

Corso di Ingegneria del Software Deliverable di progetto	<b>2023-2024</b>
---	------------------

# **“Ingegneria del Software”**

## **2023-2024**

**Docente: Prof. Angelo Furfaro**

1	
---	--

Corso di Ingegneria del Software Deliverable di progetto	2023-2024
---	-----------

# Organigramma Aziendale Builder

<b>Data</b>	10/09/2024
<b>Documento</b>	Documento Finale – D3

Team Members		
Nome e Cognome	Matricola	E-mail address
Riccardo Florio	209502	<a href="mailto:flrrcr01e20f537t@studenti.unical.it">flrrcr01e20f537t@studenti.unical.it</a>

2	
---	--

Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

## Sommario

List of Challenging/Risky Requirements or Tasks.....	5
A. Stato dell'Arte .....	8
B. Raffinamento dei Requisiti.....	10
<i>B.1 Servizi (con prioritizzazione) .....</i>	<i>10</i>
<i>B.2 Requisiti non funzionali.....</i>	<i>11</i>
<i>B.3 Scenari d'uso dettagliati .....</i>	<i>12</i>
<i>B.4 Requisiti esclusi .....</i>	<i>13</i>
<i>B.5 Assunzioni .....</i>	<i>16</i>
<i>B.6 Diagramma dei casi d'uso .....</i>	<i>19</i>
C. Architettura Software .....	20
<i>C.1 La vista statica del sistema: diagramma dei componenti .....</i>	<i>20</i>
<i>C.2 La visione dinamica dell'architettura del software: diagramma di sequenza .....</i>	<i>27</i>
D. Dati e loro modellazione (se il sistema si interfaccia con un DBMS) .....	29
E. Scelte Progettuali (Design Decisions).....	36
F. Progettazione di Basso Livello .....	40

3	
---	--

<div>Corso di Ingegneria del Software</div> <div>Deliverable di progetto</div>	<div>2023-2024</div>
--	----------------------

G. Spiegare come il progetto soddisfa i requisiti funzionali (FRs) e quelli non funzionali (NFRs)..... 46

Appendice: Prototipo..... 50

Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

## List of Challenging/Risky Requirements or Tasks

Challenging Task	Data the task is identified	Data the challenge is resolved	Explanation on how the challenge has been managed
Creazione dell'organigramma sfruttando il pattern Composite	30/08/2024	30/08/2024	Il pattern Composite è stato scelto per rappresentare la struttura gerarchica dell'organigramma. Ciò ha permesso di gestire sia le unità semplici che gli organi di gestione in modo omogeneo. Implementando l'interfaccia UnitaIF, è stato possibile trattare tutte le unità allo stesso modo, facilitando l'espansione della gerarchia aziendale.
Integrazione del pattern Observer	30/08/2024	31/08/2024	Il pattern Observer è stato utilizzato per aggiornare in tempo reale l'interfaccia grafica quando vengono apportate modifiche all'organigramma. Questo è stato gestito aggiungendo osservatori a ogni unità, assicurando che i cambiamenti si riflettessero immediatamente nella visualizzazione senza richiedere ulteriori interventi manuali.
Integrazione del pattern Iterator	31/08/2024	01/09/2024	Il pattern Iterator è stato implementato per permettere una

5	
---	--

Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

			navigazione efficiente della struttura gerarchica. È stato creato un iteratore che consente di scorrere facilmente tutte le unità dell'organigramma, facilitando l'accesso e la manipolazione dei dati in maniera scalabile, anche con strutture complesse.
Integrazione del pattern Visitor	02/09/2024	02/09/2024	Il pattern Visitor è stato implementato per serializzare l'organigramma in formato XML. Questo ha permesso di separare la logica di attraversamento della struttura dall'esportazione dei dati, rendendo il codice più modulare e mantenibile. È stato creato un Visitor che genera il file XML con tutti i dettagli delle unità e dei dipendenti associati.
Integrazione dei dipendenti nella struttura	30/08/2024	03/09/2024	L'integrazione dei dipendenti è stata complessa da realizzare, in quanto richiedeva la gestione di ruoli multipli per lo stesso dipendente in unità diverse. È stata gestita implementando una mappa all'interno di ogni unità che collega un dipendente ad uno specifico ruolo all'interno di quella stessa unità.
Creazione della GUI	03/09/2024	06/09/2024	La GUI è stata sviluppata utilizzando JGraphX per rappresentare graficamente

6	
---	--

<p>Corso di Ingegneria del Software</p> <p>Deliverable di progetto</p>	<p><b>2023-2024</b></p>
--	-------------------------

			l'organigramma e Swing per gestire l'interazione con l'utente. Sono stati inclusi elementi per la gestione dinamica delle unità, l'aggiunta di dipendenti e la modifica della struttura in tempo reale, garantendo un'interfaccia intuitiva e facile da usare.
Testing	04/09/2024	04/09/2024	Il testing è stato eseguito utilizzando JUnit per verificare il corretto funzionamento dell'iteratore in profondità (DepthFirstIterator). Sono stati testati i casi in cui l'iteratore percorre la struttura in modo corretto e lancia un'eccezione quando non ci sono più elementi da restituire. I test hanno verificato che l'ordine di iterazione sia conforme alle aspettative.

7	
---	--

Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

## A. Stato dell'Arte

---

*Analizzate sistemi esistenti, prendete spunto da ciò che esiste.*

---

Attualmente esistono numerosi strumenti software per la creazione e la gestione di organigrammi aziendali. Uno dei più popolari è **EdrawMax**, un programma utilizzato per creare diagrammi di vario tipo, inclusi gli organigrammi. EdrawMax consente di progettare strutture gerarchiche visuali in modo semplice e intuitivo. Tuttavia, pur offrendo strumenti avanzati per la rappresentazione visiva delle relazioni tra le unità aziendali, presenta alcune limitazioni importanti rispetto ai requisiti di gestione organizzativa complessa.

### Esempio di applicativo esistente

**EdrawMax** permette agli utenti di costruire e personalizzare organigrammi, consentendo di inserire nomi di unità e posizionarle in base alla gerarchia desiderata. Tuttavia, l'applicazione non supporta la gestione di informazioni aggiuntive come l'assegnazione di dipendenti a specifiche unità o la possibilità di definire tipologie dettagliate di unità aziendali (ad esempio direzioni, reparti o uffici). Ciò rende il software più adatto a scopi puramente visuali piuttosto che a una vera e propria gestione gerarchica aziendale funzionale.

8	
---	--



<p>Corso di Ingegneria del Software</p> <p>Deliverable di progetto</p>	<p><b>2023-2024</b></p>
--	-------------------------

### **Differenze con il Nostro Sistema**

Il nostro progetto si distingue proprio per la possibilità di gestire non solo la struttura gerarchica delle unità, ma anche di associare informazioni chiave come la tipologia delle unità e i dipendenti che vi appartengono, insieme ai loro ruoli specifici. Questo livello di dettaglio consente di utilizzare il software non solo per scopi di rappresentazione grafica, ma anche per la gestione funzionale e amministrativa delle risorse aziendali.

Fonte: <https://www.edrawsoft.com/ad/organizational-chart-maker-edrawmax.html>

9	
---	--

<p>Corso di Ingegneria del Software</p> <p>Deliverable di progetto</p>	<p>2023-2024</p>
--	------------------

## B. Raffinamento dei Requisiti

---

*A partire dai servizi minimali richiesti, raffinate la descrizione dei servizi offerti dal vostro applicativo. Descrivete anche I requisiti non funzionali.*

---

### B.1 Servizi (con prioritizzazione)

---

*Descrivete in dettaglio i servizi offerti dal vostro Sistema, insieme a quelli che ritenete siano le soluzioni concettuali necessarie. In questa fase, non fate riferimento ad alcuna tecnologia specifica. Se volete, intervistate stakeholder e collezionate dati dal web o da altre sorgenti. Dovete acquisire una conoscenza avanzata dei problemi associate ai vostri servizi. Assegnate un ID a ciascun servizio. Prioritizzate inoltre I servizi in base a due scale: importanza alta, media, bassa. Complessità alta, media, bassa.*

---

1. **Creazione di nuove unità:** L'utente può creare nuove unità o organi di gestione all'interno della struttura aziendale (Alta Priorità, Media Complessità).
2. **Aggiunta di dipendenti:** Possibilità di aggiungere dipendenti esistenti o crearne di nuovi all'interno di specifiche unità (Alta Priorità, Media Complessità).

10	
----	--

Corso di Ingegneria del Software Deliverable di progetto	2023-2024
---	-----------

3. **Salvataggio e caricamento organigrammi:** Supporto al salvataggio e caricamento della struttura in formato XML (Alta Priorità, Alta Complessità).
4. **Visualizzazione del grafo dell'organigramma:** Presentazione visuale della struttura aziendale (Alta Priorità, Media Complessità).
5. **Modifica delle unità e dipendenti:** Ridenominazione e gestione di unità e dipendenti esistenti (Media Priorità, Media Complessità).

