UNIVERSITÀ DEGLI STUDI DI BERGAMO
Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

Esercitazione su AMDD: iterazione 0 e 1

Corso di laurea Magistrale in Ingegneria Informatica

PROGETTAZIONE, ALGORITMI E COMPUTABILITÀ (38090-MOD1)

RELATORE
Prof.ssa Patrizia Scandurra

SEDE
DIGIP

DATA
04-10-2021

Esercizio 1: Iterazione 0 AMDD (*requirements and architecture envisioning*)

Si consideri un **sistema software per la vendita di biglietti** (vedi specifiche uetente nella slide successiva)

- Partendo dalle specifiche utente e dal documento di analisi, modellare un diagramma dei casi d'uso iniziale del sistema
  (non è necessario in prima istanza descrivere i passi dei casi d'uso)
- Fornire in notazione *free-style* o in UML (diagramma UML a componenti e di deployment) una scomposizione iniziale dell'architettura del sistema in sottosistemi/componenti e la loro allocazione/distribuzione su una topologia hardware.

Specifiche utente

- Si vuole realizzare un **software per la gestione della vendita di biglietti** di teatro e per la gestione della cassa. Tale software è utilizzato solo dagli operatori di cassa e non è pertanto previsto nessun sistema per l'acquisto dei biglietti tramite Internet.
- Il sistema deve permettere agli operatori di inserire e consultare il calendario degli eventi e di emettere i biglietti.
- L'emissione dei titoli di accesso avviene selezionando l'ordine di posto e la tariffa; a questo punto il sistema genera tramite un apposito algoritmo un **sigillo fiscale** che **identifica in maniera univoca il titolo di accesso**.
- La stampa del biglietto può avvenire contestualmente all'emissione del sigillo fiscale o in un secondo momento.

(continua nella slide successiva)

- Il processo di **emissione di un biglietto** è sostanzialmente composto da tre fasi:
  1. **la scelta del tipo di titolo** (l'ordine di posto, la posizione del posto all'interno del locale che ospita la manifestazione) e del tipo di tariffa (intero, ridotto, ecc.)
  2. **l'emissione fiscale del titolo** di accesso (generazione del sigillo fiscale)
  3. **la stampa del titolo** di accesso.

  In caso di errori in fase di emissione o di stampa del titolo di accesso,deve essere possibile **annullare il biglietto**. Tale operazione viene effettuata annullando semplicemente il corrispondente sigillo fiscale.
- Il sistema deve inoltre prevedere la **generazione dei documenti** da inviare all'Agenzia delle Entrate e alla S.I.A.E. (Società Italiana degli Autori ed Editori) ai fini fiscali e di applicazione delle norme sul diritto d'autore. In particolare devono essere prodotti:
  - il **log delle transazioni** (giornale di fondo)
  - i **riepiloghi giornalieri** e
  - i **riepiloghi mensili**

Tali documenti devono essere firmati e inviati tramite e-mail. A tale scopo la macchina sulla quale verrà installato il sistema deve essere dotata di un **lettore di smart card** nel quale verrà inserita una carta contenente il certificato di firma digitale necessario per eseguire tale operazione in maniera sicura.

Primo documento di analisi

- Il sistema è suddiviso logicamente in due sottosistemi fondamentali:
  - **Software per l'operatore**: si occupa delle **funzioni di selezione e stampa del titolo di accesso interagendo con il software fiscale per l'emissione del sigillo fiscale**; si occupa inoltre del**l'annullamento dei biglietti** interagendo con il software fiscale per l'annullamento del corrispondente sigillo fiscale.
  - **Software fiscale**: si occupa dell'emissione dei sigilli fiscali e della gestione dei log. L'emissione dei sigilli fiscali produce i dati necessari alla generazione dei riepiloghi da inviare all'Agenzia delle Entrate e alla S.I.A.E.

    Il software fiscale è a sua volta suddiviso in:
    - **Motore fiscale**: si occupa delle operazioni legate alla generazione ed emissione dei sigilli fiscali (un'operazione continua)
    - **Motore fiscale Log**: si occupa della generazione dei log, della firma elettronica e della trasmissione dei riepiloghi (operazione con una certa cadenza).
- Tutte le componenti del sistema si scambiano i dati attraverso un **database**.

