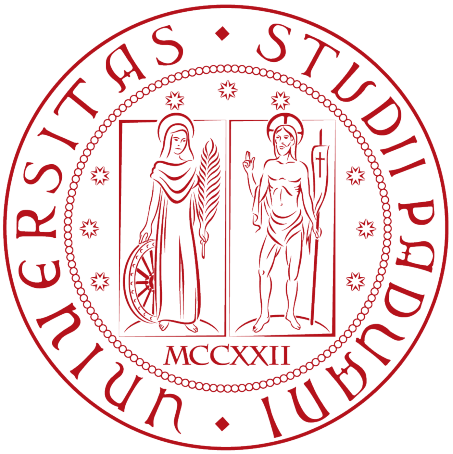
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Analisi e Implementazione di Algoritmi per problemi di Scheduling

Tesi di laurea triennale

Relatore

Prof. Luigi De Giovanni

Laureando

Beatrice Liberi

Anno Accademico 2016-2017

Sommario

Il presente documento descrive il lavoro che ho svolto durante il periodo di stage, della durata di trecento ore, presso l'azienda Trans-Cel Autotrasporti.

Lo scopo dello stage è stato lo studio di modelli per problemi di scheduling e implementazione di algoritmi euristici che restituiscano una soluzione ammissibile in tempi sufficientemente brevi per una pianificazione in tempo reale. Il progetto ha fornito a Trans-Cel Autotrasporti uno strumento che potrà essere integrato in software di pianificazione da essa sviluppati da applicare in diversi contesti per i quali sia necessaria la risoluzione di un problema di scheduling.

Gli obiettivi da raggiungere nel corso dello stage erano molteplici: in primo luogo era richiesto di assimilare i concetti di base dei problemi di scheduling, studiare la letteratura scientifica sui modelli e sugli algoritmi euristici più adatti per risolvere gli stessi; in secondo luogo era richiesto di definire formalmente con un modello matematico un problema di scheduling da risolvere, e progettare un algoritmo euristico per trovare una soluzione al problema; terzo ed ultimo obiettivo era l'implementazione dell'algoritmo euristico, integrandolo nelle librerie dell'azienda.

Indice

1 Introduzione **1**
- 1.1 L'azienda . . . 1
- 1.2 L'idea . . . 1
- 1.3 Organizzazione del testo . . . 2

2 Descrizione dello stage **3**
- 2.1 Introduzione al progetto e contestualizzazione . . . 3
- 2.2 Vincoli . . . 3
- 2.3 Obiettivi . . . 4
- 2.4 Pianificazione . . . 6
- 2.5 Ambiente di lavoro . . . 9
  - 2.5.1 Metodologia di sviluppo . . . 9
  - 2.5.2 Gestione di progetto . . . 9
  - 2.5.3 Documentazione . . . 10
  - 2.5.4 Ambiente di sviluppo . . . 10
- 2.6 Analisi preventiva dei rischi . . . 11

3 Definizione del problema **13**
- 3.1 Problemi di scheduling . . . 13
- 3.2 Scheduling nei casinò . . . 13
- 3.3 Modello . . . 13

4 Analisi dei requisiti **15**
- 4.1 Casi d'uso . . . 15
- 4.2 Tracciamento dei requisiti . . . 16

5 Progettazione e codifica **19**
- 5.1 Tecnologie e strumenti . . . 19
- 5.2 Progettazione . . . 19
- 5.3 Design Pattern utilizzati . . . 19
- 5.4 Codifica . . . 19

6 Verifica e validazione **21**
- 6.1 Generazione degli input . . . 21
- 6.2 Modello probabilistico . . . 21
- 6.3 Progettazione . . . 21
- 6.4 Esiti dei test . . . 21

7 Conclusioni **23**

7.1 Consuntivo finale 23
7.2 Raggiungimento degli obiettivi 23
7.3 Conoscenze acquisite 23
7.4 Valutazione personale 23

Glossario **25**

Acronimi **29**

Bibliografia **31**

Elenco delle figure

2.1 Diagramma di Gantt per il piano di lavoro, settimane 1-5 8
2.2 Diagramma di Gantt per il piano di lavoro, settimane 5-9 8

4.1 Use Case - UC0: Scenario principale 15

Elenco delle tabelle

2.1 Obiettivi da raggiungere a fine stage 5
2.2 Pianificazione del periodo di stage 7
2.3 Analisi preventiva dei rischi 11

4.1 Tabella del tracciamento dei requisti funzionali 17
4.2 Tabella del tracciamento dei requisiti qualitativi 17
4.3 Tabella del tracciamento dei requisiti di vincolo 17

Capitolo 1

Introduzione

In questo capitolo vengono brevemente descritte l'azienda Trans-Cel Autotrasporti presso la quale ho svolto lo stage e l'idea dalla quale è nata la necessità del progetto portato a termine durante lo stesso.
Viene inoltre presentata la suddivisione della tesi per capitoli e vengono introdotte alcune norme tipografiche che verranno utilizzate di seguito.