## ***B.2 Requisiti non funzionali***

---

*Elencare i requisiti non funzionali più importanti per il vostro Sistema*

---

1. **Performance:** Il sistema deve poter gestire strutture complesse senza compromettere la reattività dell'interfaccia utente.
2. **Usabilità:** L'interfaccia deve essere intuitiva e facilmente navigabile, consentendo una gestione semplice anche per utenti non tecnici.
3. **Portabilità:** Il sistema deve essere compatibile su qualsiasi sistema operativo che supporti Java.

11	
----	--

<p>Corso di Ingegneria del Software</p> <p>Deliverable di progetto</p>	<p><b>2023-2024</b></p>
--	-------------------------

### ***B.3 Scenari d'uso dettagliati***

---

*Descrivere gli scenari più comuni, più interessanti, o più complicati d'uso dei vostri servizi.*

---

#### **Scenario 1: Apertura e modifica di un organigramma esistente**

L'utente può aprire un file XML che rappresenta un organigramma aziendale preesistente. Una volta caricato l'utente può modificarlo, ad esempio, aggiungendo o rimuovendo dipendenti da un'unità o creando nuove sotto-unità. Una volta apportate le modifiche, l'utente può salvare nuovamente il file, aggiornando così l'organigramma aziendale.

#### **Scenario 2: Creazione di un nuovo organigramma aziendale**

In questo scenario, l'utente inizia creando un nuovo organigramma da zero. Può creare unità aziendali e specificare la loro tipologia (per esempio, **direzione** o **ufficio**), aggiungendo inoltre dipendenti e assegnando loro ruoli specifici. Alla fine del processo, l'utente salva l'organigramma in un file XML, che può poi essere riaperto o condiviso.

#### **Scenario 3: Assegnazione di ruoli diversi allo stesso dipendente**

12	
----	--

<p>Corso di Ingegneria del Software</p> <p>Deliverable di progetto</p>	<p>2023-2024</p>
--	------------------

L'utente può aggiungere lo stesso dipendente a più unità aziendali, ma con ruoli diversi per ciascuna unità. Questo riflette situazioni reali in cui un dipendente potrebbe avere compiti diversi a seconda del reparto o della sede in cui opera. L'interfaccia permette di gestire questi dettagli, mostrando chiaramente i diversi ruoli all'interno dell'organigramma.

#### **Scenario 4: Visualizzazione dinamica della struttura aziendale**

Una volta creato o caricato un organigramma, il sistema fornisce una visualizzazione dinamica della struttura aziendale. L'utente può navigare tra le diverse unità, visualizzare i dipendenti e i loro ruoli e aggiornare la struttura con facilità. Ogni modifica è riflessa immediatamente nella rappresentazione grafica, facilitando il controllo e la gestione dell'organizzazione.

### ***B.4 Requisiti esclusi***

---

*Descrivere i servizi eventualmente i esclusi, e spiegare il perché*

---

Durante lo sviluppo del sistema si è voluto garantire che le funzioni essenziali fossero implementate con successo. Tuttavia, alcune funzionalità avanzate sono state momentaneamente escluse per concentrare le risorse sui requisiti principali. In

13	
----	--

<p>Corso di Ingegneria del Software</p> <p>Deliverable di progetto</p>	<p>2023-2024</p>
--	------------------

futuro, è possibile immaginare l'integrazione di una serie di miglioramenti che potrebbero arricchire l'esperienza dell'utente e le capacità del sistema. Ecco alcuni scenari futuri che potrebbero essere presi in considerazione:

1. **Gestione avanzata dei ruoli in Unità complesse:** Al momento, un dipendente può avere un solo ruolo all'interno di una Unità o Organo di Gestione. In futuro, si potrebbe introdurre la capacità di assegnare ruoli multipli allo stesso dipendente, specialmente in contesti in cui un dipendente svolge diverse funzioni all'interno della stessa Unità. Questa funzionalità può risultare utile in aziende in cui i dipendenti coprono più aree di competenza.
2. **Specificazione di posizioni aperte e ruoli disponibili:** Un'opzione futura potrebbe permettere di associare a ciascuna Unità o Organo di Gestione una lista di ruoli ancora vacanti, permettendo così all'organigramma di fungere anche da strumento di gestione delle risorse umane. Ciò potrebbe includere la gestione delle assunzioni in corso e l'aggiornamento automatico quando un ruolo viene occupato.
3. **Compatibilità con diversi formati di file:** Attualmente il sistema supporta l'importazione e l'esportazione degli organigrammi solo in formato XML. Ampliando i formati supportati, come JSON o CSV, sarebbe possibile aumentare l'interoperabilità con altre piattaforme di gestione aziendale. Inoltre, questo renderebbe il sistema più flessibile per l'integrazione con applicazioni già esistenti all'interno delle aziende.

14	
----	--

Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

4. **Funzionalità di collaborazione multiutente:** L'attuale versione del sistema prevede la gestione dell'organigramma da un singolo utente alla volta. In un contesto aziendale, potrebbe essere utile introdurre la possibilità di collaborazione in tempo reale, dove più utenti possono modificare l'organigramma contemporaneamente, con una gestione accurata delle modifiche.
5. **Integrazione con piattaforme cloud:** Un'ulteriore evoluzione potrebbe riguardare la sincronizzazione e l'archiviazione degli organigrammi su piattaforme cloud, consentendo la gestione centralizzata e la condivisione istantanea delle modifiche in ambienti distribuiti. Questa funzionalità garantirebbe anche un backup sicuro delle informazioni aziendali e una facile accessibilità da parte dei dipendenti autorizzati.
6. **Reportistica e analisi dei dati:** È possibile prevedere una funzionalità per generare report automatici che includano analisi sulle strutture organizzative, come la distribuzione del personale tra le Unità, il numero di dipendenti in ciascun reparto o l'identificazione delle posizioni vacanti. Questi strumenti di analisi potrebbero aiutare le aziende a ottimizzare la loro gestione interna.

Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

## B.5 Assunzioni

---

*Documenta brevemente, in questa sezione, le ipotesi/decisioni sui requisiti più rilevanti che hai dovuto prendere durante il tuo progetto*

---

Di seguito le assunzioni per l'implementazione del sistema:

1. **Matricola unica per dipendente:** Ogni dipendente è identificato da una matricola univoca all'interno del sistema, garantendo che non vi siano duplicati, indipendentemente dal numero di unità a cui è assegnato.
2. **Organigramma con radice singola:** Ogni organigramma ha un solo organo di gestione come nodo radice da cui si sviluppa l'intera struttura gerarchica, rappresentando il vertice dell'organizzazione.
3. **Dipendente con ruoli diversi in più unità:** Un dipendente può appartenere a più unità o organi di gestione all'interno dell'organigramma, potendo ricoprire ruoli diversi in ciascuna di queste unità, per riflettere la sua partecipazione multiruolo nell'organizzazione.
4. **Gerarchia rigida:** Ogni unità o organo di gestione può avere una sola unità superiore, ma può avere più sotto-unità. La struttura segue una gerarchia ad albero.

16	
----	--



<p>Corso di Ingegneria del Software</p> <p>Deliverable di progetto</p>	<p>2023-2024</p>
--	------------------

5. **Nome unico per unità:** All'interno dello stesso organigramma, non possono esistere più unità con lo stesso nome per evitare confusione o ambiguità.
6. **Ruoli unici per dipendente:** Ogni dipendente può avere un solo ruolo all'interno di una specifica unità o organo di gestione. Tuttavia, può avere ruoli diversi in altre unità.
7. **Dimensione illimitata:** Non esiste un limite sul numero di dipendenti che possono essere assegnati a un'unità o un organo di gestione né sul numero di sotto-unità di un organo di gestione.
8. **Persistenza dati:** Gli organigrammi possono essere salvati in formato XML, con la possibilità di caricarli successivamente senza perdita di informazioni.
9. **Modifiche in tempo reale:** Le modifiche apportate all'organigramma, come l'aggiunta o rimozione di dipendenti o unità, sono immediatamente visibili e applicate senza bisogno di riavviare il sistema.
10. **Validazione dei dati:** È presente un sistema di validazione per impedire l'inserimento di dati errati o duplicati, come la ripetizione di un nome per una unità o l'assegnazione di un dipendente già esistente senza un ruolo valido.
11. **Rimozione condizionale:** La rimozione di un'unità o di un organo di gestione è permessa solo se tutte le sotto-unità e dipendenti sono stati rimossi, evitando la perdita di dati accidentale.
12. **Gestione centralizzata delle matricole:** Le matricole dei dipendenti vengono assegnate automaticamente e in modo incrementale, evitando

17	
----	--

Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

conflitti o duplicati anche nel caso in cui lo stesso dipendente sia assegnato a più unità.

