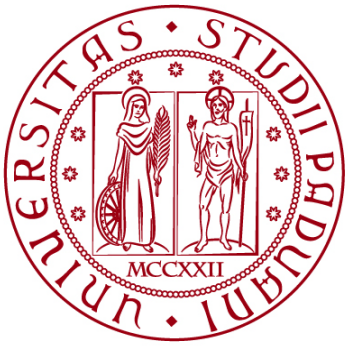
UNIVERSITÀ DEGLI STUDI DI PADOVA
DIPARTIMENTO DI INGEGNERIA CIVILE, EDILE E AMBIENTALE
Department Of Civil, Environmental and Architectural Engineering

Corso di Laurea Magistrale in Ingegneria Civile
Curriculum Progettazione tecnologia e recupero edilizio



TESI DI LAUREA

Pianificazione computazionale delle opere edili basata su modelli informativi openBIM – IFC: analisi di modelli di processo e sviluppo di algoritmi in python

Relatore: **Prof. Carlo Zanchetta**
Tutor: **Ing. Andrea Zamborlini**

Laureando: **Michele Berlato**

ANNO ACCADEMICO 2022/2023

SOMMARIO

ABSTRACT

La pianificazione delle opere edili rimane, per la maggior parte dei casi, tutt'ora in gran parte manuale dato che i sistemi di pianificazione assistita proposti negli anni non hanno soddisfatto i pianificatori. L'obiettivo della tesi è fornire ai pianificatori uno strumento che automatizzi la pianificazione dei lavori edili basandosi su modelli informativi openBIM-IFC. Il sistema creato non è fine a sé stesso, ma è la prima parte di percorso che punta a creare una pianificazione automatizzata che si possa poi reintegrare nel modello BIM. Durante la tesi si analizzano i modelli di processo che portano alla realizzazione di un'opera edile e tramite python si sviluppano degli algoritmi che permettono di automatizzare la pianificazione delle principali fasi della costruzione. Il sistema è poi testato su un caso studio che ne conferma la validità e pone buone prospettive per continuare la ricerca su questa strada in modo da migliorare il sistema e raggiungere i prossimi obiettivi prefissati.

The planning of construction works remains, for the most part, still largely manual as the assisted planning systems proposed over the years have not satisfied planners. The objective of this thesis is to provide planners with a tool that automates the planning of construction works based on openBIM-IFC information models. The system created is not an end in itself, but is the first part of a path that aims to create automated planning that can then be reintegrated into the BIM model. During the thesis, the process models leading to the realisation of a construction project are analysed and algorithms are developed using python to automate the planning of the main construction phases. The system is then tested on a case study, which confirms its validity and poses good prospects for continuing research along this path in order to improve the system and achieve the next set goals.

INTRODUZIONE

In tutti i progetti di costruzione, soprattutto in quelli più complessi, è necessario pianificare le lavorazioni in cantiere in modo da rendere il processo di costruzione efficiente e sicuro. Il progetto è fondamentale ma è altrettanto fondamentale la pianificazione della costruzione dell'edifico, la quale non può essere casuale. Infatti, una buona pianificazione può portare ad un notevole risparmio in termine di tempo e di costo, oltre che una migliore gestione della sicurezza dei lavoratori. Al giorno d'oggi quando si progetta un edificio non lo si rappresenta più in due dimensioni ma tramite dei modelli tridimensionali, in particolare questi modelli non sono semplici rappresentazioni grafiche di un edifico ma sono veri e propri contenitori di informazioni sull'edifico. Il modello informativo dell'edificio (BIM) è il punto di partenza per una buona pianificazione dei lavori in quanto esso ci fornisce tutti i dati dell'edificio, ma spesso è poco utilizzato o non utilizzato al massimo delle sue potenzialità. Questo perché, la pianificazione dei lavori edili è spesso, ancora oggi, effettuata manualmente leggendo le informazioni dal BIM ma non utilizzandole direttamente. Per una corretta pianificazione è necessario seguire una serie di passaggi e fare dei ragionamenti i quali spesso sono basati sull'esperienza del pianificatore, anche se con il giusto metodo il tutto si potrebbe automatizzare. Nonostante l'avanzamento dei mezzi informatici c'è uno scarso utilizzo di quest'ultimi per assistere il pianificatore automatizzando in parte la pianificazione. Negli anni sono state proposte delle metodologie per la pianificazione assistita dei lavori, le quali però rimangono ad oggi poco diffuse per una serie di motivi che si analizzeranno.

Per superare gli attuali limiti si propone una sistema di pianificazione delle opere edili basato su modelli openBIM-IFC. Il sistema parte da un modello BIM esportato in formato IFC, il quale è un formato dati internazionale e opensource, per utilizzare le informazioni contenute in esso in modo da sviluppare una pianificazione automatizzata dei lavori edili. I formati citati verranno approfonditi durante la tesi in modo da evidenziarne i pregi, ed eventualmente i difetti. Il sistema che si sviluppa in questa tesi non risulta però fine a sé stesso, infatti esso è parte di un'idea più grande, in cui si vuole sviluppare un sistema di pianificazione che sia in grado poi di ricollegarsi al modello BIM inserendo come attributi dei singoli elementi la durata della loro lavorazione. Di conseguenza questa è una prima tesi che inizia a trattare un argomento ampio con lo scopo di mettere le basi della pianificazione automatizzata utili agli sviluppi futuri appena discussi. Il metodo sfrutta alcuni modelli di processo, elaborati dall'autore, i quali mettono le regole di base per una pianificazione automatizzata, la quale si esegue tramite il linguaggio di programmazione Python. In pratica si elabora un algoritmo che tramite il modello BIM in formato IFC e alcune regole riesce ad esportare un cronoprogramma dei lavori edili, il quale sarà visualizzabile tramite il software opensource Project Libre.

La tesi nel capitolo primo analizza cos'è un cronoprogramma dei lavori edili, quali sono

le sue forme, le sue finalità e le sue fasi, in modo da comprendere come si esegue una pianificazione manuale e quali sono le sue ipotesi. Successivamente sempre nel primo capitolo si passa alla spiegazione di che cos'è il BIM, che cos'è IFC e cosa si intende per pianificazione 4D, poi avendo tutti gli strumenti necessari si passa a citare alcuni articoli interessanti che propongono sistemi di pianificazione assistita. Nel capitolo secondo si analizza la letteratura letta, si definiscono gli obiettivi della tesi e la metodologia da seguire per eseguirli. Nel capitolo terzo si presentano ampiamente i passaggi metodologici e un caso studio a confermare quanto spiegato nella metodologia. Si conclude con l'analisi dei risultati e le conclusioni, accompagnate dai future works.

Capitolo Primo

PIANIFICAZIONE MANUALE E PIANIFICAZIONE ASSISTITA DEI LAVORI EDILI

In questo capitolo si approfondisce il contesto disciplinare normativo e tecnologico in cui si sviluppa in primo luogo la pianificazione manuale dei lavori edili ed in secondo luogo la pianificazione assistita dei lavori edili. Infatti, per comprendere al meglio la pianificazione assistita e svilupparla è necessario partire dalla classica pianificazione manuale.

1 PIANIFICAZIONE MANUALE DEI LAVORI EDILI

Si vuole inizialmente comprendere come si sviluppa una pianificazione manuale e quali sono i suoi aspetti principali. Tipicamente una pianificazione dei lavori edili necessità di un cronoprogramma il quale è fondamentale per una buona riuscita del progetto. In questo paragrafo si va ad analizzare cos'è un cronoprogramma dei lavori edili, quali sono le sue forme, le sue finalità e le sue fasi. I paragrafi che seguiranno sono tratti dagli scritti di Angelotti Andrea (2021) e Luigi Ottaiano (2016).

1.1 Cos'è il cronoprogramma dei lavori edili?

Il cronoprogramma è la rappresentazione grafica di una serie di eventi di qualsiasi tipo, redatto con l'intento di ottimizzare i tempi del progetto generale, con l'occhio rivolto a tutte le problematiche attinenti alla sua realizzazione di cui le principali sono di ordine contrattualistico, economico e antinfortunistico. Quindi il cronoprogramma dei lavori edili è un elaborato, tipicamente di tipo grafico, adottato per la pianificazione dei lavori in cantiere, e di questi ne rappresenta lo sviluppo temporale, imposto o scelto. Esso deve comprendere tutte le fasi di realizzazione e gestione di un'opera, con le corrette tempistiche, ed inoltre è la base per una corretta gestione economico-finanziaria e per la tutela dei lavoratori. Infatti, è propedeutico allo sviluppo delle seguenti attività: gestione delle risorse (personale, materiali, mezzi d'opera); pianificazione della sicurezza; gestione degli ordini; valutazione del rendimento atteso; valutazione del BEP (Break event point = punto di pareggio) economico e finanziario.

Il cronoprogramma, in generale, è applicato in tutte le produzioni per commesse e in particolare nel settore delle costruzioni di qualsiasi tipo (edili, industriali, navali, telecomunicazioni, ecc.) dove questo elaborato è obbligatorio per tutti i progetti esecutivi di un'opera e costituisce parte integrante del contratto. Infatti, il cronoprogramma diventa un vicolo contrattualistico perché stabilisce esattamente le tempistiche di tutte le lavorazioni e di conseguenza se l'azienda non le rispetta il committente può chiedere dei risarcimenti attraverso delle cause giuridiche. Quindi l'azienda incaricata dei lavori deve stare molto attenta nel redigere il cronoprogramma in quanto può diventare un'arma a doppio taglio.

Nella realizzazione di un cronoprogramma se fossimo incaricati della sua esecuzione senza alcun limite al carico dei costi, certamente, ci orienteremmo ad eseguire il progetto nel più breve tempo possibile, viceversa se il vincolo fosse di tipo economico privilegeremmo le componenti di natura economico-finanziaria. Il punto di equilibrio tra i due casi estremi, unitamente alla presa in carico delle altre problematiche presenti nella redazione di un cronoprogramma, è determinato da una serie di compromessi attraverso i quali è consentito individuare e adottare la soluzione migliore per la pianificazione in base alle esigenze emerse, compatibilmente con le effettive possibilità di applicarla. In ultima analisi si può dire che la redazione di un cronoprogramma verte ad affermare, nell'esecuzione di un progetto, la realizzazione del maggior profitto dalle risorse introdotte con il minor sforzo possibile per il conseguimento del risultato.

1.1.1 Work Breakdown Structure (WBS)

Lo sviluppo di una pianificazione adeguata prevede, come primo passo, la redazione di una Work Breakdown Structure (WBS), nella quale avviene l'individuazione delle attività elementari (tasks) in cui viene scomposto il progetto, le quali verranno poi raccolte in macro-attività (work packages). La WBS di progetto è quindi una rappresentazione grafica "ad albero" che rappresenta in modo gerarchico la scomposizione del lavoro da svolgere, aumentandone la possibilità di controllo ma soprattutto evidenziandone i flussi di lavoro. La suddivisione di opere più o meno complesse prevede più fasi e più livelli di sottofasi il cui insieme costituisce la WBS dell'opera medesima. Maggiormente viene dettagliato il processo produttivo più agevole e precisa è la costruzione del cronoprogramma ed il suo controllo.

La WBS è una fondamentale tecnica di management messa a punto negli Stati Uniti che presenta il progetto secondo una scomposizione sistematica in parti di uguale livello di definizione. È in sostanza un diagramma a blocchi o organigramma rispetto al quale sono maggiormente specificate le funzioni di correlazione tra i diversi livelli gerarchici ed in cui possono venire definiti anche i tempi ed i costi relativi alle singole operazioni. La WBS è costituita da una serie di caselle che individuano le parti del progetto e da linee di congiunzione che stabiliscono una relazione gerarchica tra le caselle. Al vertice del diagramma sta generalmente una sola casella, che può prendere nome di master. La WBS, pertanto, non è altro che una forma immediata di rappresentazione della struttura di un progetto e costituisce in genere il supporto per analisi successive più specifiche (ad esempi analisi delle criticità degli elementi d'opera). La stesura della WBS passa attraverso tre aspetti principali:

a) La struttura di una WBS può essere orizzontale o verticale, a seconda dello spazio disponibile per la rappresentazione. Si può anche decidere di utilizzare la struttura verticale solo a partire da un certo livello del diagramma.

b) Il sistema di codificazione che può essere numerico, alfabetico o alfanumerico, serve per rendere identificabili in maniera chiara ed univoca tutti gli elementi del diagramma e consentire quindi richiami in altre parti del documento.

c) Il numero di livelli in cui può essere articolato un diagramma WBS è funzione del grado di scomposizione del progetto e quindi del grado di complessità della commessa, definito in base ad un'analisi preliminare dei rischi. Il numero di livelli comunque è generalmente non superiore a tre o quattro in quanto diagrammi troppo lunghi risultano di non immediata comprensione e richiedono troppo spazio per la rappresentazione (una corretta dimensione del diagramma è quella che può essere contenuta all'interno di un foglio di formato A4). Quando si rende necessario un numero maggiore di livelli, si può dividere il progetto in più sottoprogetti da trattare in diagrammi distinti. È necessario, a questo proposito, che i sottoprogetti siano tra di loro autonomi; una suddivisione utile può attuarsi in corrispondenza delle diverse parti dell'opera affidate in subappalto.

Il punto chiave di una WBS è la sua completezza: un diagramma incompleto in alcune delle sue parti può essere equivoco per chi lo legge. Dall'altro lato però, il rischio principale è quello di renderle troppo ampie, così da facilitare un'involuzione del processo con un'eccessiva produzione di documenti. Per evitare questi rischi occorre studiare un sistema di codifica che preventivamente tenga conto delle esigenze di ciascun livello organizzativo. Per quanto riguarda la scelta delle funzioni di scomposizione gerarchica del progetto, si può scegliere tra una delle seguenti: fase lavorativa o progettuale; collocazione spaziale delle opere, lotti esecutivi (es. prog. A-prog. B / prog. A, parte Nord -prog. A parte Sud...); discipline interessate (ad es. in base al comfort termico, acustico, resistenza meccanica...); durata dell'operazione; responsabilità; grado di conoscenza dell'attività; livello di criticità; squadre di lavoro coinvolte; subappaltatori; tecnologia impiegata; altro.

Probabilmente il principale problema connesso all'uso della WBS risiede nel fatto di non mostrare e talvolta di dissimulare le interazioni esistenti tra le diverse parti in cui viene scomposto, ad un dato livello, il progetto. Si tratta comunque di un limite insito nella natura dello strumento (in particolare della struttura ad albero) e di cui quindi si deve essere consci sin dal principio.

A completamento di quanto esposto, in Figura 1 viene proposta la WBS di una piccola costruzione, come esempio, per comprendere al meglio i concetti.

1.2 Forme del cronoprogramma

Poiché la finalità principale del cronoprogramma è quella previsionale, cioè prevedere la durata dei lavori edili, i quali sono composti da una serie di attività o fasi, gli elementi minimi indispensabili per il raggiungimento di questo scopo sono, per ciascuna fase di lavoro, le indicazioni di inizio e durata che consentono di tenere evidenza dell'andamento temporale previsto. Le tecniche che trovano un maggior impiego per la programmazione dei tempi dei lavori sono le seguenti:

- Il Time Schedule: per la programmazione temporale basata sul diagramma a barre di Gantt;

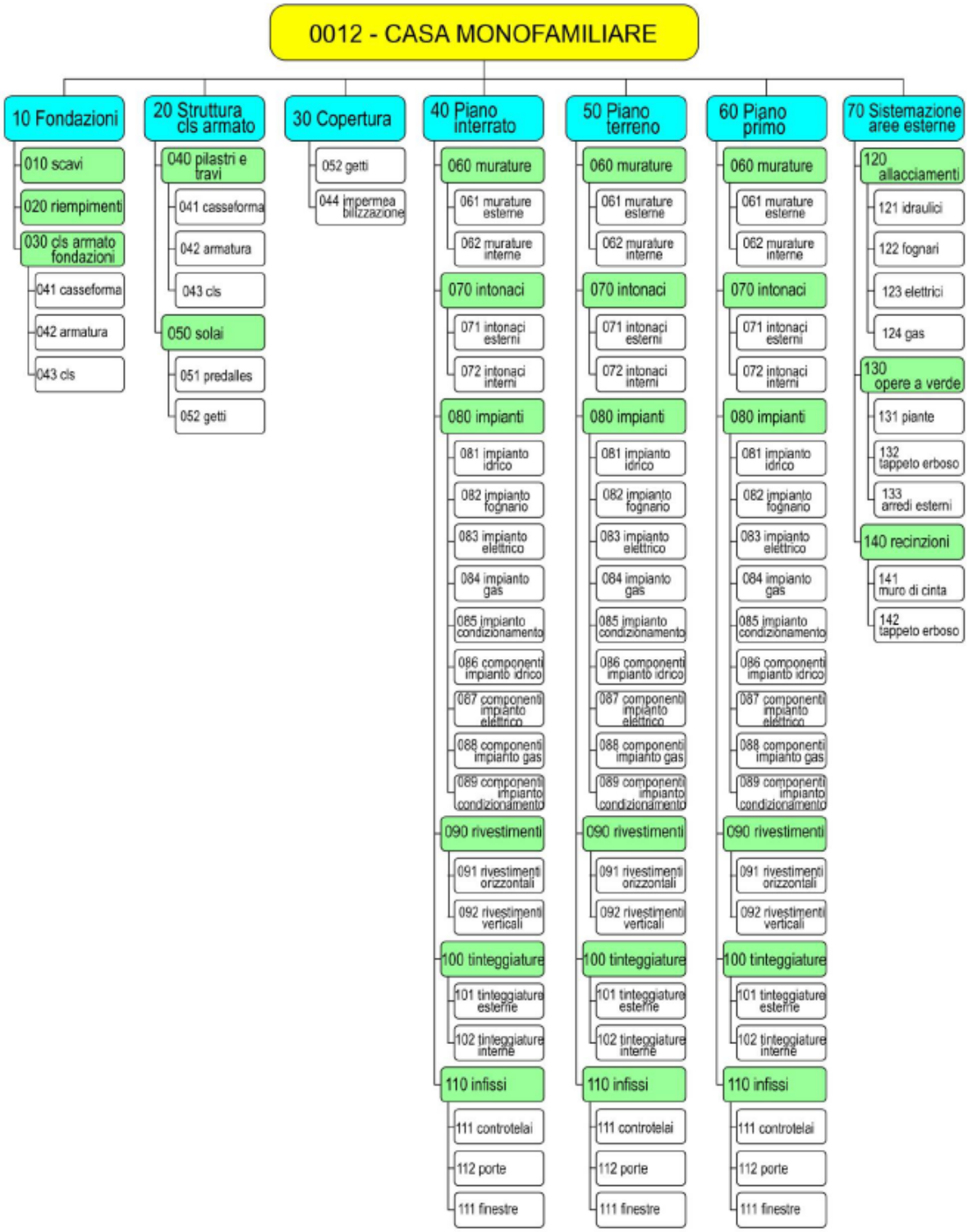


Figura 1 – Esempio di WBS (immagine tratta da Luigi Ottaiano, 2016)

- Il Programme Evaluation and Review Technique (PERT), stime dei tempi probabilistiche;

- Il Critical Path Method (CPM), stime deterministiche dei tempi (il quale non è un metodo a sé stante ma si usa spesso in supporto degli altri due metodi).

1.2.1 Diagramma di Gantt

Sul piano operativo per la predisposizione del cronoprogramma dei lavori può essere usato un diagramma di tipo lineare (Gantt) di cui la forma semplificata è quella rappresentata in Figura 2, che consente una facile ed immediata individuazione delle varie voci e dei relativi tempi di esecuzione dei lavori stessi; questo tipo di schema permette anche di visualizzare, in modo indicativo, le varie sovrapposizioni delle fasi di lavoro.

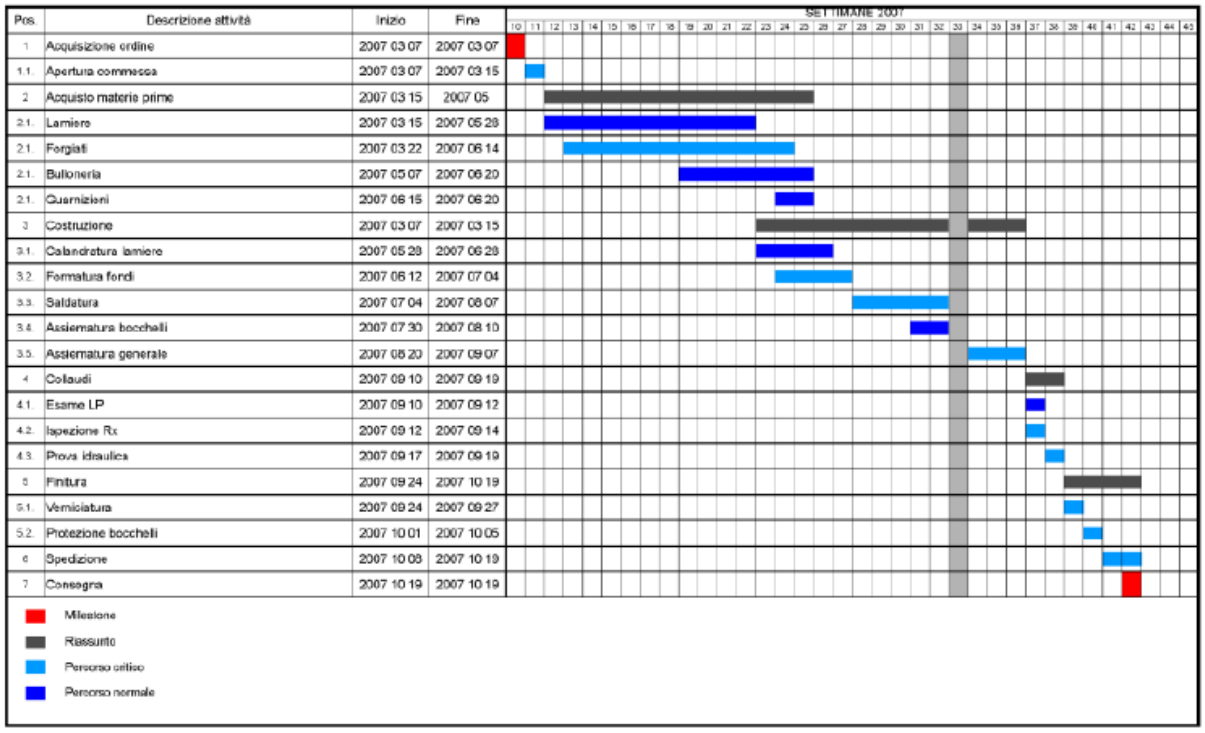


Figura 2 – Esempio di diagramma di Gantt (immagine tratta da Luigi Ottaiano, 2016)

Il diagramma di Gantt (dall'Ing. Henry Gantt che l'ha ideato) è definito come un diagramma di schedulazione a barre, che è lo strumento di rappresentazione grafica che contiene tutte le informazioni relative alla programmazione dell'opera. La tecnica Gantt di programmazione lineare è la più in uso perché di facile comprensione da parte dell'utilizzatore, ma anche dei non addetti ai lavori, perché consente un'intuitiva visualizzazione dello svolgimento delle attività e della loro collocazione temporale relativa alle altre attività che la precedono o seguono. Costituisce una rappresentazione grafica che contiene l'intero elenco delle attività che compongono il lavoro, la data di inizio e di fine e la durata di ciascuna attività. Tale rappresentazione a barre, dove la lunghezza della barra è proporzionale alla durata temporale per eseguirla, offre una chiara visione dell'intero processo attuativo e del suo svolgimento temporale e quindi un agevole controllo della scadenza programmata della fase. Il diagramma rappresenta uno strumento molto utile anche quando si vogliono evidenziare come

interagiscono diversi blocchi della WBS perché consente infatti di definire quali sono le attività che ricevono input e forniscono output ad altri blocchi. Questo tipo di elaborato, come si vede in Figura 2, è costituito da una griglia con i periodi riportati sulle ascisse (ogni colonna equivale ad un periodo: ora, giorno, settimana, mese, anno) ed i tipi di lavorazioni riportati sulle ordinate; l'incrocio dei due valori permette di individuare, in funzione della lavorazione, la quantità di periodi necessari per la sua esecuzione. Il diagramma può essere preparato, in modo molto diretto, a mano utilizzando delle griglie in bianco prestampate e colorando le varie strisce dei giorni corrispondenti ad ogni lavorazione oppure tramite computer (con il quale si possono successivamente apportare modifiche).

I vantaggi di questo tipo di diagramma sono rappresentati: a) dalla facilità con cui può essere approntato e modificato, anche negli uffici di cantiere, dagli stessi tecnici che lo utilizzano come riferimento delle varie fasi di realizzazione dell'opera; b) dalla semplicità di lettura che consente un immediato riconoscimento visivo delle singole fasi di lavoro e della loro durata.

I limiti di questo strumento, nella forma semplificata, sono costituiti dai seguenti punti: a) la possibilità di un uso appropriato solamente per lavori di piccola entità (un eccessivo numero di fasi porterebbe ad un'estensione eccessiva della griglia di riferimento); b) una notevole approssimazione nello stabilire la durata delle varie fasi in funzione della mano d'opera impiegata (spesso questo tipo di valutazione viene fatto in modo molto generico).

1.2.2 Programme Evaluation and Review Technique (Pert)

Nel caso di appalti complessi, viene usato un diagramma Pert, generato con l'uso del computer, che consente un'esatta individuazione dei tempi, delle fasi e delle interferenze tra le varie lavorazioni. La piena e corretta utilizzazione di questo strumento consente alla stazione

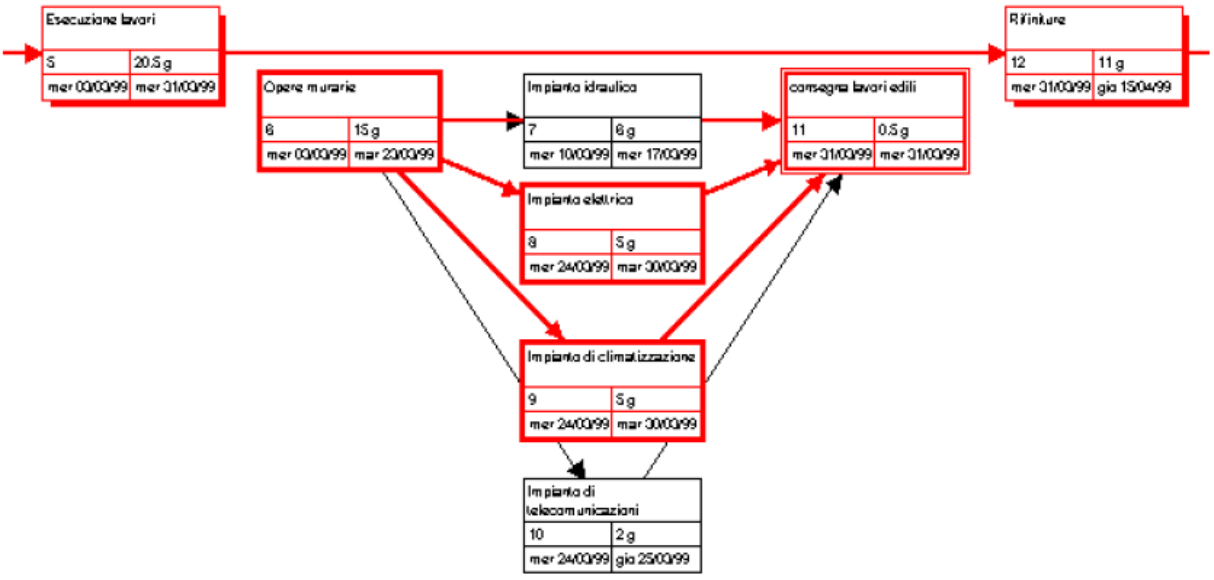


Figura 3 – Esempio di diagramma Pert (immagine tratta da Luigi Ottaiano, 2016)

appaltante, al direttore dei lavori, all'impresa esecutrice e alle figure preposte per la predisposizione delle misure antinfortunistiche di verificare prontamente tempi e durata delle varie fasi di lavoro, interferenze ed eventuali ritardi sull'andamento delle opere e di provvedere tempestivamente al recupero dei tempi contrattuali.

Il Pert (*Programme Evaluation and Review Technique*) è un metodo di programmazione utilizzato per la valutazione dei tempi e delle concatenazioni delle varie fasi di lavoro legate alla realizzazione di un'opera. Questo tipo di diagramma presuppone l'utilizzo di un computer per l'elaborazione dei vari elementi riportati nei grafici finali (esempio di Pert in Figura 3.) La caratteristica del Pert, rispetto ad altre tecniche di programmazione, consiste nella possibilità di valutare, soltanto da un punto di vista probabilistico, le concatenazioni previste per ogni fase di lavoro ed i tempi necessari per la realizzazione. La veste grafica di questo tipo di elaborazione, come si vede in Figura 3, è costituita da un reticolo o diagramma a frecce nel quale ogni evento viene individuato e legato agli altri da segmenti orientati ed il cui verso stabilisce l'ordine di precedenza di ciascun evento; la regola costante è che tutte le attività che confluiscono in un "nodo" debbano essere completate prima di iniziare qualunque altra attività che parta dallo stesso nodo.

In corrispondenza di ogni evento o nodo vengono indicati tre valori: a) il tempo assoluto (in giorni) stimato per l'esecuzione del lavoro a cui il nodo si riferisce; b) il numero (in giorni) totale, a partire dal giorno di inizio e sommato a tutti i tempi dei nodi precedenti (nell'ipotesi di durata minima del lavoro riportato nel nodo); c) il numero totale (in giorni), a partire dal giorno di inizio e sommato a tutti i tempi dei nodi precedenti (nell'ipotesi di durata massima del lavoro riportato nel nodo).

Dalla accuratezza e dal dettaglio dei dati immessi, si possono ottenere elaborati in cui sono elencati: 1) il numero di individuazione dei nodi e brevi descrizioni dei lavori di ciascun nodo; 2) il tempo previsto per l'esecuzione di ogni singolo lavoro; 3) la durata minima di esecuzione di ogni lavoro; 4) la durata massima di esecuzione di ogni lavoro; 5) l'esecutore di ogni lavoro (impresa o altro); 6) il numero di lavoratori previsti per tale lavoro; 7) eventuali vincoli, ritardi e note per l'individuazione del percorso critico.

All'interno del diagramma reticolare con i nodi ed i versi di sequenza di ciascun nodo viene riportato (sulla base delle note inserite) il percorso critico che individua la sequenza principale dei vari lavori o fasi più importanti senza la cui esecuzione non possono essere svolte tutte le opere successive. È evidente che questo tipo di strumento costituisce, dal punto di vista sia della quantificazione dei tempi che del tipo di interrelazioni tra le varie sequenze, un supporto di notevole importanza per l'organizzazione della fase esecutiva vera e propria.

Anche in questo caso, comunque, esistono dei limiti che sono rappresentati: a) dalla possibilità di modificare lo schema finale utilizzando esclusivamente dei computer; b) dalla necessità di un'esatta corrispondenza tra i dati inseriti nel computer e quelli reali (quantità di mano d'opera, mezzi, disponibilità dei materiali, ecc.).

Oggi, con l'ausilio dei computer il Pert ed il diagramma di Gantt praticamente si equivalgono o, meglio, possono essere rappresentati entrambi poiché i dati di immissione sono per la quasi totalità identici.

1.2.3 Critical Path Method (CPM)

In un determinato progetto ci possono essere centinaia di attività e moltissime dipendenze. In alcuni casi questo fatto può rendere quasi impossibile individuare i compiti più importanti; quelli che, se ritardano oppure stentano a partire, produrranno un impatto significativo sull'intero progetto e su cui occorre quindi concentrare l'attenzione per garantire il buon esito del progetto.

Il Critical Path Method è un metodo utile per la determinazione della durata minima di un progetto individuando le attività critiche che lo caratterizzano. Si tratta di uno strumento di gestione progetti sviluppato nel 1957 dalla Catalytic Construction Company per la manutenzione degli impianti della Du Pont de Nemours. È una tecnica utilizzata nei progetti in cui le risorse necessarie allo svolgimento di un'attività possono non essere considerate come vincolo e le durate delle attività di cui sono composti possono essere stimate con un adeguato livello di certezza, permettendo, quindi, di considerarle di natura deterministica. Il metodo consiste nell'individuazione del cammino critico, ovvero di quell'insieme di attività logicamente dipendenti tra di loro che collegano il nodo iniziale (attività di inizio) al nodo finale (attività di fine) e la cui somma delle durate è massima. Un ritardo in una di queste attività implica il ritardo dell'intero progetto. La tecnica essenziale di questo metodo consiste nel costruire un modello del progetto includendo i seguenti aspetti: una lista di tutte le attività richieste per completare un progetto; la durata di ogni attività; la dipendenza fra le attività.

Il Critical Path Method ha alcune caratteristiche principali, le quali sono: tiene conto del trade-off tempi/costi; cerca di ridurre i tempi al minor costo possibile; individua il cammino critico; compatta le attività del cammino critico a partire dalla meno costosa.

L'analisi dei tempi può essere suddivisa come segue: 1) passaggio in avanti (forward pass) che prevede il calcolo delle date al più presto (early start, early finish); 2) passaggio all'indietro (backward pass) che prevede il calcolo delle date al più tardi (late start, late finish); 3) calcolo degli scorrimenti totale e libero delle attività (total float, free float); 4) individuazione del percorso e quindi delle attività critiche.

Il Critical Path Method si articola in due diversi tipi di analisi: la prima analisi e la seconda analisi. La prima analisi consiste in un'analisi del reticolo delle attività del progetto simile a quella del metodo Pert, con l'unica differenza che in questo caso la stima della durata di ogni attività non è più probabilistica ma deterministica. Questo presuppone che sia possibile prevedere con sufficiente approssimazione quale sarà il tempo di esecuzione delle singole attività in cui è suddivisa la realizzazione del progetto; La seconda analisi consiste nell'approfondimento dell'analisi precedente e mira a unire i tempi e i costi al fine di ottenere un miglior controllo del binomio tempi-costi del progetto. L'aspetto più importante di questa

analisi è l'introduzione delle relazioni tempo-costo e la ricerca del costo totale minimo e della relativa durata ottima.

Il CPM è molto importante per ogni project manager ed è tipicamente associato ad altri metodi per lo sviluppo del cronoprogramma, ad esempio può essere associato al diagramma di Gantt.

1.3 Finalità del cronoprogramma

L'elaborazione del cronoprogramma, in genere a cura dell'impresa appaltatrice, obbliga questa a rispettare i tempi parziali e complessivi in esso esposti con l'obiettivo di conciliare i termini contrattuali concordati tra impresa appaltatrice e committente con le attese economiche dell'imprenditore. Le finalità del cronoprogramma sono diverse, ma in pratica sono tre quelle a cui si deve rivolgere maggiore attenzione: contrattuale, operativa, economica.

1.3.1 Finalità contrattuale

Il cronoprogramma viene assunto come impegno contrattuale da parte dell'impresa affidataria dell'esecuzione dell'opera. È compito del direttore operativo (Direzione dei lavori) curare l'aggiornamento del cronoprogramma generale e particolareggiato e segnalare al direttore dei lavori eventuali difformità. Pertanto, il cronoprogramma, obbliga l'impresa appaltatrice a rispettare i tempi parziali e complessivi in esso esposti con l'obiettivo di conciliare i termini contrattuali concordati tra essa impresa appaltatrice e il committente con le proprie attese economiche.

