr non appartiene più a *rag*
tr è stato eliminato
rag è stato rimosso da *tab*
rag è eliminato

3c.1 creaRipetizioneRaggruppamento(ragg: Raggruppamento, cadenza: testo, dataFineRipetizione: numero)

Pre-condizioni:
esiste *ragg* **appartenente** a *tab*
Post-condizioni:
per ogni *tr* **appartenente** a *tab:*
per ogni turno raggruppato *trg* **appartenente** a *ragg*:
di cui si devono creare le istanze in *ripet*, trg.data calcolato per ogni cadenza fino alla
dataFineRipetizione non è stata trovata corrispondenza con tr.data
allora:
è stata creata un'istanza *ripet* di Ripetizione.
sono state create istanze turno ripetuto *trp* in *ripet* che seguono la cadenza e la
dataFineRipetizione.
per ogni *trp* in *ripet*:
trp.OraInizio = trg.oraInizio
trp.OraFine = trg.oraFine
trp.location = trg.location
ragg **appartiene** a *ripet*
ripet **appartiene** a *tab*

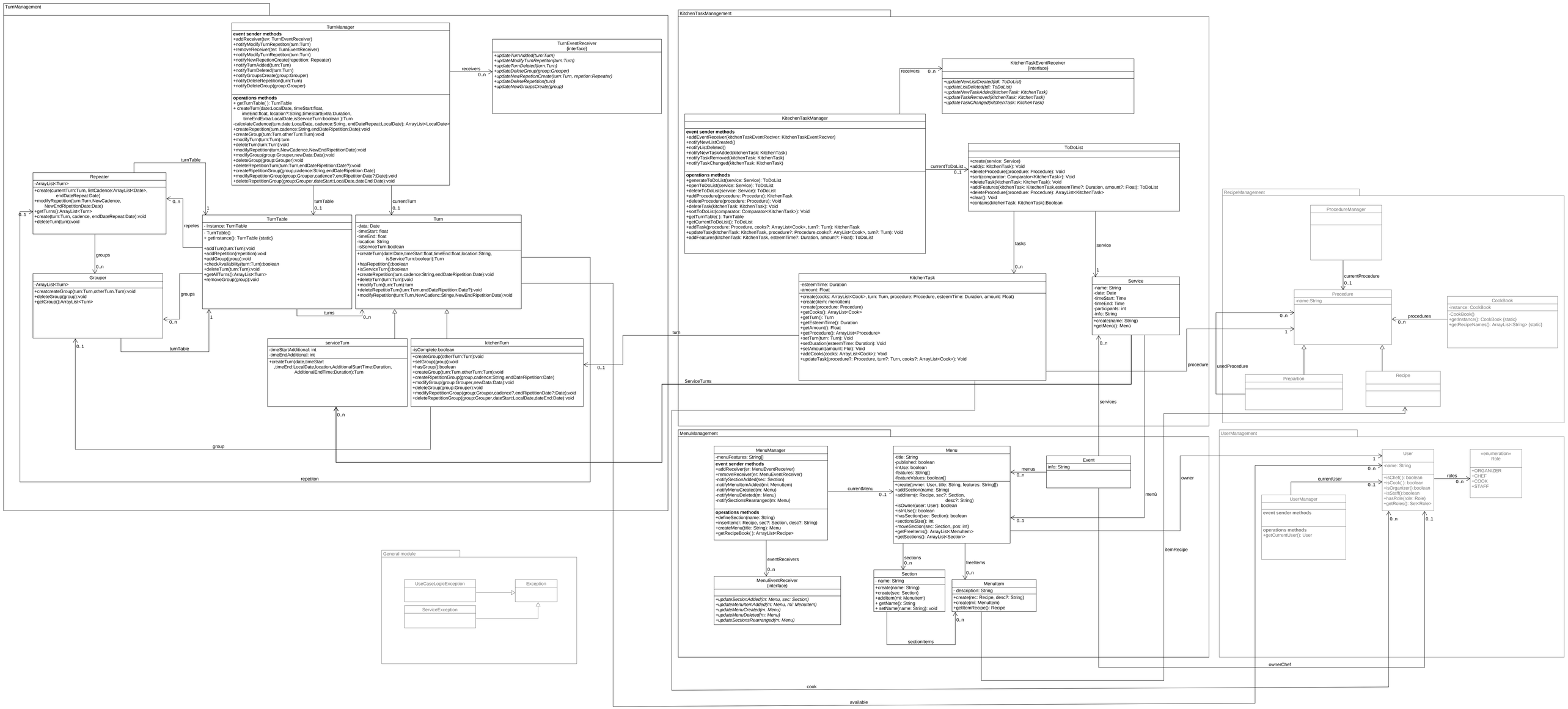
3d.1 modificaRipetizioneRaggruppamento(ragg: Raggruppamento, cadenza: testo, dataFineRipet: numero)