1.1 L'azienda

Trans-Cel Autotrasporti è un'azienda che si occupa di trasporti su gomma di merci per conto terzi su mezzi pesanti. Con una flotta di trenta camion che deve compiere carichi e scarichi in tutta l'Italia centro-settentrionale da coordinare in tempo reale, da qualche anno l'azienda ha cominciato a sviluppare, grazie ad un team di informatici e matematici, un sistema per il controllo della flotta e soprattutto per la pianificazione di viaggi, carichi e scarichi in modo da ottimizzare sia l'utilizzo dello spazio disponibile sui camion, sia i chilometri percorsi; il tutto vincolato ai tempi concordati di ritiro e consegna.
Trans-Cel Autotrasporti vede questo planning come il primo mattone di un sistema molto più grande che unirà diversi tipi di servizi utili per tutta la *supply chain*[g], non solo nel campo dei trasporti.

1.2 L'idea

Uno degli ulteriori servizi che Trans-Cel Autotrasporti vuole sviluppare consiste in un software per la gestione della *schedulazione*[g] (o *scheduling*) dei turni di lavoro.
I *problemi di scheduling*[g] ricadono generalmente nella classe *NP-Hard*[g] e risulta quindi particolarmente difficile trovare delle soluzioni ottime, e spesso anche solo soluzioni ammissibili. Inoltre, esistono diversi contesti in cui i turni devono poter essere proposti in tempi molto rapidi, ad esempio per adeguarsi a cambiamenti durante l'orizzonte di pianificazione. Uno degli utilizzi che Trans-Cel Autotrasporti potrebbe fare di questo software, nel particolare, può essere l'organizzazione delle squadre di meccanici.
Lo stage si pone dunque in questo contesto di progettazione di un framework per la risoluzione dei problemi di scheduling.

1.3 Organizzazione del testo

Il secondo capitolo descrive in dettaglio lo stage. Ne specifica il progetto da svolgere contestualizzandolo nella realtà aziendale, i requisiti richiesti, gli obiettivi da raggiungere e la pianificazione iniziale.

Il terzo capitolo approfondisce l'argomento dei problemi di scheduling e definisce nei dettagli il problema da risolvere durante lo stage.

Il quarto capitolo consiste nell'analisi dei requisiti svolta per il progetto, approfondita con diagrammi dei casi d'uso.

Il quinto capitolo presenta la progettazione svolta per il progetto, approfondita con diagrammi *UML*[g], e ne descrive la fase di codifica.

Il sesto capitolo approfondisce la fase di verifica e validazione del progetto, specificando le modalità ed i risultati ottenuti.

Nel settimo capitolo riporta le conclusioni oggettive e soggettive a cui si è giunti per il progetto.

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

* gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
* per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *parola*[g];
* i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati in *italico*.

Capitolo 2

Descrizione dello stage

Questo capitolo descrive in dettaglio lo stage. Ne specifica il progetto da svolgere contestualizzandolo nella realtà aziendale, i requisiti richiesti, gli obiettivi da raggiungere e la pianificazione iniziale.

2.1 Introduzione al progetto e contestualizzazione

Nel contesto del supply chain management in cui l'azienda opera, ha assunto una gran rilevanza l'organizzazione dei turni di singoli lavoratori o di squadre di lavoratori, in relazione alle competenze da loro possedute e richieste dai vari lavori da portare a compimento.

Il progetto di stage si prefiggeva l'obiettivo di estendere per la risoluzione dei problemi di scheduling il *framework*[g] aziendale e scrivere un'applicazione che lo utilizzasse andando a risolvere un problema specifico. L'obiettivo finale è utilizzare il framework per lo scheduling delle squadre dei meccanici, tuttavia è stato deciso che, per cominciare a studiare i problemi di scheduling e la fattibilità di risolverli con metodi *euristici*[g] e *meta-euristici*[g], fosse opportuno cominciare da un problema più semplice: l'organizzazione dei turni nei casinò. L'organizzazione dei turni dei meccanici sarà il naturale proseguimento di questo progetto, e la complessità maggiore dovuta allo schedulazione di individui singoli inseriti in un gruppo anch'esso da schedulare sarà supportata dal framework di base, robusto e già testato, sviluppato all'interno del progetto di stage. Il problema della schedulazione dei turni all'interno dei casinò è stato scelto come problema di partenza in quanto comunque di interesse per l'azienda, che ha contatti con un manager de "The Hippodrome Casino" di Londra, il quale ha sottoposto il problema in quanto, al momento, lo scheduling viene fatto completamente a mano, dovendo tenere conto di numerosi vincoli e variabili.

2.2 Vincoli

È stato concordato con Trans-Cel Autotrasporti che lo stage fosse svolto presso la sede operativa della stessa, ad Albignasego, affinché io potessi essere inserita nel contesto aziendale ed avere la possibilità di confrontarmi col resto del team di sviluppo in

maniera immediata.
Il tutor aziendale ha richiesto di scrivere ogni giorno un breve report delle attività svolte, delle difficoltà incontrate, degli obiettivi raggiunti e delle considerazioni personali su un documento condiviso su *Google Drive*[g], ha inoltre richiesto che ogni mattina prima di cominciare a lavorare si svolgesse una breve riunione con tutto il team per focalizzarsi sulle attività della giornata.
Lo stage, che prevede una durata di 300 ore, è stato pianificato su una durata di poco meno di 9 settimane prevedendo in media 6.5 ore lavorative al giorno. Tale stima è stata effettuata al ribasso per permettere una certa flessibilità per quanto riguarda l'orario di lavoro e/o eventuali giorni liberi che avrei potuto prendere per impegni universitari; l'orario di lavoro invece è stato stabilito dal lunedì al venerdì, dalle 9.00 alle 18.00 con un'ora di pausa pranzo.
Prima dell'inizio dello stage Trans-Cel Autotrasporti ha redatto un piano di lavoro in cui sono stati fissati gli obiettivi e la pianificazione settimanale per lo stage, che vengono illustrati nei paragrafi "Obiettivi" e "Pianificazione" di questo capitolo.
Per quanto riguarda i vincoli tecnologici, essi saranno trattati nel paragrafo "Ambiente di lavoro" di questo capitolo.