1.3.2 Finalità operativa

Operativamente il cronoprogramma consente il controllo di una serie di fenomeni, tra questi i principali sono: tempistica delle lavorazioni; concomitanza delle lavorazioni; ottimizzazione degli approvvigionamenti delle risorse.

1.3.2.1 Tempistica delle lavorazioni

Ai fini di un efficace controllo, vengono redatti più cronoprogramma, uno previsionale e gli altri a resoconto periodico che messi a costante confronto evidenziano le discordanze temporali tra i due tipi, notificando la necessità di adozione di misure atte a rimettere in carreggiata, nei tempi concordati, la produzione. La differenza delle tempistiche tra i due elaborati si riscontra spesso nelle lavorazioni inconsuete per l'impresa, la quale deve ipotizzare una tempistica su basi teoriche o per indicazioni date da terze parti. Fatto sta che a quel prezzo con quella tempistica i costi di produzione non consentono slittamenti, e pertanto è indispensabile introdurre tutte le capacità dell'impresa per rientrare nei tempi. In mancanza si va incontro ad una perdita parziale certa, scaturita in genere dal non aver saputo valutare l'incidenza della risorsa manodopera.

1.3.2.2 Concomitanza delle lavorazioni

Un altro aspetto operativo molto importante del cronoprogramma è quello legato alla salvaguardia della salute e della sicurezza dei lavoratori (D.lgs 494/96 – attuazione direttiva 92/57/Cee concernente le prescrizioni minime di sicurezza e di salute da attuare nei cantieri temporanei o mobili) soprattutto per quello che riguarda le concomitanze. Denominate anche sovrapposizioni, interferenze, ecc. si manifestano quando lavorazioni di tipo diverso vengono a trovarsi sul cronoprogramma nello stesso luogo e nello stesso tempo. Queste situazioni, non da evitarsi ma, da escludere categoricamente, sono regolamentate dalle norme sulla sicurezza.

Quando operazioni diverse devono essere necessariamente condotte simultaneamente vanno individuati i corretti sistemi di sincronismo per l'adozione di misure di sicurezza specificatamente dedicate. Se si rileva dal cronoprogramma che vi sono una serie di concomitanze, esse vanno analizzate sia per valutare se temporalmente le attività interessate possono essere eseguite in differita, o, nella indispensabilità di eseguirle contemporaneamente, quali criteri di sicurezza adottare, dettagliando all'occorrenza meglio le attività per rendere il cronoprogramma più comprensibile dal punto di vista della sicurezza.

Dal punto di vista, poi, della concomitanza spaziale delle lavorazioni occorre avere una cognizione di dettaglio anche delle sotto lavorazioni e dell'influenza che ha una lavorazione sull'altra contemporanea, questo soprattutto ai fini antinfortunistici, ma anche dal punto di vista di ottimizzazione delle risorse con particolare riferimento all'apporto di manodopera e di uso di mezzi d'opera affinché non si creino delle soste di precedenza nelle lavorazioni rendendo inoperose entrambe le squadre interessate.

La valutazione dei rischi da interferenze sul lavoro è un capitolo della normativa in materia di sicurezza, ampiamente trattato, dibattuto, controverso e più volte rivisto. Le interferenze spesso si presentano nel momento in cui più operatori afferenti ad aziende diverse, prestano la loro opera (contestualmente o meno) sullo stesso luogo di lavoro; è quindi il caso in cui diverse realtà lavorative con ragioni sociali e datori di lavoro differenti, lavorano nello stesso sito, nello stesso momento, o anche in alcuni casi in successione se comunque gli effetti del lavoro di chi precede possono ricadere in qualche modo su chi interviene successivamente. È presumibile quindi che ogni prestatore d'opera apporti dei rischi sul luogo di lavoro, connessi con la propria attività specifica, e che questi rischi, sommati a quelli eventualmente apportati dagli altri attori, possano in qualche modo generare delle sovrapposizioni con un aumento del livello di rischio ed una diversa tipologia di pericolo presente sul sito. Gli aspetti normativi che regolamentano questo tipo di situazioni vengono trattati nell'art. 26 del Testo Unico: "Obblighi connessi ai contratti d'appalto o d'opera o di somministrazione"; in tal senso si precisa che per "contratti di appalto" si intente la relazione tra un committente (pubblico o privato) ed un'impresa terza che riceve l'incarico per specifiche attività; nel "contatto d'opera" le attività definite dal contratto vengono svolte da un prestatore d'opera autonomo o dal titolare di una ditta individuale mentre nei "contratti di somministrazione" il fornitore assicura prestazioni periodiche o continuative di cose o servizi.

In tutte queste tipologie di rapporto di lavoro, quindi, sempre definite da un contratto tra le parti, esistono una committenza ed un contraente con rispettivi specifici oneri e responsabilità ascrivibili a entrambi i soggetti. Il datore di lavoro della committenza ha l'obbligo di verificare che i contraenti siano in possesso dei requisiti tecnico professionali per svolgere l'attività richiesta e si assume l'incarico di definire i rischi da interferenza, come precedentemente descritti, apportati da tutte le imprese e/o dai singoli lavoratori autonomi coinvolti nell'attività.

La valutazione dei rischi da interferenza si definisce con l'elaborazione del Documento Unico di Valutazione dei Rischi Interferenziali (DUVRI), un documento in cui il datore di lavoro che ha la disponibilità giuridica dei luoghi dove si svolge l'appalto, effettua la valutazione dei rischi presenti ed elabora le misure preventive e protettive rivolte ad eliminarli o ridurli al minimo. È importante sottolineare che non sussiste l'obbligo di elaborazione del DUVRI se il servizio prestato è di natura intellettuale o è una semplice fornitura di materiali ed in ogni caso per tutti i lavori la cui durata non sia superiore ai cinque uomini giorno in un arco temporale di un anno dalla data di inizio dei lavori. Il DUVRI deve essere invece sempre redatto, anche quindi per attività di durata inferiore ai cinque giorni, nei casi in cui la valutazione del rischio incendio risulti elevato o in tutte le attività svolte in ambienti confinati o in cui vi sia presenza di agenti cancerogeni, mutageni, biologici, di amianto o di atmosfere esplosive, nonché nelle attività che presentino rischi particolari elencate nell'allegato XI del D.Lgs 81/08.

1.3.2.3 Ottimizzazione degli approvvigionamenti delle risorse

Un cronoprogramma ben progettato è di ausilio all'ottimizzazione dei costi per l'impiego delle risorse:

- Manodopera: In genere il problema non si pone per il personale non qualificato che è facilmente reperibile e immediatamente utilizzabile. La difficoltà risulta evidente quando è necessario assumere personale con particolari qualifiche, dal tecnico specializzato in particolari lavorazioni a quello destinato alla conduzione di particolari attrezzature, al capocantiere o direttore di cantiere specializzati in quel tipo di opera. È in questa fase che si fa la prima distinzione di lavorazioni da fare in proprio e lavorazioni da affidare in subappalto. In genere, quando non disponibile da altri cantieri, si predilige l'assumere del personale che è adatto al tipo di specializzazione dell'impresa e che può essere utilizzato su più insediamenti aziendali. In seconda battuta si procede all'assunzione del personale con qualifiche per lavorazioni specializzate che non rientrano nel ''core business'' dell'impresa, ma che hanno una durata tale da giustificare l'assunzione di quel personale (tra l'altro se disponibile). In caso di prestazioni di breve durata si preferisce affidare quella lavorazione in subappalto (negli appalti di opere pubbliche vi sono limiti all'affidamento dei lavori in subappalto).
- Materiali: Con un cronoprogramma alla mano si semplificano le operazioni di ordinativi per acquisto merce, sapendo in precedenza la tipologia, l'epoca e il luogo di utilizzo. Lo si immagini per un'impresa che ha in funzione contemporaneamente più cantieri. Si ha

la libertà di ordinare merci tenendo conto di più fattori oltre quello temporale. In periodi di particolari contingenze, per esempio quando il prezzo sale, è conveniente ordinare oggi a prezzo prefissato per una necessità che avrò fra tre mesi, oppure con prezzi in discesa (abitualmente sinonimo di abbondanza di merce) ordinare il giorno prima per il giorno dopo, e così via. Soprattutto per i semilavorati le imprese si pongono l'obiettivo o di produrre o di commercializzare quanto più possibile all'interno dell'azienda stessa. Imprese di grandi dimensioni specializzate in lavori autostradali o in edilizia industrializzata, spesso hanno partecipazioni in stabilimenti di prefabbricazione, o in agenzie di commercializzazione di materiali edili.

- Mezzi d'opera: Anche per i mezzi d'opera la conoscenza della durata e l'epoca di impiego dà la possibilità di scelta tra il noleggio e l'acquisto, e con tempo a disposizione rispetto a all'epoca di impiego si può visionare un mercato ampio in estensione e abbondante in quantità, tale da offrire la possibilità di acquistare macchinari in buono stato d'uso che in funzione della durata di impiego richiesta potrebbe costare addirittura al di sotto del nolo. Il vantaggio a volte non si manifesta nel cantiere in corso, ma in quelli successivi. Anche per i mezzi d'opera alcune imprese ad alta specializzazione tendono o a costruire in proprio i propri macchinari o ad acquisire partecipazioni in stabilimenti di produzione.
- Trasporti: La possibilità di organizzare per tempo un trasporto magari di tipo eccezionale rende l'operazione mediamente più vantaggiosa, e quando l'impresa è così grande perché precludere la possibilità di utilizzare mezzi in proprio magari fondando o partecipando un'impresa di trasporti in proprio.

1.3.3 Finalità economica

Un cronoprogramma è un buon cronoprogramma se i dati forniti sono completi e formulati in modo corretto. Partendo da questo presupposto, il cronoprogramma è alla base del business plan, quel documento che sintetizza i contenuti e le caratteristiche di un progetto imprenditoriale che viene utilizzato sia per la pianificazione e gestione aziendale che per la comunicazione esterna, in particolare verso potenziali finanziatori o investitori a cui l'impresa si rivolge. Quindi il documento se ben fatto non è solo quel foglio quadrettato con le barre da consegnare al committente, ma utilizzato per i fini già enumerati è anche il documento di riferimento per la documentazione da produrre ad una banca per ottenere un démarrage, quel contratto bancario con il quale una banca corrisponde al cliente, una somma in denaro o costituisce una disponibilità in conto corrente per consentirgli l'avvio della costruzione, con la restituzione della somma in percentuale sugli importi degli stati di avanzamento dei lavori (SAL).

1.4 Fasi della realizzazione di un cronoprogramma

Nei paragrafi seguenti si analizzeranno le fasi di realizzazione di un cronoprogramma dei lavori utilizzando l'esempio di un piccolo garage tratto dallo scritto di Angelotti Andrea (2021). L'opera presa in oggetto dall'autore dell'esempio consiste in un garage indipendente di

49mq composto da due posti auto con relative entrate e un bagno accessibile solamente dall'interno. La struttura è composta da sei pilastri, tamponatura esterna e cordolo soprastante, il tutto sostenuto, ipotizzando di essere in presenza di un terreno poco resistente, da una fondazione a platea che grava su uno strato di magrone (in Figura 4 la vista 3D del garage).

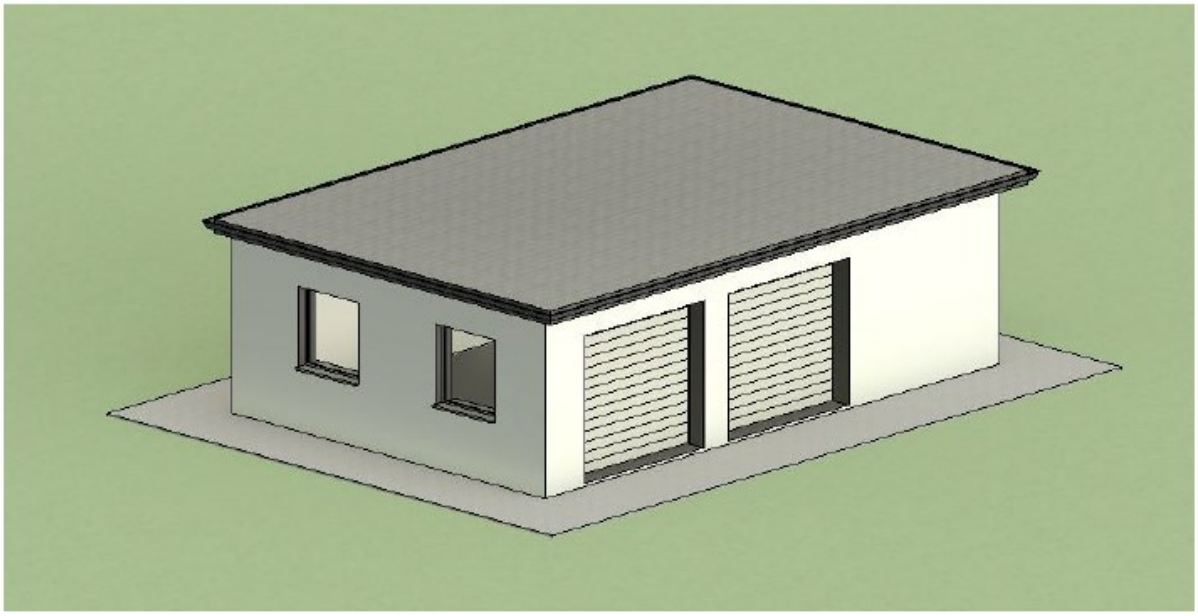


Figura 4 – Vista 3D del garage (immagine tratta da Angelotti Andrea, 2021)

1.4.1 Realizzazione della Work Breakdown Structure (WBS)

Di seguito si riporta un esempio di WBS, la cui logica si è spiegata in precedenza, e con il proseguire dei paragrafi ad ogni attività verranno associate le risorse umane e i materiali necessari, oltre al tempo necessario per completarla ed ai vincoli che la legano alle altre attività. Essa si configura tramite una struttura ad albero gerarchico orientato al prodotto e suddiviso in tutte le parti che lo compongono. È consigliabile che la scomposizione del progetto venga eseguita con logica definendo a priori il livello di disaggregazione da scegliere. Nel caso proposto da Angelotti Andrea (2021) è conveniente affidarsi a una disaggregazione per processi, ovvero la scomposizione avviene in base ai processi che portano alla realizzazione dei prodotti finali individuando le macro-attività (fondazioni, scavi, strutture, murature, impianti e finiture).

Si ricorda che la WBS è considerata efficiente quando: Ai livelli più alti prevale la logica della suddivisione per aree di progettazione; Viene adottato un unico criterio per ciascun livello, così da consentire omogeneità nelle comunicazioni.

I criteri di aggregazione degli elementi costituenti la WBS di progetto sono puramente logici, per cui essa non ordina cronologicamente le fasi ed i compiti. L'ordine cronologico, infatti, è eseguito in seguito con l'elaborazione della schedulazione generale del progetto. In Figura 5 viene riportato lo studio della WBS (in formato di lista) prodotta per la realizzazione

del garage. In rosso sono evidenziate le macro-attività che individuano le aree di progettazione richieste da questo processo edilizio in particolare, per poi suddividersi con le seguenti tasks. La gerarchia ad albero è stabilita, non solo dalla codifica numerica a più livelli ma anche dalla colorazione (rosso-arancione-giallo), favorendo un miglior impatto visivo.

1- ALLESTIMENTO CANTIERE
1.1 Delimitazione area cantiere
1.2 Impianti di alimentazione e distribuzione elettrica
1.2.1 Allestimenti quadri elettrici
1.2.2 Posizionamento cavi e linee di alimentazione
1.2.3 Esecuzione impianto di terra
1.3 Baraccamenti e apprestamenti igenico-sanitari
1.3.1 Preparazione e posa baraccamenti
1.2.2 Allacciamenti e opere fognatura
1.4 Istallazione macchine
2- SCAVI
2.1 Scavi con mezzi meccanici
2.2 Scavi eseguiti a mano
3- STRUTTURE IN C.A
3.1 Strutture di ffondazione
3.1.1 Magrone
3.1.2 Casseri fondazione
3.1.3 Getto della fondazione
3.2 Applicazione sist. Interrati di drenaggio e impermeabiliz.
3.2.1 Applicazione vespaio
3.2.2 Applicazione membrana
3.2.3 Rinterro con mezzi meccanici
3.3 Strutture in elevazione
3.3.1 Applicazione casseri per pilastri
3.3.2 Getto pilastri
3.3.3 Applicazione casseri per travi
3.3.4 Getto delle travi
3.3.5 Applicazione casseri per solaio
3.3.6 Getto solaio di copertura

4- MURATURE
4.1 Tamponatura esterna
4.2 Rivestimento
4.3 Tramezzatura interna
5- IMPIANTI
5.1 Impianto idrico
5.1.1 Montaggio impianto
5.1.2 Getto massetto autolivellante
5.2 Apparecchi sanitari
6- FINITURE
6.1 Pavimentazione
6.2 Intonacatura
6.2.1 Pareti interne
6.2.2 Pareti esterne
6.3 Montaggio infissi
6.3.1 Montaggio controtelai
6.3.2 Posa in opera infissi
6.4 Tinteggiatura
6.4.1 Pareti interne
6.4.2 Pareti esterne
6.5 Sistema di raccolta acque piovane
6.5.1 Montaggio canale di gronda
6.5.2 Montaggio di pluviale in rame

Figura 5 – Esempio di WBS in formato di lista (immagine tratta da Angelotti Andrea, 2021)

1.4.2 Associazione delle quantità alle attività

Le quantità sono informazioni che provengono dal modello BIM, più precisamente dalle proprietà di ogni elemento di cui è composta l'opera. Quindi si dovrà associare ad ogni task che compone la WBS una quantità, misurabile o verificabile sul modello BIM, del suo prodotto finale. La stima delle quantità richiede una conoscenza dettagliata del progetto e di come esso è sviluppato e quindi può essere eseguito dal planner con l'assistenza del team di progetto.

In questa sezione dell'analisi verrà eseguita l'elaborazione delle quantità degli elementi che costituiscono il garage. Le stime riguardano grandezze per le quali sono definiti alcuni parametri di quantità o di misurazione (ad esempio m2, m3 o metri lineari), mentre altre lavorazioni sono stimabili diversamente, ad esempio: a) Parte impiantistica: richiede una serie

di quadri e linee ben organizzate; b) Baraccamenti e wc: il numero è stabilito descritti dalla normativa, Il T.U. D.Lgs. 81/2008 Allegato XII, nel caso proposto è sufficiente un baraccamento e un wc; c) Installazione dei macchinari: la costruzione di un garage non necessita di molti macchinari elettrici, infatti, è sufficiente un'impastatrice di calcestruzzo e malta (betoniera).

L'autore dell'esempio proposto effettua le misurazioni secondo le metodologie del Prezzario ufficiale 2020 della Regione Marche, che fanno riferimento ai metodi di misurazione del "Capitolato speciale tipo per appalti di lavori edilizi" pubblicato dal Ministero delle Infrastrutture e dei Trasporti, Servizio tecnico centrale.

L'assegnazione delle quantità aiuta il planner nello studio delle risorse e delle durate, perciò è necessario che le stime delle quantità siano il più realistiche possibili. In questo caso dato che lo scritto da cui si è tratto l'esempio è recente, l'autore dispone del modello BIM del garage che fornisce ogni dato di ciascun prodotto finito in modo preciso. Attualmente l'utilizzo del BIM nella pianificazione è imprescindibile. In Figura 6 è riportata l'assegnazione delle quantità a ciascuna attività contenuta nella WBS del garage.

ID		QUANTITA'
	1- ALLESTIMENTO CANTIERE	
1	1.1 Delimitazione area cantiere	75 ml
	1.2 Impianti di alimentazione e distribuzione elettrica	
2	1.2.1 Allestimenti quadri elettrici	n.5
3	1.2.2 Posizionamento cavi e linee di alimentazione	50 ml
4	1.2.3 Esecuzione impianto di terra	n.1
	1.3 Baraccamenti e apprestamenti igenico-sanitari	
5	1.3.1 Preparazione e posa baraccamenti	n.2
6	1.2.2 Allacciamenti e opere fognatura	n.2
7	1.4 Istallazione macchine	n1
	2- SCAVI	
8	2.1 Scavi con mezzi meccanici	57 m3
9	2.2 Scavi eseguiti a mano	4 m3
	3- STRUTTURE IN C.A	
	3.1 Strutture di fondazione	
10	3.1.1 Magrone	10,5 m3
11	3.1.2 Casseri fondazione	41,4 ml
12	3.1.3 Getto della fondazione	31,5 m3
	3.2 Applicazione sist. interrati di drenaggio e impermeabiliz.	
13	3.2.1 Applicazione vespaio	105 m2
14	3.2.2 Applicazione membrana	22 m2
15	3.2.3 Rinterro con mezzi meccanici	61 m3
	3.3 Strutture in elevazione	
16	3.3.1 Applicazione casseri per pilastri	n.6 pil.
17	3.3.2 Getto pilastri	2,5 m3
18	3.3.3 Applicazione casseri per travi	33,2 ml
19	3.3.4 Getto delle travi	4,34 m3
20	3.3.5 Applicazione casseri per solaio	30,4 ml + 51,7m2
21	3.3.6 Getto solaio di copertura	13,9 m3
	4- MURATURE	
22	4.1 Tamponatura esterna	66,37 m2
23	4.2 Rivestimento	82 m2
24	4.3 Tramezzatura interna	14,3 m2
	5- IMPIANTI	
	5.1 Impianto idrico	
25	5.1.1 Montaggio impianto	n.1
26	5.1.2 Getto massetto autolivellante	48,71 m2
27	5.2 Apparecchi sanitari	n.2
	6- FINITURE	
28	6.1 Pavimentazione	48,1 m2
	6.2 Intonacatura	
29	6.2.1 Pareti interne	108,26
30	6.2.2 Pareti esterne	82 m2
	6.3 Montaggio infissi	
31	6.3.1 Montaggio controtelai	n.3
32	6.3.2 Posa in opera infissi	n.5
	6.4 Tinteggiatura	
33	6.4.1 Pareti interne	108,26
34	6.4.2 Pareti esterne	82 m2
	6.5 Sistema di raccolta acque piovane	
35	6.5.1 Montaggio canale di gronda	33,2 ml
36	6.5.2 Montaggio di pluviale in rame	13 ml

Figura 6 – Esempio assegnazione delle quantità alle attività (immagine tratta da Angelotti Andrea, 2021)

1.4.3 Associazione delle risorse umane alle attività

Per associazione delle risorse si intende il processo volto ad individuare le risorse da destinare allo svolgimento delle attività di un progetto. Successivamente andranno definite le diverse tipologie di risorse richieste da ogni attività e la loro quantificazione in base ad una stima. Occorre essere realistici e procedere all'allocazione delle risorse basandosi su stime accurate ed ottimizzando il tutto tenendo conto dei reali carichi di lavoro. Per eseguire le assegnazioni è estremamente importante disporre dei loro profili di competenze aggiornati per adeguare le loro competenze a quanto richiesto da determinati progetti. Ciò andrà fatto in stretta integrazione con la funzione del personale e nel rispetto delle procedure da essa previste. L'assegnazione delle risorse è una delle competenze principali del Planner. La pianificazione delle risorse umane dipende dalle competenze richieste dalle attività di progetto che, a loro volta, dipendono dall'analisi dei work packages presenti nella WBS.

L'obiettivo di questa fase è associare ad ogni attività la squadra o le squadre di lavoro composta da operatori che sappiano ricoprire le abilità richieste dall'esecuzione della lavorazione. In questo caso è sufficiente associare una squadra di lavoro ad ogni attività che compone la WBS, mentre per la composizione della squadra l'autore dell'esempio le ricava analizzando le descrizioni delle lavorazioni nel Prezzario ufficiale 2020 della Regione Marche. Ogni lavorazione del prezzario comprende una descrizione nella quale è contenuta la voce "manodopera" che descrive da quanti e quali operatori è generalmente eseguita la lavorazione. In Figura 7 sono riportate le risorse umane assegnate a ciascuna attività contenuta nella WBS del garage.

1.4.4 Associazione dei tempi alle attività

Al fine di realizzare la programmazione dell'opera, l'associazione dei tempi è essenziale; infatti, la durata di un progetto è la conseguenza della durata di ciascuna attività del progetto stesso. Il processo di stima della durata delle attività schedulate utilizza informazioni relative all'ambito del lavoro, ai tipi di risorse necessarie, alle quantità di risorse stimate e ai calendari con relative disponibilità degli operatori; inoltre deve essere stimata la quantità dell'impegno di lavoro richiesto al completamento dell'attività schedulata.

La durata di un'attività dipende dall'effort (ore/uomo necessarie per lo svolgimento dell'attività) e dal numero di risorse assegnate a quel task (maggiore è il numero di risorse, minore è la durata). L'autore dell'esempio ricava l'effort dalle descrizioni delle lavorazioni nel Prezzario ufficiale 2020 della Regione Marche che, oltre alla descrizione della composizione della squadra, rilascia una durata media in ore per ciascun componente della squadra. Essa può essere presa come base per la determinazione delle durate, poiché non è mai caratterizzata dal 100% di certezza in quanto influenzata da innumerevoli variabili, che possono presentarsi nella situazione di lavoro e/o dalla produttività dell'operatore. Per questo motivo le informazioni del Prezzario devono essere accompagnate da una stima delle durate, eseguita tramite le seguenti tecniche e strumenti:

ID		RISORSE UMANE
	1- ALLESTIMENTO CANTIERE	
1	1.1 Delimitazione area cantiere	1 sq: n.1 Op. Spec./n.1 Op. Com.
	1.2 Impianti di alimentazione e distribuzione elettrica	
2	1.2.1 Allestimenti quadri elettrici	1sq: n.2 Op. Spec.
3	1.2.2 Posizionamento cavi e linee di alimentazione	1sq: n.2 Op. Spec.
4	1.2.3 Esecuzione impianto di terra	1sq: n.1 Op. Spec.
	1.3 Baraccamenti e apprestamenti igenico-sanitari	
5	1.3.1 Preparazione e posa baraccamenti	1 sq: n.1 Op. Spec./n.1 Op. Com.
6	1.2.2 Allacciamenti e opere fognatura	1 sq: n.1 Op. Spec./n.1 Op. Com.
7	1.4 Istallazione macchine	1sq: n.1 Op. Spec.
	2- SCAVI	
8	2.1 Scavi con mezzi meccanici	1 sq: n.1 Op. Spec./n.1 Op. Com./ n.1 Op. Qualif.
9	2.2 Scavi eseguiti a mano	1 sq: n.1 Op. Spec./n.1 Op. Com.
	3- STRUTTURE IN C.A	
	3.1 Strutture di fondazione	
10	3.1.1 Magrone	1 sq: n.1 Op. Spec./n.1 Op. Com.
11	3.1.2 Casseri fondazione	1 sq: n.1 Op. Spec./n.1 Op. Com./ n.1 Op. Qualif.
12	3.1.3 Getto della fondazione	1 sq: n.1 Op. Spec./n.1 Op. Com.
	3.2 Applicazione sist. Interrati di drenaggio e impermeabiliz.	
13	3.2.1 Applicazione vespaio	1 sq: n.1 Op. Spec./n.1 Op. Com.
14	3.2.2 Applicazione membrana	1 sq: n.1 Op. Spec./n.1 Op. Com.
15	3.2.3 Rinterro con mezzi meccanici	1 sq: n.1 Op. Spec./n.1 Op. Com.
	3.3 Strutture in elevazione	
16	3.3.1 Applicazione casseri per pilastri	1 sq: n.1 Op. Spec./n.1 Op. Com./ n.1 Op. Qualif.
17	3.3.2 Getto pilastri	1 sq: n.1 Op. Spec./n.1 Op. Com.
18	3.3.3 Applicazione casseri per travi	1 sq: n.1 Op. Spec./n.1 Op. Com./ n.1 Op. Qualif.
19	3.3.4 Getto delle travi	1 sq: n.1 Op. Spec./n.1 Op. Com.
20	3.3.5 Applicazione casseri per solaio	1 sq: n.1 Op. Spec./n.1 Op. Com./ n.1 Op. Qualif.
21	3.3.6 Getto solaio di copertura	1 sq: n.1 Op. Spec./n.1 Op. Com.
	4- MURATURE	
22	4.1 Tamponatura esterna	1 sq: n.1 Op. Spec./n.1 Op. Com.
23	4.2 Rivestimento	1 sq: n.1 Op. Spec./n.1 Op. Com.
24	4.3 Tramezzatura interna	1 sq: n.1 Op. Spec./n.1 Op. Com.
	5- IMPIANTI	
	5.1 Impianto idrico	
25	5.1.1 Montaggio impianto	1sq: n.2 Op. Spec.
26	5.1.2 Getto massetto autolivellante	1 sq: n.1 Op. Spec./n.1 Op. Com.
27	5.2 Apparecchi sanitari	1 sq: n.1 Op. Com.
	6- FINITURE	
28	6.1 Pavimentazione	1 sq: n.1 Op. Spec./n.1 Op. Com.
	6.2 Intonacatura	
29	6.2.1 Pareti interne	1 sq: n.1 Op. Qualif./n.1 Op. Com.
30	6.2.2 Pareti esterne	1 sq: n.1 Op. Qualif./n.1 Op. Com.
	6.3 Montaggio infissi	
31	6.3.1 Montaggio controtelai	1 sq: n.1 Op. Qualif.
32	6.3.2 Posa in opera infissi	1 sq: n.1 Op. Qualif.
	6.4 Tinteggiatura	
33	6.4.1 Pareti interne	1 sq: n.2 Op. Spec.
34	6.4.2 Pareti esterne	1 sq: n.2 Op. Spec.
	6.5 Sistema di raccolta acque piovane	
35	6.5.1 Montaggio canale di gronda	1 sq: n.1 Op. Spec./n.1 Op. Com.
36	6.5.2 Montaggio di pluviale in rame	1 sq: n.1 Op. Spec./n.1 Op. Com.

Figura 7 – Esempio associazione delle risorse umane alle attività (immagine tratta da Angelotti Andrea, 2021)

- Parere di esperti: la durata delle attività è spesso difficile da stimare a causa del numero di fattori che possono influenzarla, come il livello o la produttività delle risorse. Il parere di esperti, supportato da informazioni storicizzate, può essere usato ove possibile. I membri del team di progetto possono inoltre fornire informazioni sulla stima delle durate o il limite massimo consigliato per le durate delle attività derivanti da progetti simili. Se tali competenze non sono disponibili, le stime delle durate sono meno sicure e più rischiose.
- Stima per analogia: la stima per analogia delle durate prevede l'utilizzo della durata effettiva di simili attività schedulate effettuate in precedenza come base per la stima della durata di una futura attività. Questa stima è utilizzata frequentemente per valutare la durata del progetto quando si dispone di scarse informazioni dettagliate su di esso. La stima per analogia utilizza i dati storici e il parere di esperti ed è più affidabile quando le attività precedenti sono simili nella sostanza e non solo nella forma e se i membri del team di progetto che predispongono le stime, posseggono le competenze necessarie.
- Stima parametrica: La base di stima per la durata delle attività può essere quantitativamente determinata moltiplicando la quantità di lavoro da eseguire per il tasso di produttività. Ad esempio, i tassi di produttività di un progetto architettonico possono essere stimati sulla base del numero di disegni moltiplicato per le ore lavorative richieste per disegno oppure l'installazione di cavi in base ai metri di cavo moltiplicati per le ore lavorative richieste per metro. Le quantità di risorse complessive sono moltiplicate per le ore lavorative per periodo lavorativo o la capacità di produzione per periodo lavorativo; quindi, si divide il risultato per il numero di risorse assegnate per determinare la durata dell'attività in periodi lavorativi.
- Stima a tre punti: L'accuratezza della stima della durata delle attività può essere incrementata prendendo in considerazione la quantità di rischio nella stima originale. Le stime a tre punti si basano sulla determinazione dei tre tipi di stima: 1) Più probabile: durata dell'attività schedulata, date le risorse che probabilmente verranno assegnate, la loro produttività, le aspettative realistiche in termini di disponibilità per l'attività schedulata, le relazioni di dipendenza da altri partecipanti e le interruzioni; 2) Ottimistico: la durata dell'attività si basa sullo scenario migliore relativamente a quanto è descritto nella stima più probabile. 3) Pessimistico: la durata dell'attività si basa sullo scenario peggiore relativamente a quanto viene descritto nella stima più probabile. Una stima della durata dell'attività può essere costruita utilizzando una media delle tre durate stimate. Questa media fornisce in genere una stima più accurata della durata dell'attività rispetto a una stima più probabile a valore singolo.