Pre-condizioni:
Esiste turno *tr*
tr **appartiene** a *ragg*
Post-condizioni:
per ogni istanza *tr* **appartenente** a *ragg* **appartenente** a *ripet*:
tr viene modificato in accordo alla nuova cadenza e dataFineRipetizione.

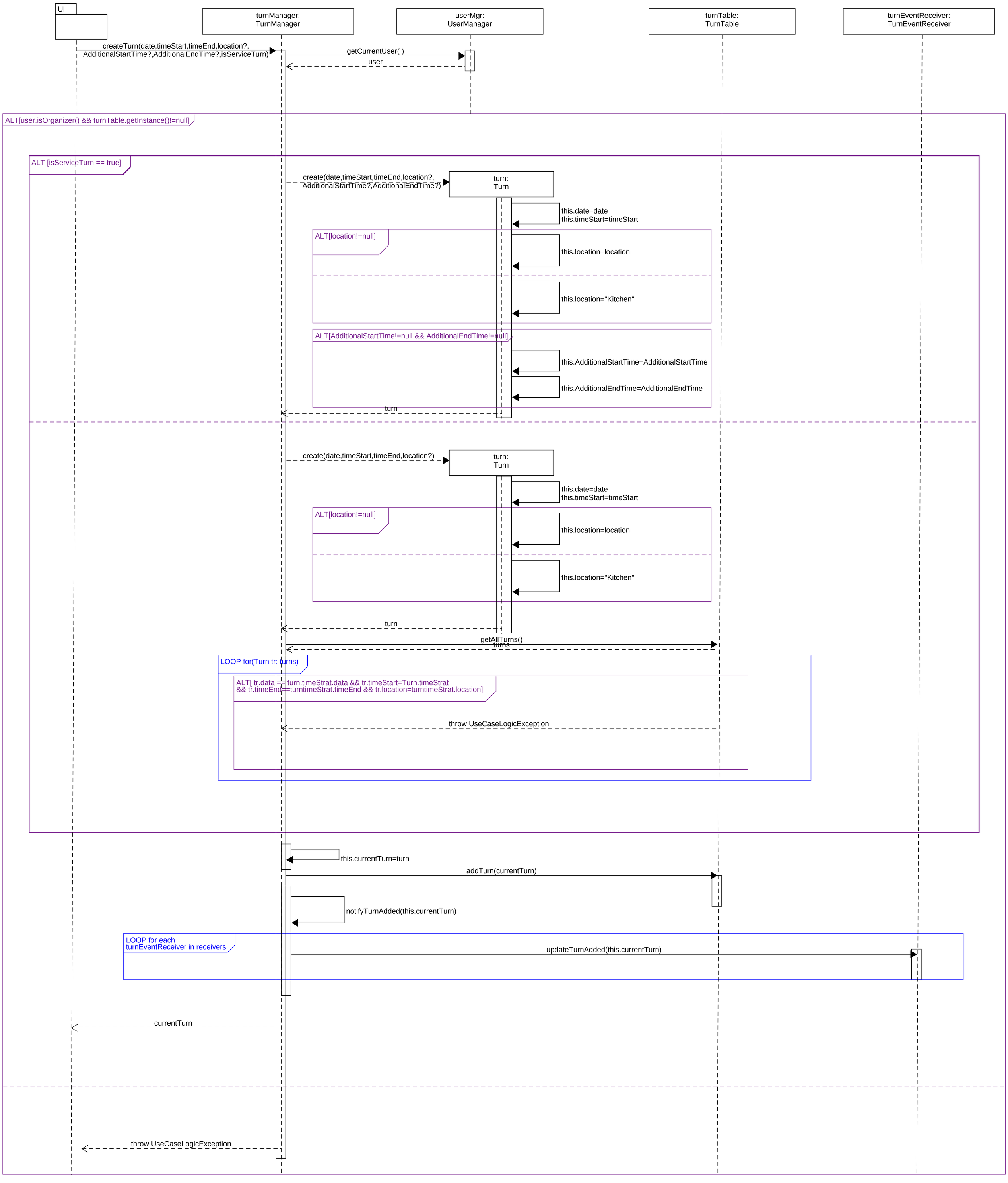
3e.1 eliminaRipetizioneRaggruppamento(ragg: Raggruppamento, dataInizioElim:numero, dataFineElim?: numero)