13. **Compatibilità del sistema:** Il sistema è progettato per funzionare su qualsiasi piattaforma che supporti Java, garantendo una portabilità completa e indipendenza dal sistema operativo.

**B.6 Diagramma dei casi d'uso**

Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

## C. Architettura Software

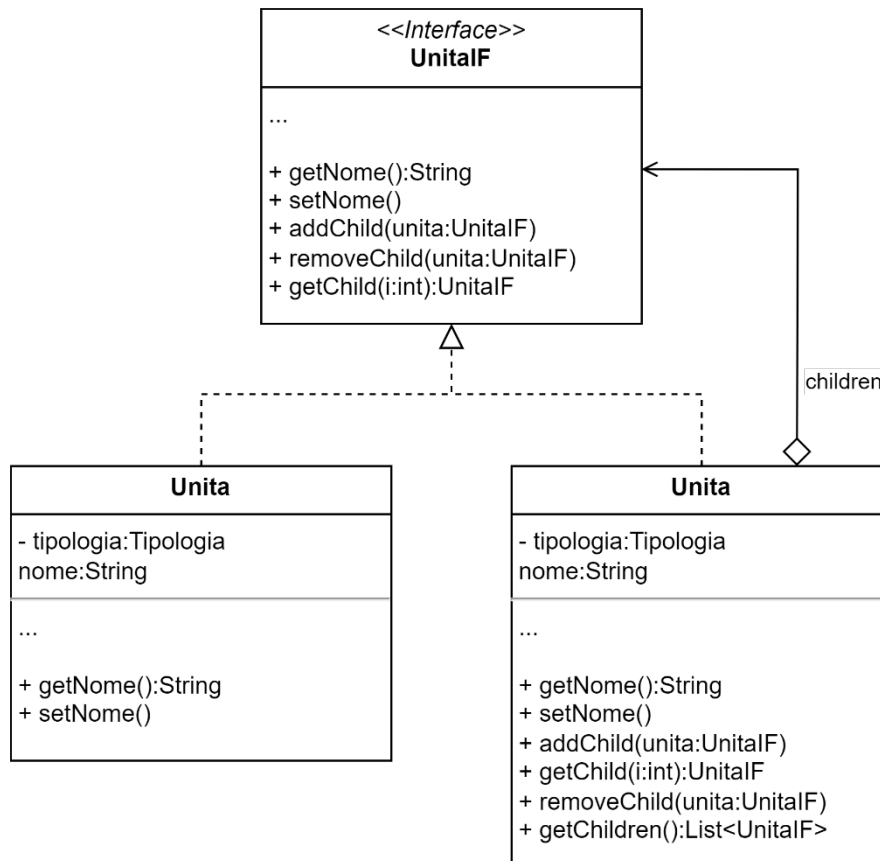
### ***C.1 La vista statica del sistema: diagramma dei componenti***

Lo sviluppo dell'applicazione ha seguito varie fasi, nelle quali è stato impiegato un pattern per soddisfare i requisiti del sistema.

#### **Fase 1: Creazione dell'organigramma sfruttando il pattern Composite**

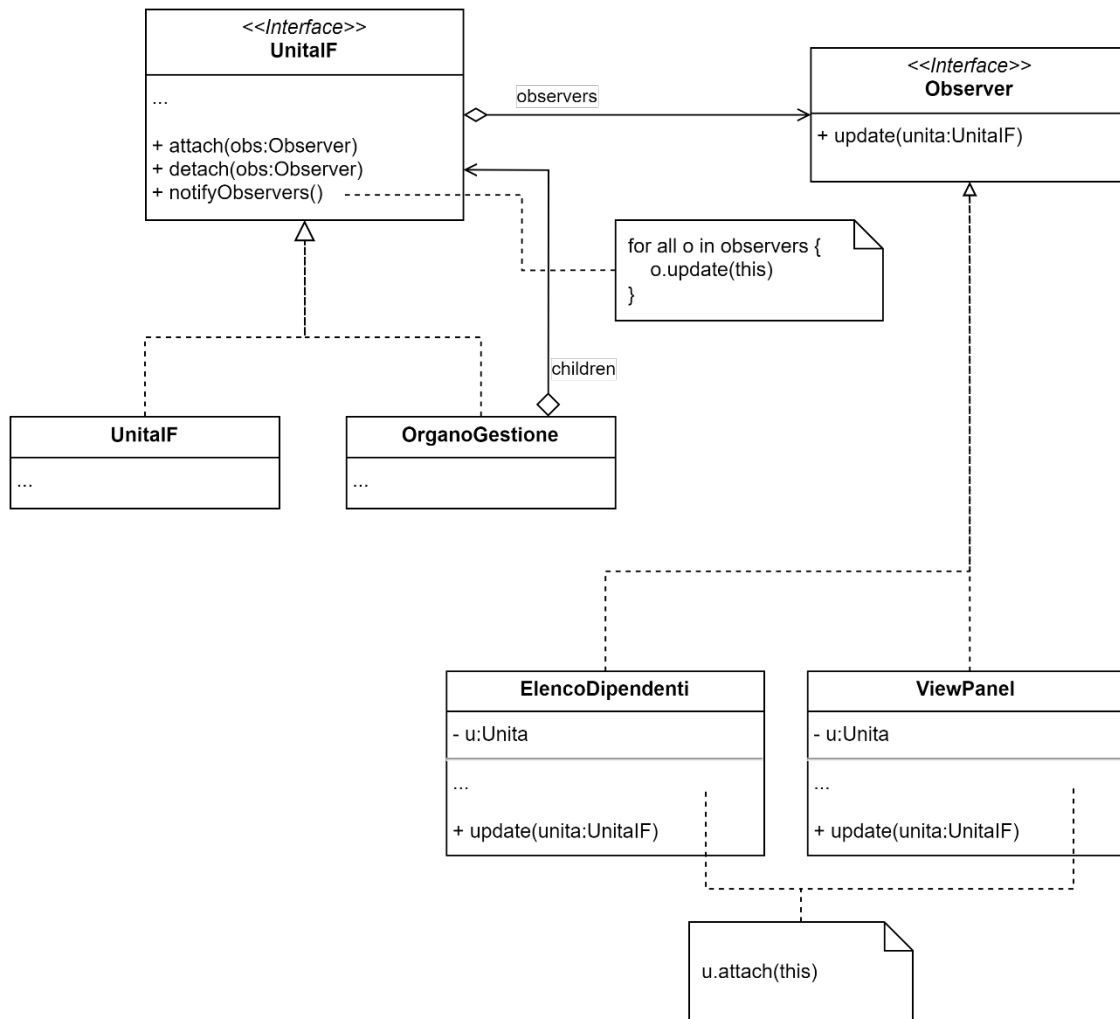
In questa fase si inizia a costruire la struttura per rappresentare l'organigramma aziendale richiesto. Per farlo viene sfruttato il pattern Composite, che si mostra particolarmente adatto allo scopo.

## Deliverable di progetto

**Fase 2: Integrazione del pattern Observer**

Essendo richiesta l'implementazione di un'interfaccia grafica, è stato integrato il pattern Observer per notificare, in un secondo momento, i cambiamenti alla struttura dati a chi gestisce la visualizzazione/modifica di tale struttura.

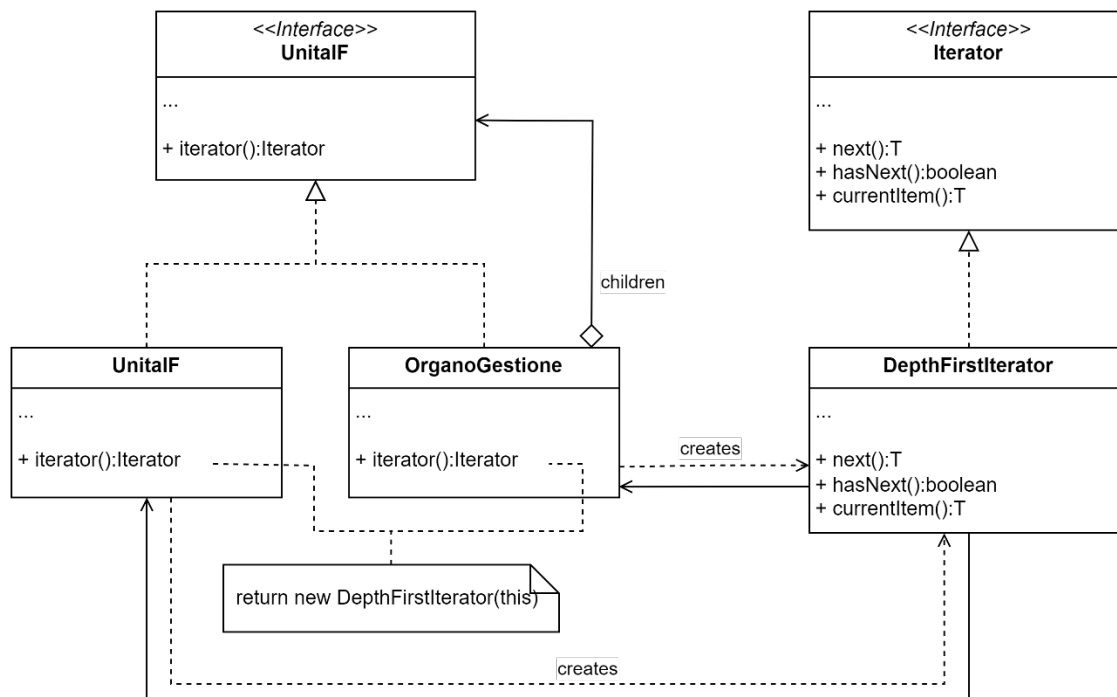
## Deliverable di progetto



Si noti che si sta usando il pattern Observer per un osservatore con più Subjects e che la classe **ElencoDipendenti** è una classe privata interna alla classe **EditUnitaDialog**.