ID			RISORSE UMANE	QUANTITA'
	1- ALLESTIMENTO CANTIERE	TOT: 17,5h		
1	1.1 Delimitazione area cantiere	4h	1 sq: n.1 Op. Spec./n.1 Op. Com.	75 ml
	1.2 Impianti di alimentazione e distribuzione elettrica			
2	1.2.1 Allestimenti quadri elettrici	4h	1sq: n.2 Op. Spec.	n.5
3	1.2.2 Posizionamento cavi e linee di alimentazione	3h	1sq: n.2 Op. Spec.	50 ml
4	1.2.3 Esecuzione impianto di terra	1h	1sq: n.1 Op. Spec.	n.1
	1.3 Baraccamenti e apprestamenti igenico-sanitari			
5	1.3.1 Preparazione e posa baraccamenti	2 h	1 sq: n.1 Op. Spec./n.1 Op. Com.	n.2
6	1.2.2 Allacciamenti e opere fognatura	2,5h	1 sq: n.1 Op. Spec./n.1 Op. Com.	n.2
7	1.4 Istallazione macchine	1h	1sq: n.1 Op. Spec.	n1
	2- SCAVI	TOT: 16h		
8	2.1 Scavi con mezzi meccanici	6h	1 sq: n.1 Op. Spec./n.1 Op. Com./ n.1 Op. Qualif.	57 m3
9	2.2 Scavi eseguiti a mano	10h	1 sq: n.1 Op. Spec./n.1 Op. Com.	4 m3
	3- STRUTTURE IN C.A	TOT: 78h		
	3.1 Strutture di fondazione			
10	3.1.1 Magrone	2h	1 sq: n.1 Op. Spec./n.1 Op. Com.	10,5 m3
11	3.1.2 Casseri fondazione	8h	1 sq: n.1 Op. Spec./n.1 Op. Com./ n.1 Op. Qualif.	41,4 ml
12	3.1.3 Getto della fondazione	4h	1 sq: n.1 Op. Spec./n.1 Op. Com.	31,5 m3
	3.2 Applicazione sist. Interrati di drenaggio e impermeabiliz.			
13	3.2.1 Applicazione vespaio	5h	1 sq: n.1 Op. Spec./n.1 Op. Com.	105 m2
14	3.2.2 Applicazione membrana	3h	1 sq: n.1 Op. Spec./n.1 Op. Com.	22 m2
15	3.2.3 Rinterro con mezzi meccanici	6h	1 sq: n.1 Op. Spec./n.1 Op. Com.	61 m3
	3.3 Strutture in elevazione			
16	3.3.1 Applicazione casseri per pilastri	4h	1 sq: n.1 Op. Spec./n.1 Op. Com./ n.1 Op. Qualif.	n.6 pil.
17	3.3.2 Getto pilastri	2h	1 sq: n.1 Op. Spec./n.1 Op. Com.	2,5 m3
18	3.3.3 Applicazione casseri per travi	6h	1 sq: n.1 Op. Spec./n.1 Op. Com./ n.1 Op. Qualif.	33,2 ml
19	3.3.4 Getto delle travi	2h	1 sq: n.1 Op. Spec./n.1 Op. Com.	4,34 m3
20	3.3.5 Applicazione casseri per solaio	32h	1 sq: n.1 Op. Spec./n.1 Op. Com./ n.1 Op. Qualif.	30,4 ml + 51,7m2
21	3.3.6 Getto solaio di copertura	4h	1 sq: n.1 Op. Spec./n.1 Op. Com.	13,9 m3
	4- MURATURE	TOT: 71h		
22	4.1 Tamponatura esterna	40h	1 sq: n.1 Op. Spec./n.1 Op. Com.	66,37 m2
23	4.2 Rivestimento	25h	1 sq: n.1 Op. Spec./n.1 Op. Com.	82 m2
24	4.3 Tramezzatura interna	6h	1 sq: n.1 Op. Spec./n.1 Op. Com.	14,3 m2
	5- IMPIANTI	TOT: 11h		
	5.1 Impianto idrico			
25	5.1.1 Montaggio impianto	3h	1sq: n.2 Op. Spec.	n.1
26	5.1.2 Getto massetto autolivellante	4h	1 sq: n.1 Op. Spec./n.1 Op. Com.	48,71 m2
27	5.2 Apparecchi sanitari	4h	1 sq: n.1 Op. Com.	n.2
	6- FINITURE	TOT: 184h		
28	6.1 Pavimentazione	16h	1 sq: n.1 Op. Spec./n.1 Op. Com.	48,1 m2
	6.2 Intonacatura			
29	6.2.1 Pareti interne	45h	1 sq: n.1 Op. Qualif./n.1 Op. Com.	108,26
30	6.2.2 Pareti esterne	40h	1 sq: n.1 Op. Qualif./n.1 Op. Com.	82 m2
	6.3 Montaggio infissi			
31	6.3.1 Montaggio controtelai	4h	1 sq: n.1 Op. Qualif.	n.3
32	6.3.2 Posa in opera infissi	6h	1 sq: n.1 Op. Qualif.	n.5
	6.4 Tinteggiatura			
33	6.4.1 Pareti interne	35h	1 sq: n.2 Op. Spec.	108,26
34	6.4.2 Pareti esterne	24h	1 sq: n.2 Op. Spec.	82 m2
	6.5 Sistema di raccolta acque piovane			
35	6.5.1 Montaggio canale di gronda	5h	1 sq: n.1 Op. Spec./n.1 Op. Com.	33,2 ml
36	6.5.2 Montaggio di pluviale in rame	2h	1 sq: n.1 Op. Spec./n.1 Op. Com.	13 ml

Figura 8 – Esempio associazione dei tempi alle attività, sono riportati anche i dati relativi alle risorse umane e alle quantità (immagine tratta da Angelotti Andrea, 2021)

Nell'esempio preso dallo scritto di Angelotti Andrea (2021), per determinare la durate di ciascuna attività, l'autore prende in considerazione le ore/uomo che fornisce il Prezzario, cercando di produrre una stima il più realistica possibile utilizzando anche il parere di esperti tramite l'impresa Edil Pro Ristrutturazioni s.r.l. e, ove possibile, un confronto per analogia tra Task simili. In Figura 8 sono riportate le attività che compongo il WBS, con l'associazione delle durate ad ogni task.

1.4.5 Limiti, vincoli e predecessori

Tutti i progetti hanno dei vincoli ovvero delle restrizioni o delle limitazioni, sia interna che esterna al progetto, che influisce quindi sulle prestazioni. Sono fattori che influiscono sul modo di gestire il progetto da parte del Planner e del team. Questi vincoli sono una parte inevitabile di qualsiasi progetto e una volta scoperti e classificati, non possono essere trascurati. Infatti, se il Planner non li considera e non presta la dovuta attenzione ai limiti imposti, essi potrebbero ripercuotersi negativamente sulla pianificazione in quanto non solo potrebbe verificarsi un blocco delle attività, ma una pianificazione inadeguata potrebbe costringere a interrompere totalmente il progetto.

Nel caso proposto da Angelotti Andrea (2021), essendo un progetto non reale non si hanno limiti o vincoli che incidono nella pianificazione e nella realizzazione dell'opera, per cui per rendere lo studio il più realistico possibile l'autore ha attribuito dei vincoli fittizi e dei limiti al progetto che possono presentarsi nella realtà compatibilmente con un progetto di tali dimensioni, come per esempio: 1) Data inizio lavori lunedì 30/05/2022; 2) L'Impresa Esecutrice non lavora il sabato e la domenica; 3) Orario di Lavoro degli operatori è di 8 ore al giorno cad., 4 mattina e 4 pomeriggio. Questi vincoli fittizi sono vincoli fondati su una modalità organizzativa che varia da impresa a impresa.

Individuati i vincoli generali si dovranno individuare i vincoli di natura fisica ovvero quei vincoli che dipendono dal processo naturale edilizio. Ad esempio, l'intonaco di una muratura andrà necessariamente steso soltanto in una fase successiva alla tamponatura (vincolo di successione fisica interno al processo). Una volta individuate le attività che fanno parte del WBS, si potrà procedere con l'assegnazione dei predecessori a ciascuna attività. La precedenza indica al software di schedulazione quale attività precede l'attività in questione, la quale può iniziare solo quando quella precedente è terminata (Finish To Start); questa relazione è dettata da un vincolo fisico del processo edilizio. Nell'esempio tratto dallo scritto di Angelotti Andrea (2021), le precedenze, che andranno inserite nel Gantt per ciascuna attività della WBS, sono riportate nell'apposita colonna della tabella in Figura 9.

Essendo un'opera di piccole dimensioni, nella maggior parte dei casi le attività sono in sequenza e non sono presenti precedenze molto complicate da dover gestire da parte del Planner. Ad ogni task che compone la WBS, ovvero tutte le attività da svolgere, viene associato un codice ID (in magenta) utile a riconoscere le attività durante le fasi della schedulazione. Questa fase di associazione dei vincoli aiuta ad attribuire in maniera semplice tutti i

predecessori delle attività che si trovano nell'apposita colonna (gialla).

ID			RISORSE UMANE	QUANTITA'	PREDECESSORI
	1- ALLESTIMENTO CANTIERE	TOT: 17,5h			
1	1.1 Delimitazione area cantiere	4h	1 sq: n.1 Op. Spec./n.1 Op. Com.	75 ml	
	1.2 Impianti di alimentazione e distribuzione elettrica				
2	1.2.1 Allestimenti quadri elettrici	4h	1sq: n.2 Op. Spec.	n.5	1
3	1.2.2 Posizionamento cavi e linee di alimentazione	3h	1sq: n.2 Op. Spec.	50 ml	2
4	1.2.3 Esecuzione impianto di terra	1h	1sq: n.1 Op. Spec.	n.1	3
	1.3 Baraccamenti e apprestamenti igenico-sanitari				
5	1.3.1 Preparazione e posa baraccamenti	2 h	1 sq: n.1 Op. Spec./n.1 Op. Com.	n.2	4
6	1.2.2 Allacciamenti e opere fognatura	2,5h	1 sq: n.1 Op. Spec./n.1 Op. Com.	n.2	5
7	1.4 Istallazione macchine	1h	1sq: n.1 Op. Spec.	n1	6
	2- SCAVI	TOT: 16h			
8	2.1 Scavi con mezzi meccanici	6h	1 sq: n.1 Op. Spec./n.1 Op. Com./ n.1 Op. Qualif.	57 m3	6
9	2.2 Scavi eseguiti a mano	10h	1 sq: n.1 Op. Spec./n.1 Op. Com.	4 m3	8
	3- STRUTTURE IN C.A	TOT: 78h			
	3.1 Strutture di fondazione				
10	3.1.1 Magrone	2h	1 sq: n.1 Op. Spec./n.1 Op. Com.	10,5 m3	9
11	3.1.2 Casseri fondazione	8h	1 sq: n.1 Op. Spec./n.1 Op. Com./ n.1 Op. Qualif.	41,4 ml	10
12	3.1.3 Getto della fondazione	4h	1 sq: n.1 Op. Spec./n.1 Op. Com.	31,5 m3	11
	3.2 Applicazione sist. Interrati di drenaggio e impermeabiliz.				
13	3.2.1 Applicazione vespaio	5h	1 sq: n.1 Op. Spec./n.1 Op. Com.	105 m2	12
14	3.2.2 Applicazione membrana	3h	1 sq: n.1 Op. Spec./n.1 Op. Com.	22 m2	13
15	3.2.3 Rinterro con mezzi meccanici	6h	1 sq: n.1 Op. Spec./n.1 Op. Com.	61 m3	14
	3.3 Strutture in elevazione				
16	3.3.1 Applicazione casseri per pilastri	4h	1 sq: n.1 Op. Spec./n.1 Op. Com./ n.1 Op. Qualif.	n.6 pil.	12
17	3.3.2 Getto pilastri	2h	1 sq: n.1 Op. Spec./n.1 Op. Com.	2,5 m3	16
18	3.3.3 Applicazione casseri per travi	6h	1 sq: n.1 Op. Spec./n.1 Op. Com./ n.1 Op. Qualif.	33,2 ml	17
19	3.3.4 Getto delle travi	2h	1 sq: n.1 Op. Spec./n.1 Op. Com.	4,34 m3	18
20	3.3.5 Applicazione casseri per solaio	32h	1 sq: n.1 Op. Spec./n.1 Op. Com./ n.1 Op. Qualif.	30,4 ml + 51,7m2	19
21	3.3.6 Getto solaio di copertura	4h	1 sq: n.1 Op. Spec./n.1 Op. Com.	13,9 m3	20
	4- MURATURE	TOT: 71h			
22	4.1 Tamponatura esterna	40h	1 sq: n.1 Op. Spec./n.1 Op. Com.	66,37 m2	19
23	4.2 Rivestimento	25h	1 sq: n.1 Op. Spec./n.1 Op. Com.	82 m2	22
24	4.3 Tramezzatura interna	6h	1 sq: n.1 Op. Spec./n.1 Op. Com.	14,3 m2	21
	5- IMPIANTI	TOT: 11h			
	5.1 Impianto idrico				
25	5.1.1 Montaggio impianto	3h	1sq: n.2 Op. Spec.	n.1	24
26	5.1.2 Getto massetto autolivellante	4h	1 sq: n.1 Op. Spec./n.1 Op. Com.	48,71 m2	25
27	5.2 Apparecchi sanitari	4h	1 sq: n.1 Op. Com.	n.2	28
	6- FINITURE	TOT: 184h			
28	6.1 Pavimentazione	16h	1 sq: n.1 Op. Spec./n.1 Op. Com.	48,1 m2	26
	6.2 Intonacatura				
29	6.2.1 Pareti interne	45h	1 sq: n.1 Op. Qualif./n.1 Op. Com.	108,26	24
30	6.2.2 Pareti esterne	40h	1 sq: n.1 Op. Qualif./n.1 Op. Com.	82 m2	22
	6.3 Montaggio infissi				
31	6.3.1 Montaggio controtelai	4h	1 sq: n.1 Op. Qualif.	n.3	29
32	6.3.2 Posa in opera infissi	6h	1 sq: n.1 Op. Qualif.	n.5	31
	6.4 Tinteggiatura				
33	6.4.1 Pareti interne	35h	1 sq: n.2 Op. Spec.	108,26	32
34	6.4.2 Pareti esterne	24h	1 sq: n.2 Op. Spec.	82 m2	32
	6.5 Sistema di raccolta acque piovane				
35	6.5.1 Montaggio canale di gronda	5h	1 sq: n.1 Op. Spec./n.1 Op. Com.	33,2 ml	34
36	6.5.2 Montaggio di pluviale in rame	2h	1 sq: n.1 Op. Spec./n.1 Op. Com.	13 ml	35

Figura 9 – Esempio predecessori, sono riportati anche i dati relativi ai tempi, alle risorse umane e alle quantità (immagine tratta da Angelotti Andrea, 2021)

1.4.6 Gestione dei tempi

L'autore dell'esempio proposto ha deciso di utilizzare il diagramma di Gantt. Per gestire i tempi di un lavoro attraverso il diagramma di Gantt occorre effettuare tre operazioni fondamentali, le quali sono già state sviluppate precedentemente, ovvero: 1) Elencare tutte le attività necessarie per poter portare a compimento il progetto; 2) Determinare il tempo necessario per eseguire ciascuna operazione; 3) Realizzare un programma costituito da una successione logica delle operazioni considerate.

Il diagramma prevede di rappresentare la lista delle attività di progetto in uno schema grafico nel quale sull'asse verticale sono rappresentate le attività e sull'asse orizzontale il tempo, la cui unità di misura è definita dal calendario di progetto o più in generale dalle esigenze di reportistica del Planner. All'interno di questo schema le attività sono rappresentate come barre orizzontali proporzionali alla durata delle attività che, a seconda del loro posizionamento, indicano la data massima o minima di inizio e di fine attività. Si possono inserire le date minime di inizio e di fine progetto, come anche le date massime di inizio e di fine in modo da calcolare lo scorrimento totale. Pertanto, da questo genere di report tabellare si ricava che, se un'attività risulta priva di scorrimento è un'attività critica.

La fase realizzativa del diagramma di Gantt, come per quasi tutte le pratiche lavorative attuali, sfrutta i diversi applicativi con i quali è possibile supportare, organizzare la gestione del lavoro. La scelta di alcune piattaforme rispetto ad altre è ovviamente frutto di molteplici variabili. Come per esempio l'interoperabilità, il costo e le licenze messe a disposizione dall'ateneo. Al fine della realizzazione del programma lavori l'autore dell'esempio ha deciso di utilizzare il software GanttProject. Il grafico redatto dal software è leggibile e fruibile su Navisworks, inoltre GanttProject è ben strutturato per quanto riguarda la realizzazione del grafico a barre, le impostazioni e personalizzazioni degli input e del calendario. L'unico svantaggio che evidenzia è l'impossibilità di poter imporre delle durate in ore o differenti al giorno intero. Invece di stabilire manualmente le date d'inizio di tutte le attività, si possono utilizzare i vincoli di precedenza e lasciare formulare lo scheduling al programma. Per una lettura più semplice e intuitiva sono stati riportate le attività con gli stessi colori della WBS. In Figura 10 è riportato il cronoprogramma dei lavori edili dell'esempio proposto.

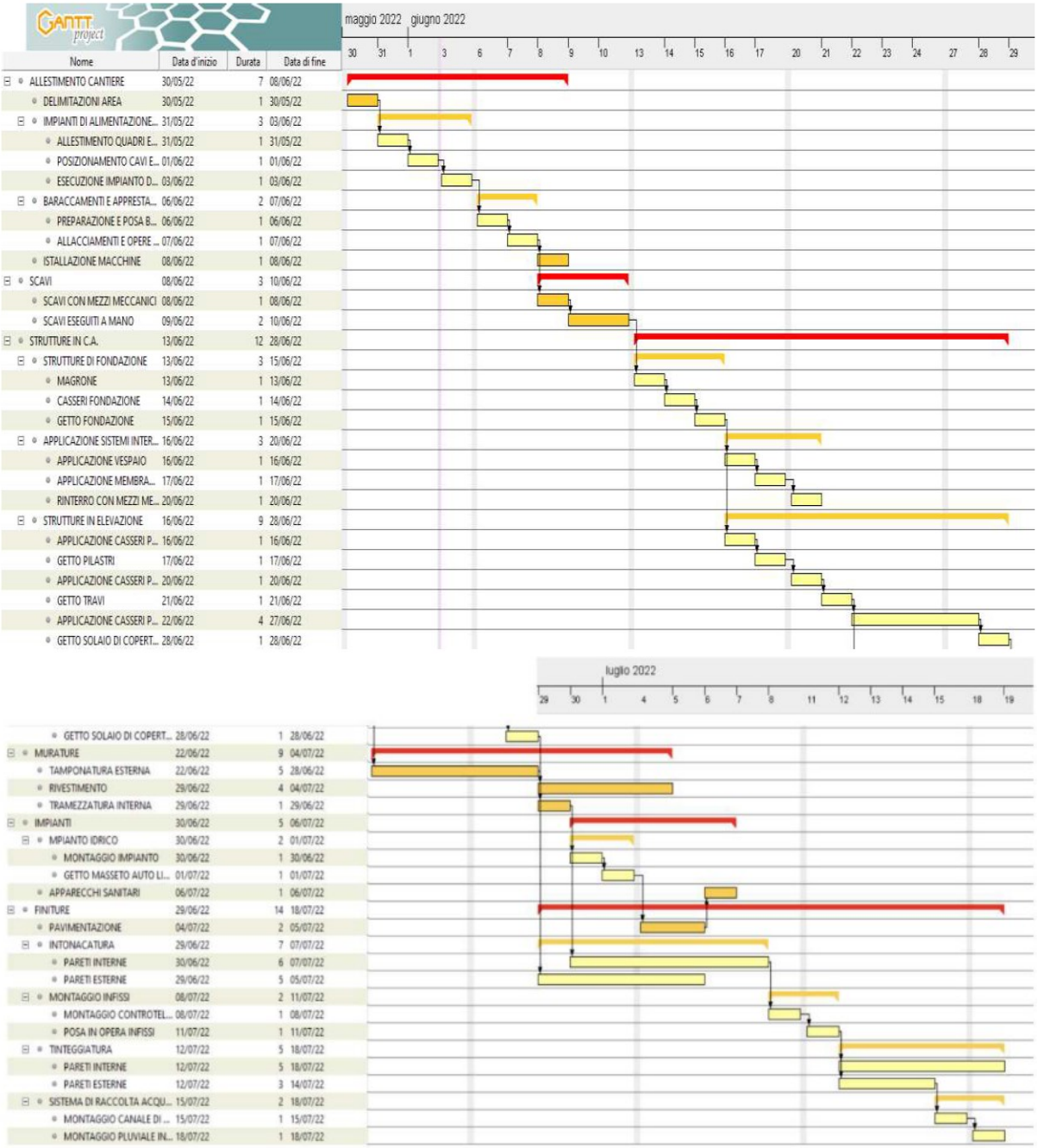


Figura 10 – Esempio cronoprogramma dei lavori edili (immagine tratta da Angelotti Andrea, 2021)

2 PIANIFICAZIONE ASSISTITA DEI LAVORI EDILI TRAMITE STRUMENTI COMPUTAZIONALI

Negli anni si sono effettuati numerosi studi per provare ad automatizzare la pianificazione dei lavori edili, lo scopo di questo paragrafo è riassumere gli strumenti più interessanti per la pianificazione assistita derivanti dalla letteratura Prima però per avere gli strumenti necessari a comprendere gli studi proposti si chiariscono i concetti e le utilità del BIM, del formato IFC e della pianificazione 4D.

2.1 BIM - Building Information Modeling

Per molti anni la questione sul come gestire gli edifici secondo i principi di efficienza ed efficacia ha posto una notevole sfida per il mondo delle costruzioni. Questo dibattito ha avuto rinnovata attenzione a seguito dell'avvento del Building Information Modeling (BIM). Un modello BIM contiene informazioni riguardanti l'edificio o le sue parti come la localizzazione geografica, la geometria, le proprietà dei materiali e degli elementi tecnici, le fasi di realizzazione, le operazioni di manutenzione. Consente di integrare in un modello le informazioni utili in ogni fase della progettazione, da quella architettonica a quella esecutiva, (strutture, impianti, sicurezza, manutenzione, prestazioni energetiche, ecc.) e gestionale (computi metrici, distinte fornitori, ecc.). In pratica il BIM con un unico modello permette di gestire ed integrare tutte le fasi della costruzione, dall'idea alla demolizione dell'opera, vedi Figura 11 (Berchet, 2016).

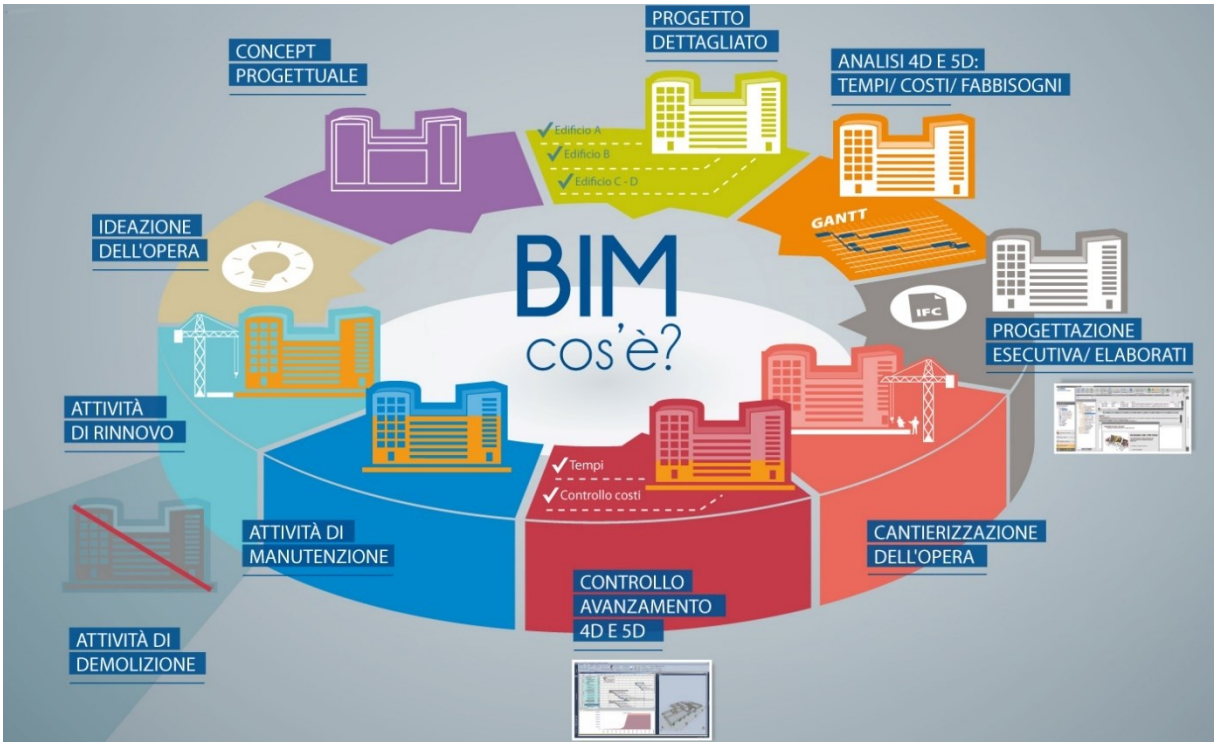


Figura 11 – Attività dell'industria delle costruzioni che sfruttano il BIM, dall'idea alla demolizione (immagine tratta da Berchet, 2016)

L'adozione di questa metodologia, basata sull'impiego di un unico sistema coerente di modelli 3D anziché su disegni di progetto separati, favorisce la collaborazione tra le figure coinvolte nell'intero ciclo di vita di un'opera (architetti, ingegneri, appaltatori, operai edili, produttori di materiali e componenti, eccetera) e assicura l'accesso a informazioni sempre aggiornate, riducendo la percentuale di errori e il numero di modifiche. Il BIM però non è semplicemente un nuovo formato di rappresentazione 3D o un software, ma una tecnologia che permette di creare un modello informativo, multidisciplinare e condiviso contenente le informazioni utili in ogni fase della progettazione (Namirial, 2022).

L'adozione di questo sistema di progettazione permette anche un miglior lavoro in team, all' interno del quale progettisti, strutturisti, impiantisti ed impresa edile possono collaborare su un progetto centralizzato occupandosi ognuno della propria disciplina. I vantaggi derivanti dall'uso di un sistema di progettazione BIM sono molti: risulta essere più ricco, completo e preciso rispetto all' equivalente redatto con i tradizionali sistemi CAD bidimensionali. La partecipazione attiva, l'accesso a informazioni sempre aggiornate, e il coordinamento imposto dalla condivisione di un unico modello riduce infatti drasticamente il tasso percentuale di errori e incongruenze, abbassando come conseguenza il numero di modifiche e, in ultima istanza, i costi di progettazione, rendendo il tutto più economicamente sostenibile (Berchet, 2016).

Esistono quattro diversi livelli di collaborazione condivisa in un progetto, noti come livelli di maturità BIM:

1) Livello 0 – CAD standardizzato: richiede l'organizzazione di un lavoro tradizionale intorno a un sistema di standard in cui la produzione e la condivisione delle informazioni avviene tramite documenti cartacei;
2) Livello 1 – BIM Solitario (lonely BIM) o non collaborativo: si usa il metodo di progettazione parametrica e gestione dei dati all'interno del proprio workflow ma non si instaura nessun tipo di collaborazione con gli altri professionisti. In questo caso viene utilizzato un Common Data Environment (CDE), è un archivio condiviso online in cui vengono raccolti, archiviati e organizzati tutti i dati del progetto;
3) Livello 2 – BIM Collaborativo: in questo caso c'è collaborazione tra i soggetti coinvolti nella progettazione, coordinata dal BIM Leading consultan, e le informazioni sono condivise attraverso un formato file comune;
4) Livello 3 – BIM Condiviso: tutti i professionisti lavorano contemporaneamente allo stesso modello in modo da recepire gli aggiornamenti in tempo reale.

Con la Direttiva 2014/24/EU, l'Unione Europea incoraggia il ricorso al sistema BIM per accrescere l'efficacia e la trasparenza delle procedure di appalto. Nell'ordinamento italiano, in particolare, modalità e tempi per l'introduzione del BIM negli appalti pubblici per opere edilizie e infrastrutture sono stati definiti dal Decreto del Ministero delle Infrastrutture numero 560 del 1º dicembre 2017. Con il decreto il ricorso al Building Information Modeling è diventato

obbligatorio a partire dall'anno 2019 per le opere di importo a base di gara pari o superiore a 100 milioni di euro. Tale importo è destinato a diminuire progressivamente negli anni (Namirial, 2022). Dal 1° gennaio 2022, invece, è diventato obbligatorio l'uso delle metodologie BIM per le opere pubbliche di valore pari o superiore a 15 milioni di euro. Ed è per questo che oggi sono numerosi gli operatori economici (società di progettazione, grandi imprese, stazioni appaltanti, committenze pubbliche e private) che si sono "evoluti al BIM", investendo nell'innovazione e nella digitalizzazione. Il decreto 2 agosto 2021, n. 312 modifica il precedente dm 560/2017 sul BIM e introduce una diversa tempistica sull'obbligatorietà dell'uso di metodi e strumenti elettronici di modellazione per l'edilizia e le infrastrutture negli appalti pubblici. In Figura 12 vediamo gli obblighi di adozione del BIM negli anni, dove gli importi in rosso sono riferiti alla direttiva del 2019 che è stata superata (BibLus-net, 2022).

OBBLIGO DI ADOZIONE DEL BIM
nelle opere pubbliche

BibLus-net

TIPOLOGIA DI LAVORI	IMPORTO	DATA
Lavori complessi	≥ **100 milioni** di euro	dal 1° gennaio 2019
	≥ **50 milioni** di euro	dal 1° gennaio 2020
	≥ **15 milioni** di euro	dal 1° gennaio 2021
Opere di **nuova costruzione** ed interventi su **costruzioni esistenti**, fatta **eccezione per le opere di manutenzione ordinaria**	≥ **15 milioni** di euro ~~(≥ 5,35 milioni di euro)~~	dal 1° gennaio 2022
Opere di **nuova costruzione** ed interventi su **costruzioni esistenti**, fatta **eccezione per le opere di manutenzione ordinaria e straordinaria**	≥ **5,35 milioni** di euro ~~(≥ 1 milione di euro)~~	dal 1° gennaio 2023
Opere di **nuova costruzione** ed interventi su **costruzioni esistenti**, fatta **eccezione per le opere di manutenzione ordinaria e straordinaria**	≥ **1 milione** di euro ~~(≤ 1 milione di euro)~~	dal 1° gennaio 2025

Figura 12 – Obbligo di adozione del BIM nelle opere pubbliche (immagine tratta da BibLus-net, 2022)

Inoltre, la normativa italiana UNI 11337-6 individua sette dimensioni del Building Information Modeling che fanno riferimento ai diversi livelli di informazione che si possono trovare in un BIM model:

1D: Concept design;

2D: Elaborati 2D come piante, prospetti e sezioni;

3D: Rappresentazione tridimensionale del prodotto;

4D: Analisi della durata o tempi (programmazione);

5D: Gestione economica (analisi dei costi, stime e valutazioni economiche);

6D: Fase di gestione dell'opera (uso, manutenzione e dismissione);

7D: Valutazione della sostenibilità (sociale, economica e ambientale) (Namirial, 2022).

2.2 IFC - Industry Foundation Classes

Come detto nel paragrafo precedente il BIM consente l'interoperabilità e l'interscambio di dati in modo sicuro, senza errori e perdita di informazioni, rispetto al vecchio metodo dello scambio di disegni 2D (vedi Figura 13). I vari professionisti hanno però a disposizione un ampio ventaglio di software BIM da utilizzare per svolgere il loro compito, e ogni software ha il proprio formato proprietario che non è sempre compatibile con gli altri software. Per questo motivo è nato il formato IFC (Industry Foundation Class) che è un formato dati aperto, non controllato da un singolo operatore, promosso dal concetto di openBIM e nato per facilitare l'interoperabilità. In questo modo un modello BIM è compatibile con tutti i software BIM.

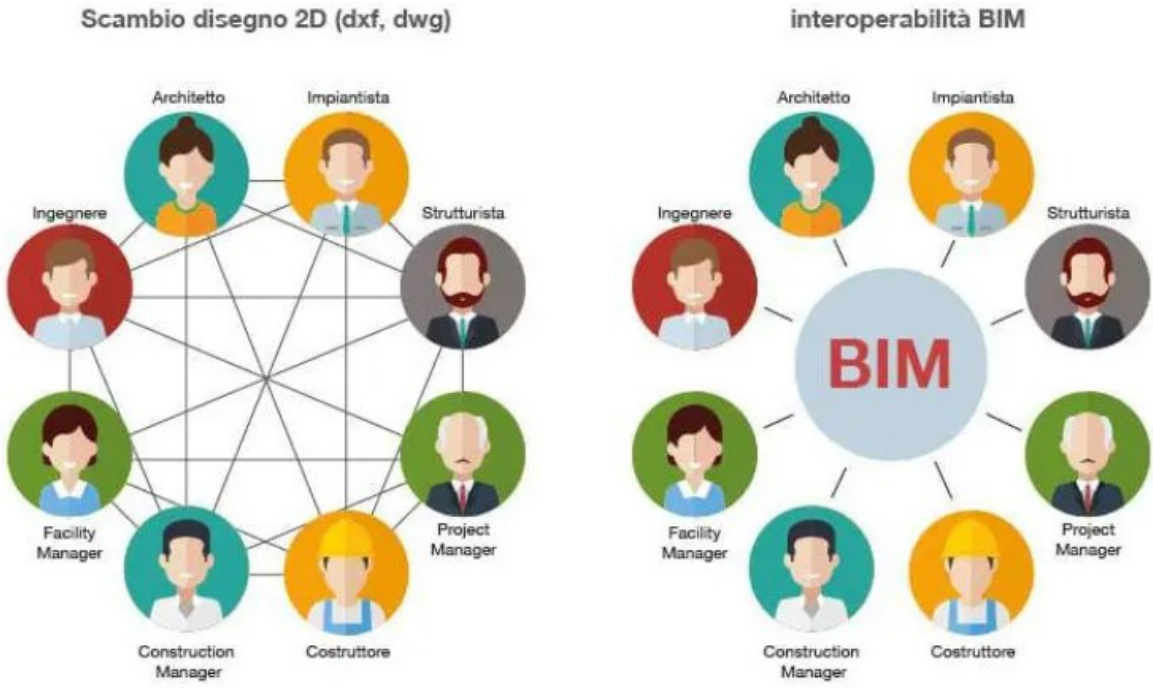


Figura 13 – Interoperabilità BIM (immagine tratta da Furcolo Nicola, 2016)

L'iniziativa IFC ha avuto inizio nel 1994, quando un consorzio industriale investì nello sviluppo di un insieme di classi C ++ in grado di supportare lo sviluppo di applicazioni integrate. Dodici società statunitensi aderirono al consorzio. Queste aziende inizialmente chiamarono il consorzio "Industry Alliance for Interoperability", nel settembre 1995 l'Alleanza aprì l'adesione a tutte le parti interessate e nel 1997 cambiò il suo nome in "International Alliance for Interoperability". La nuova alleanza fu ricostituita come organizzazione non-profit, con

l'obiettivo di sviluppare e promuovere "l'Industry Foundation Class" (IFC) come modello dati neutro di prodotto dell'edilizia utile a raccogliere informazioni lungo tutto il ciclo di vita di un edificio/impianto. Dal 2005 l'Alleanza porta avanti le proprie attività tramite "BuildingSMART International" che opera a livello internazionale attraverso il lavoro congiunto dei suoi capitoli nazionali. La specifica del modello dati IFC è stata riconosciuta e registrata dalla ISO come norma internazionale ISO 16739:2013 (IBIMI, n.d.).

BuildingSMART International è dunque l'attuale ente internazionale che mira a migliorare lo scambio di informazioni tra le applicazioni software utilizzate nel settore delle costruzioni, e per questo svolge due funzioni principali:

- Mantiene e sviluppa la libreria "BuildingSMART Data Dictionary", la quale è fondamentale per orientarsi all'interno del formato IFC.
- Gestisce e sviluppa il modello di dati IFC, indipendente da ogni software.

La nascita di IFC fa parte di un lungo percorso di concettualizzazione costituito da quattro principi:

- Un vocabolario comune con il principio di "IS_A" costituito dalla lista delle classi/entità necessarie a definire gli oggetti, esempio "IfcWall".
- Le classi sono poi legate dal principio di "IS_KIND_OF" che permette di associare gli elementi in superclassi e sottoclassi, esempio "IfcWall" e "IfcWindow" sono sottoclassi di "IfcBuildingElement", cioè un muro e una finestra sono nello stesso livello all'interno degli elementi dell'edificio. Lo stesso vale per un'apertura nel muro (IfcOpening) la quale può essere riempita da una porta (IfcDoor), da una finestra (IfcWindow) o rimanere una semplice apertura.
- Gli oggetti possono essere composti da più parti, secondo il principio di "IS_PART_OF", esempio questo permette di costruire la relazione tra una facciata continua "IfcCurtainWall" e le sue componenti: pannello (IfcPlate), montanti (IfcMember).
- Capacità dello schema IFC di rappresentare sequenze logiche grazie alle classi di relazione, esempio l'appartenenza ad una struttura spaziale (IfcRelContainedInSpatialStructure) (Paolo Borin & Carlo Zanchetta, 2020).