Pre-condizioni:
Esiste Turno *tr*
tr **appartiene** a *ragg*
ragg **appartiene** a *ripet* di Ripetizione
Post-condizioni:
[Se specificato dataFineElim]*:*
per ogni istanza *tr* **appartenente** a *ragg*, con tr.data>=dataInizioElim e tr.data<= dataFineElim:
tr non appartiene più a *ragg*
tr è stato eliminato
ragg è eliminato
ripet è stato rimosso da tab
ripet è eliminato
[Se non specificato dataFineElim]*:*
per ogni istanza *tr* **appartenente** a *ragg*, con tr.data>=dataInizioElim:
tr non appartiene più a *ragg*
tr è stato eliminato
ragg è eliminato
ripet è stato rimosso da tab
ripet è eliminato

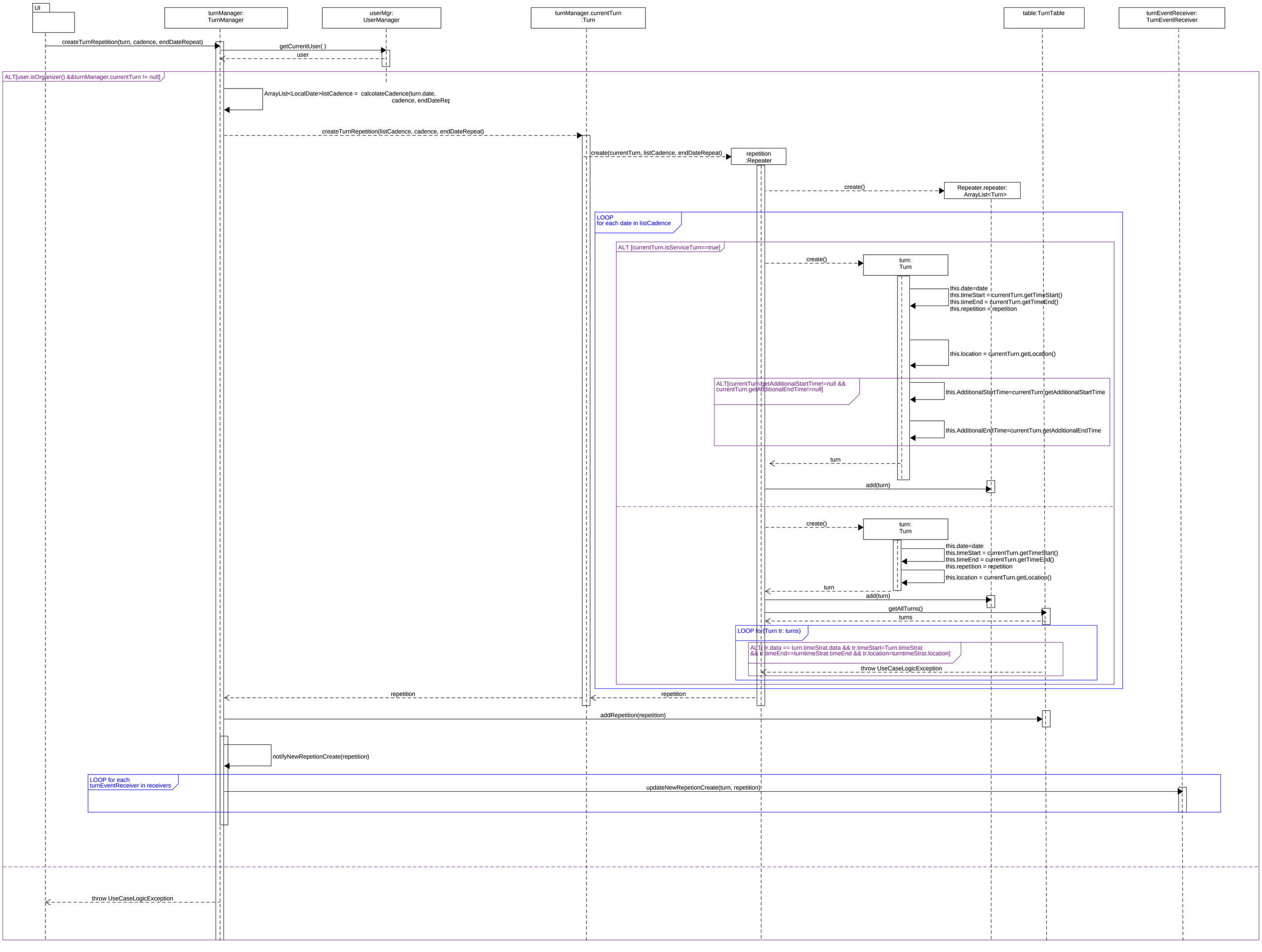
Design Class Diagram (DCD)