**Fase 3: Integrazione del pattern Iterator**

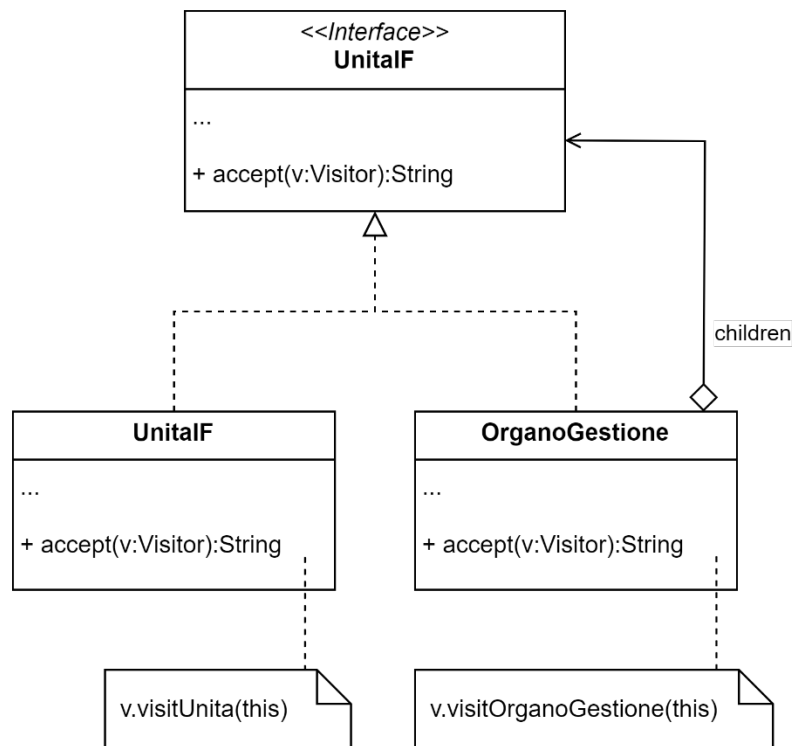
Questo pattern è utile in questo contesto dal momento che permette di definire un iteratore per scandire la struttura che è stata definita prima. Essendo una struttura ad albero, si implementa l'iteratore sfruttando una ricerca in profondità.

**Fase 4: Integrazione del pattern Visitor**

Dovendo implementare una soluzione che permette di salvare l'organigramma in memoria, su richiesta della traccia, decido di usare il pattern Visitor per creare una

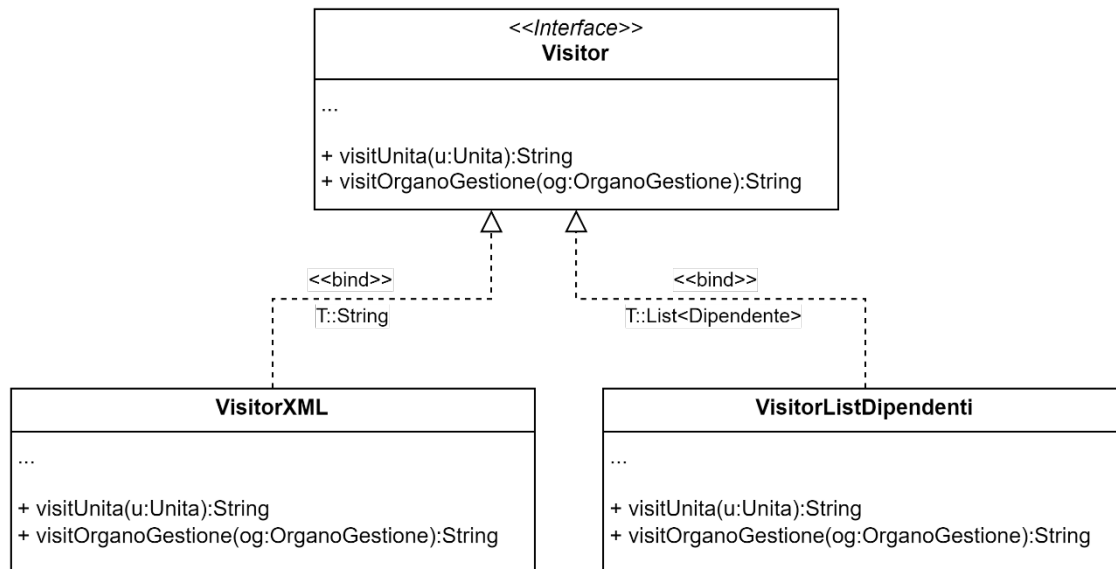
## Deliverable di progetto

stringa formattata secondo i canoni del linguaggio XML, il quale è appropriato per rappresentare strutture ad albero come quella che stiamo implementando.