La potenza del linguaggio IFC si trova nella sua capacità di standardizzare e codificare univocamente in maniera gerarchica le seguenti componenti di un modello BIM:

- identità e semantica, ovvero il riconoscimento dell'oggetto in maniera meccanica tramite un identificativo univoco che ne riassume il nome, il tipo di oggetto e la sua funzione;
- caratteristiche e attributi, ad esempio le informazioni sui materiali e l'intero set di proprietà fisiche, chimiche e termiche, nonché le informazioni legate al colore;
- relazioni in essere (in termini di posizionamento, collegamenti/connessioni ecc.): tra i diversi oggetti che compongono il modello, come travi e colonne in sistemi strutturali a

telaio; tra i concetti astratti, come le attività di analisi attuabili attraverso il modello BIM (performance, costi, funzionamento impianti); tra i processi (ad esempio attività manutentive oppure installazione dei componenti); tra gli stakeholder che collaborano all'interno del progetto.

Il risultato finale dell'IFC è quello di poter trasmettere (e archiviare) il modello informativo mantenendo intatte le logiche e le informazioni geometrico-documentali a esso connesse (vedi Figura 14). Perché ciò avvenga, è necessario avere una sorta di "dizionario" che consenta la traduzione di ognuno degli aspetti sopra descritti nella corretta codifica leggibile a qualsiasi software compatibile. Un programma di modellazione tridimensionale non è considerabile un software di BIM Authoring, se si trova privo della capacità di import/export dei file in formato IFC, la cui operatività sia stata verificata da BuildingSMART (Adhox, 2022).

Ulteriori dettagli su IFC verranno forniti nel Capitolo Terzo.

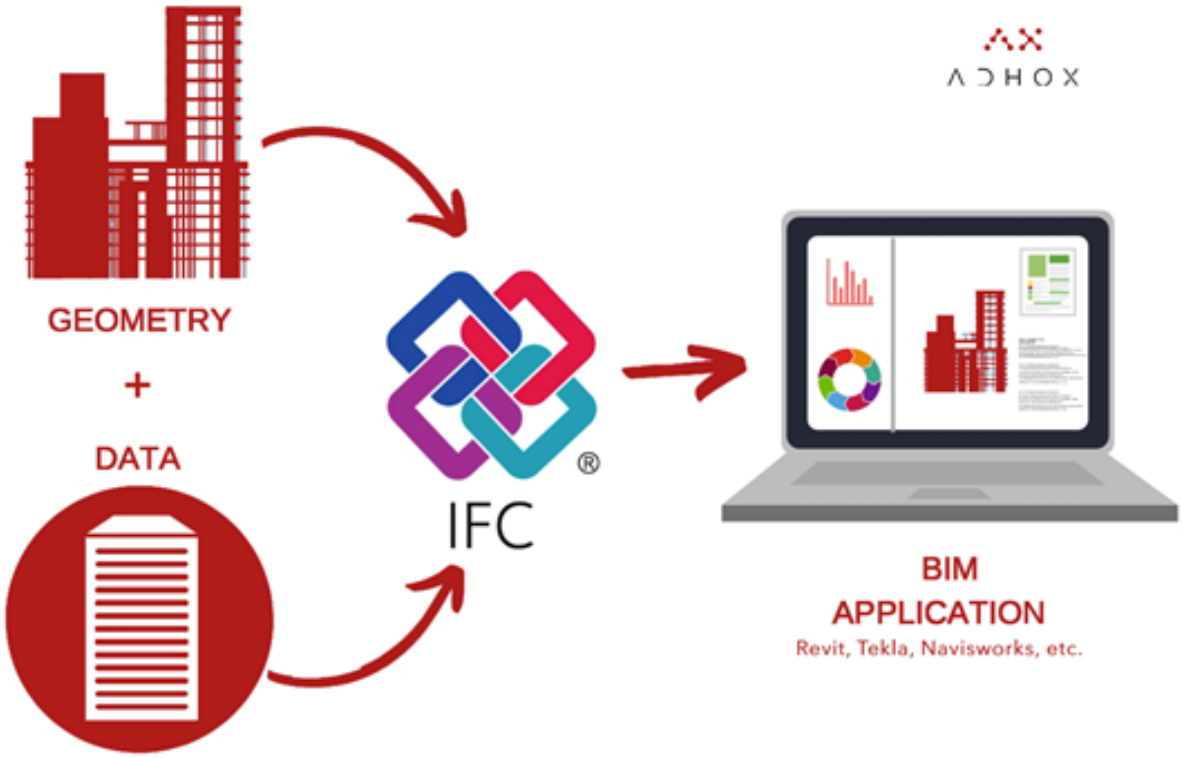


Figura 14 – Il modello IFC contiene le informazioni geometrico-documentali che si possono visualizzare tramite un software BIM (immagine tratta da Adhox, 2022)

2.3 Pianificazione 4D

Come si è detto, una delle dimensioni del BIM è il 4D il quale riguarda l'analisi e la programmazione dei tempi di lavoro. Infatti, la programmazione dei tempi di realizzazione di un'opera è un aspetto indispensabile per ottimizzare le risorse in fase di cantiere. A corredo di un progetto è sempre opportuno, se non obbligatorio (come negli appalti pubblici), redigere il cronoprogramma dei lavori (di cui si è parlato ampiamente nei paragrafi precedenti) per la

gestione della durata di un cantiere. Per committenti, progettisti e appaltatori è d'aiuto poter "visualizzare il cantiere" già durante la fase di progettazione, per prevedere e pianificare tutte le attività in funzione del tempo. Il BIM, e in particolare il BIM 4D, può essere la soluzione a queste esigenze e può rappresentare un valido supporto per ridurre i ritardi e le inefficienze in cantiere.

Figura 15 – Modello 3D BIM + cronoprogramma = modello 4D BIM (immagine tratta da Fuccelli & Fabio, 2017)

La pianificazione 4D o BIM 4D è l'unione tra il modello 3D dell'opera e le informazioni relative ai tempi di esecuzione dei vari lavori necessari per realizzare l'opera, cioè il cronoprogramma (vedi Figura 15). Il risultato è un modello informativo completo che può essere utilizzato anche per realizzare simulazioni realistiche del processo di realizzazione dell'opera in funzione del tempo. Lo scopo è identificare tutte le attività di cantiere (come in un tradizionale cronoprogramma), visualizzarne lo stato di avanzamento nel tempo e offrire agli operatori l'opportunità di identificare, analizzare e prevenire i problemi relativi agli aspetti sequenziali, spaziali e temporali del processo di cantierizzazione di un'opera edile. Questo rende il processo sicuro, migliora il controllo del rilevamento dei conflitti tra le diverse attività, limita l'insorgere di spiacevoli imprevisti durante il cantiere e conseguenti sprechi di tempo e risorse (BibLus-BIM, 2021).

I modelli 4D, come detto, sono molto utili però sono tipicamente fatti manualmente dal progettista che modella il 3D, compila il cronoprogramma, tramite i software disponibili sul mercato, ed infine gli unisce tramite dei visualizzatori 4D come Navisworks.

2.4 Strumenti di pianificazione assistita

Negli anni sono stati fatti numerosi studi per provare ad automatizzare la pianificazione dei lavori in cantiere, Desgagné-Lebeuf et al. (2019) hanno fatto una revisione sistematica degli strumenti di pianificazione assistita tramite computer dal 2008 al 2018, la quale viene presa come riferimento per il periodo citato, mentre per i successivi anni si effettua una ricerca classica. Di seguito si propone un riassunto degli articoli più interessanti in ordine cronologico, mentre le considerazioni verranno fatte nel Capitolo Secondo. L'elemento comune dei seguenti strumenti proposti è che almeno una fase della pianificazione è automatizzata.

Feng et al. (2010) hanno proposto di utilizzare il modello MD CAD (multi-dimensional CAD) per sviluppare la programmazione integrata tempo-costo per progetti di costruzione. Il primo passo del metodo, partendo dal progetto, è creare il modello MD CAD il quale sarà la

fonte di tutte le informazioni per la pianificazione. Poi la metodologia si divide in due fasi: 1- Matrice di sequenziamento degli oggetti, cioè le relazioni fisiche tra gli oggetti sono utilizzate per formulare delle sequenze di costruzione (non c'è un'unica sequenza ma sono possibili diverse soluzioni); 2- Applicazione degli algoritmi genetici, tramite gli algoritmi genetici si cerca la sequenza di costruzione migliore, e del Meccanismo evolutivo, cioè si sceglie il meccanismo con cui gli algoritmi genetici arrivano alla soluzione ottimale (in questo caso: relazione fisica tra gli oggetti e continuità di costruzione). Dall'applicazione del metodo ad un caso studio si è visto che esso ha le potenzialità per ridurre in modo significativo gli sforzi, gli errori della pianificazione dei lavori e migliorarla. In particolare, gli algoritmi genetici sono molto utili per fornire in poco tempo una pianificazione efficiente e flessibile. La gestione della costruzione in termini di tempi e costi è importante sviluppare una programmazione dettagliata. Il modello MD CAD il quale genera tutte le informazioni richieste, dato che il modello contiene anche le risorse ed i costi, è essenziale per una pianificazione integrata tempo-costo. Sono state fatte alcune semplificazioni in termini di spazi di lavoro, produttività e risorse.

Song et al. (2012) hanno proposto un sistema di ottimizzazione e simulazione del framework strutturale basato su BIM per gestire la programmazione di complessi processi edili (vedi Figura 16). Il metodo si divide in due fasi principali. La prima fase è la definizione della struttura gerarchica del framework strutturale: 1- Modellazione 3D basata sulla WBS; 2- Uso del formato IFC per rappresentare le informazioni geometriche e strutturali; 3- Configurazione della struttura di lavoro e mappatura dei dati quantificativi; 4- Definizione dei dati di specificazione delle risorse basata sulle proprietà per ciascun livello. La seconda fase è il calcolo per l'ottimizzazione della costruzione del framework strutturale: 1- Processo di calcolo; 2-

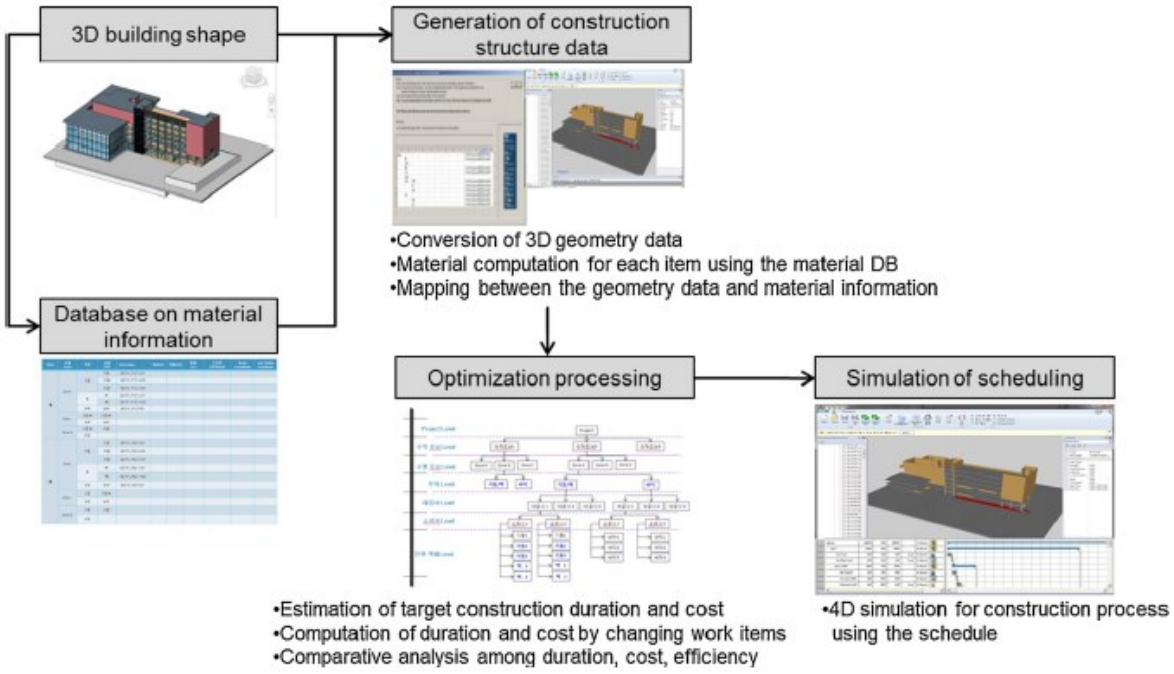


Figura 16 – Schema complessivo del processo (immagine tratta da Song et al., 2012)

Definizione del calcolo per dati quantificativi e la sua formula; 3- Definizione delle proprietà delle risorse umane, dell'equipaggiamento, dei materiali e la loro formula; 4- Definizione della formula tra lavoro su piccola scala e lavoro di livello superiore; 5- Definizione del calcolo delle proprietà aggiuntive. Infine, si ottiene un modello di simulazione della pianificazione 4D. Il sistema è stato applicato ad un caso studio e si è visto che il sistema facilita la pianificazione 4D. Gli utenti possono valutare varie pianificazioni 4D e decidere qual è più adatta ai vincoli, come la durata ed il costo di costruzione, che si sono posti all'inizio della pianificazione.

König et al. (2012) hanno proposto un metodo per la pianificazione intelligente della costruzione basata sul BIM utilizzando la simulazione di eventi discreti. L'obiettivo è proporre un approccio di pianificazione intelligente per: assegnare modelli di processo e definire le interdipendenze in modo automatico; gestire i diversi scenari disponibili e le modifiche. Il metodo segue i seguenti passaggi: Si utilizza un approccio multi modello per incorporare tutte le risorse di dati; Si definiscono dei modelli di processo per gli elementi da costruire; Si modellano le interdipendenze complesse tra i diversi elementi, livelli o sezioni; Si applica una simulazione basata su dei vincoli per generare delle pianificazioni valide. I modelli di processo e le interdipendenze sono classificati in dei templates in base al metodo di costruzione e alla logica di costruzione, e vengono selezionati in base al caso. Infine, si comparano le pianificazioni ottenute per selezionare la migliore. Il metodo è stato applicato ad un caso studio e si è visto che il base ai modelli di processo scelti si possono ottenere pianificazioni alternative. Si seleziona poi la pianificazione più adatta a rispettare i vincoli che il progetto pone. Un elemento interessante sono i templates dei modelli di processo e delle interdipendenze, i quali una volta definiti velocizzano e migliorano il processo.

H. Kim et al. (2013) hanno proposto un metodo per la pianificazione della costruzione attraverso l'estrazione automatica dei dati utilizzando la tecnologia BIM. Il metodo è scomposto in cinque fasi: A- Preparazione del modello BIM (modellazione, assegnazione dei materiali e strati, esportazione in formato IFCXML); B- Analisi dei dati IFCXML (estrarre la geometria, la posizione e le informazioni sui materiali degli elementi); C- Trasformare i dati analizzati in attività con la propria durata; D- Generare la pianificazione, D- Rifinire il processo (modifica manuale di alcune parti della pianificazione se necessario). Il metodo è stato applicato ad un caso studio e si è ottenuta una pianificazione per gli elementi di base (muri, solai, pavimenti, porte, finestre, copertura). Il file finale è stato prodotto per essere letto da Microsoft Project. Il metodo si concentra sullo scambio di dati utilizzando uno schema standard di dati internazionale consolidato, cioè IFCXML. Il metodo soffre delle limitazioni del formato IFCXML che è scalabile e complesso.

Wang et al. (2014) hanno proposto un metodo per utilizzare i dati del computo delle quantità forniti dal BIM per supportare le stime probabilistiche della durata delle attività lavorative coinvolgendole in una simulazione dei lavori in cantiere, per produrre alla fine una pianificazione della costruzione. Il metodo proposto di divide in cinque fasi principali (vedi Figura 17): 1- Sviluppo del modello BIM e calcolo delle quantità; 2- Modulo di interfaccia per la

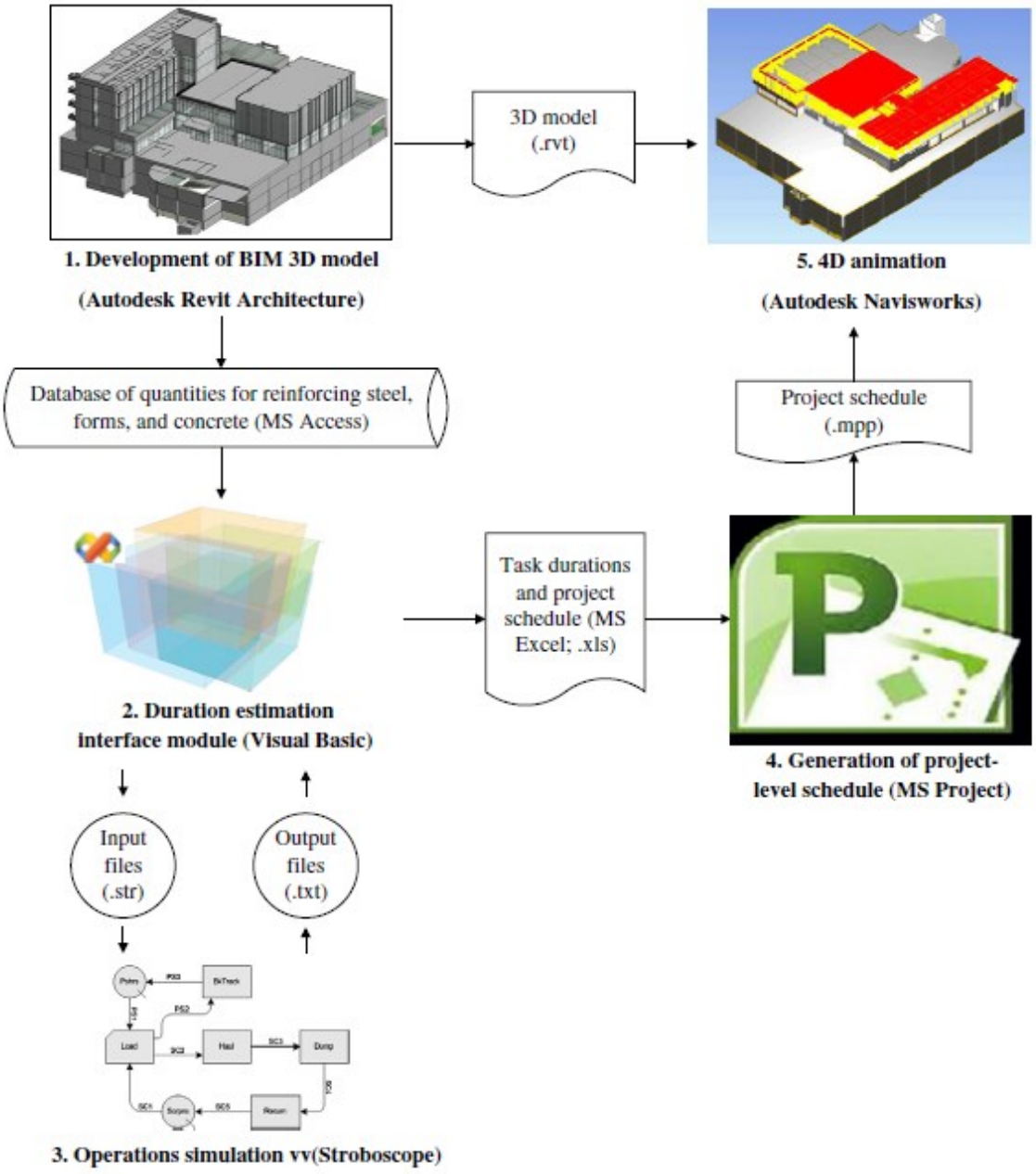


Figura 17 – Schema del metodo proposto (immagine tratta da Wang et al., 2014)

stima della durata (DEI), il quale si divide in due componenti: la componente di cattura delle quantità dei materiali e la componente di sviluppo delle funzioni principali (inserimento delle condizioni di costruzione, attivazione delle operazioni di stima delle durate tramite Stroboscope, fuoriuscita dei risultati della stima); 3- Simulazione delle operazioni di cantiere tramite Stroboscope; 4- Generazione della panificazione; 5- Animazione 4D unendo modello 3D con la pianificazione. Il metodo è stato applicato ad un caso studio e si sono ottenute 48 pianificazioni alternative. Tra le varie opzioni si è scelta l'alternativa 25 perché permette di ottenere la durata voluta senza spendere troppo e senza turni straordinari. L'applicazione positiva del metodo dimostra la sua efficacia. Questo lavoro ha dimostrato che l'integrazione

dei computi quantitativi del BIM e una simulazione delle operazioni è fattibile attraverso lo sviluppo di un sistema di interfaccia. Il BIM può supportare le simulazioni in Stroboscope fornendo le quantità materiali degli elementi costruttivi per valutare la durata delle attività in modo tempestivo e accurato. La simulazione delle operazioni consente la valutazione di varie strategie di allocazione delle risorse e considera la concorrenza tra le risorse nel generare un programma adeguato. Il modello 4D sviluppato presenta sequenze di lavoro affidabili delle attività di costruzione.

Faghihi et al. (2014) hanno proposto uno strumento per il supporto alla pianificazione della costruzione utilizzando gli algoritmi genetici basati sulle informazioni derivanti dal BIM, in cui i progettisti possano visualizzare in modo interattivo la comunicazione tra il processo di pianificazione e il modello 3D ad ogni incremento temporale. Il primo passo del metodo è quello di prendere il modello 3D dell'edificio ed esportarlo tramite il formato IFC, grazie a questo si genera la matrice MoCC, spiegata in un precedente articolo, la quale contiene tutti i dati di input dell'edifico da inserire poi nell'algoritmo genetico. Quest'ultimo è utilizzato per produrre una pianificazione della costruzione che permetta la costruibilità del progetto, ed è formato dai seguenti passi: creazione del genoma; selezione di membri elite; definizione della funzione di fitness; definizione del metodo di selezione; crossover; mutazione; validazione del genoma; soluzioni. Il metodo è stato poi validato modificando delle variabili di processo e monitorando l'effetto delle modifiche. Si è modificata la complessità del modello 3D, inserendo 3 modelli di complessità differente, e si sono modificati dei parametri dell'algoritmo genetico, ottenendo 7 cambi di settaggio. In totale quindi per validare il metodo si sono applicati 21 diversi scenari, combinazioni dei parametri modificati, e tutti hanno dato riscontro positivo in quanto si è ottenuta una pianificazione che permette la costruibilità del progetto, ovviamente con diversi tempi di esecuzione. Il metodo però ha considerati solo gli elementi strutturali.

Liu et al. (2015) hanno proposto un approccio integrato basato sul BIM per la pianificazione dettagliata della costruzione avendo vincoli di risorse. L'approccio proposto segue i seguenti passi: 1- creazione del modello BIM, il quale include tutte le informazioni sugli elementi presenti nel progetto; 2- creazione della WBS in un database MS Access; 3- processo di simulazione della costruzione, il quale riceve come input i dati BIM e WBS e ha il compito di stilare il processo di costruzione nel dettaglio; 4- integrazione del BIM e del processo di simulazione, tramite entità informative arricchite, 5- ottimizzazione della pianificazione della costruzione, con l'obiettivo di ottenere la minima durata del progetto con dei vincoli di risorse e per farlo si utilizza l'algoritmo PSO; 6- integrazione del processo di ottimizzazione e del processo di simulazione, per ottenere la pianificazione finale. Il metodo proposto è stato implementato per le costruzioni a pannelli (vedi Figura 18), le quali usano la tecnologia di realizzazione LGS (light gauge steel), ed il risultato è stato positivo convalidando il metodo.

Kim & Cho (2015) hanno proposto un sistema di ragionamento geometrico chiamato CSIR il quale deriva l'interpretazione spaziale specifica della costruzione dal modello BIM, in base alle relazioni geometriche tra le facce dei componenti, per supportare la pianificazione

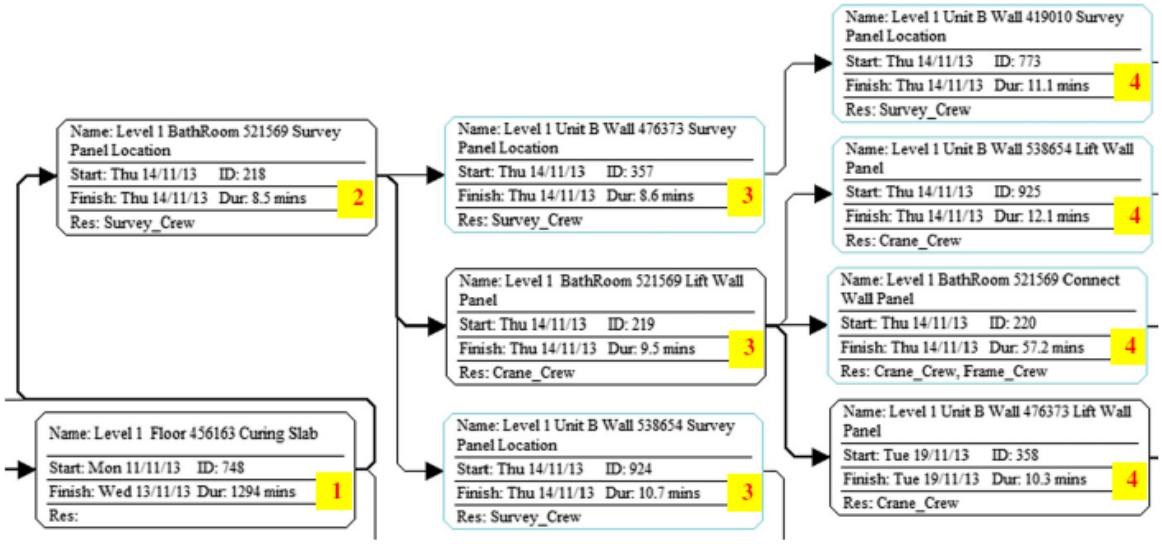


Figura 18 – Porzione di cronoprogramma generato dall'approccio proposto (immagine tratta da Liu et al., 2015)

automatizzata della costruzione. Il sistema proposto CSIR (construction spatial information reasoner) analizza le relazioni spaziali tra i componenti edilizi tramite degli algoritmi i quali hanno tre parti principali: 1- Struttura dati di relazione, la quale consente di memorizzare l'adiacenza geometrica di un elemento con altri elementi; 2- Valutazione delle relazioni di adiacenza, la quale tramite un algoritmo analizza l'adiacenza geometrica tra le facce degli elementi; 3- Generazione ed interpretazione del grafico di adiacenza, i quali permettono di ricavare informazioni spaziali specifiche sulla costruzione. Avendo tutte le informazioni geometriche e applicando al tipo di attività le proprie caratteristiche si ricava la pianificazione. Viene presentato un caso di studio per dimostrare il potenziale del sistema CSIR il quale ricava le informazioni dal BIM in modo da assistere la pianificazione di alcune attività interne selezionate (muri interni e controsoffitto). Il sistema analizza automaticamente le informazioni geometriche dell'edificio, applica le conoscenze di pianificazione e realizza una pianificazione realistica convalidando il metodo (vedi Figura 19). Il sistema proposto permette di superare le analisi manuali e soggettive ricavando le informazioni geometriche in modo automatico direttamente dal modello BIM. Il sistema ha alcune limitazioni: è applicabile solo per superfici rettangolari e planari, non considera la relazione tra i muri e gli impianti.

Ivson et al. (2018) hanno proposto un nuovo sistema, detto CasCADe, per la visualizzazione 4D della pianificazione virtuale della costruzione. Il sistema CasCADe presentato segue uno sviluppo basato sulla metodologia della progettazione mettendo al centro l'utente, infatti alcuni esperti sono coinvolti nello sviluppo del sistema. Il principio del sistema è usare una delle 3 coordinate spaziali del modello 3D per rappresentare il tempo, cioè gli oggetti sono traslati verticalmente lungo l'asse z in accordo con la loro data di completamento. Mentre le coordinate x e y mantengono l'informazione spaziale dell'oggetto in cantiere. Il sistema è progettato per visualizzare cinque elementi fondamentali della pianificazione 4D: 1- Programma generale; 2- Task prioritarie; 3- Quantità numeriche; 4- Relazioni tra gli oggetti; 5-

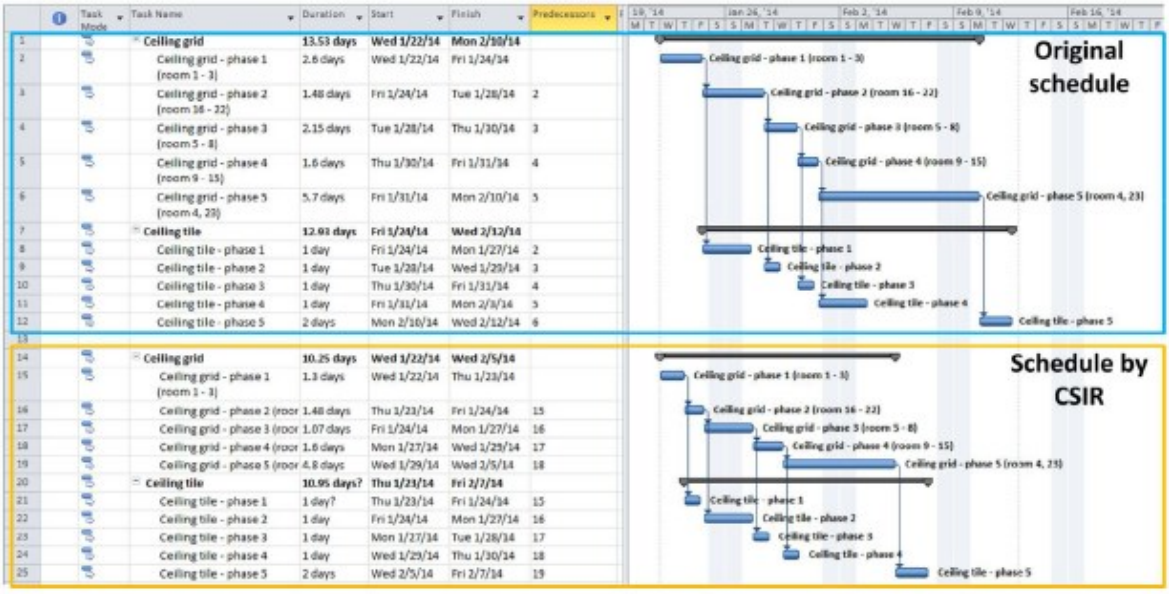


Figura 19 – Cronoprogramma originale manuale vs Cronoprogramma con il sistema CSIR (immagine tratta da Kim & Cho, 2015)

Layout del cantiere. Il sistema è stato testato da tre ingegneri collaboratori che lo hanno utilizzato per la pianificazione della costruzione di un impianto di lavorazione del petrolio e del gas (vedi Figura 20), essi hanno trovato alcune incertezze ma il sistema è stato approvato e convalidato. Il sistema CasCADe migliora l'efficacia della visualizzazione 4D, infatti esso mappa il tempo come una dimensione spaziale per creare un effetto di geometrie a cascata in una visualizzazione 3D esplosa. Viene a crearsi così un ambiente intuitivo dove lo spazio e il tempo sono simultaneamente e immediatamente evidenti.

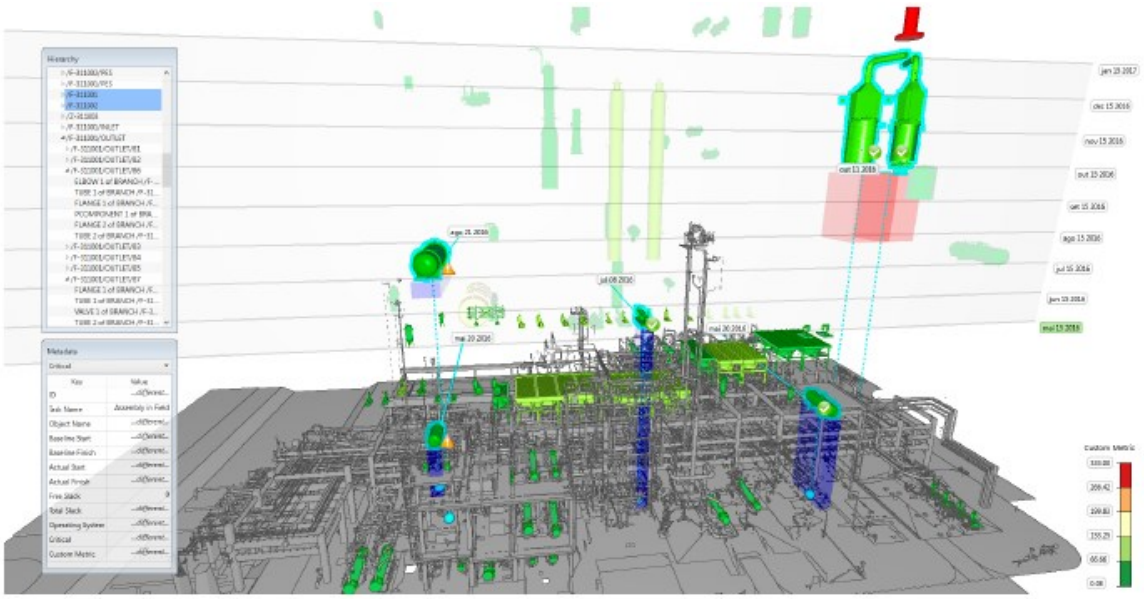


Figura 20 – L'esclusiva visualizzazione 4D di CasCADe che combina la sequenza delle attività dei diagrammi PERT o Gantt con la consapevolezza spaziale trasmessa dai modelli CAD 3D (immagine tratta da Ivson et al. 2018)

Hatori et al. (2021) hanno proposto un sistema di simulazione 4D basato sul BIM per impostare in modo automatico la data di inizio e fine dei lavori di ciascun elemento utilizzando regole di pianificazione derivate da esperti. Il metodo proposto per creare in modo automatico la pianificazione dei lavori segue tre passaggi: 1- Ricavare le informazioni spaziali dal modello BIM; 2- Includere nel modello BIM le informazioni riguardanti la costruzione che non sono già incluse, in questo caso specifico, ad esempio, si crea una funzione per creare in modo automatico il modello delle impalcature; 3- Basandosi sul parere di esperti, definire una sequenza dei lavori. È stato sviluppato un software che integra tutte le informazioni della costruzione e utilizza Navisworks per visualizzare la pianificazione 4D finale. Si è valutato il metodo proposto in comparazione con l'impostazione manuale classica e il risultato è che con il metodo proposto si riduce il tempo di sviluppo della pianificazione dell'80%.

Amer & Golparvar-Fard (2021) hanno proposto un metodo per la modellazione dinamica della pianificazione dei lavori da costruzione tramite il machine learning (apprendimento automatico) sequenziale derivato da pianificazioni esistenti, per risolvere i problemi di complessità della pianificazione manuale. Il metodo inizia con l'acquisizione dei dati derivanti da pianificazioni esistenti, i dati verranno analizzati per ricavare le descrizioni delle attività e le relazioni di dipendenza. Poi si passa alla formalizzazione dell'attività, che permette di riconoscere e raggruppare le attività simili, che si divide in tre step: 1- Incorporamento di parole per le descrizioni delle attività pianificate; 2- Etichettatura delle attività nel POA (part-of-activity); 3- Incorporamento delle attività. Infine, si passa alla formalizzazione della logica di pianificazione e delle dipendenze tra le attività ed il tutto viene archiviato nei DPT (Dynamic Process Templates) che permettono la pianificazione automatica. Per convalidare il metodo vengono raccolti 32 progetti reali ed elaborati secondo il procedimento descritto. Il risultato convalida il metodo che è in grado di elaborare i predecessori ed i successori per ciascuna attività. I DPT si sono dimostrati in grado di apprendere tra il 76% ed il 98% delle sequenze dei progetti inseriti e il metodo ha dimostrato la sua flessibilità nella gestione delle diverse espressioni di linguaggio per indicare le attività. Lo strumento di pianificazione dei lavori basato sul machine learning sviluppato permette di prendere i dati dalle pianificazioni derivanti da progetti esistenti e le traduce in template per le pianificazioni future.

