Software Engineering

Module: Model Based Software Engineering

Progetti Modelica - Requisiti progetto Prodigit

AY 2020/21

Enrico Tronci
Computer Science Department, Sapienza University of Rome
Via Salaria 113 - 00198 Roma - Italy

tronci@di.uniroma1.it

Version: 2020-12-29

1 Outline

Il progetto *Prodigit* proposto dal Prof. Francesco Parisi-Presicce per la sua parte del corso può anche essere utilizzato per lo sviluppo del progetto Modelica relativo alla parte del corso svolta dal Prof. Enrico Tronci.

In questo documento si riporta il progetto *Prodigit* (Sezione 2) e si danno indicazioni su un suo possibile sviluppo come progetto Modelica (Sezione 3).

Si noti che si tratta solo di un *possibile* sviluppo del progetto. Potete senz'altro proporre diverse impostazioni del progetto.

Per ogni dubbio potete inviarmi una email: tronci@di.uniroma1.it.

2 Testo progetto Prodigit

Progettare un sistema di prenotazione di posto a lezione, nel rispetto della (nuova) capienza delle aule (caratterizzate da un codice).

La presenza degli studenti è organizzata in turni prenotabili in base alle ultime due cifre del numero di matricola (da 00 a 49 e da 50 a 99) a settimane alterne.

La prenotazione per ciascun turno può essere fatta (e cancellata, per permettere a studenti in lista d'attesa di frequentare) solo dal lunedì a venerdì della settimana precedente.

Al termine della prenotazione, il sistema produce una ricevuta (con aula, matricola ed orario) da scaricare/salvare.

Per accedere al sistema servono le credenziali di Sapienza.

2.1 Da consegnare

- Analisi di contesto
- Specifica dei requisiti

- Analisi
  - Nomi/verbi
  - Schede CRC
  - Classi di analisi (stereotipi entity/boundary/control)
  - Organizzazione in package
- Progetto
  - Specifica dell'architettura
  - Classi di progetto, sottosistemi, componenti
  - Realizzazione dei casi d'uso (sequence diagrams)
  - Cicli di vita di classi significative

3 Possibili requisiti per Modelica

Per quanto riguarda il progetto Modelica il focus è sull'architettura di sistema e l'interazione tra i sottosistemi.

Come al solito dobbiamo individuare le quattro componenti che definiscono il sistema:

- Environment (cioè tutti i possibili casi d'uso o scenari operativi)
- Modello del sistema
- Requisiti funzionali
- Requisiti non funzionali

3.1 Environment

L'environmnet consiste dei seguenti elementi il cui comportamento non è controllabile dal nostro software.

- Aule: in ogni istante di tempo un aula può cambiare stato e passare da agibile ad inagibile o viceversa. Se si vuole si può anche considerare il caso in cui diventi parzialmente inagibile, cioè dimuisce la sua capienza covid. Il sistema prodigit acquisisce tale informazione su richiesta attraverso il GOMP.
- Studenti: uno studente può prenotarsi, cancellare la sua prenotazione, etc.

Come al solito si può modellare ogni elemento dell'environment con una Catena di Markov.

3.2 Modello del sistema

Il sistema prodigit interagisce con il GOMP dal quale riceve le informazioni relative alle aule (e.g., agilbità, capienza covid, etc).

È importante modellare questa interazione poichè un disallineamento con le informazioni del GOMP potrebbe portare ad usare un aula inagibile ovvero lasciare inutilizzate aule agibili.

Intuitivamente, fare il download dei dati GOMP molto spesso garantisce un allineamento, ma potrebbe essere inefficiente. Farlo troppo di rado, potrebbe generare un disallineamento. Fare il check ad ogni richiesta di prenotazione potrebbe compromettere la disponibilità del sistema prodigit poiché un malfunzionamento del GOMP si ripercuoterebbe su prodigit.

L'architettura del sistema definirà un bilanciamento tra queste esigenze contrastanti.

Per la parte Modelica potete ignorare le procedure di login e concentrarvi solo sulle operazioni di lettura/scrittura da parte degli utenti, sulla sincronizzazione dei dati con il gomp e sull'interazione con il mondo esterno (mi pare solo aule e studenti). In breve, potete limitarvi a considerare solo gli aspetti rilevanti ai fini dell'interazione tra sistemi diversi (incluso l'ambiente).

3.3 Requisiti funzionali

Alcuni esempi di requisiti funzionali:

- *Safety*: Non fare mai *overbooking* (cioè, prenotare per un aula un numero di studenti superiore alla capienza covid dell'aula).
- *Liveness*: Non rifiutare una prenotazione se c'è posto nell'aula.

3.4 Requisiti non-funzionali

Alcuni esempi di requisiti non funzionali:

- Si vuole che prodigit sia *down* al più per il 80% del tempo per cui il GOMP è *down*. Quindi, ad esempio, se il GOMP è down per un ora al giorno, prodigit potrà essere down al più per 48 minuti al giorno. Si noti la similarità d questo requisito con quello relativo alla availability del software per la clinica pschiatrica descritto nel libro di testo.
- Il tempo per effettuare una prenotazione non deve essere superiore ad un minuto. Questo esclude una sincronizzazione con il GOMP ad ogni richiesta di prenotazione poichè una failure del GOMP bloccherebbe tutto il sistema prodigit.