2.3 Obiettivi

Sono riportati di seguito gli obiettivi fissati per lo stage con rispettivo identificativo, importanza e breve descrizione.

L'identificativo (in breve, Id) è la sigla che identifica ogni requisito e rispetta la notazione *[Importanza][Identificativo]*, dove l'importanza è la sigla **Ob** / **De** / **Op** a seconda che l'obiettivo sia obbligatorio, desiderabile o opzionale; l'identificativo è un numero progressivo che identifica in modo univoco l'obiettivo.

Un obiettivo è classificato, secondo l'importanza, come:

* **Obbligatorio:** è l'importanza attribuita agli obiettivi il cui soddisfacimento dovrà necessariamente avvenire in quanto sono di importanza fondamentale per la riuscita del progetto;
* **Desiderabile:** è l'importanza attribuita agli obiettivi il cui soddisfacimento non è necessario, tuttavia desiderabile;
* **Opzionale:** è l'importanza attribuita agli obiettivi il cui soddisfacimento è del tutto opzionale in quanto renderebbero il progetto più completo.

Nel settimo capitolo è riportato un consuntivo che riporta quanti degli obiettivi riportati di seguito sono stati soddisfatti nel corso dello stage.

Tabella 2.1: Obiettivi da raggiungere a fine stage

ID	IMPORTANZA	DESCRIZIONE
Ob1	Obbligatorio	Assimilazione dei concetti di base dei problemi di scheduling
Ob2	Obbligatorio	Definizione del problema di scheduling da risolvere
Ob3	Obbligatorio	Studio della letteratura scientifica sui modelli e i metodi per problemi di schedulazione e indagine sugli algoritmi euristici più adatti al tipo di problema in esame
Ob4	Obbligatorio	Studio del framework aziendale dalla relativa documentazione
Ob5	Obbligatorio	Progettazione di un algoritmo euristico per la soluzione del problema
Ob6	Obbligatorio	Implementazione prototipale dell'algoritmo euristico
Ob7	Obbligatorio	Integrazione dell'algoritmo euristico nelle librerie dell'azienda
Ob8	Obbligatorio	Test preliminare su istanze benchmark di problemi di scheduling
Ob9	Obbligatorio	Produzione di documentazione sui moduli sviluppati
Ob10	Obbligatorio	Produzione di un manuale di utilizzo dei moduli sviluppati
De1	Desiderabile	Analisi prestazionale su diverse istanze del problema
De2	Desiderabile	Esplorazione di altri contesti applicativi
Op1	Opzionale	Definizione di un modello di programmazione matematica del problema
Op2	Opzionale	Implementazione con AMPL (o altro linguaggio di modellazione matematica) del modello di programmazione lineare intera per la soluzione esatta del problema di scheduling

2.4 Pianificazione

La pianificazione del lavoro da svolgere durante lo stage è stata costruita sulla base delle ore di lavoro previste dallo stage curricolare.
Il progetto è stato suddiviso in otto periodi:

1. **Analisi del problema.** Periodo di assimilazione dei concetti di base dei problemi di scheduling, per comprenderne contesto e applicazione; approccio al problema dello scheduling nei casinò, con definizione dei vincoli a cui si deve sottostare.

2. **Analisi dello stato dell'arte.** Periodo di approfondimento dello studio dei problemi di scheduling, sia nella teoria che nella pratica, con relativi metodi di risoluzione.

3. **Ideazione e progettazione di un algoritmo euristico.** Periodo di ricerca di algoritmi euristici per la risoluzione dei problemi di scheduling e di definizione, sulla base di questi ultimi, dello scheletro di un algoritmo per lo scheduling nei casinò.

4. **Implementazione dell'algoritmo euristico.** Periodo di familiarizzazione con il framework aziendale, di progettazione dell'architettura da integrare con lo stesso per estenderlo con la risoluzione dei problemi di scheduling, e conseguente implementazione.

5. **Integrazione e test dell'algoritmo.** Periodo di implementazione dell'applicazione per il problema specifico dello scheduling nei casinò utilizzando il framework aziendale già esteso; si sono inoltre svolti i primi test semplificati.

6. **Confronto con tecniche esatte.** Periodo di studio di modelli matematici per problemi di scheduling classici e definizione di un modello matematico per lo scheduling nei casinò, con conseguente implementazione in *AMPL*[g] e confronto fra *metodo esatto*[g] e euristica prodotta.

7. **Verifica e testing.** Periodo trasversale ai periodi di Implementazione dell'algoritmo euristico, Integrazione e test dell'algoritmo, Confronto con tecniche esatte. Prevede il testing del prodotto.