Capitolo Secondo

ANALISI CRITICA DELLA LETTERATURA, DEFINIZIONE DEGLI OBIETTIVI E ASPETTI METODOLOGICI

In questo capitolo si sintetizzano le criticità messe in luce dalla analisi del contesto di riferimento. Si definiscono, quindi, gli obiettivi della tesi e gli aspetti metodologici per metterli in atto, in modo da superare i limiti attuali.

1 ANALISI CRITICA DELLA LETTERATURA

Nonostante gli sforzi della letteratura scientifica nello sviluppo di sistemi e strumenti di pianificazione assistita basata sull'automatizzazione del processo, molti progetti nell'industria delle costruzioni sono ancora pianificati manualmente, in tutte le fasi con l'eccezione dello sviluppo del modello 3D BIM in quale si sta diffondendo sempre di più (se non altro per l'obbligatorietà sopra certi importi negli appalti pubblici). Il modello 3D BIM è di certo il punto di partenza ma non è sufficiente da solo per considerare la pianificazione come assistita. Infatti, come visto nei primi paragrafi del Capitolo Secondo, in cui si riportano le fasi della realizzazione di un cronoprogramma, anche utilizzando il modello 3D BIM dell'edificio (che assiste nell'individuazione delle quantità), comunque, la pianificazione rimane fortemente manuale.

La pianificazione manuale è principalmente effettuata da pianificatori esperti, i quali basano il tutto sulla loro esperienza lavorativa, questo aspetto è uno dei principali problemi. Infatti, oltre al fatto che è difficile per un nuovo pianificatore entrare nelle logiche dettate dall'esperienza (i panificatori esperti sanno più di quello che riescono a spiegare), proprio quest'ultime sono spesso spiegate utilizzando pianificazioni precedenti e non tramite una conoscenza strutturata e basata su principi.

Molte aziende nell'industria delle costruzioni continuano con il metodo manuale perché come si può comprendere dalla letteratura esaminata e dallo scritto di Amer et al. (2021) ci sono alcuni problemi che limitano l'adozione di sistemi di pianificazione automatizzata:

1. In primo luogo, la ricerca ha rivelato che gli schemi di dati utilizzati per rappresentare e archiviare la conoscenza della costruzione nei sistemi di pianificazione esistenti sono rigidi. Queste rappresentazioni richiedono un elenco esaustivo e la modellazione di tutte le operazioni di costruzione, degli oggetti del modello e delle loro interazioni.
2. In secondo luogo, la ricerca ha dimostrato che la creazione e il mantenimento delle basi di conoscenza avviene interamente manualmente. Data la grande

quantità di modelli di costruzione richiesti, il compito manuale di crearli è estremamente noioso e, in alcuni casi, irrealizzabile.

3. In terzo luogo, non esistono metodi di apprendimento automatico che consentano di creare automaticamente basi di conoscenza della costruzione a partire dai dati esistenti, sebbene tutte le informazioni richieste siano già disponibili tramite programmi di progetto esistenti e piani di lavoro settimanali. Ciò è dovuto anche alla rigidità dei modelli di lavoro, come discusso in precedenza. Sebbene esistano metodi che consentono di acquisire i casi di pianificazione attuali da riutilizzare in futuro, questi metodi non sono completamente automatizzati e richiedono comunque un input manuale da parte degli utenti
4. In quarto luogo, la convalida dei sistemi di pianificazione automatizzati esistenti è stata limitata. Questi sistemi, nella maggior parte dei casi, non sono stati testati su progetti di costruzione reali. Questa semplificazione eccessiva del compito di pianificazione rende i sistemi sviluppati meno attraenti per i professionisti dell'industria. Ad esempio, manca un sistema di pianificazione e programmazione intelligente in grado di tenere conto simultaneamente dei lavori meccanici, elettrici, idraulici, antincendio e di finitura, pianificando e programmando anche le operazioni strutturali di base e del muro a secco interno.
5. In quinto luogo, i motori di pianificazione e programmazione esistenti sono sufficientemente maturi per ragionare sull'ambito del progetto direttamente dai modelli BIM.
6. In sesto luogo, sebbene i metodi esistenti possano pianificare e programmare le attività automaticamente, la ricerca sull'ottimizzazione della costruzione è separata dalla ricerca sull'automazione della pianificazione. Pochissimi motori di pianificazione automatizzati sfruttano le capacità di ottimizzazione. Perché l'ottimizzazione è difficile da eseguire manualmente da parte di un pianificatore umano, offrire funzionalità di ottimizzazione con motori di pianificazione automatizzati offre un valore aggiunto ai professionisti del settore. Il disaccoppiamento di queste due linee di ricerca sta creando un costo opportunità che può essere sfruttato in futuro.

2 DEFINIZIONE DEGLI OBIETTIVI E ASPETTI METODOLOGICI

Lo scopo della tesi è muovere i primi passi verso lo sviluppo di un sistema per la pianificazione automatizzata il quale superi le difficoltà degli attuali sistemi. L'obiettivo generale è ottenere un cronoprogramma che sia reintegrabile all'interno del modello BIM in modo che le attività e le tempistiche per la costruzione di un singolo elemento siano direttamente attributi di quell'elemento nel modello BIM. Per far questo sono tre i passi principali (vedi Figura 21):

- creare un cronoprogramma in modo automatico tramite degli algoritmi prendendo i dati direttamente dal modello BIM;
- visualizzare, controllare e modificare il cronoprogramma creato;
- reintegrare il modello BIM con i dati del cronoprogramma tramite IFC in modo da avere direttamente i valori come attributi degli elementi.

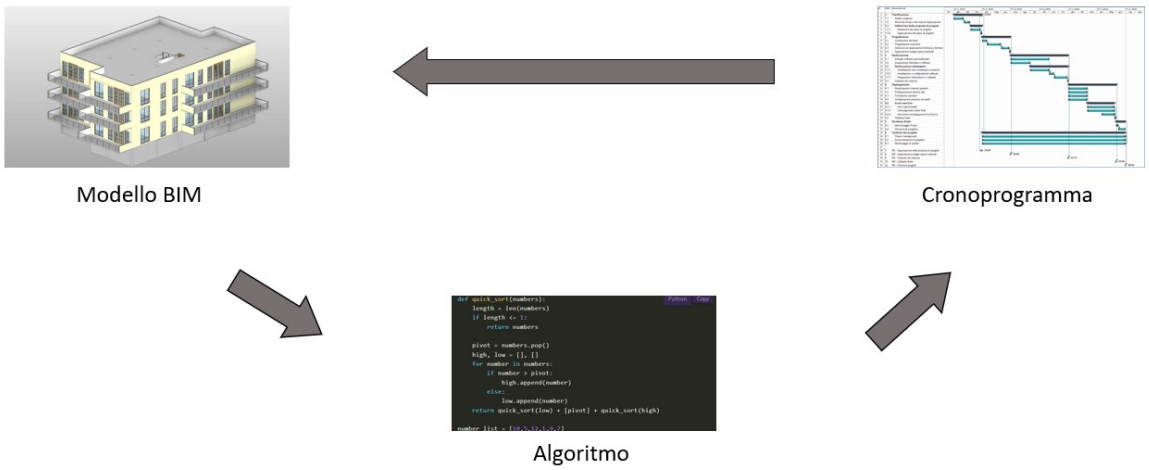


Figura 21 – Schema dell'obiettivo generale (immagine elaborata dall'autore della tesi)

In pratica l'obiettivo finale sarebbe unire il database del progetto ed il database della pianificazione in un unico database tramite il formato IFC. Ovviamente essendo questa la prima tesi che affronta l'argomento, l'obiettivo non è completare tutti i passaggi sopra descritti ma iniziare a muovere i primi passi per lo sviluppo di un sistema di pianificazione automatizzata. L'obiettivo della tesi è quindi creare un algoritmo o una serie di algoritmi che siano in grado di generare una programmazione automatica dei lavori per un semplice caso studio derivando i dati direttamente dal modello BIM dell'edificio tramite il formato IFC.

La metodologia che si sviluppa è fortemente incentrata sul modello BIM e sul formato IFC in quanto si ritiene che per il futuro sia fondamentale utilizzare questi strumenti. Con il diffondersi dell'obbligatorietà, normativa e tecnologica, del BIM è sempre più necessario il formato IFC. Quest'ultimo è un formato dati aperto che consente l'interoperabilità e che ormai è affermato, quasi tutti se non tutti i principali software utilizzati nell'industria delle costruzioni

sono in grado di leggerlo. Per questi motivi una metodologia sviluppata tramite IFC consente di essere disponibile per tutti e più appetibile rispetto ad altri formati.

La metodologia parte dalla creazione e dall'esportazione del modello BIM 3D dell'edifico tramite il formato IFC. Il file viene poi inserito in un ambiente di programmazione, il quale sfrutta Python (tramite la libreria IFCopenshell) per ricavare le quantità e di conseguenza assegnare le tempistiche a ciascuna lavorazione in modo da poter esportare un cronoprogramma dei lavori, tramite il formato XML, visualizzabile in Project Libre. Ovviamente essendo un primo approccio, la tesi si basa su delle ipotesi che poi andranno approfondite nei successivi sviluppi. La sperimentazione, quindi, si basa tutta sullo sviluppo di algoritmi che intendono aprire la prospettiva su una possibilità concreta di poter sviluppare un cronoprogramma in modo automatico a partire da un modello BIM 3D in formato IFC.

Capitolo Terzo

PROPOSTA DI UNA METODOLOGIA E APPLICAZIONE AD UN CASO STUDIO

In questo capitolo viene presentata la metodologia per lo sviluppo della sperimentazione, la descrizione del caso studio e quindi i risultati ottenuti.

1 PROPOSTA DI UNA METODOLOGIA E CRITERI DI VALUTAZIONE DEI RISULTATI

Il questo paragrafo si esamina la metodologia della tesi per arrivare alla creazione dell'algoritmo che permette di ottenere in automatico un cronoprogramma dei lavori edili. Il metodo si divide in sei passaggi (vedi Figura 22): sviluppo del modello 3D BIM con l'utilizzo del software revit; esportazione del modello 3D BIM in formato IFC4; definizione dei metodi di costruzione, delle tempistiche e delle risorse; creazione dell'algoritmo per l'estrazione delle quantità in Python; creazione dell'algoritmo per l'esportazione della pianificazione in formato XML in Python; visualizzazione della pianificazione in Project Libre.

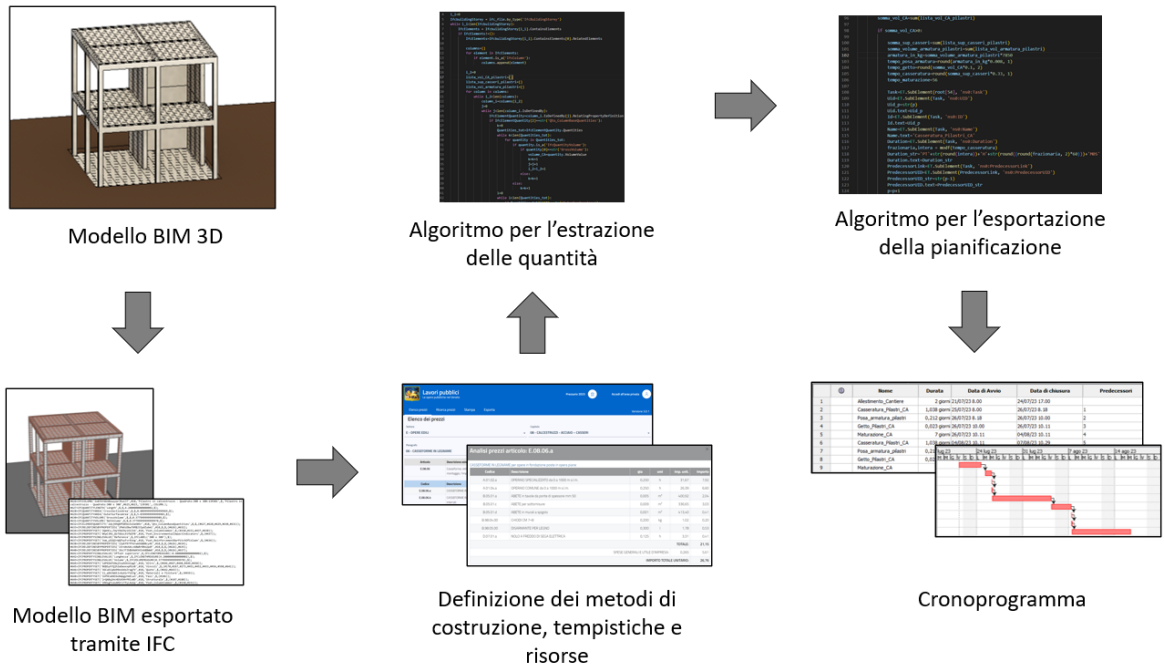


Figura 22 – Schema della metodologia (immagine elaborata dall'autore della tesi)

1.1 Sviluppo del modello 3D BIM con revit

Per lo sviluppo del modello 3D BIM, il quale contiene tutte le informazioni dimensionali dell'edificio, si utilizza il software Autodesk Revit, il quale verrà presentato brevemente di seguito. Inoltre, per la creazione del modello si devono seguire alcune semplici regole di

"corretta modellazione" le quali verranno specificate nel paragrafo dedicato.

1.1.1 Autodesk Revit

Autodesk Revit è un software dalle infinite possibilità che permette di gestire tutte le fasi che caratterizzano un progetto di costruzione di edifici o infrastrutture, ma va anche oltre, rivelandosi un ottimo strumento che permette la collaborazione di più utenti con ruoli diversi.

Revit non nasce come software Autodesk ma viene sviluppato dalla Revit Technologies Inc. nel 2000 come programma dotato di una vasta gamma di strumenti idonei alla gestione della costruzione di qualsiasi tipo di infrastrutture. Le potenzialità di questo software erano tali da spingere Autodesk a comprarlo ad appena 2 anni dalla sua nascita, nel 2002 per la cifra di 133 milioni di dollari. Un investimento più che azzeccato, dato l'impatto rivoluzionario che, da quel momento in poi, Revit ebbe sulla figura del progettista. Alla nota Multinazionale, oltre alla lungimiranza nell'acquisto del software, va dato il merito di aver contribuito significativamente nel suo sviluppo e nella sua ottimizzazione, fino a renderlo uno dei prodotti per la progettazione BIM e CAD più diffusi e richiesti sul mercato.

Revit serve a realizzare progetti e disegni nell'ambito BIM e CAD e grazie alla sua piattaforma BIM multidisciplinare permette di gestire e pianificare tutte le fasi della vita di un edificio. Per permettere la collaborazione tra più utenti con ruoli diversi, Revit si divide in tre sezioni principali:

- Revit Architecture: rivolto ad architetti e progettisti edili, consente l'utilizzo del prodotto per rendere originale e fattibile un'idea, a partire dal progetto concettuale fino alla documentazione della costruzione, con analisi e ottimizzazione delle prestazioni dell'edificio;
- Revit structure: rivolto ad ingegneri strutturali, permette la progettazione strutturale e l'uso di modelli intelligenti per il confronto e l'analisi relativa delle componenti dell'edificio, in modo da valutare la conformità alle normative esistenti dal lato strutturale e della sicurezza.
- Revit MEP: rivolto ingegneri meccanici, elettrici e termotecnici, permette l'analisi e la progettazione degli elementi più specifici dell'edificio, come gli impianti (Pedago, 2021).

Quando si accede al programma c'è la possibilità di selezionare il modello da utilizzare in base alla disciplina che si vuole modellare (strutturale, architettonica, impiantistica). In Figura 23 possiamo vedere i principali modelli disponibili, dove in particolare il modello di costruzione permette di visualizzare tutte le discipline in un unico modello. Quando si modella un edificio tipicamente si inizia con il modello di costruzione, dove si modella la parte architettonica, e poi si aprono separatamente un modello strutturale ed un modello di sistemi nei quali si modella, appunto, la parte strutturale e la parte impiantistica. Il modello strutturale ed impiantistico saranno poi linkati all'interno del modello di costruzione in modo da avere una visione completa dell'edificio. Chiaramente esistono altre possibilità, ad esempio si può anche modellare tutto

all'interno del modello di costruzione oppure gli impianti si possono modellare in modo separato in base alla disciplina (meccanica, elettrica, idraulica). Generalmente in base alla complessità del progetto si decide come agire, per un progetto di piccole dimensioni, avente un solo operatore, esso può decidere di modellare tutto in un unico modello, al contrario per un progetto di grandi dimensioni è necessario dividere in più modelli, in modo da lavorare in più operatori. Per questa tesi si utilizza il primo metodo illustrato in cui si modella la parte architettonica nel modello di costruzione e si linkano i modelli strutturali e impiantistico.

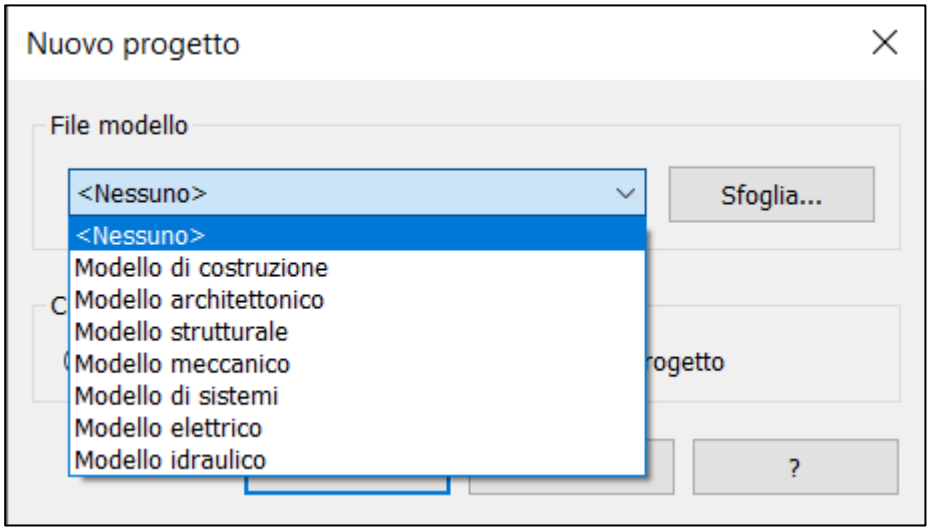


Figura 23 – Finestra di selezione del modello (immagine tratta da Autodesk Revit 2024)

Appena selezionato il modello si aprirà un'interfaccia come in Figura 24:

- in alto si ha la barra di acceso rapido, dove tramite delle icone si selezionano gli oggetti da inserire, in modo molto intuitivo, oppure si modificano gli oggetti presenti nel modello;
- a destra si ha la finestra delle proprietà, dove si visualizzano le proprietà del livello o dell'oggetto selezionato;
- a sinistra si ha la finestra del browser di progetto in cui si possono visualizzare i livelli, gli abachi, le tavole, i file linkati, ecc.;
- in centro si ha l'area di modellazione.

Il setup presente in Figura 24 è elaborato in parte dall'autore della tesi, di conseguenza sono disponibili altre disposizioni. Dopo aver illustrato brevemente la schermata principale non si ritiene necessario approfondire con ulteriori dettagli sul funzionamento del software, quindi per ulteriori informazioni si rimanda ai numerosi tutorial presenti online.

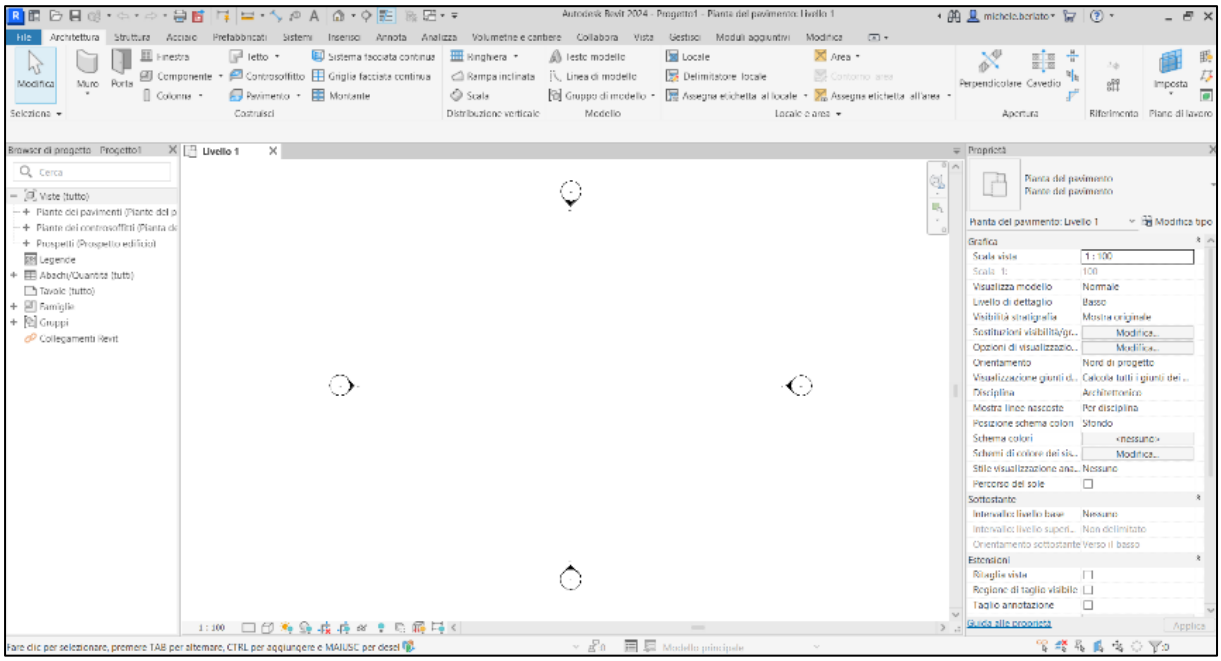


Figura 24 – Schermata principale di Autodesk Revit (immagine tratta da Autodesk Revit 2024)

1.1.2 Regole di corretta modellazione

Dopo aver brevemente illustrato il software si dichiarano alcune regole di corretta modellazione, secondo il parere dell'autore della tesi, da seguire per modellare un'opera senza incorrere in problemi di vario genere ed in modo da rendere utilizzabile l'algoritmo. Le regole sono:

- La parte architettonica e la parte strutturale dell'opera si devono modellare in due modelli separati, modello di costruzione e modello strutturale, i quali successivamente vanno linkati per aiutarsi nella modellazione ma esportati separatamente;
- Si devono impostare dei livelli di lavoro in base alla disciplina nei rispettivi modelli (livello 0-strutturale; livello 0-architettonico, ecc.)
- Si deve evitare di usare un livello per elementi che non poggiano o sono "appesi" effettivamente su quel livello (esempio fondazione e magrone richiedono due livelli differenti in quanto gli IFCSLAB sono appesi e non appoggiati al livello);
- Si devono dividere i muri e i pilastri in base ai livelli e non fare elementi unici per tutta l'altezza dell'edificio;
- Quando si modella la parte architettonica, i muri completi si devono fare solo tra i pilastri, mentre in corrispondenza dei pilastri o dei muri strutturali vanno modellati dei rivestimenti esterni ed interni in modo da non sovrastimare la quantità dei materiali (esempio in Figura 25);

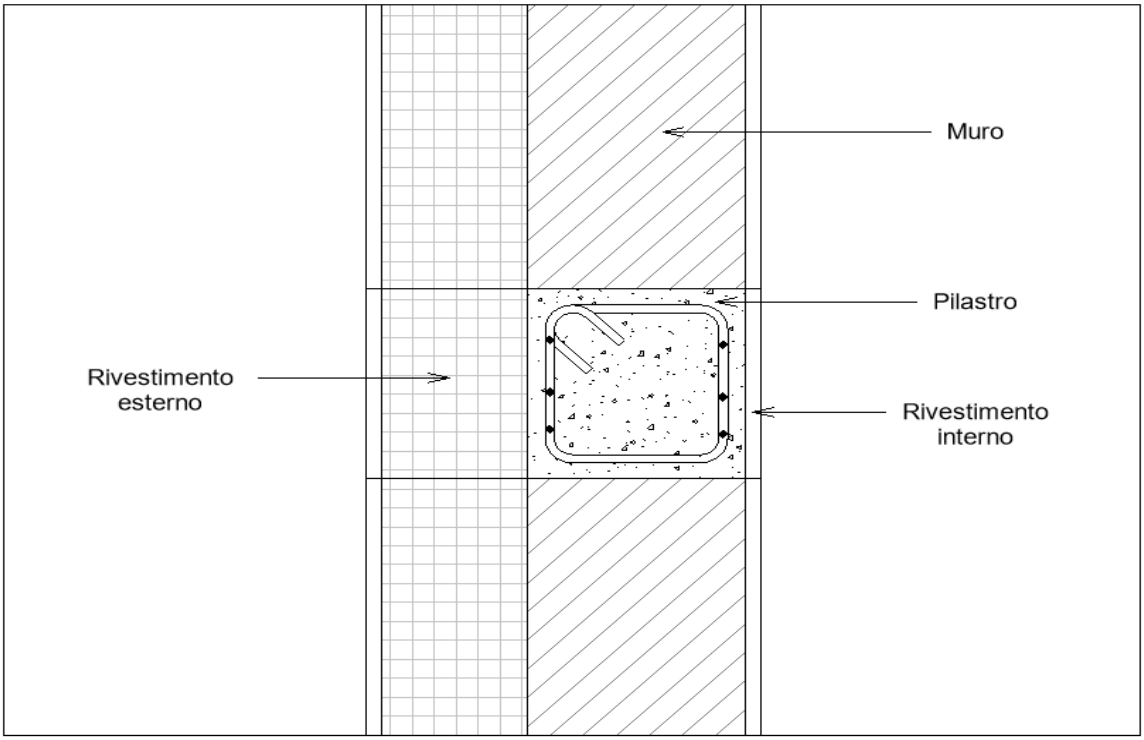


Figura 25 – Esempio modellazione muri intorno al pilastro (immagine elaborata dall'autore della tesi)

- Durante la modellazione si devono seguire dei metodi di costruzione precisi e fedeli alla realtà (in questo caso quelli presentati al paragrafo "Definizione dei metodi di costruzione, tempistiche e risorse");
- Il livello architettonico del piano terra deve essere posizionato alla quota 0m, prendendo come riferimento il terreno, ed eventualmente il pavimento alzato di qualche centimetro in modo da essere più alto del terreno, perché lo scavo viene calcolato dalla quota zero (se si vuole fare un piano terra rialzato allora vanno specificati due livelli diversi, uno per la quota zero e uno per il piano rialzato);
- Le armature negli elementi strutturali vanno modellate il più possibile uguale alla realtà in modo da rispettare le quantità.

1.2 Esportazione del modello 3D BIM in formato IFC4

Revit come la maggior parte dei programmi di modellazione 3D permette di esportare il modello 3D BIM nel formato IFC, il quale si è presentato in precedenza (paragrafo 2.2). Ci sono varie versioni di IFC sviluppate negli anni, per questa tesi si scegli di utilizzare IFC4 di cui si fa un focus prima di spiegare la modalità di esportazione del modello.

1.2.1 IFC4

Nel 2013 è avvenuto il rilascio dell'edizione 4, il cui nome si è scelto di semplificare in IFC4. Tale aggiornamento porta numerose novità. Innanzitutto, l'edizione 4 contiene il data set che consente l'utilizzo dei modelli in attività legate agli ambiti del 4D e 5D BIM, oltre che

l'implementazione per analisi di performance energetica e ambientale. Vengono inoltre aggiunti numerosi elementi allo schema degli elementi IFC e risulta ora possibile mappare il design di un edificio all'interno di un sistema GIS e viceversa. Uno dei maggiori avanzamenti della release IFC4 è però rappresentata dall'implementazione della descrizione geometrica degli elementi, arrivando a supportare la B.rep o Boundary representation (rappresentazione vettoriale di un solido per mezzo primariamente dei suoi vertici e spigoli, con una generazione secondaria delle superfici) e le NURBS – Non Uniform Rational Basis-Splines, ovvero superfici geometriche curve descritte da una rappresentazione matematica e solitamente utilizzate per descrivere geometrie free-form. Il beneficio principale, rispetto all'interpretazione delle superfici solo tramite mesh della precedente versione, è quello di semplificare le forme e contestualmente diminuire la dimensione dei file. Attualmente la versione IFC4 continua a evolvere con continui aggiornamenti: l'inclusione di categorie di oggetti legate all'ambito infrastrutturale quali ponti, strade, ferrovie, porti e aeroporti (precedentemente non supportare) permette di allargare lo schema di scambio informativo su scale di progetto precedentemente impensabili (come le grandi opere civili) e consente così l'interoperabilità di un maggior numero di software settoriali specifici. Tutti questi dati sono in genere codificati su uno dei tre formati disponibili:

- .ifc: formato di file predefinito basato sullo standard ISO-STEP;
- .ifcxml: codifica basata sul linguaggio XML;
- .ifczip: archivio compresso di uno di questi formati, che possono contenere anche materiale aggiuntivo, come PDF o immagini (Adhox, 2022).

1.2.2 Esportazione del modello e visualizzazione del file IFC

Una volta modellata l'opera si passa alla fase di esportazione in formato IFC, selezionando esporta in formato IFC, su Revit, compare una schermata come quella in Figura 26. La schermata permette di selezionare il luogo di salvataggio del file, di entrare nella finestra di "modifica configurazione", per selezionare le varie impostazioni, e di visualizzare le impostazioni principali, riassunte direttamente nella schermata. Selezionando "modifica configurazione" si apre la schermata in Figura 27, dove si può scegliere principalmente: la versione IFC con cui esportare il modello; il livello di dettaglio; le proprietà da esportare. Inoltre, si possono selezionare altre impostazioni di dettaglio, le quali non verranno approfondite.