Iterazione 1 AMDD (*early architecture design*)

Best practice per la modellazione *agile*

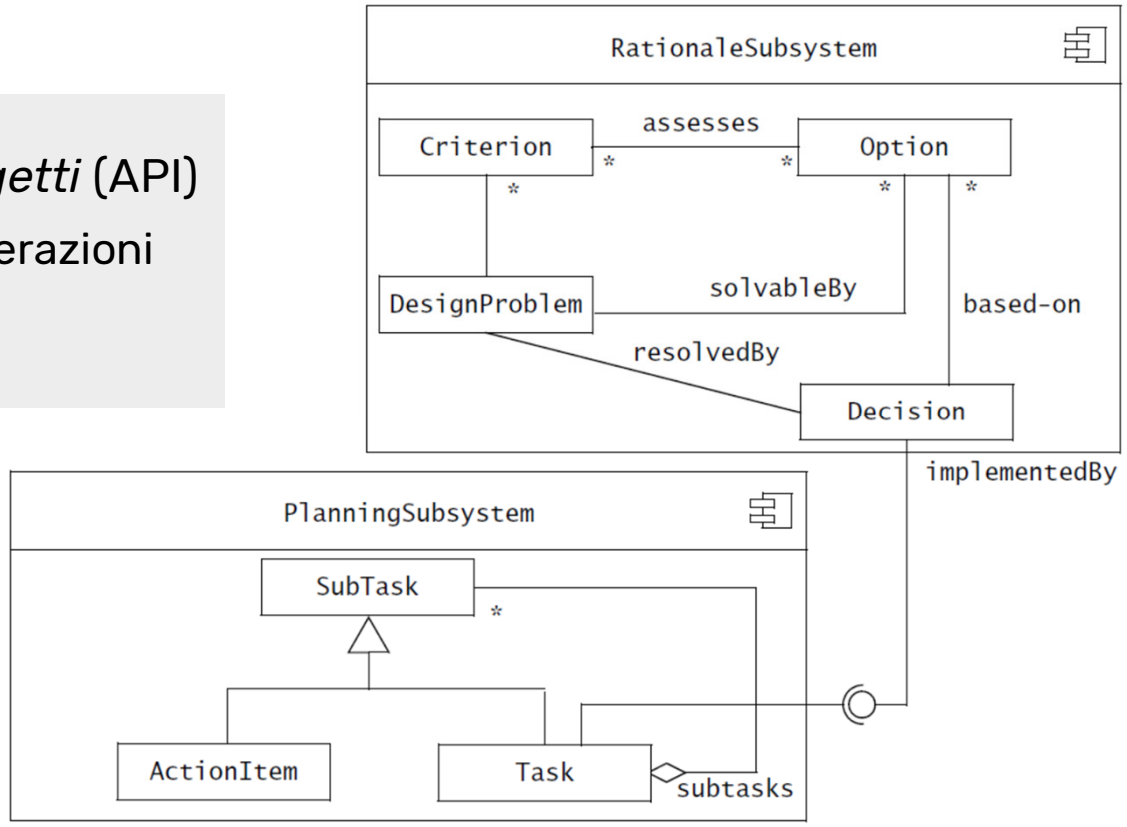
già all'iterazione 1 si può implementare codice: ciò andrà fatto dopo aver eseguito un matching tra design dell'architettura del programma e architettura imposta dal sistema che si utilizzerà per distribuire il codice.