8. **Produzione di documenti, manuali e relazioni.** Periodo trasversale ai periodi di Ideazione e progettazione di un algoritmo euristico, Implementazione dell'algoritmo euristico, Integrazione e test dell'algoritmo, Confronto con tecniche esatte, Verifica e testing. Prevede la scrittura di documentazione per il codice, comprendente il manuale sviluppatore e il manuale utente; la scrittura di relazioni sui test effettuati e sullo studio del problema.

Di seguito vengono riportate in dettaglio le attività previste per ogni periodo, specificando il numero di ore previste per ognuna e il periodo in cui ne è pianificato lo svolgimento.

Tabella 2.2: Pianificazione del periodo di stage

ATTIVITÀ	ORE	DAL	AL
1. Analisi del problema			
Assimilazione dei concetti di base per problemi di scheduling	10	10/07	11/07
Definizione delle caratteristiche del problema di scheduling da risolvere	10	11/07	12/07
2. Analisi dello stato dell'arte			
Studio della letteratura di base su problemi di scheduling	15	13/07	14/07
Ricerca ed analisi degli algoritmi più adatti	15	17/07	18/07
3. Ideazione e progettazione di un algoritmo euristico			
Individuazione dei blocchi di base da algoritmi pre-esistenti	15	19/07	20/07
Integrazione dei blocchi	10	21/07	24/07
Definizione dell'algoritmo di soluzione	15	24/07	25/07
4. Implementazione dell'algoritmo euristico			
Studio ed assimilazione del framework aziendale	15	26/07	27/07
Progettazione dei moduli	15	28/07	31/07
Implementazione	40	01/08	07/08
5. Integrazione e test dell'algoritmo			
Integrazione dell'algoritmo nelle librerie dell'azienda	30	08/08	11/08
Test preliminari su istanze semplificate	15	21/08	22/08
6. Confronto con tecniche esatte			
Studio dei modelli in letteratura	10	23/08	24/08
Definizione di un modello matematico	10	24/08	25/08
Implementazione del modello in AMPL	10	28/08	29/08
Confronto con i risultati dell'euristica e analisi delle presentazioni	10	29/08	30/08
7. Verifica e testing	20	01/08	04/09
8. Produzione di documenti, manuali e relazioni	35	17/07	15/09