Il risultato dell'esportazione è un file IFC che si può aprire tramite vari programmi oppure visualizzare anche come file di testo. In Figura 29 si vede un esempio dell'apertura di un modello BIM 3D strutturale esportato in formato IFC e aperto con il visualizzatore BIMvision. Vediamo sulla destra nella finestra "struttura IFC" come gli oggetti siano classificati all'interno del file IFC, cioè si ha un progetto che contiene un sito che contiene un edifico che contiene dei piani che contengono degli oggetti e così via. In particolare, poi se si seleziona un oggetto come il pilastro evidenziato in verde nella finestra in basso a destra si possono vedere tutte le sue proprietà che si dividono in varie sezioni, le quali al loro interno hanno tutti i dati dell'oggetto

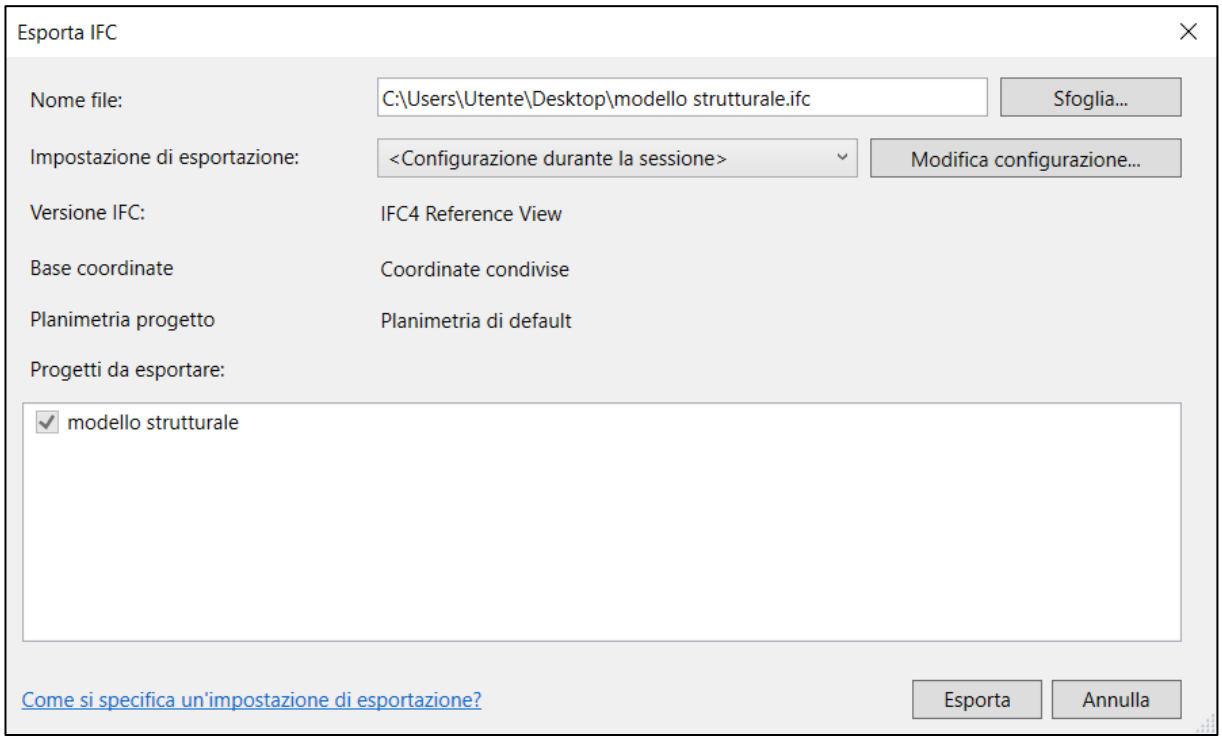


Figura 26 – Schermata esportazione modello in IFC (immagine tratta da Autodesk Revit 2024)

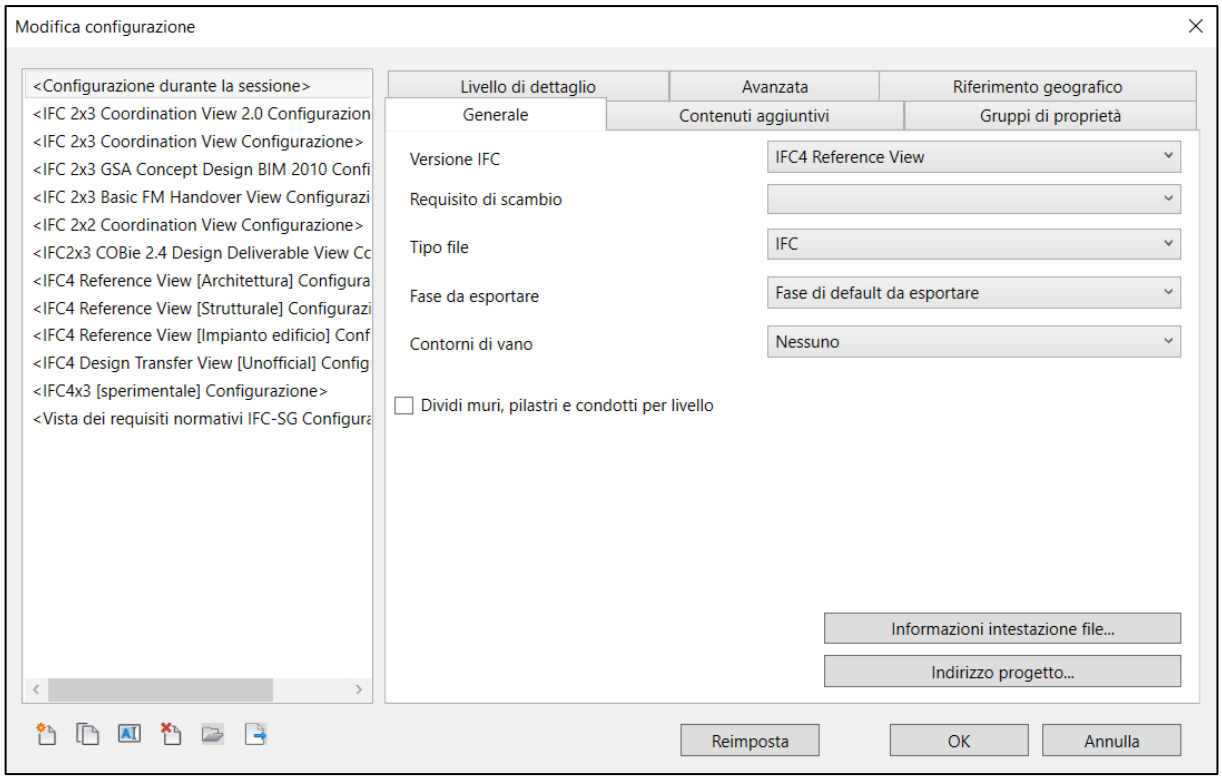


Figura 27 – Schermata “modifica configurazione” per l’esportazione del modello in IFC (immagine tratta da Autodesk Revit 2024)

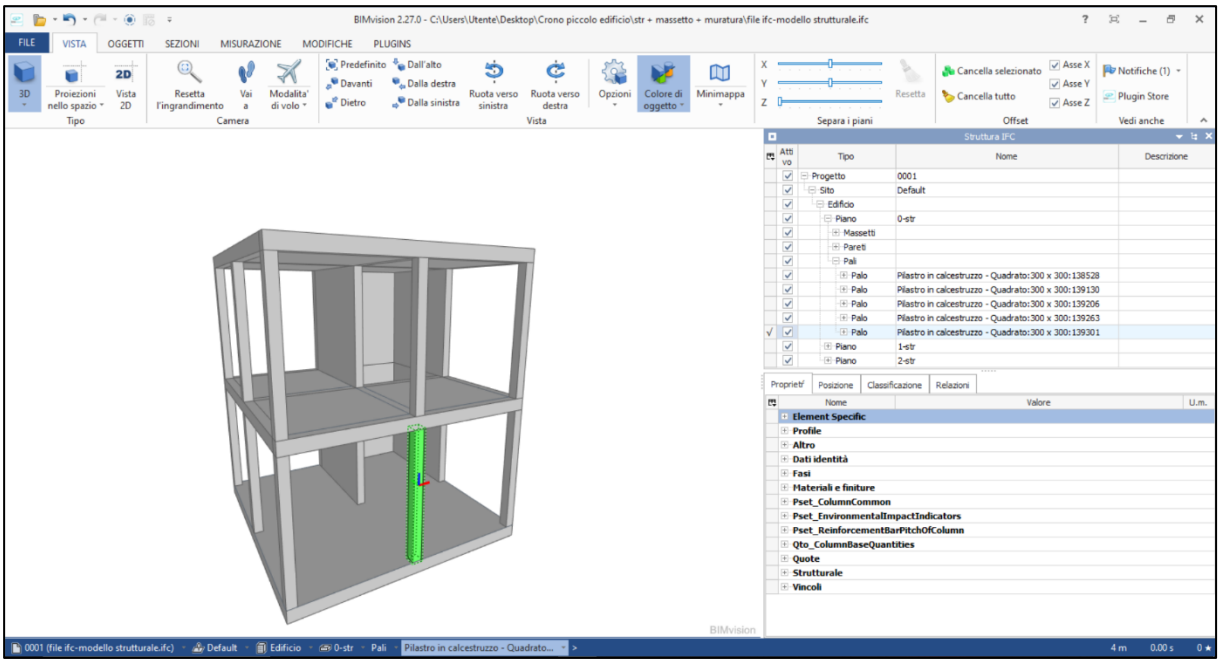


Figura 29 – Apertura di un file IFC con il visulizzatore BIMvision (immagine tratta da BIMvision)

evidenziato, che si possono visualizzare aprendo le tendine (le quali in questo caso sono chiuse per poter visualizzare tutte le sezioni delle proprietà del pilastro). Un altro modo per visualizzare il file IFC è aprirlo in un file di testo come in Figura 28 dove si vede una porzione dell'intero testo, la quale si riferisce sempre al pilastro evidenziato in verde su BIMvision. In questo caso i dati sono espliciti essendo un testo ma sono poi gli stessi che si possono visualizzare su BIMvision. Si vedono ad esempio all'interno delle varie categorie IFC le dimensioni del pilastro richiamate all'interno di "Qto_ColumnBaseQuantities" che sono le stesse che si possono vedere aprendo la stessa tendina in BIMvision.

```
#626=IFCCOLUMN('2x0JVr66n0ouyywrJkuiJ7',#18,'Pilastro in calcestruzzo - Quadrato:300 x 300:139301',$,'Pilastro in
calcestruzzo - Quadrato:300 x 300',#625,#623,'139301',.COLUMN.);
#627=IFCQUANTITYLENGTH('Length',$,$,4.2000000000000002,$);
#628=IFCQUANTITYAREA('CrossSectionArea',$,$,0.089999999999999969,$);
#629=IFCQUANTITYAREA('OuterSurfaceArea',$,$,5.0399999999999983,$);
#630=IFCQUANTITYVOLUME('GrossVolume',$,$,0.37799999999999989,$);
#631=IFCQUANTITYVOLUME('NetVolume',$,$,0.37799999999999978,$);
#632=IFCELEMENTQUANTITY('2dyjHhQDPh8hbs5xSeO85r',#18,'Qto_ColumnBaseQuantities',$,$,(#627,#628,#629,#630,#631));
#633=IFCRELDEFINESBYPROPERTIES('2PWVzDbw7XM$23IpdZubm1',#18,$,$,(#626),#632);
#634=IFCPROPERTYSET('2QeHCu_FmyYOAZ9yiO1lS4',#18,'Pset_ColumnCommon',$,(#150,#151,#437,#438));
#635=IFCPROPERTYSET('0KpC3Rx_$xTGGsLKtfWIPB',#18,'Pset_EnvironmentalImpactIndicators',$,(#437));
#636=IFCPROPERTYSINGLEVALUE('Reference',$,IFCLABEL('300 x 300'),$);
#637=IFCPROPERTYSET('3oW_yD2QIrWQfnyFvrEUng',#18,'Pset_ReinforcementBarPitchOfColumn',$,(#636));
#638=IFCRELDEFINESBYPROPERTIES('1Ipkf97fF6rwmSG$08cy4E',#18,$,$,(#626),#634);
#639=IFCRELDEFINESBYPROPERTIES('15roNvG4Lv6BWRrBRo2pdF',#18,$,$,(#626),#635);
#640=IFCRELDEFINESBYPROPERTIES('2bzJTIb$bXWX5Hlnb8$WWX',#18,$,$,(#626),#637);
#641=IFCPROPERTYSINGLEVALUE('Offset superiore',$,IFCLENGTHMEASURE(-0.40000000000000002),$);
#642=IFCPROPERTYSINGLEVALUE('Lunghezza',$,IFCLENGTHMEASURE(4.2000000000000002),$);
#643=IFCPROPERTYSINGLEVALUE('Volume',$,IFCVOLUMEMEASURE(0.37799999999999978),$);
#644=IFCPROPERTYSET('1dPKDeFK8qjSvphOnnCpgI',#18,'Altro',$,(#446,#447,#448,#449,#450));
#645=IFCPROPERTYSET('0G$SyhTQjh2aOwoxxpMioB',#18,'Vincoli',$,(#170,#267,#273,#451,#452,#453,#456,#598,#641));
#646=IFCPROPERTYSET('3GCuKCqXe99esb4xZrag7V',#18,'Quote',$,(#642,#643));
#647=IFCPROPERTYSET('11_q9AJWAt1vXy61rfiDJg',#18,'Materiali e finiture',$,(#459));
#648=IFCPROPERTYSET('1kPHCabNI6obWpgyhmELw3',#18,'Fasi',$,(#186));
#649=IFCPROPERTYSET('2rQABq2HcHDS45MrPRlx0D',#18,'Strutturale',$,(#187,#188));
#650=IFCPROPERTYSET('1MHsgYLmsAMZrzTfyLAoUp',#18,'Pset_ColumnCommon',$,(#150,#151));
```

Figura 28 – Apertura file IFC con il Blocco note di Windows (immagine tratta dal Blocco note di Windows)

1.3 Definizione dei metodi di costruzione, tempistiche e risorse

Prima di passare alla parte di creazione dell'algoritmo è necessario fare un focus sui metodi di costruzione, sulle tempistiche e sulle risorse da impiegare. Di seguito, come primo passo, si tratteranno i metodi di costruzione in modo separato dalle tempistiche e le risorse per poter spiegare al meglio i concetti. Mentre come secondo passo le tematiche verranno unite per ottenere una visione completa.

1.3.1 Definizione dei metodi di costruzione

Si parte dal presupposto che un cronoprogramma si riferisce sempre ad un determinato metodo di costruzione perché, se ad esempio ho i pilastri gettati in opera sarà necessario posizionare l'armatura, casserare e poi gettare il calcestruzzo mentre se ho i pilastri prefabbricati queste operazioni non saranno necessarie. Di conseguenza per questa tesi si utilizza l'ipotesi di conoscere il metodo di costruzione per poter automatizzare la pianificazione, in sviluppi futuri la questione potrà essere analizzata in modo più efficace.

In pratica quello che si intende è che la pianificazione automatizzata che si ottiene tramite questa tesi va bene per un qualsiasi edificio ma il quale deve essere eseguito secondo alcuni metodi di costruzione, i quali poi potranno anche essere mescolati tra di loro. Esempio se ho un edificio a telaio in calcestruzzo armato poi posso scegliere di fare il solaio a piastra in calcestruzzo armato oppure fare un solaio a travetti e pignatte. Quindi se uso il primo metodo userò determinati script, mentre se uso il secondo lo script del solaio sarà diverso, ma quello per gli elementi del telaio rimarrà uguale. Lo stesso vale ad esempio se decido di posizionare l'isolante esternamente o internamente al muro, il che varierà il metodo di costruzione del singolo elemento muro ma non quello degli elementi del telaio che ospita il muro.

In conclusione, quello che si fa è raggruppare le lavorazioni per ottenere gli elementi dell'edificio a seconda del metodo di costruzione scelto. In relazione a questo ogni gruppo di lavorazioni che genera un certo metodo di costruzione, come può essere ad esempio pilastri in calcestruzzo armato gettati in opera, avrà il suo script parte dell'algoritmo finale. Poi basterà avere lo script adatto al singolo elemento all'interno dell'algoritmo per far sì che esso venga pianificato in modo automatico.

1.3.2 Definizione delle tempistiche e delle risorse

Data la vastità dell'argomento trattato non c'è modo di approfondire le tempistiche necessarie per eseguire le lavorazioni e di conseguenza le risorse in termini di squadra di lavoro. Per questo motivo, per questa tesi, si utilizzano le squadre e le tempistiche secondo il prezziario della regione Veneto del 2023 (*Prezzario Regione Veneto*, 2023). Le quali però come spiegato ai paragrafi 1.4.3 e 1.4.4 andrebbero aggiustate tramite il parere di esperti e la stima per analogia utilizzando cronoprogrammi esistenti.

Innanzitutto, si riporta un esempio di lettura delle informazioni utili sul prezziario della regione Veneto, il quale consente una consultazione dinamica direttamente online. Aprendo il

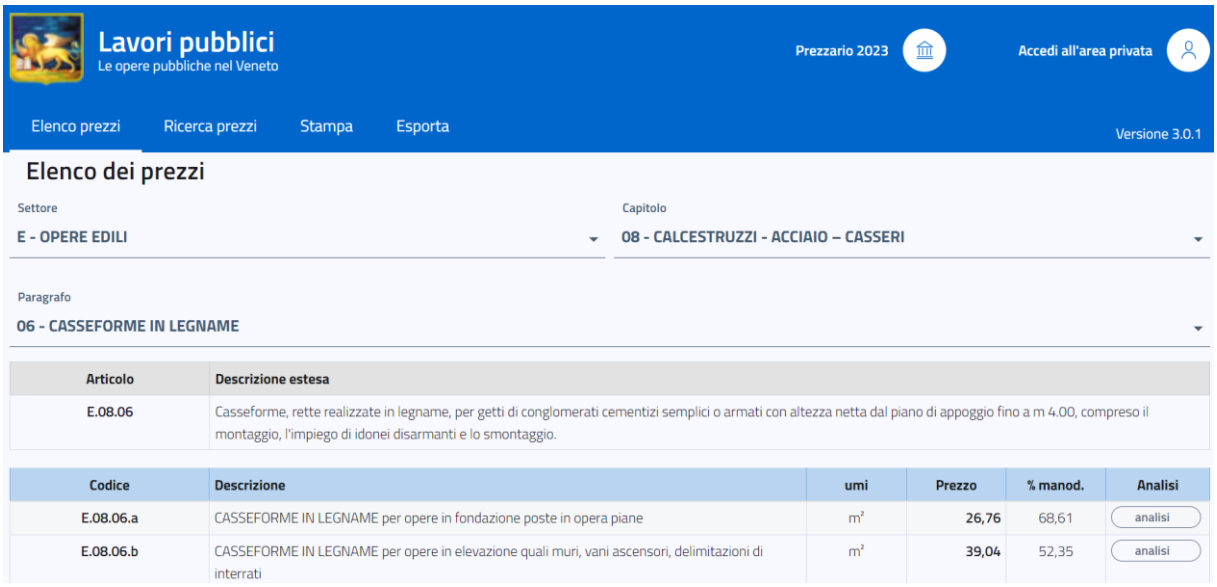


Figura 30 – Schermata per la consulatazione dinamica del prezziario Regionale della regione Veneto (immagine tratta dal Prezzario Regione Veneto, 2023)

prezziario (vedi Figura 30) per prima cosa si chiede di selezionare il settore, in questo caso "E – opere edili", poi si chiede di selezionare il capitolo, in questo caso "08 – Calcestruzzi – Acciaio Casseri", ed infine si chiede di selezionare il paragrafo, in questo caso "06 – Casseforme in legname". Si aprono, quindi, una serie di articoli (in Figura 30 sono visualizzabili i primi due, ma scorrendo nel sito se ne possono visualizzare altri), in questo caso si seleziona il primo: "Casseforme in legname per opere in fondazione poste in opera piane". Quello che ci interessa è l'unità di misura, cioè in questo caso "m^2", e quello che c'è scritto all'interno dell'analisi. Aprendo l'analisi come si può vedere in Figura 31 si hanno tutte le quantità in termini di operai, materiali e attrezzature. Quello che interessa per questa tesi è il numero di operai e il tempo impiegato da essi per la lavorazione dell'unità elementare, la quale non è altro che l'unità di misura. In questo caso si ha che due operai ci mettono 0,25 [h] a installare 1 [m^2] di casseri, con l'ipotesi, valida per questa tesi, di avere il materiale subito a disposizione (per questo primo approccio sarebbe inutile complicare la questione inserendo anche la questione della disponibilità dei materiali e degli stoccaggi).

Per tutte le lavorazioni di questa tesi si usa questo metodo "del prezziario" per avere una prima stima delle tempistiche e delle risorse. Poi i valori sono aggiustati tramite delle stime per analogia utilizzando cronoprogrammi esistenti. Non si ritiene necessario approfondire ulteriormente la questione in quanto per ora è sufficiente avere dei dati, più o meno precisi, da cui partire per realizzare una programmazione automatica dei lavori. In successivi sviluppi si potrebbe approfondire la questione magari ragionando su dei metodi per ricavare delle tempistiche il più possibile aderenti alla realtà.

Inoltre, anche per quanto riguarda la squadra di lavoro si utilizza quella standard del prezziario, però anche qui in futuro si potrebbe approfondire la questione in modo da poter

Analisi prezzi articolo: E.08.06.a

CASSEFORME IN LEGNAME per opere in fondazione poste in opera piane

Codice	Descrizione	qta	umi	Imp. unit.	Importo
A.01.02.a	OPERAIO SPECIALIZZATO da 0 a 1000 m s.l.m.	0,250	h	31,67	7,92
A.01.04.a	OPERAIO COMUNE da 0 a 1000 m s.l.m.	0,250	h	26,39	6,60
B.05.01.a	ABETE in tavole da ponte di spessore mm 50	0,005	m³	400,92	2,04
B.05.01.c	ABETE per sottomisure	0,009	m³	336,65	3,03
B.05.01.d	ABETE in murali a spigolo	0,001	m³	413,40	0,41
B.98.04.00	CHIODI CM 7÷8	0,200	kg	1,02	0,20
B.98.05.00	DISARMANTE PER LEGNO	0,300	l	1,78	0,53
D.07.01.a	NOLO A FREDDO DI SEGA ELETTRICA	0,125	h	3,31	0,41
				TOTALE:	21,15
			SPESE GENERALI E UTILE D'IMPRESA:	0,265	5,61
				IMPORTO TOTALE UNITARIO:	26,76

Figura 31 – Esempio di scheda di Analisi dei prezzi delle casseforme in legname (immagine tratta dal Prezzario Regione Veneto, 2023)

aumentare il numero di lavoratori per ridurre le tempistiche o diminuirlo per una questione di sicurezza e sovrapposizioni.

1.3.3 Metodi, tempistiche e risorse

Con le ipotesi discusse si creano delle tabelle Excel in cui si definiscono le sequenze tipo dei metodi di costruzione, per ottenere gli elementi scelti per questa tesi, unite alla durata ed alla squadra di lavoro necessaria, ricavate come detto nel paragrafo precedente. In questo modo si hanno tutte le info che serviranno per creare l'algoritmo, in modo che da qualsiasi edificio costruito con gli elementi selezionati per questa tesi si possa ottenere la programmazione automatica dei lavori. Per evitare di appesantire troppo lo scorrere della spiegazione della metodologia le tabelle vengono riportate tutte in Appendice 1.

Pilastri in CA gettati in opera					
#	Attività	Risorse	Durata in ore (h)	Quantità unitaria	U.d.M
1	Posa armatura pilastri	3	0,02	1	Kg
2	Casseratura pilastri	2	0,33	1	m^2
3	Getto Cls pilastri	2	0,8	1	m^3
4	Maturazione Cls	\	32	\	\

Figura 32 – Screenshot tabella metodo di costruzione dei pilastri in CA gettati in opera (immagine ricavata dall'appendice 1)

Si riporta in Figura 32, come esempio, la tabella dei pilastri in CA gettati in opera, la quale fa parte delle tabelle di cui si parlava sopra. Si utilizza questa tabella come esempio per tutte le altre, le quali seguono gli stessi principi spiegati di seguito per questa. Il titolo della tabella indica l'elemento o gli elementi che si vogliono costruire, mentre le attività, le quali sono

messe in ordine temporale, indicano le lavorazioni per costruire l'elemento o gli elementi secondo il metodo costruttivo scelto. Poi in base all'attività si ha il numero di operai coinvolti (risorse) e la durata in ore per la quantità unitaria (come già spiegato in precedenza). Si nota in questo caso un'ultima attività denominata "maturazione cls", con questa si intende il tempo minimo che deve intercorrere tra il getto e l'inizio della fase successiva, come ad esempio in questo caso le travi ed il solaio sovrastante. Come indicazione generale si utilizzano i giorni indicati in Figura 33, i quali però si basano sull'ipotesi che la temperatura dell'aria in cui avviene la maturazione sia intorno ai 20°C, per questo motivo i tempi vengono aumentati di un giorno per pilastri e le altre strutture verticali e di due giorni per i solai. Di conseguenza si considera 4 giorni lavorativi (32 ore) per i pilastri e i setti e 12 giorni lavorativi (96 ore) per solai (e di conseguenza anche per le travi in quanto si gettano in contemporanea seguendo le tabelle con i metodi di costruzione). Per i solai dopo 12 giorni non si tolgono completamente tutti i puntelli ma un po' si lasciano per maggior sicurezza. Inoltre, dato che è stato aumentato il tempo l'operazione di disarmo dei casseri si considera compresa nella maturazione.

Tipi di elementi strutturali	Calcestruzzo di cemento con resistenza iniziale ordinaria (giorni)	Calcestruzzo di cemento a elevata resistenza iniziale (giorni)
Sponde dei casseri di travi e pilastri	3	2
Casserature di solette di luce modesta	10	4
Puntelli e centine di travi, archi e volte...	24	12
Elementi strutturali a sbalzo	28	14

Figura 33 – Tempo di attesa minimo per il disarmo dei casseri in base all'elemento strutturale e al tipo di calcestruzzo con una temperatura dell'ambiente intorno ai 20°C (immagine tratta da Mecos, 2020)

1.4 Creazione dell'algoritmo per l'estrazione delle quantità in Python

Si vuole creare un algoritmo che sia in grado di estrarre le quantità degli elementi dell'edificio, direttamente dal file IFC del modello BIM, in modo da poterle moltiplicare per le durate per quantità unitaria definite nel paragrafo precedente così da ottenere le tempistiche di esecuzione di ogni lavorazione. Prima di tutto però va fatto un focus sul linguaggio di programmazione utilizzato e sul software per l'utilizzo del linguaggio, poi si spiega come estrarre le quantità, che verranno utilizzate per ricavare le tempistiche, come spiegato.

1.4.1 Visual Studio Code e Python

Visual Studio Code (o più semplicemente VS Code) è un editor di codice sorgente sviluppato da Microsoft per Windows, Linux e MacOS. Include il supporto per debugging, un controllo per Git integrato, syntax highlighting, IntelliSense, snippet e refactoring del codice. Sono personalizzabili il tema dell'editor, le scorciatoie da tastiera e le preferenze, e permette di installare estensioni che aggiungono ulteriori funzionalità. È un software libero e gratuito per uso personale e commerciale, anche se la versione ufficiale è sotto una licenza proprietaria. Visual Studio Code è basato su Electron, un framework con cui è possibile sviluppare applicazioni Node.js, e può essere utilizzato con i principali linguaggi di programmazione, tra

cui python installando il pacchetto dedicato. Il software incorpora un insieme di funzioni variano a seconda del linguaggio che si sta usando (Wikipedia, 2023).

Python è un linguaggio di programmazione dinamico orientato agli oggetti utilizzabile in numerose applicazioni, come ad esempio sviluppo di siti o applicazioni Web e desktop, realizzazione di interfacce grafiche, amministrazione di sistema, calcolo scientifico e numerico, database, giochi, grafica 3D, eccetera. Il suo vantaggio principale è che è un linguaggio semplice che può essere imparato in poco tempo, questo ne ha aiutato la diffusione in questi anni, dato che ormai in tutti i campi è necessario utilizzare linguaggi di programmazione. I principali punti di forza di python sono:

- Facilità di scrittura e lettura del codice: Python è un linguaggio di programmazione ad alto livello ma, come detto, è molto facile da imparare. Chiunque può impararlo in pochi giorni, nelle sue funzioni base, mentre imparare a padroneggiare Python e tutti i suoi concetti avanzati, pacchetti e moduli ovviamente potrebbe richiedere più tempo. Tuttavia, imparare la sintassi di base di Python è molto facile, rispetto ad altri linguaggi popolari come C, C++ e Java. Il codice Python sembra composto da semplici parole in inglese, e per questo è molto intuitivo.
- Gratuito ed Open-Source: Python è sviluppato sotto una licenza open-source approvata dalla OSI. Quindi, è completamente gratuito da usare, anche per scopi commerciali. Non costa nulla scaricare Python dal sito ufficiale.
- Libreria Standard Robusta: Python ha a disposizione una vasta libreria standard per chiunque la voglia utilizzare. Ciò significa che i programmatori non devono scrivere il proprio codice per ogni singola cosa, a differenza di altri linguaggi di programmazione. Ci sono librerie per la manipolazione delle immagini, i database, il testing delle unità, le espressioni e molte altre funzionalità. Oltre alla libreria standard, c'è anche una crescente raccolta di migliaia di componenti, tutti disponibili nell'indice dei pacchetti Python.
- Portatile: Python è portatile nel senso che lo stesso codice può essere utilizzato su diverse macchine. Supponiamo di scrivere un codice Python su un Mac. Se si vuole eseguirlo successivamente su Windows o Linux, non è necessario apportare alcuna modifica o relativamente poche. Non c'è bisogno di scrivere un programma più volte per diverse piattaforme(*Python*, n.d.).

1.4.2 Estrazione delle quantità

Come detto in precedenza IFC contiene tutte le informazioni sul modello creato, dove le informazioni non sono slegate ma sono tutte connesse tra loro. Infatti, si parte da un contenitore grande che è il progetto che contiene tutto e si può andare sempre più in profondità tramite varie relazioni tra le entità finché non si arriva alla singola caratteristica, ad esempio la lunghezza, il materiale, ecc. IFC classifica tutte le entità presenti in un possibile

progetto di costruzione, sarebbe inutile e dispendioso spiegare tutti i concetti alla base delle varie classificazioni o di come le entità si relazionano quindi ci si limiterà a spiegare i concetti utili per questa tesi.

La prima cosa da fare, dopo aver aperto VS Code, è aprire un nuovo file di lavoro selezionando il linguaggio Python. Dopo di che si deve importare la libreria "IFCopenshell", la quale deve essere scaricata dal relativo sito (*Downloads - IfcOpenShell* , n.d.). Con quest'ultima è possibile aprire il file IFC (vedi Figura 34), a patto che il file IFCopenshell e il file python siano all'interno della stessa cartella altrimenti non funziona "l'import". IFCopenshell aiuta a sviluppare piattaforme digitali per l'industria delle costruzioni, infatti permette di leggere, scrivere e modificare i modelli BIM salvati in formato IFC.

```
1   import ifcopenshell
2   ifc_file=ifcopenshell.open('C:/Users/Utente/Desktop/Script/File IFC.ifc')
```

Figura 34 – Script per importare la libreria ifcopenshell e aprire il file ifc (immagine tratta dall'algoritmo creato dall'autore della tesi)

A questo punto è possibile lavorare sul file IFC, con l'obiettivo di estrarre le quantità utili per il calcolo dei tempi. Si riporta un esempio, in Figura 35 e Figura 36, nel quale si vogliono estrarre le quantità utili per poi poter definire la sequenza delle lavorazioni di un pilastro in calcestruzzo armato gettato in opera. Dunque, seguendo ciò che è stato detto al paragrafo 1.3 ed in particolare l'esempio riportato in Figura 32, è necessario estrarre: il volume dell'armatura (che poi verrà trasformato in Kg), la superficie laterale del pilastro (per la superficie dei casseri), il volume del pilastro (per la quantità di calcestruzzo da gettare).

Analizzando l'algoritmo proposto in Figura 35 si vede che la prima cosa che si fa è estrarre l'elenco dei piani ("IfcBuldingStorey") dal file IFC, il quale servirà per procedere per livelli dal basso verso l'alto per estrarre le quantità. Infatti, l'algoritmo finale sarà in grado di estrarre le quantità necessarie partendo dal piano più basso in modo da avere già una sequenza generale delle lavorazioni, la quale poi sarà aggiustata in base all'elemento in questione. L'elenco dei piani si ottiene grazie alla funzione "by_type" la quale permette di richiamare tutte le entità, contenute all'interno del file IFC, le quali hanno lo stesso nome che viene inserito all'interno della parentesi dopo la funzione. Di conseguenza la variabile prima del simbolo dell'uguale, che in modo molto intuitivo è stata chiamata esattamente come gli elementi che contiene cioè IfcBuldingStorey, conterrà l'elenco dei piani.

Si utilizza poi un ciclo "while" il quale finché la condizione scritta in fianco è verificata continua a ripetersi. La condizione dice che finché la variabile "i_i", che inizialmente è zero, è minore della lunghezza della lista si può continuare il ciclo. Poi all'interno di ogni ciclo si farà aumentare la variabile "i_i" con un "+1" per ciclo. Questo serve per prendere un piano alla volta dal primo nella lista fino all'ultimo, cioè dal più basso al più alto secondo IFC. Infatti "IfcBuldingStorey[i_i]" serve per indicare il singolo piano all'interno della lista.

```
i_i=0
IfcbuildingStorey = ifc_file.by_type('IfcBuildingStorey')
while i_i<len(IfcbuildingStorey):
    IfcElements = IfcbuildingStorey[i_i].ContainsElements
    if IfcElements!=():
        IfcElements=IfcbuildingStorey[i_i].ContainsElements[0].RelatedElements

        columns=[]
        for element in IfcElements:
            if element.is_a('IfcColumn'):
                columns.append(element)

        i_2=0
        lista_vol_CA_pilastri=[]
        lista_sup_casseri_pilastri=[]
        lista_vol_armatura_pilastri=[]
        for column in columns:
            while i_2<len(columns):
                column_i=columns[i_2]
                j=0
                while j<len(column_i.IsDefinedBy):
                    IfcElementQuantity=column_i.IsDefinedBy[j].RelatingPropertyDefinition
                    if IfcElementQuantity[2]==str('Qto_ColumnBaseQuantities'):
                        k=0
                        Quantities_tot=IfcElementQuantity.Quantities
                        while k<len(Quantities_tot):
                            for quantity in Quantities_tot:
                                if quantity.is_a('IfcQuantityVolume'):
                                    if quantity[0]==str('GrossVolume'):
                                        volume_CA=quantity.VolumeValue
                                        k=k+1
                                        j=j+1
                                        i_2=i_2+1
                                    else:
                                        k=k+1
                                else:
                                    k=k+1
                        l=0
                        while l<len(Quantities_tot):
                            if Quantities_tot[l][0]==str('OuterSurfaceArea'):
                                somma_aree_laterali=Quantities_tot[l][3]
                                l=l+1
                            else:
                                l=l+1
                    else:
                        j=j+1

```

Figura 35 – Prima parte dello script per ricavare le quantità utili per il calcolo dei tempi di un pilastro in calcestruzzo armato (immagine tratta dall'algorimto creato dall'autore della tesi)

Il passaggio successivo serve per ricavare tutti gli elementi che sono associati al singolo piano, cioè tutti gli elementi contenuti nel piano. Secondo la logica IFC il termine corretto per indicare gli elementi sarebbe "IfcProduct", nell'algoritmo però la variabile contenete gli elementi è stata chiamata "IfcElements", il nome è comunque molto intuitivo. Per ottenere gli

```
                j_a=0
                while j_a<len(column_i.IsDefinedBy):
                    IfcPropertySet=column_i.IsDefinedBy[j_a].RelatingPropertyDefinition
                    if IfcPropertySet.Name==str('Strutturale'):
                        k=0
                        Properties=IfcPropertySet.HasProperties
                        while k<len(Properties):
                            for property in Properties:
                                if property.Name==str('Volume armatura stimato'):
                                    Volume_armatura=property.NominalValue[0]
                                    j_a=j_a+1
                                    k=k+1
                                else:
                                    k=k+1
                    else:
                        j_a=j_a+1

                lista_vol_armatura_pilastri.append(Volume_armatura)

                lista_vol_CA_pilastri.append(volume_CA)
                superficie_casseri=somma_aree_laterali
                lista_sup_casseri_pilastri.append(superficie_casseri)
            i_2=i_2+1

        i_i=i_i+1
    else:
        i_i=i_i+1
```

Figura 36 – Seconda parte dello script per ricavare le quantità utili per il calcolo dei tempi di un pilastro in calcestruzzo armato (immagine tratta dall'algorimto creato dall'autore della tesi)

IfcProduct (che sono gli IfcElements in Figura 35) si segue lo schema in Figura 37 (buildingSMART, 2017), in particolare la parte di schema tra "IfcBuldingStorey" e "IfcProduct". Si vede infatti che per passare da "IfcBuldingStorey" a "IfcProduct" è necessario utilizzare prima "ContainsElements" e poi "RelatedElements" nella modalità mostrata in Figura 35. Si ottiene così l'elenco degli elementi presenti nel piano, denominati con la variabile "IfcElements", escludendo i piani senza elementi grazie alla funzione "if" la cui condizione permette l'avanzamento dello script solo se il piano contiene elementi.

A questo punto dalla lista degli elementi si vanno ad estrarre gli oggetti differenziandoli per categorie. Per questo esempio si considerano solo i pilastri ("IfcColumn") in quanto il procedimento per gli altri elementi è il medesimo concettualmente, varia solo qualche passaggio in base all'oggetto.

Il primo passaggio è creare una lista dei pilastri tramite la funzione "for" e la funzione "if", cioè per ogni elemento contenuto nella lista degli elementi ("for element in IfcElements") se quell'elemento è un pilastro ("if element.is_a('IfcColumn')") allora inseriscilo nella lista dei pilastri ("columns.append(element)"). Come si vede da questi passaggi l'algoritmo è molto intuitivo e facile da comprendere anche solo leggendo, questa è una caratteristica dell'intero algoritmo creato. La variabile "column=[]", posta all'inizio dei passaggi sopra spiegati, serve per

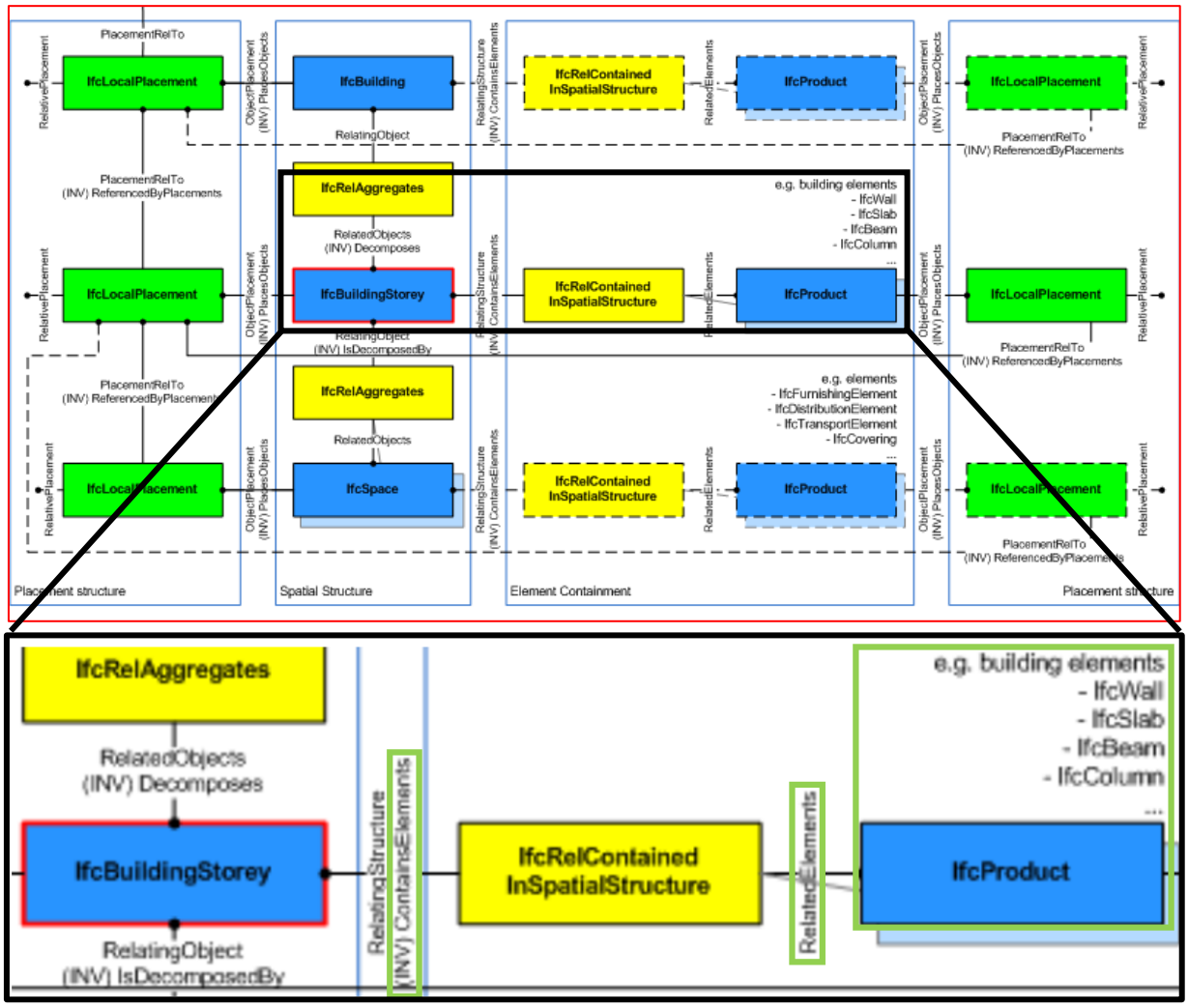


Figura 37 – Schema "Building storey composition" con l'ingradimento della parte tra "IfcBuldingStorey" e "IfcProduct" (immagine tratta da buildingSMART, 2017)

creare una lista vuota iniziale dove verranno inseriti i pilastri (lo stesso vale per le altre liste vuote create nello script).