Good design principle:
Il sistema come un insieme di *Interfacce di oggetti* (API)

- Publicazione di un servizio (= insieme di operazioni pubbliche) fornite da un sottosistema

Euristiche di early design:
Un insieme di linee guida per identificare le component iniziali (**early design**) dal modello dei *casi d'uso e da un'eventuale modello di analisi*

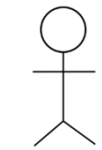
(vedi slide successiva)



Euristiche di *early design*: dai casi d'uso/modelli di analisi alla early software architecture

1. **System**: Introduce a high-level abstract component (the system) viewing **the whole system as one component** (instance level!)
2. **Actors**: Each actor identified in the use case diagram defines an **external entity** – do not represent it explicitly!
3. **Groups of use cases**: **Group use cases** together according to some kind of affinity (e.g., use cases related to each other by generalization relations) and **introduce** a ***control* component** for each group; if the group of use cases persist data, introduce also a ***data component*** for the group (logica MVC)
   - Alternatively, introduce the **same control/data/boundary components from the analysis model**
   - Candidate **classes** inside components or **other components** may come from as refinement of the classes of the analysis model
4. **Actor-use case interaction**: For each use case that has a direct interaction with an external actor introduces an **interface** of externally visible operations, *provided* or *required* by the corresponding component depending on the direction of the interaction
   - Each input variable from an actor (the environment!) specified in the description of a use case defines an **input parameter of the corresponding interface's operation**
   - Each **output variable for an actor** specified in the description of a use case defines a **return parameter of the corresponding interface's operation** (if *synchronous* mode) or an **input parameter of an operation of a *callback interface*** (if *asynchronous* mode) of the actor
5. **Subsystem decomposition and distribution**: Based on the physical deployment of the system (*hardware/software mapping* – given in terms of a UML deployment diagram or *topology diagram in free-style notation*), you may also decompose the whole system into **subsystems**, and therefore distribute your components into these subsystems

E-commerce system: alcuni casi d'uso



user

Payment server

LDAP Authentication server

Requirements

- Address must always have Street, City, State, Zipcode, and Country
- Contact Address is defined as an address.
- Shipping Address is defined as an address.
- The Billing System shall produce pet purchase invoices

Use Cases

UC-1: Sign-in User.
1-1: User provides emailId and password to the eStore authentication module.
1-2: eStore authentication module authenticates via LDAP authentication server.
UC-2: Logout User.
2-1: User submits logout request.
2-2: eStore authentication module removes active user session.
UC-3: Create new User Account.
UC-4: Manage existing User Account.
UC-5: Browse Product by Category in eStore.
UC-6: Search Product by keywords.
UC-7: Append ShoppingCart with items.
7-1: System maintains a ShoppingCart option in the User Account.
7-2: A user adds a ProductItem to a ShoppingCart from a product detail view.

Use Cases

UC-8: Modify ShoppingCart.
8-1: A user opens his ShoppingCart.
8-2: The system shows the list of ProductItems, quantity and price in the ShoppingCart.
8-3: The system saves the content of ShoppingCart in User Account.
UC-9: Checkout ShoppingCart items from eStore.
UC-10: Provide Purchase Payment information.
UC-11:Generate Purchase Invoice.
11-1: System receives payment confirmation from user.
11-2: System sends credit card information and amount to be charged to the Payment module.
11-3: Payment module sends purchase confirmation to the system.
11-4: System creates an invoice for the user consisting of the ProductItems, total cost, date of purchase, and shipping address details.

Esercizio 2: Iterazione 1 AMDD (*early architecture design)*

Applicare le euristiche di design ai casi d'uso UML del sistema di e-commerce per scomporre il sistema in sotto-sistemi e ottenere le componenti e le interfacce iniziali

- usare i **diagrammi UML di package** per definire la *scomposizione logica* (struttura logica) del sistema
- usare i diagrammi **UML delle componenti** e la notazione *ball-and-socket* per **assemblare le componenti** (sotto-sistemi e componenti)
  - Diagramma *top-level* delle macro-componenti/sotto-sistemi per l'intera architettura di sistema
  - Vari diagrammi per mostrare la composizione interna di una macro-componente
- usare un **diagramma UML delle classi** per definire le **interfacce** incluse le operazioni e i parametri di input/output delle operazioni
- usare un **diagramma UML delle classi** to define **data type classes**
- definire un **diagramma UML di deployment** e allocare le componenti del sistema in base alla topologia ottenuta in output dall'iterazione 0 AMDD (*architecture envisioning*) – *scomposizione fisica*

Online UML reference:
https://www.uml-diagrams.org/component-diagrams-reference.html