Figura 2.1: Diagramma di Gantt per il piano di lavoro, settimane 1-5

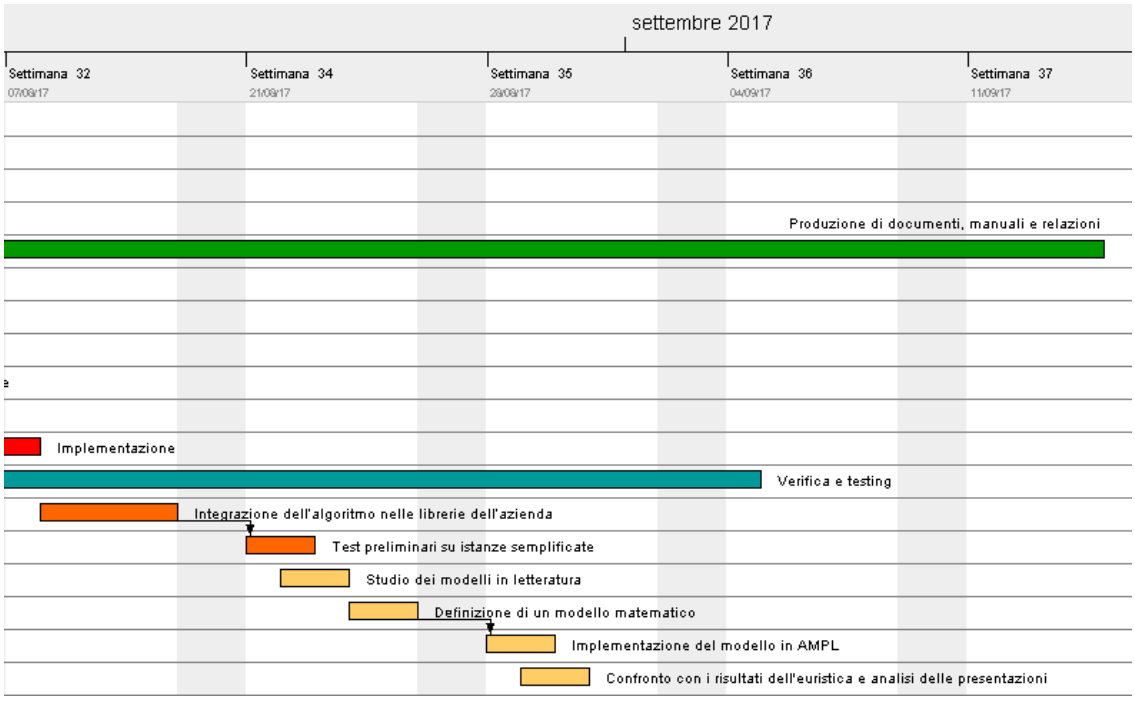


Figura 2.2: Diagramma di Gantt per il piano di lavoro, settimane 5-9

2.5 Ambiente di lavoro

2.5.1 Metodologia di sviluppo

La metodologia di sviluppo adottata in Trans-Cel Autotrasporti non si riconduce ad un modello in particolare, nonostante presenti delle affinità sia con il *modello incrementale*[g] che con il metodo *agile*[g] di tipo *scrum*[g]. Infatti, per la pianificazione del lavoro a lungo termine sono stati definite delle fasi, affini idealmente a degli incrementi, associate a *milestones*[g] ben definite; tuttavia all'interno di ogni incremento l'approccio allo sviluppo diventa più simile al metodo agile, in quanto il Project Manager, ma anche i membri del team di sviluppo stesso, ad inserire e assegnare/assegnarsi i task del backlog. Affine al metodo scrum è anche il controllo quotidiano dell'avanzamento dei lavori rappresentato dalla breve riunione mattutina con tutti i membri del team. Questa metodologia di sviluppo funziona all'interno di Trans-Cel Autotrasporti poiché il team sviluppa un software per l'azienda stessa e quindi il colloquio e l'interazione fra sviluppatori e stakeholders è estremamente facilitato.

2.5.2 Gestione di progetto

Versionamento

Per quanto riguarda il versionamento, l'azienda ha un *repository*[g] *Git*[g] privato su Bitbucket[1], a cui mi è stato dato l'accesso in lettura. Il ramo master di tale repository costituisce il codice del framework e delle applicazioni aggiornato all'ultima versione e funzionante; mentre per tutte le modifiche in via di sviluppo o le estensioni, i membri del team lavorano su dei branch personali. Anche per me ne è stato creato uno, nel quale ho potuto modificare il framework e scrivere la mia applicazione.

Ticketing

Le attività da svolgere nel tempo medio-breve, che vanno a costituire il backlog, sono gestite tramite la piattaforma di project management "Taiga"[2]. I membri del team possono aggiungere attività al backlog che inizialmente ricadono nella categoria "to do", quando vengono assegnate a qualcuno che incomincia a occuparsene nella categoria "In progress" e poi successivamente in "To test" e "Complete". Alla stessa attività possono venire assegnati più membri del team. Anche per me è stata creata un'utenza con accesso ai ticket.

Altri strumenti per la gestione di progetto

- **Google Drive**[3]: per la condivisione veloce di file come guide, documentazioni, riferimenti utili e documenti informali.

- **Google Calendar**[4]: per la gestione degli eventi ed impegni.

[1] https://bitbucket.org/
[2] https://taiga.io/
[3] https://www.google.com/intl/it_ALL/drive/
[4] https://www.google.com/calendar

2.5.3 Documentazione

Per quanto riguarda la produzione di documenti, manuali e relazioni richiesti, l'azienda non ha uno standard e mi ha dunque delegato la scelta di che software utilizzare per la loro produzione.

Doxygen

Per quanto riguarda la documentazione del codice, la mia scelta è ricaduta su Doxygen[5]. Doxygen è uno strumento per la generazione di documentazione a partire dal codice commentato per molti linguaggi di programmazione, fra i quali il C++. Genera documentazione sia in HTML che in LaTeX, corredandola con diagrammi delle classi ed estraendo i commenti direttamenti dai sorgenti per commentare attributi e metodi delle classi.

Diagrammi UML e Use Case

Per la progettazione è stato necessario modellare con UML, la mia scelta è ricaduta su Astah Professional[6], disponibile in versione gratuita grazie alla licenza accademica, perché identificato come il software più completo fra quelli dedicati. Astah Professional supporta sia i costrutti di UML 1.x che di UML 2.x e permette di modellare casi d'uso, i diagrammi delle classi, degli oggetti, delle attività e di sequenza.

2.5.4 Ambiente di sviluppo

Il framework aziendale da estendere è programmato in C++ ed offre una base per l'implementazione di algoritmi di ottimizzazione. Consiste di librerie per l'utilizzo di grafi, di algoritmi euristici di tipo *greedy*[g] e di meta-euristiche, ad esempio *Tabu Search*[g], che il l'estensione del framework andrà a sfruttare.

IDE

Il progetto originale è nato su Visual Studio, perciò è consigliato utilizzare tale *IDE*[g] per lo sviluppo in quanto i file di progetto con estensione *.vcxproj* che contengono le informazioni necessarie alla compilazione sono già presenti.
Trattandosi comunque di codice scritto in C++, può essere utilizzato qualsiasi IDE ne fornisca il supporto.

Sistemi operativi

Poiché il progetto è nato su Visual Studio, la scelta più naturale per il sistema operativo di scelta è rappresentata da Windows; ciononostante i membri possono lavorare indifferentemente anche su Linux o Mac OS.

[5] http://www.stack.nl/~dimitri/doxygen/
[6] http://astah.net/editions/professional

2.6 Analisi preventiva dei rischi

Nella seguente tabella viene riportata l'analisi dei rischi in cui si può incorrere nel corso del progetto.
Nel settimo capitolo è riportata un'attualizzazione dei rischi che si sono effettivamente verificati.