A questo punto con dei ragionamenti molto simili a quelli fatti finora si prendere un pilastro alla volta dalla lista dei pilastri, la traduzione letteraria del codice sarebbe: per ogni pilastro all'interno della lista dei pilastri finché la variabile "i_2" è minore della lunghezza della lista, cioè finché non sono finiti i pilastri, prendi il pilastro "i_2". Così si è creato un algoritmo che prende i pilastri piano per piano una alla volta, dato che si fa aumentare la variabile con un "+1" ogni ciclo, e con i passaggi successivi vedremo come estrarre le quantità volute ragionando sul singolo pilastro.

Avendo quindi il singolo pilastro ("column_i") si hanno di seguito due cicli "while" dove quello in Figura 35 serve a ricavare il volume del pilastro e la superfice laterale totale, mentre quello in Figura 36 serve a ricavare la quantità di armatura. Partendo dal primo ciclo (Figura 35) si fa riferimento allo schema in Figura 38 dove si sono rappresentati i passaggi da fare per

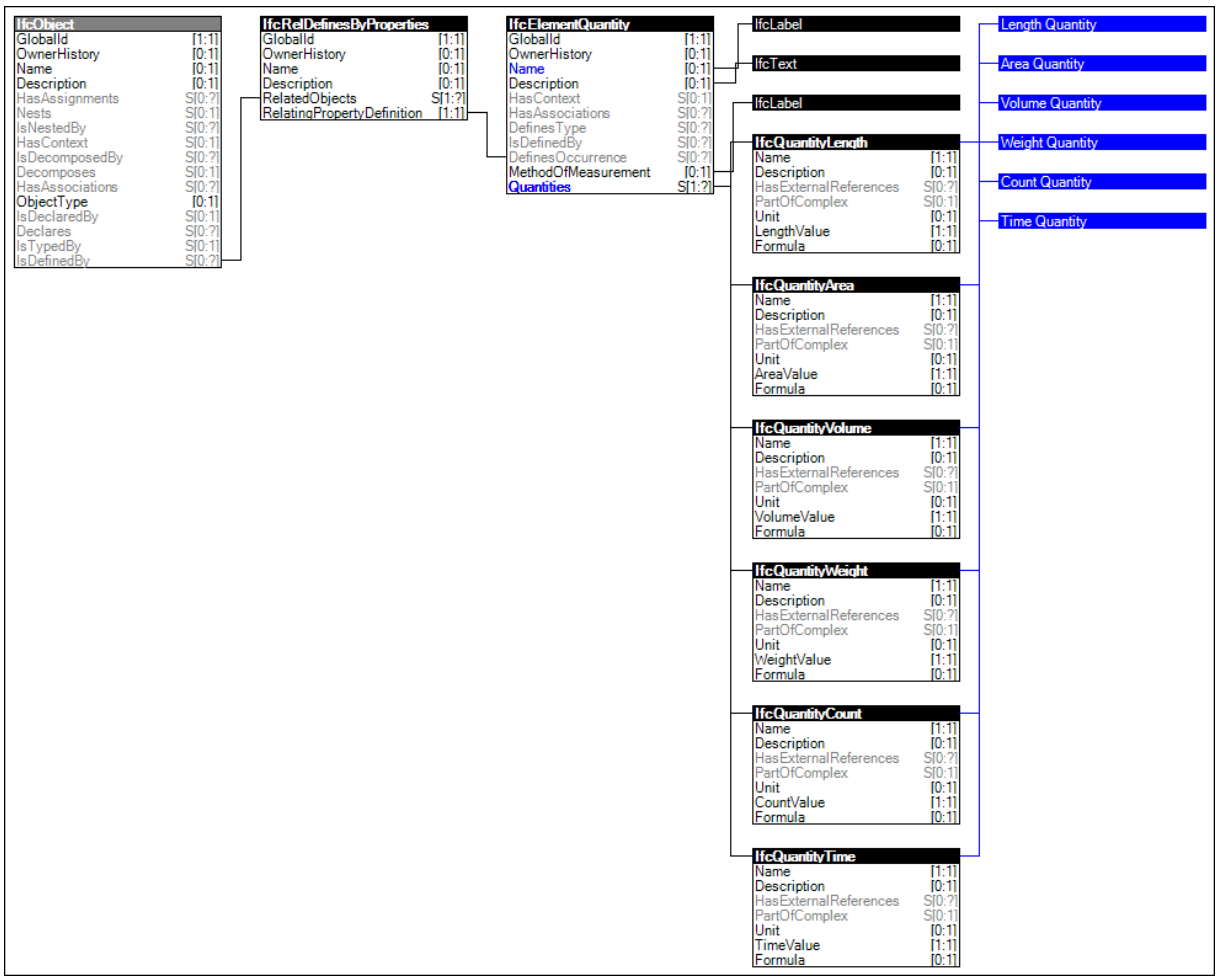


Figura 38 – Schema "Quantity Sets" (immagine tratta da buildingSMART, 2017)

passare da un oggetto generico, in questo caso il pilastro ("IfcColumn"), alle sue dimensioni, ad esempio lunghezza, area e volume. Di conseguenza utilizzando "while", "for" e "if" e le indicazioni in Figura 38 si arriva ad avere il volume di calcestruzzo e la superficie laterale del pilastro. Non si ritiene necessario spiegare tutti i singoli passaggi in quanto essi seguono gli stessi ragionamenti visti per le prime parti dell'algoritmo, si specifica solo che si va a prendere il contenitore "Qto_ColumnBaseQuantities", il quale contiene tutte le quantità di base del singolo pilastro, ed al suo interno si prendono il volume ("IfcQuantityVolume") e l'area delle superfici laterali ("OuterSurfaceArea").

Una volta ottenuto il volume e la superficie laterale bisogna ricavare il volume dell'armatura, per far questo si deve seguire lo schema in Figura 39 in quanto per ricavare la quantità di armatura si deve seguire un percorso diverso rispetto alle quantità ricavate in precedenza, le quali erano quantità base del pilastro. Infatti, come si vede in Figura 36 (che è la continuazione dell'algoritmo in Figura 35), dove sono presentati i passaggi per ricavare il volume di armatura, si passa per le proprietà strutturali all'interno delle quali è presente il volume di armatura. Il procedimento è il medesimo delle altre quantità ma seguendo un diverso schema. Infine, si fa notare in Figura 36, essendo la continuazione dell'algoritmo in Figura 35,

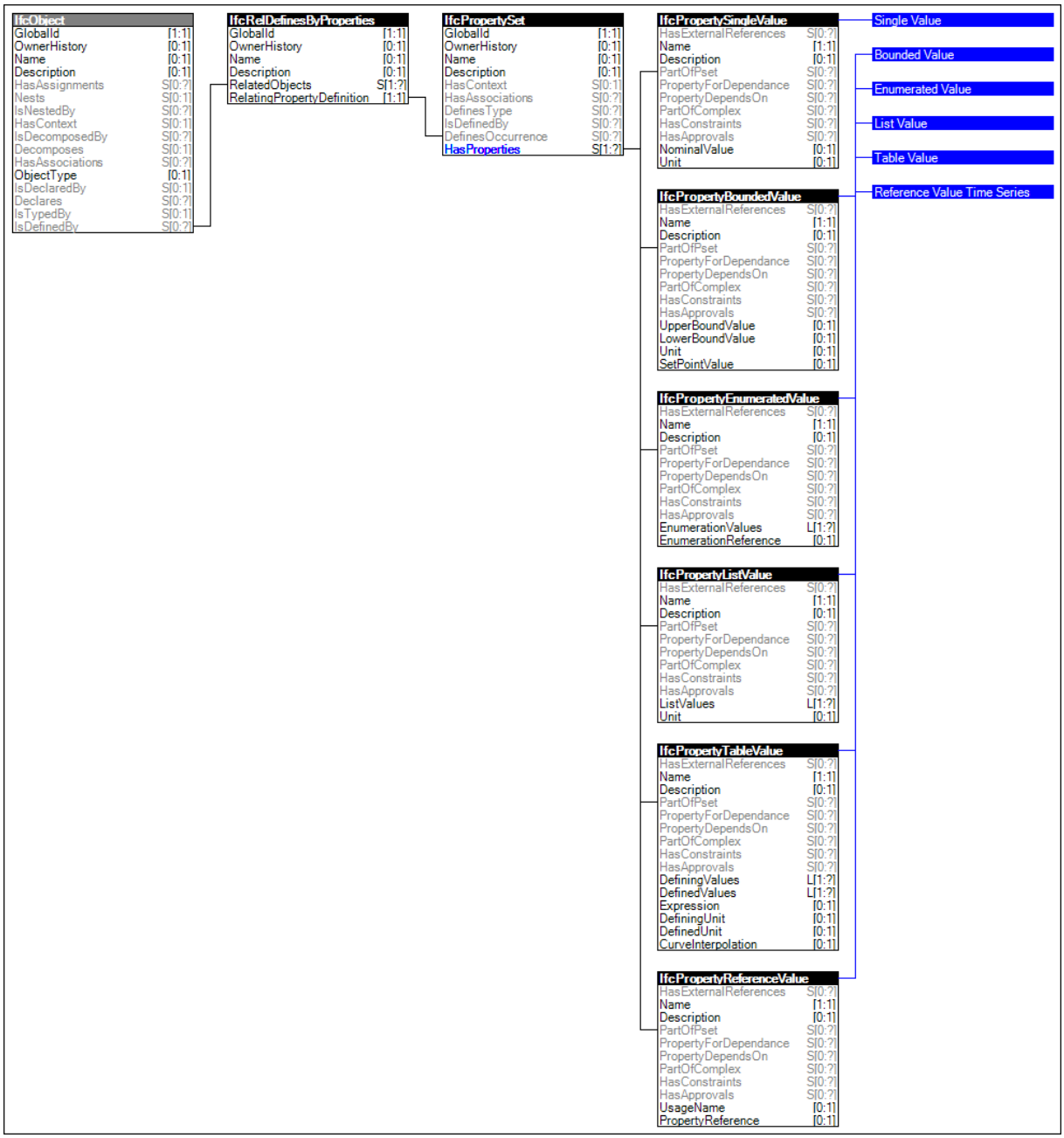


Figura 39 – Schema "Property Sets for Objects" (immagine tratta da buildingSMART, 2017)

si parte allo stesso livello della porzione di algoritmo che definisce il volume e le superfici laterali, quindi si sta lavorando sempre sullo stesso pilastro, in parallelo.

In Figura 36 si hanno anche i passaggi finali dell'intero script, tra cui si evidenziano le righe per inserire il volume di calcestruzzo armato, la superficie laterale dei pilastri e il volume di armatura nelle rispettive liste. Facendo poi la somma dei vari elementi per ogni lista si ottiene la quantità totale sul determinato piano che poi moltiplicata per le durate per unità daranno delle tempistiche per ciascuna lavorazione. Per comodità il calcolo delle tempistiche sarà all'interno dell'algoritmo presentato nel prossimo paragrafo.

1.5 Creazione dell'algoritmo in Python per l'esportazione della pianificazione in formato XML

Dopo aver spiegato la porzione di algoritmo per l'estrazione delle quantità si passa a spiegare come tramite Python è possibile esportare il cronoprogramma finale. Si vuole specificare che la spiegazione di questo algoritmo e del precedente sono fatti in capitolo separati per una maggiore chiarezza espositiva ma in realtà l'algoritmo finale è l'unione dei due paragrafi. Prima di passare alla parte di spiegazione dell'algoritmo è necessario fare un ragionamento sui predecessori.

1.5.1 Ragionamento sui predecessori

Prima dei successivi passaggi è necessario fare un focus su qual è l'ordine delle lavorazioni/attività, cioè i predecessori. Innanzitutto, bisogna distinguere le categorie strutturale, architettonica e impiantistica perché i ragionamenti differiscono in base all'ambito. Nello schema in Figura 40 si può vedere rappresentato il ragionamento sui predecessori che verrà fatto in questo paragrafo.

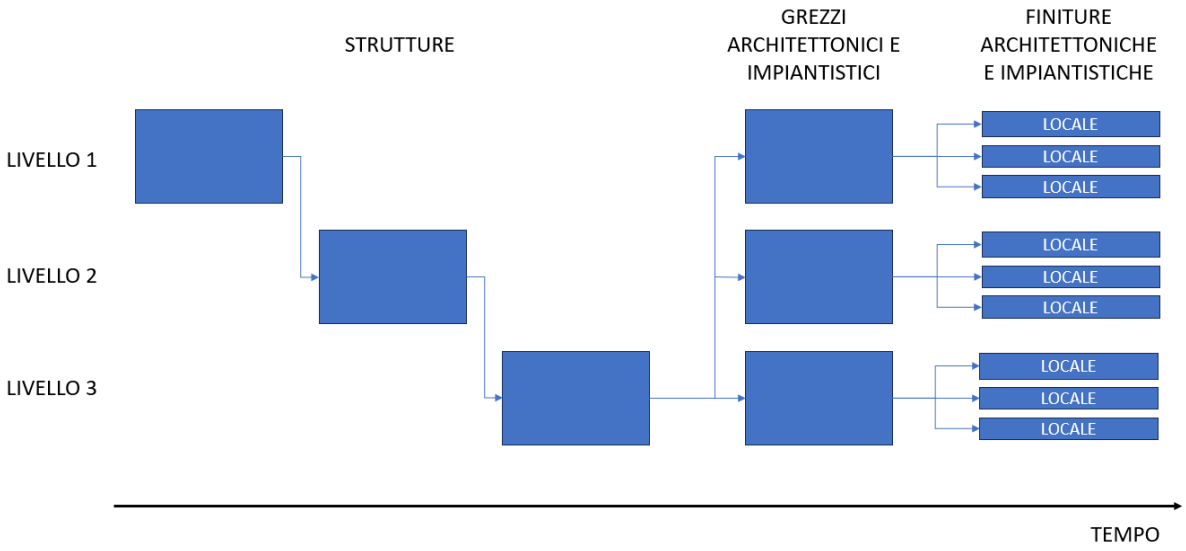


Figura 40 – Schema sulla differenza dei predecessori in base alle categorie e alla distinzione tra grezzi e finiture, basato sul tempo e sui livelli (immagine elaborata dall'autore della tesi)

Nella costruzione della struttura non si potrà lavorare contemporaneamente su due livelli per un limite fisico, cioè se non ho costruito i pilastri del piano terra e il solaio sopra di essi non posso pensare di iniziare quelli del primo piano. È per questo motivo che la struttura segue una logica dei predecessori che si dice "per edificio" nel senso che solamente le strutture di due edifici diversi possono essere fatte contemporaneamente. Questo significa che, per quanto riguarda i predecessori della parte strutturale, si parte dalla struttura più in basso, tipicamente il magrone e la fondazione, e poi si va verso l'alto con pilastri, muri o setti, travi, solaio e così via per il livello successivo. Vediamo quindi in Figura 40 come i livelli della parte strutturale siano temporalmente in fasi diverse. Si specifica che i pilastri e i muri in calcestruzzo

armato (setti) possono essere costruiti in contemporanea, essendo sullo stesso livello e avendo lo stesso compito di sostegno del piano superiore. Infatti, questo non rientra nella logica dei predecessori che si sta spiegando, ma si stanno analizzando le singole categorie all'interno dell'ambito, in questo esempio "piano strutturale", le quali possono seguire ragionamenti specifici a seconda del caso.

Per quanto riguarda le categorie architettoniche si segue una logica dei predecessori "per livello", cioè le parti architettoniche di due livelli separati si possono fare in contemporanea. Esempio per gettare il massetto al piano primo non è necessario che siano state posizionate le predisposizioni impiantistiche e i tamponamenti anche al piano terra, questo significa che, se si hanno le adeguate risorse, i lavori possono essere fatti in contemporanea tra i piani. Questo discorso fatto per le categorie architettoniche vale anche per gli impianti, anche se in realtà per essere più precisi si dovrebbe fare un ulteriore distinzione tra impianti/architettura al grezzo e impianti/architettura di finitura. Infatti, se si parla della fase "al grezzo", cioè quella in cui si fanno i primi lavori impiantistici/architettonici subito dopo la fine della fase strutturale, si può appunto parlare di logica "per livello". Però se si passa alla fase "di finitura", in cui si sono già delineate le divisioni interne e quindi le stanze si può parlare di logica dei predecessori "per locale". Infatti, se ho due stanze diverse posso fare contemporaneamente i lavori di finitura all'interno delle due stanze, sia finiture impiantistiche che finiture architettoniche. Quindi si possono appunto distinguere due momenti che sono grezzo e finitura, i quali si differenziano facendo riferimento al momento in cui si può iniziare a lavorare per locale, cioè dopo aver fatto i tamponamenti, le predisposizioni impiantistiche, il massetto e le partizioni interne. Come si può vedere in Figura 40 si hanno dunque i grezzi impiantistici e architettonici di ogni livello nella stessa fase temporale e le finiture impiantistiche e architettoniche in cui ogni locale è nella stessa fase temporale.

1.5.2 Esportazione della pianificazione in formato XML

Definite le logiche dei predecessori si passa alla spiegazione di come esportare la pianificazione, prima dell'esportazione è necessario calcolare le tempistiche ma per comodità questo verrà fatto lungo la spiegazione. L'obiettivo è esportare la durata di ogni attività secondo l'ordine delle lavorazioni e secondo i ragionamenti dei predecessori in modo che il file esportato possa essere aperto da un programma per visualizzare la pianificazione. Per far questo si utilizza il formato XML, in pratica si decide di prendere un file XML base derivato dal software open source Project Libre e di sovrascriverlo tramite Python in modo che il file XML finale sia leggibile dallo stesso software. Con file XML base si intende che si apre il programma e si salva un file in formato XML, senza aver fatto nessuna modifica, e in questo modo si avrà un file XML con le impostazioni del software ma con gli spazi vuoti dove si può inserire la pianificazione.

Il primo passo è quello di aprire Project Libre ed esportare un file vuoto in formato XML, come detto. Poi si passa su python e si va a modificare l'algoritmo presentato in precedenza per il pilastro in modo da inserire le parti di codice mancanti per consentire l'esportazione del cronoprogramma dei pilastri. Quello che si fa, come si vede in Figura 41, è importare la libreria

"xml.etree.ElementTree" la quale serve per scrivere o modificare un file XML e poi si importa il file vuoto tramite la libreria, esattamente come fatto nel paragrafo precedente, infine si scrivere la funzione per ricavare le posizioni delle sezioni all'interno del file XML ("root=tree.getroot()").

```
import xml.etree.ElementTree as ET
tree = ET.parse('c:/Users/Utente/Desktop/Scrittura File XML/File XML.xml')
root = tree.getroot()
```

Figura 41 – Script per importare la libreria "xml.etree.ElementTree", aprire il file XML e modificarlo (immagine tratta dall'algoritmo creato dall'autore della tesi)

Per creare un'attività all'interno del file XML vuoto è necessario seguire i passaggi in Figura 42, nella quale si crea l'attività iniziale la quale serve per indicare il giorno di inizio dei lavori e per considerare le attività di preparazione iniziale del cantiere. Il primo passaggio è inserire il task dell'allestimento cantiere all'interno della sezione dell'XML dedicata ai task, per far questo tramite "ET.SubElement" si crea come sotto elemento nella posizione "root[54]" (cioè la posizione dedicata ai task) un nuovo task. Sfruttando lo stesso principio all'interno del nuovo task, appena creato, si vanno ad inserire le informazioni utili all'identificazione del task da parte di Project Libre, cioè: i numeri identificativi, il nome, il giorno di inizio e la durata (che si è ipotizzata di due giorni per l'allestimento del cantiere, cioè 16 ore lavorative).

```
10  Task=ET.SubElement(root[54], 'ns0:Task')
11  Uid=ET.SubElement(Task, 'ns0:UID')
12  Uid.text='1'
13  Id=ET.SubElement(Task, 'ns0:ID')
14  Id.text='1'
15  Name=ET.SubElement(Task, 'ns0:Name')
16  Name.text='Allestimento_Cantiere'
17  Start=ET.SubElement(Task, 'ns0:Start')
18  Start.text='2023-07-21T8:00:00'
19  Duration=ET.SubElement(Task, 'ns0:Duration')
20  Duration.text='PT16H0M0S'
```

Figura 42 – Porzione di algoritmo dedicata all'esportazione del task allestimento cantiere (immagine tratta dall'algoritmo creato dall'autore della tesi)

Tornando alle attività del pilastro precedentemente discusso, una volta ricavate tutte le quantità come spiegato nel paragrafo precedente è necessario calcolare le tempistiche. Come primo passo, come si può vedere in Figura 43, si somma il volume dei pilastri in modo da avere la quantità di calcestruzzo da gettare, ma soprattutto di poter usare la funzione "if" per attivare l'esportazione delle attività legate al pilastro solo nel caso in cui ci siano effettivamente dei

```
96          somma_vol_CA=sum(lista_vol_CA_pilastri)
97
98          if somma_vol_CA>0:
99
100             somma_sup_casseri=sum(lista_sup_casseri_pilastri)
101             somma_volume_armatura_pilastri=sum(lista_vol_armatura_pilastri)
102             armatura_in_kg=somma_volume_armatura_pilastri*7850
103
104             tempo_posa_armatura=round(armatura_in_kg*0.02, 1)
105             tempo_getto=round(somma_vol_CA*0.8, 2)
106             tempo_casseratura=round(somma_sup_casseri*0.33, 1)
107             tempo_maturazione=32
108
109             Task=ET.SubElement(root[54], 'ns0:Task')
110             Uid=ET.SubElement(Task, 'ns0:UID')
111             Uid_p=str(p)
112             Uid.text=Uid_p
113             Id=ET.SubElement(Task, 'ns0:ID')
114             Id.text=Uid_p
115             Name=ET.SubElement(Task, 'ns0:Name')
116             Name.text='Posa_armatura_pilastri'
117             Duration=ET.SubElement(Task, 'ns0:Duration')
118             frazionaria,intera = modf(tempo_posa_armatura)
119             Duration_str='PT'+str(round(intera))+'H'+str(round((round(frazionaria, 2)*60)))+'M0S'
120             Duration.text=Duration_str
121             PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
122             PredecessorUID=ET.SubElement(PredecessorLink, 'ns0:PredecessorUID')
123             PredecessorUID_str=str(p-1)
124             PredecessorUID.text=PredecessorUID_str
125             p=p+1
```

Figura 43 – Porzione dell'algoritmo dedicata all'esportazione del task della posa dell'armatura dei pilastri (immagine tratta dall'algoritmo creato dall'autore della tesi)

pilastri, cioè quanto il volume è maggiore di zero. Dopo di che si sommano anche le quantità di superficie laterale e di armatura in modo da avere le quantità totali, per l'armatura è poi necessario passare anche dal volume al peso in Kg tramite il peso specifico. A questo punto si possono calcolare i tempi delle varie lavorazioni moltiplicando per la durata come già spiegato. Poi con gli stessi passaggi spiegati prima per l'esportazione dell'allestimento cantiere, più la parte legata ai predecessori, come si può vedere in Figura 43, si scrive il codice per l'esportazione della posa dell'armatura per i pilastri. In particolare, ci si avvale della libreria "math" attraverso la quale si importa "modf" il quale consente di dividere la tempistica dell'attività, la quale è calcolata in ore, in modo da poter indicare la durata in ore e minuti come voluto da Project Libre. Inoltre, indicando una variabile iniziale "p" la si può utilizzare per mettere in sequenza le attività mettendole semplicemente in ordine all'interno dello script, inserendo la variabile diminuita di 1 all'interno dei predecessori e aumentandola di 1 dopo ogni lavorazione. In Figura 43 si riporta solo la posa dell'armatura ma, come si può intuire, appena sotto sono presenti anche gli altri task di cui sono stati calcolati i tempi.

Alla fine di tutto algoritmo, come si può vedere in Figura 44 è necessario inserire il comando "tree.write()" perché il file XML sia sovrascritto e salvato.

```
186     tree.write('cronoprogramma.xml')
```

Figura 44 – Porzione di codice finale necessaria per l'esportazione della pianificazione (immagine tratta dall'algoritmo creato dall'autore della tesi)

1.6 Visualizzazione della pianificazione in Project Libre

Seguendo i passaggi spiegati in precedenza si ottiene quindi un algoritmo che avviato esporta un file XML, il quale aperto con Project Libre ci dà il risultato visibile in Figura 45. Nella quale si può visualizzare il cronoprogramma completo per la costruzione dei pilastri dell'edificio rappresentato in precedenza (paragrafo 1.2.2), in sequenza per entrambi i livelli, secondo la logica dell'algoritmo creato. Chiaramente questo cronoprogramma è infattibile dato che considera solo i pilastri dell'edificio i quali non si possono fare in sequenza, infatti è solo un esempio per dimostrare qual è il risultato finale.

	Nome	Durata	Data di Avvio	Data di chiusura	Predecessori
1	Allestimento_Cantiere	2 giorni	24/07/23 8.00	25/07/23 17.00	
2	Posa_armatura_pilastri liv. 0-str	0,537 giorni	26/07/23 8.00	26/07/23 13.18	1
3	Casseratura_Pilastri_CA liv. 0-str	1,038 giorni	26/07/23 13.18	27/07/23 13.36	2
4	Getto_Pilastri_CA liv. 0-str	0,19 giorni	27/07/23 13.36	27/07/23 15.07	3
5	Maturazione_CA liv. 0-str	4 giorni	27/07/23 15.07	02/08/23 15.07	4
6	Posa_armatura_pilastri liv. 1-str	0,537 giorni	02/08/23 15.07	03/08/23 10.25	5
7	Casseratura_Pilastri_CA liv. 1-str	1,038 giorni	03/08/23 10.25	04/08/23 10.43	6
8	Getto_Pilastri_CA liv. 1-str	0,19 giorni	04/08/23 10.43	04/08/23 13.14	7
9	Maturazione_CA liv. 1-str	4 giorni	04/08/23 13.14	10/08/23 13.14	8

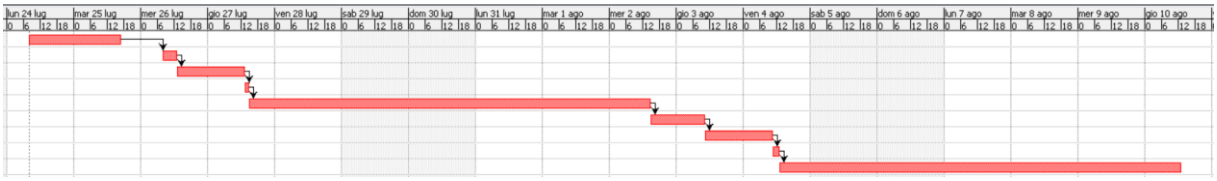


Figura 45 – Cronoprogramma pilastri (immagine tratta da Project Libre)

In Figura 45 possiamo vedere il numero ed il nome delle singole attività (con l'indicazione dei livelli di appartenenza), la loro durata in giorni, la data di avvio, la data di chiusura ed i predecessori. Inoltre, poi il cronoprogramma è rappresentato tramite un diagramma di Gantt (spiegato in precedenza). Ripetendo i passaggi dei paragrafi precedenti per tutti gli elementi dell'edificio si ottiene un cronoprogramma automatico dei lavori edili, come vedremo nel paragrafo successivo in cui verrà presentato un caso studio.

Si specifica inoltre che il file base ricavato da Project Libre, come si vede dal cronoprogramma, è impostato per una settimana lavorativa formata da 5 giorni, escludendo il sabato e la domenica, e da una giornata lavorativa di 8 ore, dalle 8 alle 12 e dalle 13 alle 17. Questi sono vincoli base che non si modificano in quanto non sono importanti per il momento ma potranno essere modificati in sviluppi futuri.

2 CASO STUDIO

Per validare la metodologia proposta si presenta un semplice caso studio riguardante un piccolo edificio elaborato dall'autore della tesi il quale, per scelta, non si basa su un progetto reale ma vuol essere il più elementare possibile per comprendere pregi e difetti del metodo. In ogni caso tutto quello che si fa per questo caso studio è applicabile anche per un edificio reale in quanto si sono seguiti standard di costruzione reali. Si presenta di seguito l'edificio facendo una distinzione tra la parte architettonica e la parte strutturale (non si è modellata la parte impiantistica dell'edifico), poi si presenta l'algoritmo, aggiungendo delle considerazioni sulla parte architettonica, ed infine si visualizzano i risultati del caso studio.

2.1 Presentazione dell'edificio

Si è deciso si modellare l'edificio in due modelli separati uno per la parte strutturale e uno per la parte architettonica, la quale si modella sul modello strutturale linkato. Si sono modellate sia la parte strutturale che quella architettonica per dimostrare l'efficacia multidisciplinare del metodo, escludendo, per questa tesi, la parte impiantistica per mantenere il modello il più semplice possibile. Questo non significa che la metodologia non sia valida anche per la parte impiantistica, la quale è tranquillamente inseribile in sviluppi futuri. Come spiegato nella metodologia i modelli BIM 3D sono modellati tramite il software Autodesk Revit. Nella presentazione dell'edificio non si ritiene necessario andare nel dettaglio con sezioni, piante, prospetti, ecc. ma ci si limita ad una presentazione generale, appunto perché essendo un caso studio elementare e non legato ad un caso reale non ha senso approfondirlo. Di seguito vediamo presentati il modello strutturale il modello architettonico.

2.1.1 Modello strutturale

Per la parte strutturale si è deciso di modellare un telaio in calcestruzzo armato ,formato da travi e pilastri, con setti in calcestruzzo armato a formare il vano scala (vedi Figura 46).

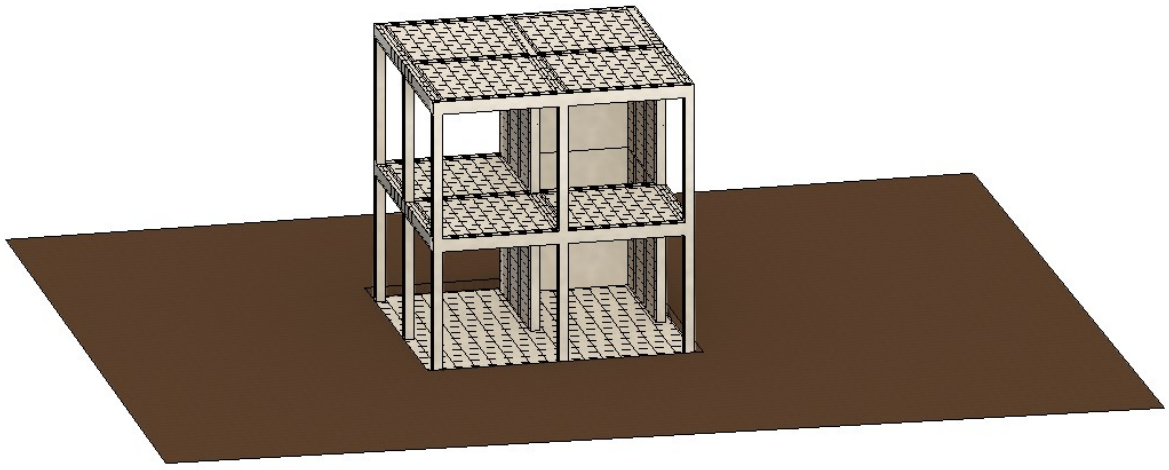


Figura 46 – Modello strutturale BIM (immagine tratta da Autodesk Revit)

Per un maggiore dettaglio si vede la Figura 47 dove partendo dal basso si ha: un magrone in calcestruzzo appeso al relativo piano; una platea in calcestruzzo armato appesa al livello 0-strutturale; uno spazio libero per l'inserimento del pavimento architettonico. Il solaio strutturale è appeso a 4 metri dal livello zero (cioè dal terreno), non ci sono ragioni specifiche per l'altezza della stanza che è stata scelta casualmente in quanto poco influente. Il solaio è anch'esso in calcestruzzo armato per semplificare al massimo la struttura, in quanto un solaio a travetti e pignatte sarebbe stato più complesso da modellare e avrebbe richiesto un maggiore sforzo computazionale nella creazione dell'algoritmo. Con questo non si vuol dire che non sia possibile automatizzare la pianificazione per un solaio a travetti e pignatte, ma dato che l'obiettivo della tesi è partire da un caso semplice, per dimostrare come si può automatizzare la pianificazione, si è scelto un solaio a piastra in calcestruzzo armato. Negli sviluppi futuri si potrà provvedere a creare la parte di algoritmo adatta ad un solaio a travetti e pignatte.