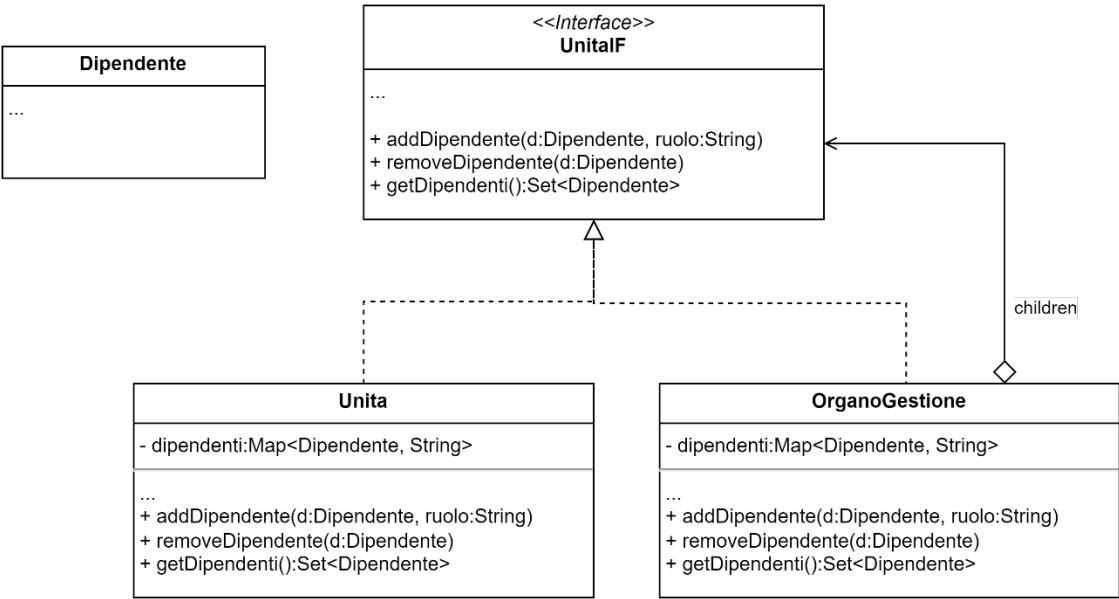




## Deliverable di progetto

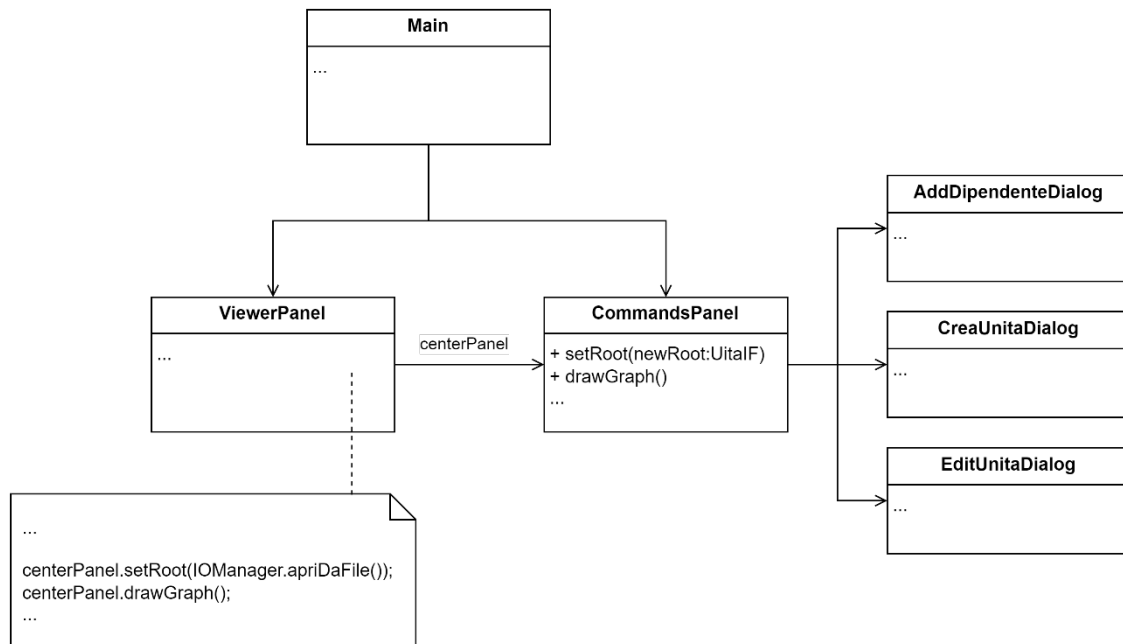
**Fase 5: Integrazione dei dipendenti nella struttura**

Una funzionalità richiesta dalla traccia è quella di poter aggiungere dipendenti ad un'unità/organi di gestione. Creo quindi una classe `Dipendente` e una struttura per associare i dipendenti ai rispettivi ruoli.



**Fase 6: Creazione GUI**

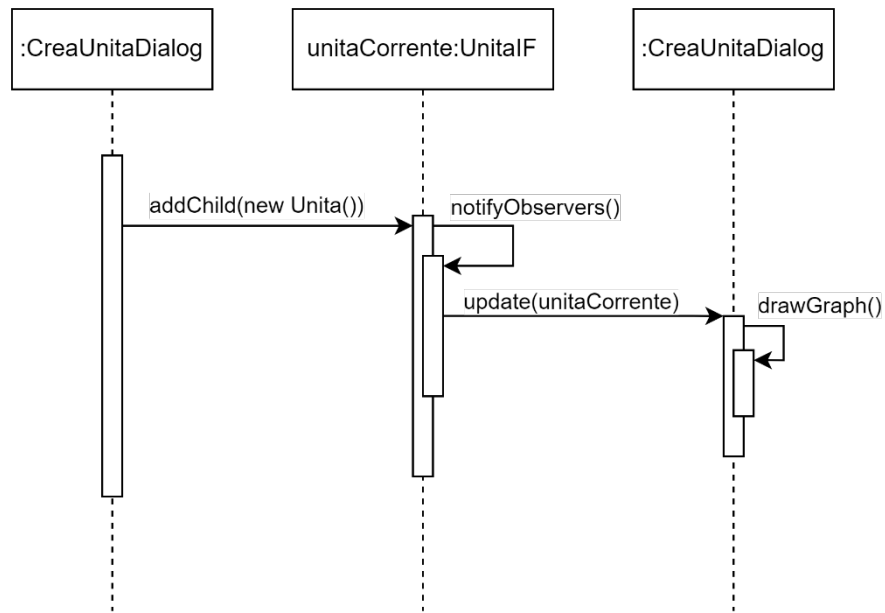
È stato già notato come alcune componenti grafiche interagiscano con il resto del sistema. Di seguito è illustrata, invece, l’interazione tra tali componenti.



## C.2 La visione dinamica dell'architettura del software: diagramma di sequenza

Di seguito un diagramma di esempio che raffigura l'aggiunta di una unità semplice all'organigramma. Dopo la chiamata al metodo *addChild()* per aggiungere l'unità, viene richiamato il metodo *notifyObservers()* per avvisare gli osservatori del cambiamento. In questo modo l'aggiornamento del grafico dell'organigramma è automatico.

## Deliverable di progetto



Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

## D. Dati e loro modellazione (se il sistema si interfaccia con un DBMS)

---

*Definite le sorgenti di dati a voi necessarie per realizzare I servizi di cui sopra. Modellate tali dati tramite un ER o similari. Specificate se e quali di tali dati sono già forniti da applicativi esistenti.*

---

Il sistema utilizza XML come formato per la memorizzazione e il caricamento dei dati relativi all'organigramma aziendale. Questo approccio consente di rappresentare in modo chiaro e leggibile sia le unità organizzative (unità semplici e organi di gestione) che i dipendenti, mantenendo informazioni cruciali come nome, matricola e ruolo, oltre alla gerarchia dell'organigramma stesso.

### Estrarre XML dall'organigramma con il Pattern Visitor

Il **VisitorXML** è un'implementazione del pattern *Visitor*, che permette di visitare ciascun nodo dell'organigramma e tradurlo in un formato XML. Il pattern *Visitor* viene utilizzato qui per separare la logica di elaborazione dei nodi dalla struttura dati dell'organigramma stesso.

<p>Corso di Ingegneria del Software</p> <p>Deliverable di progetto</p>	<p>2023-2024</p>
--	------------------

### Passi chiave nell'implementazione di VisitorXML:

1. **Visitare le unità semplici:** Quando il visitor incontra un'unità (ovvero un nodo semplice dell'organigramma) crea una rappresentazione XML con attributi come il nome e la tipologia. L'unità può anche contenere dipendenti, che vengono rappresentati come sotto-elementi `<dipendente>` con informazioni su matricola, nome e ruolo.

```
public String visitUnita(Unita u) {
    StringBuilder sb = new StringBuilder();
    sb.append(String.format("<unita nome='%s'
tipologia='%s'>%n", u.getNome(),
u.getTipologia().toString()));
    sb.append("<dipendenti>%n");
    // Ciclo per aggiungere dipendenti
    sb.append("</dipendenti>%n");
    sb.append("</unita>%n");
    return sb.toString();
}
```

2. **Visitare gli organi di gestione:** Se il visitor incontra un organo di gestione, viene creato un elemento XML `<organogestione>`, simile all'unità, ma con la

30	
----	--

Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

possibilità di contenere sia dipendenti che altre unità o organi di gestione figli. Questo viene gestito in modo ricorsivo: ogni organo di gestione figlio o unità viene a sua volta visitato.

```
public String visitOrganoGestione(OrganoGestione og) {
    StringBuilder sb = new StringBuilder();
    sb.append(String.format("<organogestione nome='%s'
tipo='%s'>%n", og.getNome(), og.getTipologia().toString()));
    // Aggiungi dipendenti
    for (UnitàIF u : og.getChildren())
        sb.append(u.accept(this)); // Chiamata ricorsiva
    sb.append("</organogestione>%n");
    return sb.toString();
}
```

Il **VisitorXML** permette dunque di navigare in profondità l'intera gerarchia e generare un file XML che rappresenta l'intero organigramma aziendale, pronto per essere salvato o trasferito.

31	
----	--

Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

### Leggere e trasformare XML in organigramma con XMLParser

Per leggere un file XML e trasformarlo nuovamente in una struttura di dati gerarchica, è stata implementata la classe **XMLParser**. Questa utilizza il *Document Object Model (DOM)* per leggere e navigare il file XML. La classe costruisce dinamicamente l'organigramma durante il parsing dell'XML, creando oggetti *OrganoGestione*, *Unità* e *Dipendente* secondo le informazioni contenute nel file.

#### Passi chiave nell'implementazione di XMLParser:

1. **Caricamento del file XML:** La classe inizia creando un oggetto *Document* attraverso il DOM, che rappresenta l'intero albero XML. Dopodiché verifica se la radice è un `<organogestione>`, altrimenti scarta il file.

```
public static UnitàIF parseXMLToOrganigramma(File file) {
    DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
    Document doc = builder.parse(file);
    Element rootElement = doc.getDocumentElement();
    if (!rootElement.getNodeName().equals("organogestione"))
return null;
    return parseOrganoGestione(rootElement);
}
```

32	
----	--



Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

2. **Parsing ricorsivo:** Il parsing è ricorsivo, significa cioè che un organo di gestione può contenere altre unità o altri organi di gestione. Ogni organo di gestione viene trattato come una radice temporanea, e per ciascun nodo si controlla se è un'unità o un organo di gestione figlio.

```
private static OrganoGestione parseOrganoGestione(Element
ogElement) {
    OrganoGestione organoGestione = new OrganoGestione(nome,
UnitaIF.Tipologia.valueOf(tipologia));
    // Parsing ricorsivo di figli (unità e altri organi di
gestione)
    if (childElement.getNodeName().equals("unita")) {
        organoGestione.addChild(parseUnita(childElement));
    } else if
(childElement.getNodeName().equals("organogestione")) {
        organoGestione.addChild(parseOrganoGestione(childElement));
    }
    return organoGestione;
}
```

33	
----	--

<p>Corso di Ingegneria del Software</p> <p>Deliverable di progetto</p>	<p>2023-2024</p>
--	------------------

3. **Parsing dei dipendenti:** Sia le unità che gli organi di gestione possono avere dipendenti. Ogni dipendente viene rappresentato come un nodo `<dipendente>` con attributi `matricola`, `nome` e `ruolo`. Il parser crea un oggetto `Dipendente` per ciascuno di questi nodi e lo aggiunge alla relativa unità o organo di gestione.

```
private static Dipendente parseDipendente(Element
dipendenteElement) {
    String nome = dipendenteElement.getAttribute("nome");
    int matricola =
Integer.parseInt(dipendenteElement.getAttribute("matricola")
);
    return new Dipendente(nome, matricola); // Creazione del
dipendente
}
```

Grazie a questa implementazione, il sistema può:

- **Salvare l'organigramma** in un formato XML leggibile e facilmente trasferibile.