1 createTurn

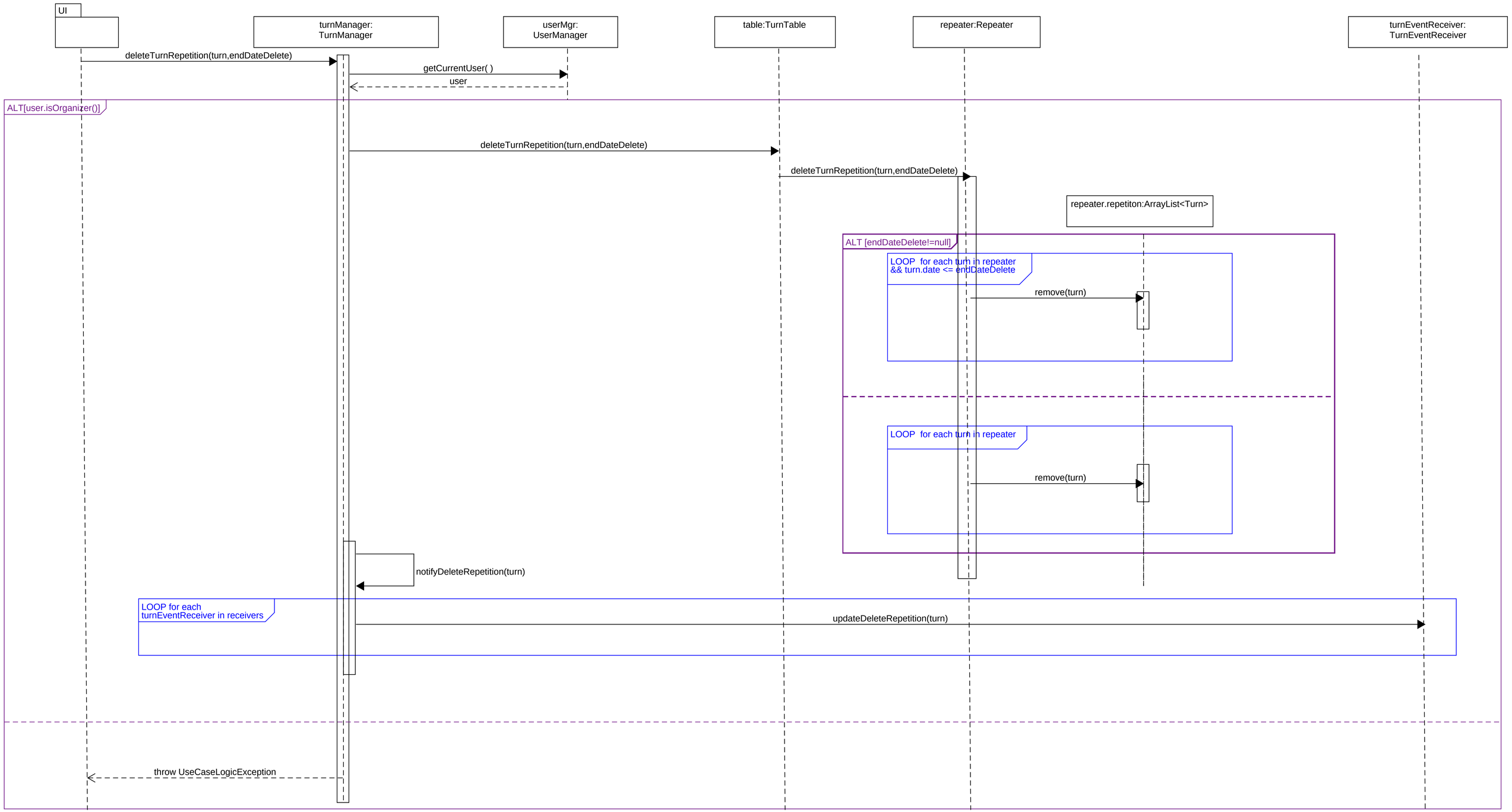


1c.1 createTurnRepetition

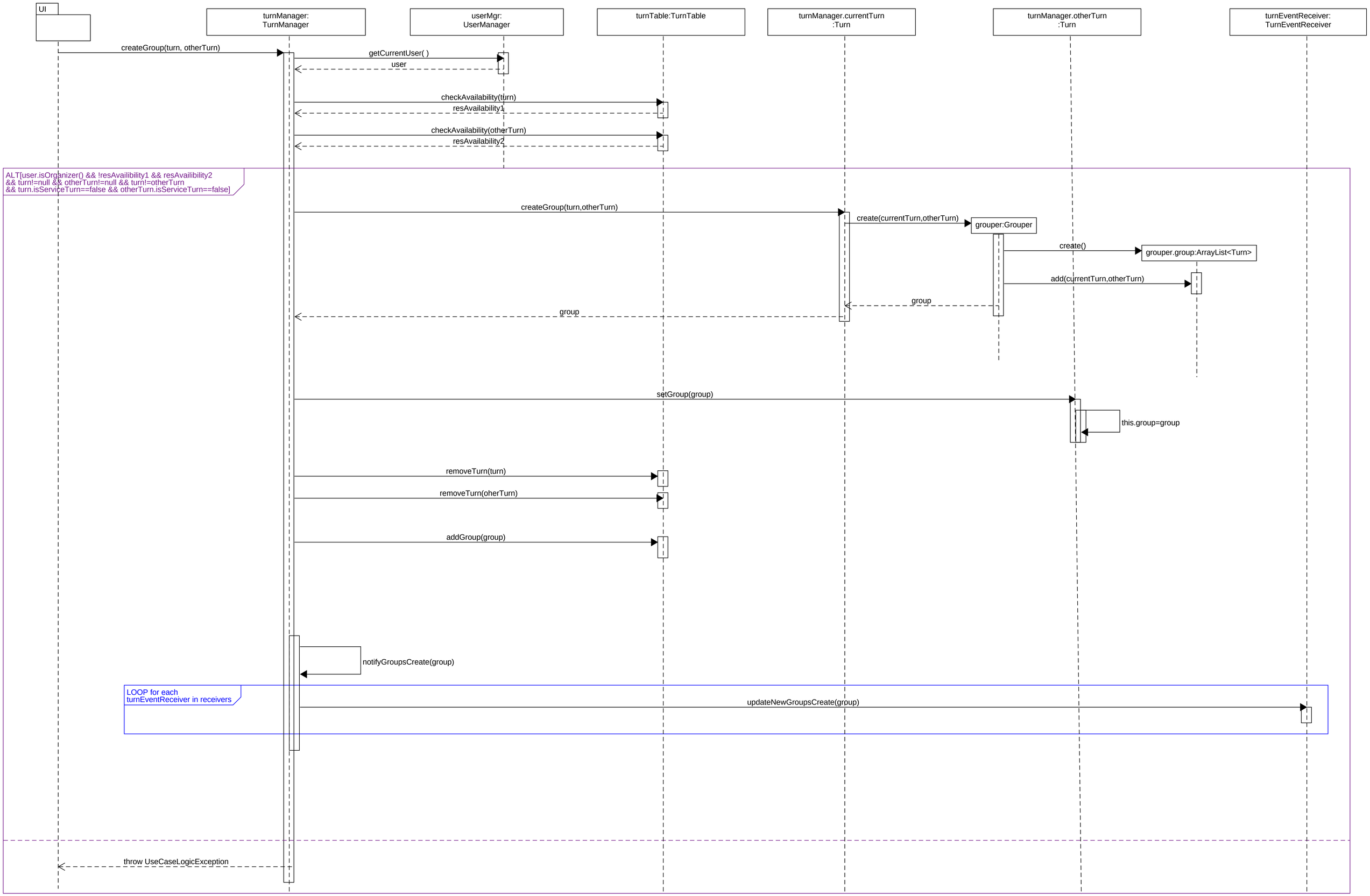


1d.1 modifyTurnRepetition

1e.1 deleteTurnRepetition



3. createGroup



3a.1 deleteGroup