Tabella 2.3: Analisi preventiva dei rischi

Descrizione	Trattamento	Rischio
Scelte di bad design nella progettazione del framework aziendale esistente, che possono rendere difficoltosa l'estensione.	Se verificato, si potrà avere un confronto con gli sviluppatori del framework per cercare assieme una soluzione.	Occorrenza: Alta
		Pericolosità: Medio-Alta
Bug all'interno del framework poiché alcune sue parti si trovano ancora in fase di sviluppo.	Se verificato, si potrà avere un confronto con gli sviluppatori del framework per cercare assieme una soluzione.	Occorrenza: Bassa
		Pericolosità: Alta

Capitolo 3

Definizione del problema

Questo capitolo approfondisce l'argomento dei problemi di scheduling e definisce nei dettagli il problema da risolvere durante lo stage.

3.1 Problemi di scheduling

3.2 Scheduling nei casinò

3.3 Modello

Capitolo 4

Analisi dei requisiti

consiste nell'analisi dei requisiti svolta per il progetto, approfondita con diagrammi dei casi d'uso.

4.1 Casi d'uso

Per lo studio dei casi di utilizzo del prodotto sono stati creati dei diagrammi. I diagrammi dei casi d'uso (in inglese *Use Case Diagram*) sono diagrammi di tipo Unified Modeling Language (UML) dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso. Essendo il progetto finalizzato alla creazione di un tool per l'automazione di un processo, le interazioni da parte dell'utilizzatore devono essere ovviamente ridotte allo stretto necessario. Per questo motivo i diagrammi d'uso risultano semplici e in numero ridotto.

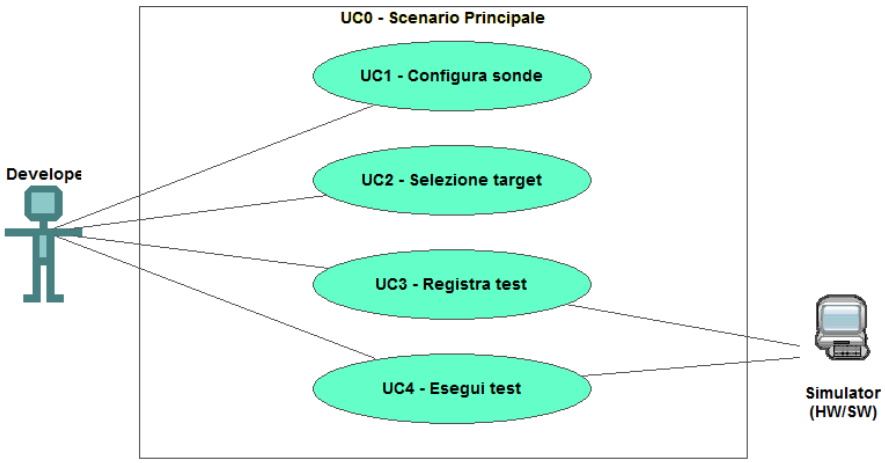


Figura 4.1: Use Case - UC0: Scenario principale

UC0: Scenario principale

Attori Principali: Sviluppatore applicativi.

Precondizioni: Lo sviluppatore è entrato nel plug-in di simulazione all'interno dell'IDE.

Descrizione: La finestra di simulazione mette a disposizione i comandi per configurare, registrare o eseguire un test.

Postcondizioni: Il sistema è pronto per permettere una nuova interazione.

4.2 Tracciamento dei requisiti

Da un'attenta analisi dei requisiti e degli use case effettuata sul progetto è stata stilata la tabella che traccia i requisiti in rapporto agli use case.
Sono stati individuati diversi tipi di requisiti e si è quindi fatto utilizzo di un codice identificativo per distinguerli.
Il codice dei requisiti è così strutturato R(F/Q/V)(N/D/O) dove:

R = requisito

F = funzionale

Q = qualitativo

V = di vincolo

N = obbligatorio (necessario)

D = desiderabile

Z = opzionale

Nelle tabelle 4.1, 4.2 e 4.3 sono riassunti i requisiti e il loro tracciamento con gli use case delineati in fase di analisi.

Tabella 4.1: Tabella del tracciamento dei requisti funzionali

Requisito	Descrizione	Use Case
RFN-1	L'interfaccia permette di configurare il tipo di sonde del test	UC1

Tabella 4.2: Tabella del tracciamento dei requisiti qualitativi

Requisito	Descrizione	Use Case
RQD-1	Le prestazioni del simulatore hardware deve garantire la giusta esecuzione dei test e non la generazione di falsi negativi	-

Tabella 4.3: Tabella del tracciamento dei requisiti di vincolo

Requisito	Descrizione	Use Case
RVO-1	La libreria per l'esecuzione dei test automatici deve essere riutilizzabile	-

Capitolo 5

Progettazione e codifica

presenta la progettazione svolta per il progetto, approfondita con diagrammi UML[g]*, e ne descrive la fase di codifica.*

5.1 Tecnologie e strumenti

Di seguito viene data una panoramica delle tecnologie e strumenti utilizzati.

Tecnologia 1

Descrizione Tecnologia 1.

Tecnologia 2

Descrizione Tecnologia 2

5.2 Progettazione

Namespace 1

Descrizione namespace 1.

Classe 1: Descrizione classe 1

Classe 2: Descrizione classe 2

5.3 Design Pattern utilizzati

5.4 Codifica

Capitolo 6

Verifica e validazione

approfondisce la fase di verifica e validazione del progetto, specificando le modalità ed i risultati ottenuti.