34	
----	--

<p>Corso di Ingegneria del Software</p> <p>Deliverable di progetto</p>	<p><b>2023-2024</b></p>
--	-------------------------

- **Caricare l'organigramma** da un file XML e ripristinare la struttura gerarchica e i dipendenti associati. L'uso del *Pattern Visitor* per la generazione dell'XML e del *DOM* per il parsing rende l'architettura flessibile e scalabile, con un'elevata separazione delle responsabilità tra la gestione della logica e la rappresentazione dei dati.

Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

## E. Scelte Progettuali (Design Decisions)

---

*Documenta qui le 5 decisioni di design più importanti che hai dovuto prendere. È possibile utilizzare sia una specifica testuale che una diagrammatica.*

---

Le cinque scelte progettuali fondamentali sono state guidate dalla necessità di creare un sistema modulare, flessibile e facilmente estendibile, che consentisse di gestire strutture complesse come organigrammi aziendali in modo intuitivo e scalabile. Di seguito, una descrizione più approfondita di ciascuna scelta, con particolare enfasi sul perché sono state adottate.

1. **Pattern Composite:** la scelta di utilizzare il *Pattern Composite* per gestire le unità e gli organi di gestione come una gerarchia flessibile è risultata cruciale per rappresentare correttamente le strutture organizzative aziendali. Questo pattern permette di trattare in modo uniforme oggetti composti (come un organo di gestione) e oggetti semplici (come una singola unità). La flessibilità del *Composite* consente all'applicazione di gestire l'aggiunta, la rimozione e la navigazione tra unità e organi di gestione senza doversi preoccupare della loro complessità interna. La struttura di un organigramma aziendale si presta perfettamente all'uso di questo pattern perché consente di trattare gruppi e singole unità allo stesso modo, semplificando la gestione e

Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

rendendo facile l'espansione di nuove funzionalità. Inoltre, facilita l'adozione di modifiche future senza stravolgere l'architettura esistente.

2. **Pattern Observer:** implementare il *Pattern Observer* è stato essenziale per mantenere sincronizzata la UI con i cambiamenti effettuati alla struttura dell'organigramma. L'uso di questo pattern consente di aggiornare automaticamente tutte le viste, come il pannello di visualizzazione del grafo (il *ViewerPanel*) o la lista dei dipendenti (il *ElencoDipendenti*), ogni volta che una parte dell'organigramma viene modificata. Questo garantisce che l'interfaccia utente rifletta sempre lo stato corrente dei dati sottostanti. Il *Pattern Observer* è stato scelto per facilitare una separazione chiara tra la logica dei dati e la presentazione, mantenendo la UI responsiva e reattiva ai cambiamenti della struttura aziendale senza richiedere un aggiornamento manuale da parte dell'utente. Questo approccio riduce le possibilità di errore e migliora l'esperienza utente, garantendo una visualizzazione sempre coerente.
3. **Pattern Iterator:** il *Pattern Iterator* è stato scelto per permettere una navigazione efficiente e flessibile della struttura dell'organigramma, che è essenzialmente un albero gerarchico. Implementare un iteratore con una ricerca in profondità (*DepthFirstIterator*) consente di esplorare l'intero organigramma senza esporre la sua struttura interna, favorendo una maggiore modularità. Questo pattern è essenziale per eseguire operazioni che

37	
----	--

<p>Corso di Ingegneria del Software</p> <p>Deliverable di progetto</p>	<p>2023-2024</p>
--	------------------

richiedono di visitare ogni nodo, come la visualizzazione grafica dell'intera struttura. Il *Pattern Iterator* facilita la separazione della logica di navigazione dall'implementazione delle unità stesse, rendendo il codice più manutenibile e facile da estendere. Questo si rivela particolarmente utile quando l'organigramma cresce in complessità, permettendo di attraversare la struttura senza preoccuparsi dei dettagli specifici della sua implementazione.

4. **Salvataggio in XML:** il formato XML è stato scelto per la sua semplicità e leggibilità. Grazie al design di tipo gerarchico, l'XML si presta perfettamente a rappresentare la struttura ad albero dell'organigramma. Inoltre, l'XML è un formato ampiamente supportato e standardizzato, che permette al sistema di essere interoperabile con altre applicazioni. L'uso del *Visitor Pattern* per esportare l'organigramma in XML consente di mantenere la logica di salvataggio separata dalla logica del dominio, rendendo il sistema più modulare. La scelta dell'XML è stata guidata dal bisogno di avere un formato di salvataggio standard e facilmente comprensibile, che permetta una gestione semplice dei dati senza la necessità di usare un database. Questo facilita anche il debug e il controllo della correttezza dei dati, poiché il file XML può essere ispezionato manualmente. La sua struttura gerarchica riflette perfettamente quella dell'organigramma, rendendo il mapping dei dati naturale e intuitivo.

38	
----	--

Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

5. **Utilizzo di JGraphX:** la scelta di JGraphX come strumento per la visualizzazione grafica si è rivelata estremamente efficace per rappresentare visivamente strutture complesse come l'organigramma aziendale. JGraphX fornisce un'interfaccia flessibile per costruire grafi e offre un set di strumenti per gestire il layout automatico, rendendo la visualizzazione della struttura intuitiva e dinamica. È stato utilizzato per garantire che ogni unità e organo di gestione sia rappresentato in modo chiaro e ordinato, con un layout che si adatta automaticamente alle modifiche. La rappresentazione grafica di un organigramma è fondamentale per una corretta comprensione della struttura aziendale. JGraphX fornisce un supporto robusto e flessibile per disegnare e aggiornare dinamicamente il grafo, rendendo più facile per l'utente finale visualizzare e interagire con la struttura. La possibilità di personalizzare il layout e i nodi del grafo ha permesso di creare una rappresentazione visiva coerente e facile da usare.

Queste scelte progettuali hanno contribuito a creare un sistema che non solo soddisfa i requisiti funzionali, ma è anche facilmente estendibile e mantenibile, grazie all'adozione di pattern di progettazione consolidati.

39	
----	--

Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

## F. Progettazione di Basso Livello

Di seguito, una spiegazione dettagliata della struttura e organizzazione dell'applicazione.

### Composite Pattern

- **Classi Coinvolte:** *UnitaIF*, *Unita*, *OrganoGestione*.
- **Descrizione:** come già spiegato in precedenza, il pattern Composite è stato utilizzato per gestire la gerarchia delle unità aziendali. L'interfaccia *UnitaIF* funge da contratto comune per tutte le unità. Le classi *Unita* e *OrganoGestione* implementano questa interfaccia, permettendo così di trattare unità semplici (*Unita*) e organi di gestione più complessi (*OrganoGestione*) nello stesso modo.
  - **OrganoGestione:** Può avere sotto-unità o altri organi di gestione come figli, gestendo una struttura ricorsiva e flessibile.
  - **Unita:** Rappresenta unità di livello più basso, come uffici o reparti, che non possono avere sotto-unità.
- **Motivazione:** Questo approccio permette di costruire una struttura gerarchica complessa, in cui le unità aziendali possono essere composte o scomposte, garantendo flessibilità e riutilizzabilità.

40	
----	--



Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

## Observer Pattern

- **Classi Coinvolte:** *UnitaIF, ElencoDipendenti, ViewerPanel, Observer.*
- **Descrizione:** Il pattern Observer viene utilizzato per mantenere sincronizzata l'interfaccia grafica con i dati modificati. Ogni volta che una unità subisce una modifica (aggiunta di un dipendente, cambiamento del nome, ecc.), gli osservatori associati vengono notificati per aggiornare la visualizzazione.
  - **ElencoDipendenti:** Questa classe implementa l'interfaccia Observer aggiornando la lista dei dipendenti visualizzata nell'interfaccia ogni volta che un dipendente viene aggiunto o rimosso.
  - **ViewerPanel:** Anche il ViewerPanel è un osservatore che aggiorna il grafico visivo dell'organigramma quando si verificano modifiche a una delle unità.
- **Motivazione:** L'uso del pattern Observer garantisce che l'interfaccia utente rimanga coerente e aggiornata in tempo reale senza dover effettuare refresh manuali. Questo rende l'esperienza utente più fluida.