6.1 Generazione degli input

6.2 Modello probabilistico

6.3 Progettazione

6.4 Esiti dei test

Capitolo 7

Conclusioni

riporta le conclusioni oggettive e soggettive a cui si è giunti per il progetto.

7.1 Consuntivo finale

7.2 Raggiungimento degli obiettivi

7.3 Conoscenze acquisite

7.4 Valutazione personale

Glossario

Agile In ingegneria del software, è un insieme di metodi di sviluppo del software emersi a partire dai primi anni 2000 e fondati su insieme di principi comuni, direttamente o indirettamente derivati dai principi del "Manifesto per lo sviluppo agile del software"[1]. Tale manifesto si può riassumere in quattro punti:

1. le persone e le interazioni sono più importanti dei processi e degli strumenti;
2. è più importante avere software funzionante che documentazione;
3. bisogna collaborare con i clienti oltre che rispettare il contratto;
4. bisogna essere pronti a rispondere ai cambiamenti oltre che aderire alla pianificazione.

Un esempio di metodo di sviluppo di tipo agile è il metodo *scrum*[g]. 9

Algoritmo euristico I metodi di risoluzione dei problemi di ottimizzazione combinatoria ricadono in due categorie: i metodi esatti (vd. *Metodo esatto*) e metodi euristici. In matematica e informatica un metodo euristico è un particolare tipo di algoritmo progettato per risolvere un problema di ricerca operativa più velocemente (qualora i metodi esatti siano troppo lenti) o per trovare una soluzione approssimata (qualora i metodi esatti falliscano nel trovare una soluzione esatta). Il risultato viene ottenuto cercando di equilibrare ottimalità, completezza, accuratezza e velocità di esecuzione.
Possono essere applicati quando la soluzione è data selezionando il miglior sottoinsieme di un dato insieme di elementi e sono costruttivi, ovvero partono da una soluzione vuota e ad ogni iterazione aggiungono un elemento applicando dei criteri di selezione[2]. 3

Algoritmo Greedy Un algoritmo greedy è un algoritmo euristico che ad ogni iterazione compie una scelta "golosa", ovvero la migliore (in termini di impatto sulla funzione obiettivo) da prendere in quel momento. 10

AMPL È un linguaggio ad alto livello per descrivere e risolvere grossi e complicati problemi di programmazione matematica (per esempio problemi di ottimizzazione e di scheduling). 6

Framework Nello sviluppo software, il framework è un'architettura logica di supporto su cui un software può essere progettato e realizzato; la sua funzione è quella di creare una infrastruttura generale, lasciando al programmatore il contenuto vero e proprio dell'applicazione. 3

[1] *Manifesto per lo sviluppo agile del software.* URL: http://agilemanifesto.org/iso/it/manifesto.html.

[2] L. De Giovanni. *Method and Models for Combinatorial Optimization.*

Git Software di controllo versione distribuito utilizzabile da interfaccia a riga di comando. È uno dei pù diffusi strumenti di controllo versione.. 9

Google Drive Servizio fornito da Google in ambiente cloud computing che offre funzioni di condivisione e hosting di file, permettendone la modifica collaborativa su uno spazio gratuito di 15 GB.. 4

IDE È un software che, in fase di programmazione, aiuta il programmatore nello sviluppo del codice sorgente fornendo un controllo sintassi in tempo reale e una serie di strumenti e funzionalità di supporto allo sviluppo e al debugging. 10

Meta-euristica In matematica e informatica è una strategia per esplorare in maniera efficiente lo spazio delle soluzioni per trovare delle soluzioni vicine all'ottimo e scappare dagli ottimi locali. Non sono algoritmi costruttivi e necessitano quindi di una soluzione per poterne trovarne un'altra migliore. 3

Metodo esatto I metodi di risoluzione dei problemi di ottimizzazione combinatoria ricadono in due categorie: i metodi esatti e metodi euristici (vd. *Algoritmo euristico*). I primi sono metodi che riescono a fornire una soluzione ottima, a differenza dei secondi che riescono a fornire una soluzione ammissibile ma senza la garanzia che sia ottimale.[3] Esempi di algoritmi esatti sono il simplesso o il branch-and-bound. 6

Milestone Traguardi intermedi ed importanti nello svolgimento di un progetto; sono spesso fissati in fase di pianificazione. 9

Modello incrementale In ingegneria del software, è una metodologia di sviluppo software in cui il cliente identifica, a grandi linee, i requisiti fondamentali e quelli desiderabili del prodotto software che vuole ottenere; viene poi deciso il numero di incrementi da effettuare, tenendo conto del fatto che ogni singolo incremento costituisce un sottoinsieme delle funzionalità del prodotto software. Una volta che gli incrementi sono stati identificati, si definiscono in dettaglio i requisiti che devono essere soddisfatti dagli incrementi prima di procedere allo sviluppo.. 9

NP-Hard vd. *teoria della complessità computazionale*[g]. 1

Problema di Schedulazione/Scheduling Un problema di schedulazione definisce le variabili decisionali, una regione ammissibile entro la quale le variabili decisionali possono variare e una funzione obiettivo per determinare la migliore *schedulazione*[g] ammissibile. 1

Repository Ambiente di un sistema informativo in cui vengono gestiti i metadati attraverso tabelle relazionali. L'insieme di tabelle, regole e motori di calcolo tramite cui si gestiscono i metadati prende il nome di metabase.. 9

Schedulazione/Scheduling La schedulazione è una forma di processo decisionale che consiste nell'allocare risorse finite in modo tale che un dato obiettivo venga ottimizzato, sincronizzando e tempificando la sequenza delle operazioni. 1, 26

[3] Giovanni, *Method and Models for Combinatorial Optimization*.

Scrum Metodo di sviluppo software che rientra fra i metodi agile. Prevede di dividere il progetto in blocchi rapidi di lavoro (Sprint) alla fine di ciascuno dei quali creare un incremento del software. Esso indica come definire i dettagli del lavoro da fare nell'immediato futuro e prevede vari meeting con caratteristiche precise per creare occasioni di ispezione e controllo del lavoro svolto. 9, 25

Supply chain La supply chain (in italiano, "catena di distribuzione") è il sistema di organizzazioni, persone, attività, informazioni e risorse coinvolte nel portare un prodotto o un servizio dal fornitore al cliente. 1

Tabu Search La Tabu Search è una tecnica meta-euristica utilizzata per la soluzione di numerosi problemi di ottimizzazione, tra cui problemi di scheduling. Essa consiste nel partire da una soluzione iniziale ed eseguire una serie di "mosse" che portano ad una nuova soluzione all'interno del vicinato per la quale la funzione obiettivo f assume un valore migliore del valore attuale. Per sfuggire gli ottimi locali, la Tabu Search permette un certo numero di mosse peggioranti rendendo tabù, proibite, le ultime mosse eseguite per evitare di "tornare indietro". 10

Teoria della complessità computazionale In informatica, la teoria della complessità computazionale è una branca della teoria della computabilità che studia le risorse minime necessarie (principalmente tempo di calcolo e memoria) per la risoluzione di un problema. I problemi sono classificati in differenti classi di complessità, in base all'efficienza del migliore algoritmo noto in grado di risolvere quello specifico problema:

* $\mathcal{P}$ (*Polynomial-time problems*) - Contiene tutti i problemi decisionali che possono essere risolti da una macchina di Turing deterministica in tempo polinomiale. La classe $\mathcal{P}$ può essere caratterizzata come quella classe di problemi per i quali si è in grado di trovare una soluzione in tempo polinomiale.
* $\mathcal{NP}$ (*Nondeterministic polynomial-time problems*) - Contiene tutti i problemi decisionali che possono essere risolti da una macchina di Turing non deterministica in tempo polinomiale. La classe $\mathcal{NP}$ può essere caratterizzata come quella classe di problemi per i quali non si è in grado di trovare una soluzione in tempo polinomiale, ma si è in grado di verificare se una possibile soluzione è effettivamente tale in tempo polinomiale.
  In particolare, $\mathcal{P} \subseteq \mathcal{NP}$.
* $\mathcal{NP} - \mathcal{C}$ o NP-Completi - Sono i più difficili problemi nella classe $\mathcal{NP}$: se si trovasse un algoritmo in grado di risolvere in tempo polinomiale un qualsiasi problema in $\mathcal{NP} - \mathcal{C}$, allora si potrebbe usarlo per risolvere in tempo polinomiale ogni problema in $\mathcal{NP}$.
  In particolare, $\mathcal{NP} - \mathcal{C} \subseteq \mathcal{NP}$.
* $\mathcal{H}$ o NP-Hard (*Nondeterministic polynomial-time hard problems*) - Classe di problemi (non limitata ai soli problemi di decisione, ma estesa anche ai problemi di ottimizzazione, ad esempio) che può essere definita informalmente come la classe dei problemi non meno difficili dei problemi $\mathcal{NP} - \mathcal{C}$, che a loro volta sono per definizione i più difficili della classe $\mathcal{NP}$. Si noti che non è sempre vero che $\mathcal{H} \subseteq \mathcal{NP}$.

(*Teoria della complessità computazionale*. URL: https://it.wikipedia.org/wiki/Teoria_della_complessit%C3%A0_computazionale.) . 26

UML in ingegneria del software *UML, Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. L'*UML* svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico. 2

Acronimi

AMPL A Mathematical Programming Language. 25

IDE Integrated Developement Environment. 26

UML Unified Modeling Language. 15, 27

Bibliografia

Riferimenti bibliografici

A Hybrid Heuristic Ordering and Variable Neighbourhood Search for the Nurse Rostering Problem. School of Computer Science e Information Technology, University of Nottingham, 2005.

Blöchliger, Ivo. *Modeling staff scheduling problems. A tutorial*. European Journal of Operational Research, 2003.

Broos Maenhout, Mario Vanhoucke. *A Hybrid Scatter Search Heuristic for Personalized Crew Rostering in the Airline Industry*. Faculteit Economie en Bedrijfskunde, 2007.

Giovanni, L. De. *Method and Models for Combinatorial Optimization* (cit. alle pp. 25, 26).

Siti web consultati

Manifesto per lo sviluppo agile del software. URL: http://agilemanifesto.org/iso/it/manifesto.html (cit. a p. 25).

Teoria della complessità computazionale. URL: https://it.wikipedia.org/wiki/Teoria_della_complessit%C3%A0_computazionale (cit. a p. 27).