## Iterator Pattern

- **Classi Coinvolte:** *UnitaIF, DepthFirstIterator, Iterator.*
- **Descrizione:** Il pattern Iterator è utilizzato per navigare la struttura gerarchica dell'organigramma. Implementando un iteratore che attraversa la

41	
----	--

<p>Corso di Ingegneria del Software</p> <p>Deliverable di progetto</p>	<p>2023-2024</p>
--	------------------

gerarchia in profondità (Depth First), è possibile percorrere tutte le unità aziendali, sia semplici che complesse, in maniera ordinata.

- **DepthFirstIterator:** Questa classe fornisce un iteratore, implementando l'interfaccia Iterator, che attraversa la struttura a partire dalla radice e visita ogni nodo figlio prima di passare ai successivi.
- **Motivazione:** L'implementazione dell'iteratore permette di scorrere facilmente tutta la gerarchia aziendale senza dover conoscere i dettagli della sua struttura interna, utile per operazioni come l'aggiornamento dinamico del grafico o l'analisi dei dati.

## Visitor Pattern

- **Classi Coinvolte:** *UnitaIF*, *OrganoGestione*, *Unita*, *Visitor*, *VisitorXML*, *VisitorListDipendenti*.
- **Descrizione:** Il pattern Visitor è utilizzato per separare l'operazione da eseguire dalla struttura degli oggetti su cui agisce. In questo progetto, il Visitor è utilizzato principalmente per l'esportazione in XML e per raccogliere informazioni sulla struttura gerarchica.
  - **VisitorXML:** Implementa l'interfaccia Visitor per navigare attraverso l'organigramma e generare un documento XML

42	
----	--

Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

rappresentante la struttura gerarchica. Ogni unità (Unità o OrganoGestione) accetta un oggetto Visitor che ne visita i componenti e genera una rappresentazione XML.

- **VisitorListDipendenti:** È un'altra implementazione dell'interfaccia Visitor che visita le unità aziendali per estrarre e restituire una lista di tutti i dipendenti contenuti nelle varie unità dell'organigramma.
- **Motivazione:** Il Visitor consente di estendere facilmente nuove operazioni su una gerarchia di classi senza modificarle direttamente. In questo progetto, l'uso del Visitor permette di implementare la serializzazione in XML e altre funzionalità senza toccare la logica interna delle unità (UnitàIF e sue implementazioni), mantenendo il codice separato e più facilmente manutenibile.

In sintesi, il **Visitor Pattern** facilita l'aggiunta di nuove funzionalità, come la serializzazione e l'analisi, senza alterare la struttura delle classi stesse.

### Archiviazione in XML

- **Classi Coinvolte:** *IOManager*, *VisitorXML*.
- **Descrizione:** Il progetto utilizza il formato XML per la persistenza dei dati dell'organigramma. Il pattern Visitor viene applicato per serializzare la struttura gerarchica delle unità in un file XML. Le unità e i loro dipendenti

43	
----	--

<p>Corso di Ingegneria del Software</p> <p>Deliverable di progetto</p>	<p>2023-2024</p>
--	------------------

vengono visitati e trasformati in un formato strutturato che può essere facilmente salvato e caricato successivamente sfruttando i metodi della classe di utilità IOManager.

- **Motivazione:** La scelta del formato XML è dovuta alla sua semplicità, leggibilità e portabilità. È un formato standardizzato che consente una facile integrazione con altre applicazioni e la gestione dei dati strutturati.

### Utilizzo di JGraphX per la Visualizzazione

- **Classi Coinvolte:** *ViewerPanel*.
- **Descrizione:** Per la visualizzazione grafica dell'organigramma, l'applicazione utilizza JGraphX, una libreria che permette di creare grafi visuali. Questo strumento rende possibile rappresentare in maniera chiara la struttura dell'organigramma con nodi e connessioni che rappresentano le unità aziendali e le loro relazioni.
  - **ViewerPanel:** Utilizza mxGraph per disegnare graficamente la struttura gerarchica, posizionando le unità come nodi e le connessioni gerarchiche come archi.
- **Motivazione:** JGraphX è stato scelto per la sua capacità di gestire grafi complessi e per le sue funzionalità di layout automatico, che permettono di visualizzare le strutture in modo leggibile anche in caso di grandi organigrammi.

44	
----	--

<p>Corso di Ingegneria del Software</p> <p>Deliverable di progetto</p>	<p><b>2023-2024</b></p>
--	-------------------------

In sintesi, la progettazione di basso livello del progetto segue una struttura ben organizzata che utilizza i pattern fondamentali per garantire flessibilità, modularità e facilità di gestione dei dati, oltre a garantire una visualizzazione efficace della struttura gerarchica aziendale.

Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

## G. Spiegare come il progetto soddisfa i requisiti funzionali (FRs) e quelli non funzionali (NFRs)

---

*Spiega in questa sezione come il design architettónico e di basso livello che hai prodotto soddisfa i FR e i NFR*

---

### Requisiti Funzionali (FR)

1. **Creazione e gestione dinamica dell'organigramma:** grazie all'implementazione del *Pattern Composite*, il sistema permette di gestire la gerarchia aziendale in modo dinamico. L'interfaccia *UnitaIF* consente di trattare in modo unificato sia le unità semplici (*Unita*) che le unità complesse come gli organi di gestione (*OrganoGestione*). Ciò facilita l'aggiunta di nuove unità o sotto-unità in modo flessibile, rispettando il requisito di creare e modificare l'organigramma aziendale in modo dinamico.
2. **Visualizzazione grafica dell'organigramma:** il sistema utilizza la libreria *JGraphX* per rappresentare graficamente l'organigramma aziendale, visualizzando in modo chiaro la struttura gerarchica. Questo consente all'utente di esplorare e visualizzare i dettagli dell'organizzazione attraverso un'interfaccia grafica chiara e interattiva. Il *Pattern Observer* è integrato per

Corso di Ingegneria del Software	2023-2024
Deliverable di progetto	

garantire che la visualizzazione sia sempre aggiornata, riflettendo in tempo reale le modifiche apportate all'organigramma.

3. **Aggiunta di dipendenti:** il sistema permette di aggiungere dipendenti esistenti o di crearne di nuovi all'interno di unità o organi di gestione. L'integrazione del *Pattern Composite* facilita l'aggiunta di dipendenti in qualsiasi nodo della gerarchia aziendale, rispettando la flessibilità richiesta dai requisiti funzionali.
4. **Salvataggio e caricamento in XML:** il sistema consente di salvare l'organigramma aziendale in formato XML e di ricaricarlo successivamente. Il *Pattern Visitor* è utilizzato per esportare l'intera struttura organizzativa, serializzandola in XML in modo da mantenere la gerarchia delle unità e la relazione tra i dipendenti. Questo soddisfa il requisito di conservare e ripristinare facilmente la struttura dell'organizzazione.
5. **Modifica di unità e dipendenti:** l'utente ha la possibilità di rinominare unità e modificare i ruoli dei dipendenti, mantenendo una visione aggiornata del sistema. Questo è reso possibile grazie all'uso del *Pattern Observer*, che permette di notificare e aggiornare tutti i componenti del sistema in caso di modifiche.

47	
----	--

<p>Corso di Ingegneria del Software</p> <p>Deliverable di progetto</p>	<p>2023-2024</p>
--	------------------

### Requisiti Non Funzionali (NFR)

1. **Performance:** l'architettura progettata assicura che il sistema sia in grado di gestire strutture complesse senza compromettere le prestazioni. L'uso del *Pattern Iterator* permette una navigazione efficiente della struttura, consentendo di percorrere in modo sistematico e scalabile le unità aziendali, anche in contesti complessi. Questo garantisce che la visualizzazione e l'aggiornamento del grafo siano eseguiti in modo performante, indipendentemente dalla dimensione dell'organigramma.
2. **Usabilità:** l'interfaccia grafica è progettata per essere intuitiva e facile da usare. Il *Pattern Observer* garantisce che l'interfaccia sia sempre aggiornata e in sincronia con i dati sottostanti, evitando confusione per l'utente. L'utilizzo di dialoghi semplici e chiari per l'aggiunta di unità e dipendenti migliora l'esperienza utente, permettendo di eseguire operazioni complesse con un'interazione minima.
3. **Portabilità:** il sistema è progettato in Java, garantendo la sua portabilità su tutte le piattaforme che supportano la JVM. Inoltre, il formato di salvataggio in XML garantisce che i dati possano essere esportati, condivisi e utilizzati su diversi sistemi e applicazioni che supportano questo standard.
4. **Modularità e Manutenibilità:** il design modulare, basato su pattern di progettazione ben noti (Composite, Observer, Visitor, Iterator), rende il sistema facilmente estensibile e manutenibile. Aggiungere nuove

48	
----	--



<p>Corso di Ingegneria del Software</p> <p>Deliverable di progetto</p>	<p><b>2023-2024</b></p>
--	-------------------------

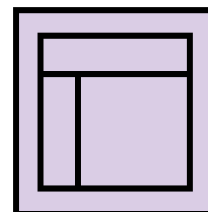
funzionalità o modificare quelle esistenti può essere fatto con interventi localizzati, senza influenzare negativamente l'intera applicazione.

Il progetto soddisfa pienamente i requisiti funzionali e non funzionali grazie a un'architettura solida e modulare, che consente di gestire in modo efficiente strutture complesse mantenendo un'interfaccia utente semplice e intuitiva.

<p>Corso di Ingegneria del Software</p> <p>Deliverable di progetto</p>	<p>2023-2024</p>
--	------------------

## Appendice: Prototipo

*Fornisci un breve rapporto sul tuo prototipo e, in particolare: informazioni su ciò che hai implementato, in che modo l'implementazione copre l'FR e l'NFR, in che modo i prototipi dimostrano la correttezza del tuo progetto rispetto all'FR e all'NFR. Puoi aggiungere alcuni screenshot per descrivere ciò che è richiesto sopra. Sii pronto a mostrare il tuo prototipo durante l'esame orale.*



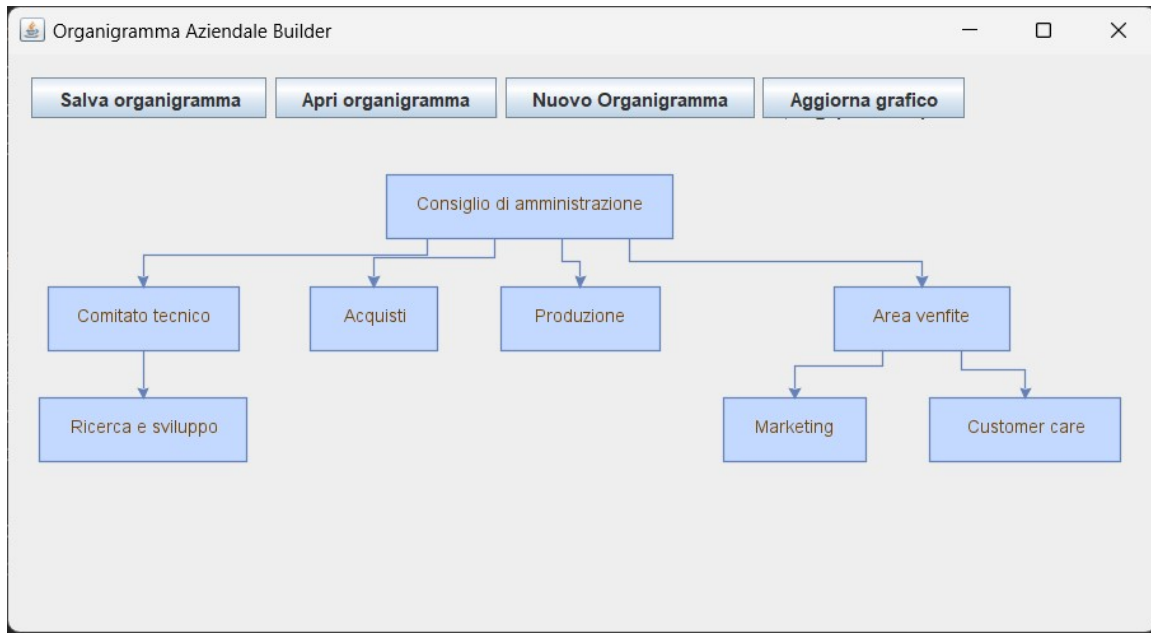
Il prototipo implementa le funzionalità chiave, tra cui:

- Creazione e gestione di organigrammi.
- Salvataggio e caricamento dei dati in formato XML.
- Interfaccia grafica per visualizzare l'organigramma in modo dinamico e intuitivo.

Di seguito sono riportati alcuni screenshot che dimostrano il funzionamento del sistema.

50	
----	--

## Deliverable di progetto

*Figura 1 - Finestra principale*

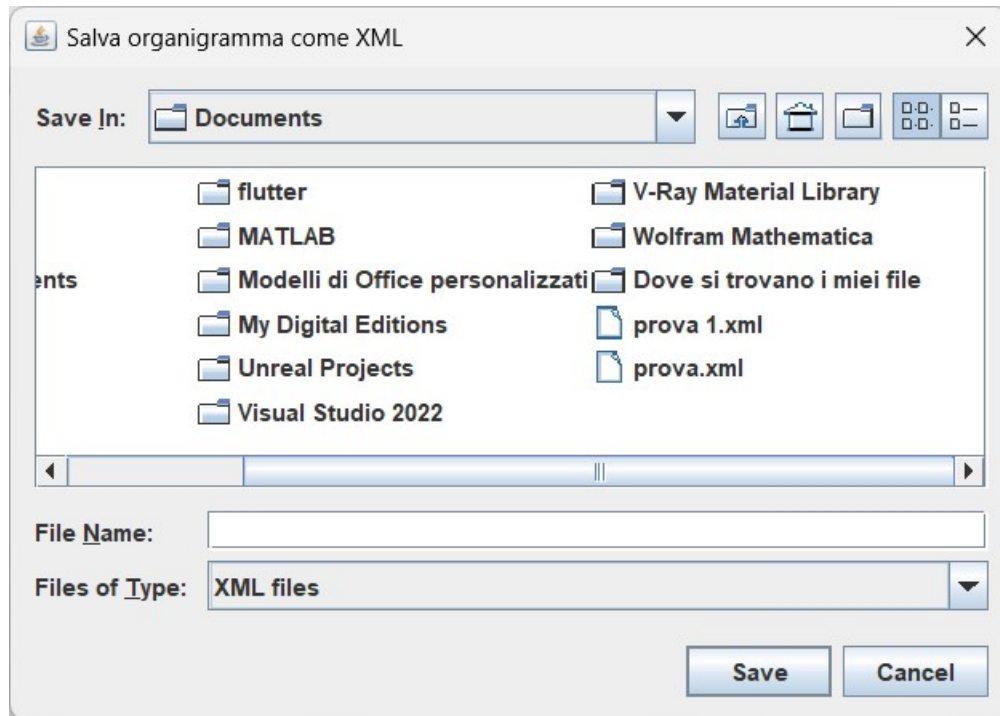


Figura 2 – Si apre quando l'utente clicca su "Salva organigramma"

Modifica unità - Consiglio di amministrazione

Rinomina unità   Aggiungi sotto-unità   Aggiungi dipendente   Elimina unità

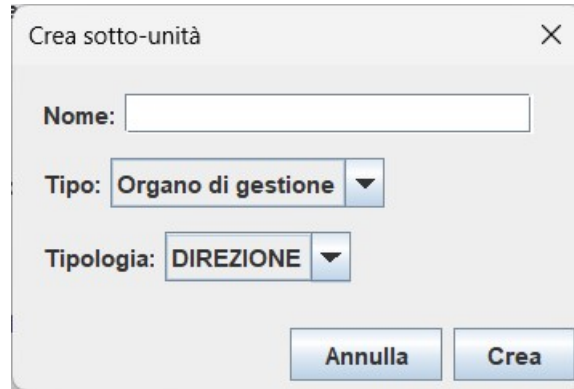
Tipo: Organo di gestione  
Tipologia: DIREZIONE  
Dipendenti:

Nome: giovanni, Matricola: 0, Ruolo: direttore   Elimina

Nome: antonio, Matricola: 1, Ruolo: venditore   Elimina

Nome: riccardo, Matricola: 2, Ruolo: venditore   Elimina

Figura 3 - Finestra "Modifica unità". Si apre al doppio click su una unità



Crea sotto-unità

Nome:

Tipo: Organo di gestione ▼

Tipologia: DIREZIONE ▼

Annulla Crea

*Figura 4 - Finestra "Crea sotto-unità". Si apre al click di "Aggiungi sotto-unità"*

Aggiungi dipendente

☒ Aggiungi dipendente esistente

Dipendenti esistenti: Nome: giovanni, Matricola: 0 ▼

Ruolo dipendente esistente:

☐ Crea nuovo dipendente

Nome del nuovo dipendente:

Ruolo del nuovo dipendente:

Annulla Aggiungi

*Figura 5 - Finestra "Aggiungi dipendente"*