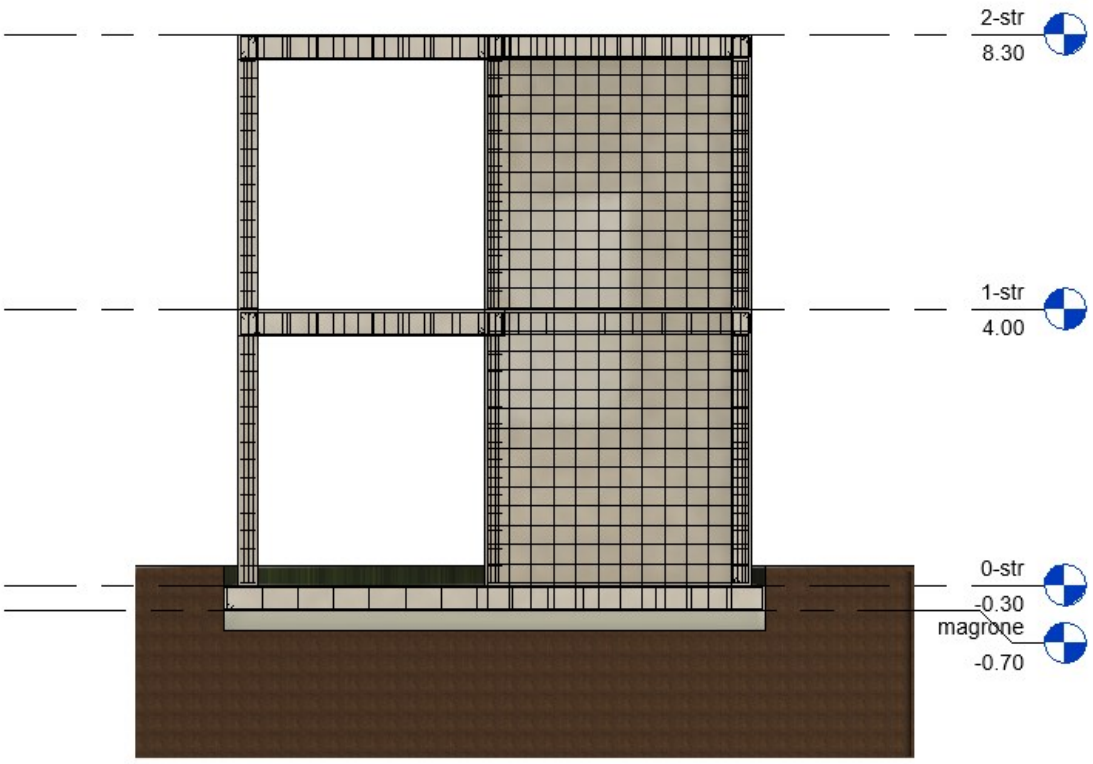


Figura 47 – Sezione modello strutturale BIM (immagine tratta da Autodesk Revit)

Ovviamente come si può vedere si sono modellate le armature per ciascun elemento strutturale (escluso il magrone che non ne necessita). Le armature inserite sono dello stesso tipo per tutti gli elementi e non sono frutto di calcoli dimensionali (come già specificato l'edificio non è un caso reale), ma sono inserite solamente per dimostrare che l'algoritmo creato è in grado di definire nella programmazione finita la loro posa in opera, rispettando i reali metodi di costruzione.

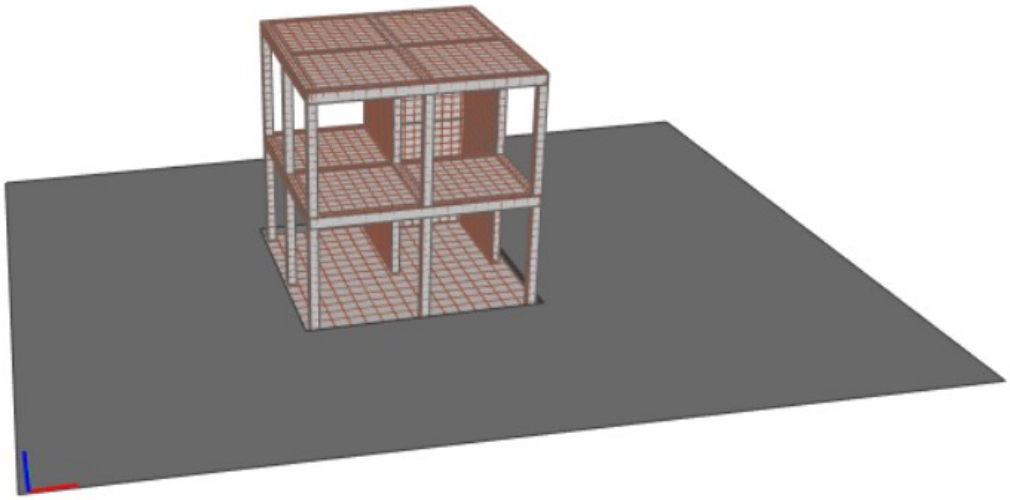


Figura 48 – Modello strutturale in formato IFC (immagine tratta da BIMvision)

Il modello strutturale una volta esportato tramite IFC e aperto in BIMvision si presenta come si vede in Figura 48, dove in particolare vediamo che le armature sono rappresentate in rosso. Si è già spiegato nei paragrafi riguardanti la metodologia che tramite BIMvision è possibile accedere a tutte le informazioni del modello BIM esportato in IFC, e questo risulta molto utile per controllare che tutto sia stato esportato correttamente.

2.1.2 Modello architettonico

Per modellare la parte architettonica si parte dal modello strutturale, lo si linka in un altro file dove si inseriscono le parti architettoniche, cioè muri, pavimenti architettonici, tetto, scale e infissi. In Figura 49 possiamo vedere il modello architettonico.

Per un maggiore dettaglio possiamo vedere Figura 50 dove si vede che sopra le fondazioni o i solai strutturali sono stati modellati dei pavimenti architettonici formati da quattro strati: massetto, isolante, sottofondo, pavimentazione. I muri sono in mattone con isolante esterno e finitura interna ed esterna. Come già specificato in presenza di setti o pilastri si è modellato l'isolante e finitura esterni e la finitura interna come due muri separati. Poi si hanno dei semplici infissi inseriti nei muri architettonici e una scala non gettata ma assemblata in opera. Infine, si ha un tetto leggermente inclinato per consentire lo scolo dell'acqua considerabile comunque come tetto piano. Sopra la parte strutturale il tetto si compone di isolante, sottofondo e finitura.

Volutamente anche in questo caso, come per la parte strutturale, la parte architettonica si è modellata in modo da essere la più basilare possibile, infatti come spiegato pavimenti, muri e il tetto sono molto basilari e nell'edifico non è stato aggiungo nessun elemento superfluo,

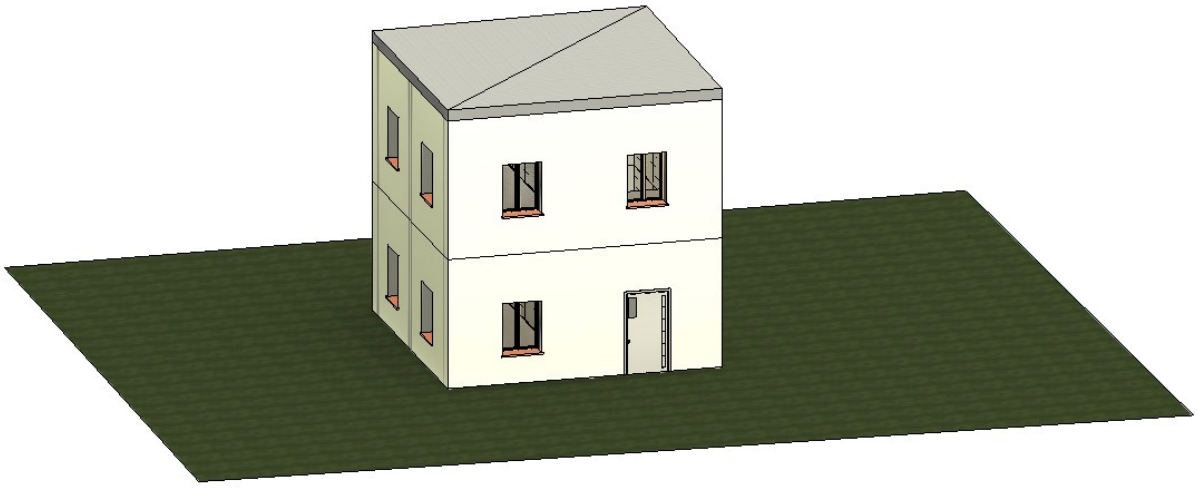


Figura 49 – Modello architettonico BIM, modellato sopra il modello strutturale (immagine tratta da Autodesk Revit)

cioè con il fine di migliorare l'estetica.

Infine, come per la parte strutturale, in Figura 51 si presenta il modello architettonico esportato tramite IFC e aperto con BIMvision, dove in particolare notiamo le separazioni dei

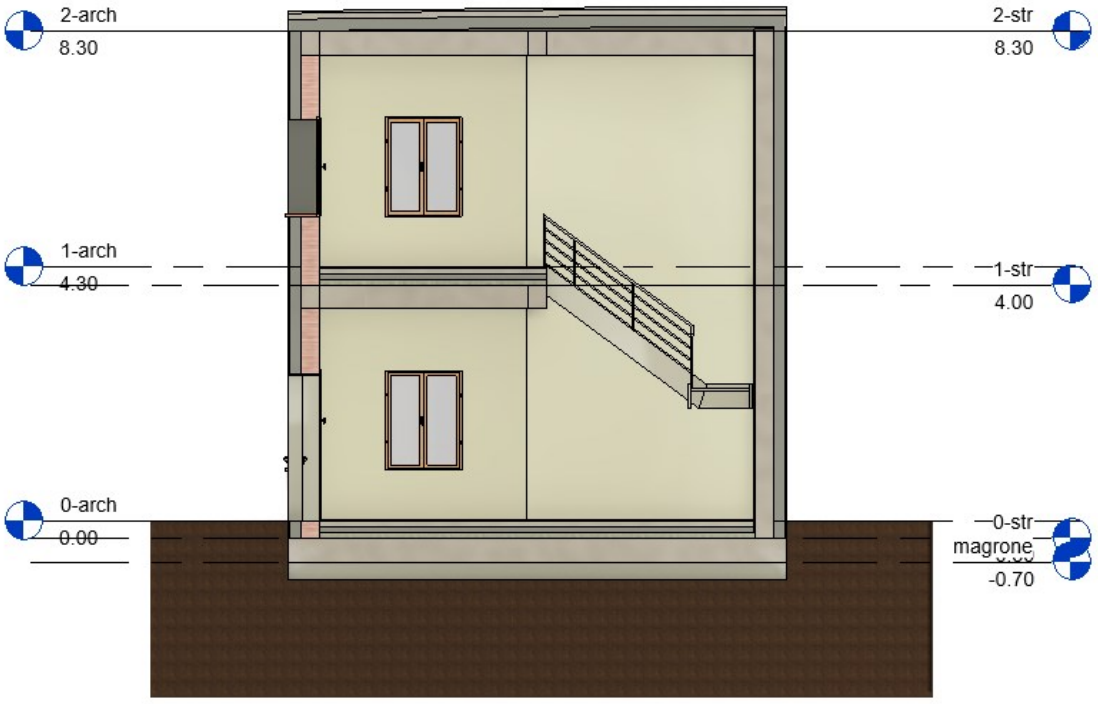


Figura 50 – Sezione modello architettonico BIM (immagine tratta da Autodesk Revit)

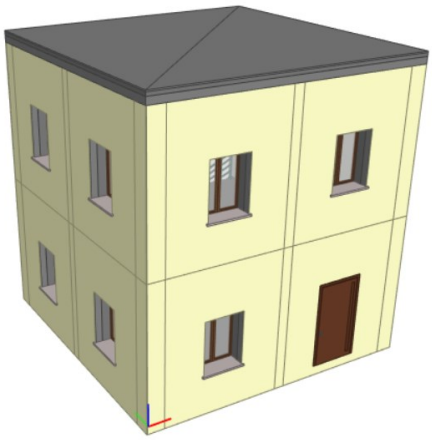


Figura 51 – Modello architettonico in formato IFC
(immagine tratta da BIMvision)

muri in corrispondenza dei pilastri, per il motivo già spiegato. Inoltre, non si nota il terreno perché esso fa parte del modello strutturale.

2.2 Presentazione dell'algoritmo

Dopo aver presentato l'edifico si passa alla presentazione dell'algoritmo il quale si riporta per intero in Appendice 2, perché essendo lungo andrebbe ad appesantire troppo lo scorrere della presentazione del caso studio.

L'algoritmo si presenta come spiegato nel capitolo della metodologia, quindi in questo paragrafo non si spiegheranno le parti di codice come fatto in precedenza ma ci si limiterà ad analizzare i passaggi principali. Nella prima parte dell'algoritmo si inseriscono i file IFC strutturale e architettonico. Dopo di che l'algoritmo è diviso in due parti, cioè due grossi cicli "while" di cui il primo serve a prendere i piani strutturali uno per volta ed il secondo a prendere quelli architettonici uno alla volta. In questo modo la prima parte analizzerà l'intero modello strutturale un piano alla volta secondo l'ordine di costruzione degli elementi, cioè: IfcBeam, IfcSlab, IfcWall, IfcColumn. Questo vuol dire che l'algoritmo parte dal livello più basso, che in questo caso è il magrone, e prende in considerazione prima le travi, poi le piastre, poi i setti e poi i pilastri. Ovviamente dato che a quel livello c'è solo il magrone verranno inserite nell'algoritmo solo le operazioni necessarie all'esecuzione del magrone. Inoltre, l'algoritmo è stato impostato in modo che prima di eseguire il magrone, riconoscendo quest'ultima entità inserisca anche le operazioni per lo scavo. Poi andando avanti riconoscerà fondazione, setti, pilastri, travi e solaio inserendo le lavorazioni una alla volta, una dopo l'altra senza sovrapposizioni, se non per i setti e i pilastri come spiegato in precedenza.

Per quanto riguarda la parte architettonica l'algoritmo è un po' più complesso nella sua disposizione degli elementi in quanto, come detto, la parte architettonica segue una logica dei

predecessori per livello (in questo esempio segue sempre una logica per livello, non essendoci partizioni interne che potrebbero far passare la logica da livello a locale). Innanzitutto, si specifica che si usa più di una variabile per indicare il corretto predecessore così da avere che in automatico i due piani architettonici vengono programmati per essere realizzati in contemporanea. Dopo di che la logica dei predecessori tra gli elementi all'interno del singolo piano segue lo schema riportato in Figura 52. Nella quale si nota che è presente l'operazione di finitura del muro senza l'operazione di intonacatura, questo perché per mantenere l'esempio elementare si uniscono le due lavorazioni in un'unica operazione finale, in sviluppi futuri si potranno dividere per rendere il modello più dettagliato.

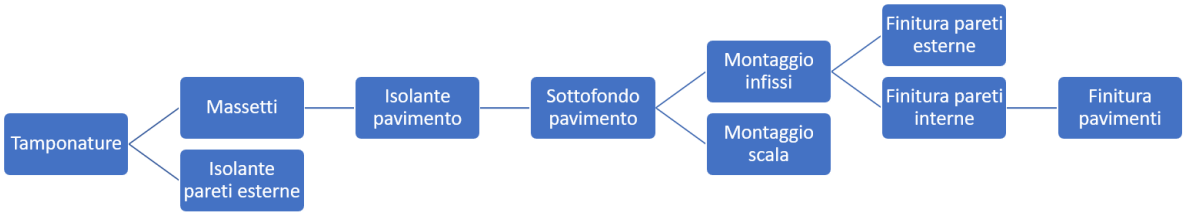


Figura 52 - Schema predecessori nel singolo piano architettonico (immagine elaborata dall'autore della tesi)

Per ricavare le quantità architettoniche si seguono le stesse logiche spiegate per la parte strutturale nei paragrafi dedicati alla metodologia, c'è solo una sostanziale differenza, cioè gli elementi architettonici sono spesso composti da più di uno strato a differenza degli elementi strutturali. Questo comporta il fatto che se come in questo caso ho muri con gli stessi materiali ma differenti strati, devo riconoscere i muri con gli strati che fanno parte di una singola lavorazione. Ad esempio, se ho un muro formato da muratura, isolante e finitura ed un altro muro formato da isolante e finitura, io devo essere in grado di riconoscere entrambi i muri se sono nella lavorazione di posa dell'isolante mentre solo il primo se sono nella lavorazione di posa della muratura. Questo perché come si può vedere nelle tabelle dei metodi di costruzione in Appendice 1 mi basta conoscere l'area del muro per le lavorazioni architettoniche del muro. Per risolvere questo problema in questa prima tesi si è dovuto usare il "naming system", cioè per riconoscere i materiali degli strati del muro si deve conoscere il nome del materiale assegnato dall'utente su revit. Questa semplificazione del problema va bene per questa singola tesi ma nei successivi sviluppi bisogna provare ad eliminarla in modo da rendere l'algoritmo indipendente dai nomi e basato solo su parametri di IFC. Altro caso è se io ho un muro con finitura esterna e finitura interna devo essere in grado di riconoscere qual è quella interna e qual è quella esterna, questo argomento merita un approfondimento ma per ragioni di tempistiche in questa tesi si tratta come il resto degli elementi architettonici. Di conseguenza in conclusione se si vuole utilizzare l'algoritmo proposto in questa tesi ed applicarlo a casi più complessi sarà necessario chiedere a chi modella l'edificio di consegnare oltre al modello BIM anche una lista dei materiali utilizzati per i vari elementi. Si ribadisce però che sarebbe utile sviluppare il tema per togliere questo vincolo legato al conoscere il nome dei materiali.

2.3 Visualizzazione dei risultati

Avviando l'algoritmo presentato in precedenza si esporta un file XML, il quale contiene tutte le informazioni sulla pianificazione. Aprendo il file XML con Project Libre si ottiene il risultato visibile in Figura 53, nella quale è visualizzabile il cronoprogramma completo dell'edificio proposto. Si può subito vedere che, come spiegato, la prima parte cioè quella strutturale, segue un andamento lineare diagonale, tranne nel caso di pilastri e setti dove vediamo una sovrapposizione delle operazioni in colore blu. Mentre nella seconda parte, cioè quella architettonica possiamo vedere che la pianificazione è sovrapposta tra i due piani, dato che i piani architettonici seguono una logica per "livello". Il colore rosso sta ad indicare la sequenza delle operazioni critiche (CPM), nella quale se ci dovesse essere qualche ritardo causerebbe un ritardo dell'intero progetto.

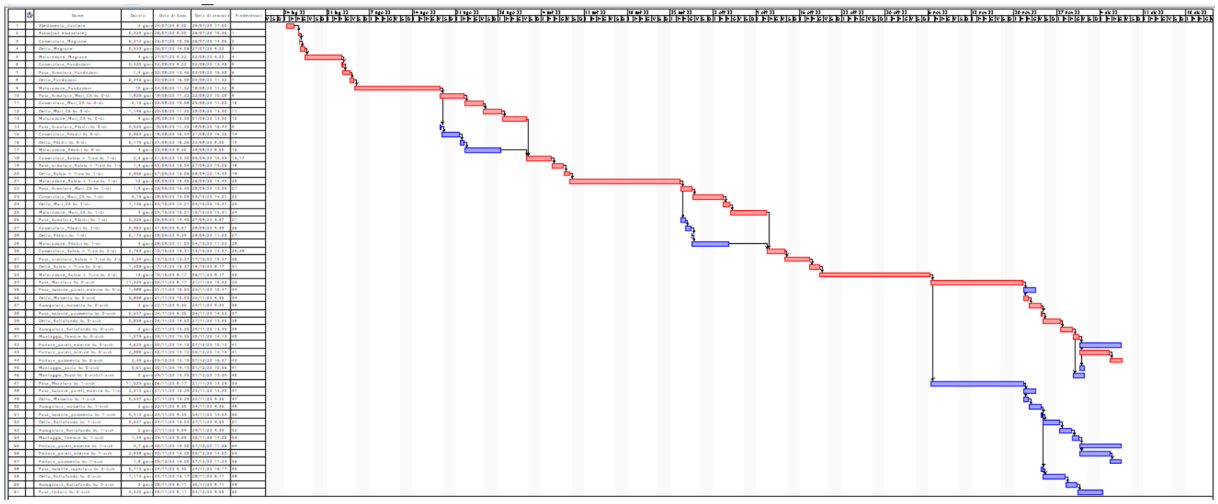


Figura 53 – Cronoprogramma completo dei lavori edili dell'edificio proposto nel caso studio (immagine tratta da Project Libre)

Di seguito si può vedere il cronoprogramma diviso in tre parti in modo da riuscire a leggere il nome, la durata, la data di avvio, la data di chiusura e i predecessori di ogni singola attività. L'inizio della costruzione si è fissato per il 24/07/2023 e la fine sarà il 04/12/2023, per un totale di circa 5 mesi di lavorazione per l'intera costruzione dell'edificio rispettando i vincoli imposti.

Il punto fondamentale è che tramite questo caso studio si è dimostrato che si può ottenere la pianificazione dei lavori in modo automatico, estraendo direttamente le quantità dal BIM, per poi passare ai successivi sviluppi della metodologia spiegati in precedenza.

	Nome	Durata	Data di Avvio	Data di chiusura	Predecessori
1	Allestimento_Cantiere	2 giorni	24/07/23 8.00	25/07/23 17.00	
2	Scavo(con escavatore)	0,325 giorni	26/07/23 8.00	26/07/23 10.36	1
3	Casseratura_Magrone	0,312 giorni	26/07/23 10.36	26/07/23 14.06	2
4	Getto_Magrone	0,533 giorni	26/07/23 14.06	27/07/23 9.22	3
5	Maturazione_Magrone	4 giorni	27/07/23 9.22	02/08/23 9.22	4
6	Casseratura_Fondazioni	0,425 giorni	02/08/23 9.22	02/08/23 13.46	5
7	Posa_Armatura_Fondazioni	1,4 giorni	02/08/23 13.46	03/08/23 16.58	6
8	Getto_Fondazioni	0,446 giorni	03/08/23 16.58	04/08/23 11.32	7
9	Maturazione_Fondazioni	10 giorni	04/08/23 11.32	18/08/23 11.32	8
10	Posa_Armatura_Muri_CA liv. 0-str	1,825 giorni	18/08/23 11.32	22/08/23 10.08	9
11	Casseratura_Muri_CA liv. 0-str	3,15 giorni	22/08/23 10.08	25/08/23 11.20	10
12	Getto_Muri_CA liv. 0-str	1,146 giorni	25/08/23 11.20	28/08/23 13.30	11
13	Maturazione_Muri_CA liv. 0-str	4 giorni	28/08/23 13.30	01/09/23 13.30	12
14	Posa_Armatura_Pilastri liv. 0-str	0,525 giorni	18/08/23 11.32	18/08/23 16.44	9
15	Casseratura_Pilastri liv. 0-str	0,963 giorni	18/08/23 16.44	21/08/23 16.26	14
16	Getto_Pilastri liv. 0-str	0,175 giorni	21/08/23 16.26	22/08/23 8.50	15
17	Maturazione_Pilastri liv. 0-str	4 giorni	22/08/23 8.50	28/08/23 8.50	16
18	Casseratura_Solaio + Travi liv. 1-str	2,3 giorni	01/09/23 13.30	05/09/23 15.54	13;17
19	Posa_armatura_Solaio + Travi liv. 1-str	1,9 giorni	05/09/23 15.54	07/09/23 15.06	18
20	Getto_Solaio + Travi liv. 1-str	0,956 giorni	07/09/23 15.06	08/09/23 14.45	19
21	Maturazione_Solaio + Travi liv. 1-str	12 giorni	08/09/23 14.45	26/09/23 14.45	20

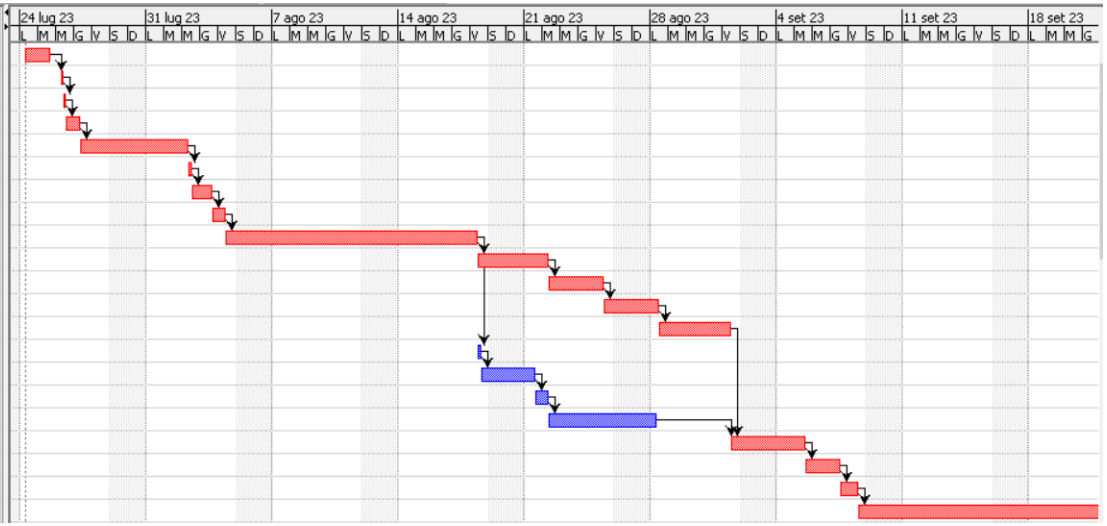


Figura 55 – Prima parte del cronoprogramma finale (immagine tratta da Project Libre)

	Nome	Durata	Data di Avvio	Data di chiusura	Predecessori
22	Posa_Armatura_Muri_CA liv. 1-str	1,8 giorni	26/09/23 14.45	28/09/23 13.09	21
23	Casseratura_Muri_CA liv. 1-str	3,15 giorni	28/09/23 13.09	03/10/23 14.21	22
24	Getto_Muri_CA liv. 1-str	1,146 giorni	03/10/23 14.21	04/10/23 15.31	23
25	Maturazione_Muri_CA liv. 1-str	4 giorni	04/10/23 15.31	10/10/23 15.31	24
26	Posa_Armatura_Pilastri liv. 1-str	0,525 giorni	26/09/23 14.45	27/09/23 9.57	21
27	Casseratura_Pilastri liv. 1-str	0,963 giorni	27/09/23 9.57	28/09/23 9.39	26
28	Getto_Pilastri liv. 1-str	0,175 giorni	28/09/23 9.39	28/09/23 11.03	27
29	Maturazione_Pilastri liv. 1-str	4 giorni	28/09/23 11.03	04/10/23 11.03	28
30	Casseratura_Solaio + Travi liv. 2-str	2,763 giorni	10/10/23 15.31	13/10/23 13.37	25;29
31	Posa_armatura_Solaio + Travi liv. 2-str	2,25 giorni	13/10/23 13.37	17/10/23 15.37	30
32	Getto_Solaio + Travi liv. 2-str	1,208 giorni	17/10/23 15.37	19/10/23 8.17	31
33	Maturazione_Solaio + Travi liv. 2-str	12 giorni	19/10/23 8.17	06/11/23 8.17	32
34	Posa_Muratura liv. 0-arch	11,325 giorni	06/11/23 8.17	21/11/23 10.53	33
35	Posa_isolante_pareti_esterne liv. 0-arch	1,988 giorni	21/11/23 10.53	23/11/23 10.47	34
36	Getto_Massetto liv. 0-arch	0,838 giorni	21/11/23 10.53	22/11/23 9.35	34
37	Asciugatura_massetto liv. 0-arch	2 giorni	22/11/23 9.35	24/11/23 9.35	36
38	Posa_isolante_pavimento liv. 0-arch	0,537 giorni	24/11/23 9.35	24/11/23 14.53	37
39	Getto_Sottofondo liv. 0-arch	0,838 giorni	24/11/23 14.53	27/11/23 13.35	38

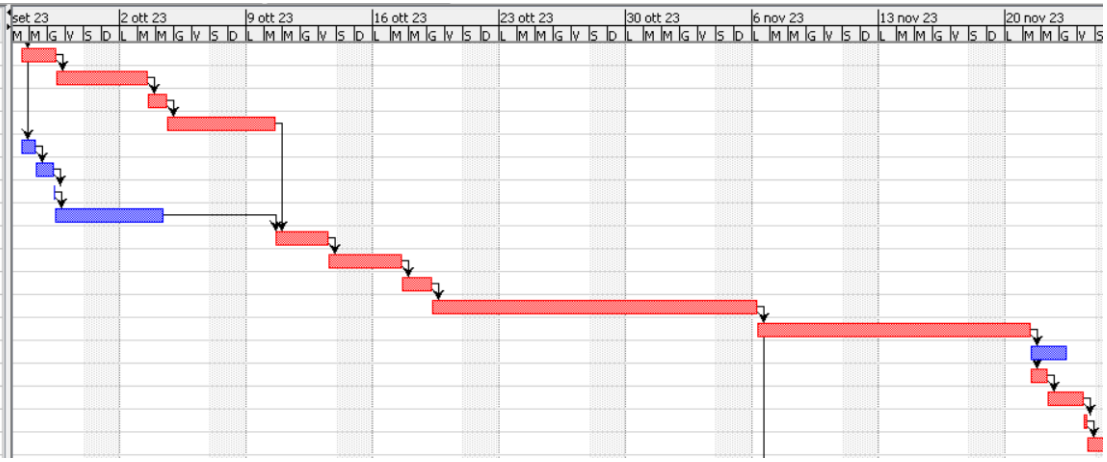
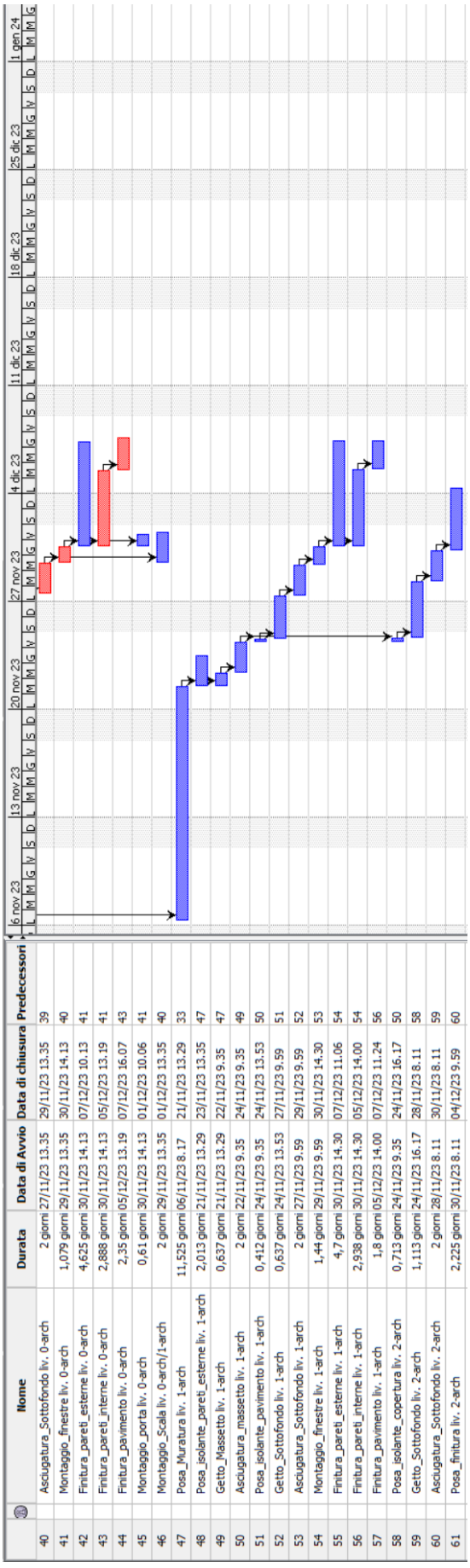


Figura 54 – Seconda parte del cronoprogramma finale (immagine tratta da Project Libre)



	Nome	Durata	Data di Avvio	Data di chiusura	Predecessori
40	Asciugatura_Sottofondo liv. 0-arch	2 giorni	27/11/23 13.35	29/11/23 13.35	39
41	Montaggio_finestre liv. 0-arch	1,079 giorni	29/11/23 13.35	30/11/23 14.13	40
42	Finitura_pareti_esterne liv. 0-arch	4,625 giorni	30/11/23 14.13	07/12/23 10.13	41
43	Finitura_pareti_interne liv. 0-arch	2,888 giorni	30/11/23 14.13	05/12/23 13.19	41
44	Finitura_pavimento liv. 0-arch	2,35 giorni	05/12/23 13.19	07/12/23 16.07	43
45	Montaggio_porta liv. 0-arch	0,61 giorni	30/11/23 14.13	01/12/23 10.06	41
46	Montaggio_Scala liv. 0-arch/1-arch	2 giorni	29/11/23 13.35	01/12/23 13.35	40
47	Posa_Muratura liv. 1-arch	11,525 giorni	06/11/23 8.17	21/11/23 13.29	33
48	Posa_isolante_pareti_esterne liv. 1-arch	2,013 giorni	21/11/23 13.29	23/11/23 13.35	47
49	Getto_Massetto liv. 1-arch	0,637 giorni	21/11/23 13.29	22/11/23 9.35	47
50	Asciugatura_massetto liv. 1-arch	2 giorni	22/11/23 9.35	24/11/23 9.35	49
51	Posa_isolante_pavimento liv. 1-arch	0,412 giorni	24/11/23 9.35	24/11/23 13.53	50
52	Getto_Sottofondo liv. 1-arch	0,637 giorni	24/11/23 13.53	27/11/23 9.59	51
53	Asciugatura_Sottofondo liv. 1-arch	2 giorni	27/11/23 9.59	29/11/23 9.59	52
54	Montaggio_finestre liv. 1-arch	1,44 giorni	29/11/23 9.59	30/11/23 14.30	53
55	Finitura_pareti_esterne liv. 1-arch	4,7 giorni	30/11/23 14.30	07/12/23 11.06	54
56	Finitura_pareti_interne liv. 1-arch	2,938 giorni	30/11/23 14.30	05/12/23 14.00	54
57	Finitura_pavimento liv. 1-arch	1,8 giorni	05/12/23 14.00	07/12/23 11.24	56
58	Posa_isolante_copertura liv. 2-arch	0,713 giorni	24/11/23 9.35	24/11/23 16.17	50
59	Getto_Sottofondo liv. 2-arch	1,113 giorni	24/11/23 16.17	28/11/23 8.11	58
60	Asciugatura_Sottofondo liv. 2-arch	2 giorni	28/11/23 8.11	30/11/23 8.11	59
61	Posa_finitura liv. 2-arch	2,225 giorni	30/11/23 8.11	04/12/23 9.59	60

Figura 56 – Terza parte del cronoprogramma finale (immagine tratta da Project Libre)

Capitolo Quarto

ANALISI CRITICA DEI RISULTATI

In questo capitolo si evidenziano le semplificazioni fatte per arrivare al risultato, in modo da dare degli spunti di miglioramento e chiarire il quadro generale, per poi analizzare i risultati ottenuti

Innanzitutto, si può dire che l'obiettivo della tesi è stato raggiunto, riuscendo ad ottenere una pianificazione automatizzata dei lavori edili direttamente dal modello BIM esportato in formato IFC. Per arrivare all'obiettivo si sono dovute fare delle semplificazioni del problema tramite alcune ipotesi e regole, esse sono:

- Regole di corretta modellazione;
- Metodi di costruzione predefiniti;
- Tempistiche delle lavorazioni e squadra tipo stimate con il prezziario della regione Veneto e/o per analogia;
- Materiali disponibili immediatamente per ogni nuova lavorazione;
- Risorse umane disponibili per eseguire i piani architettonici in contemporanea;
- Montaggio dei ponteggi o delle protezioni anti-caduta non considerato.

Essendo questa una prima tesi sull'argomento le semplificazioni/ipotesi fatte sono state necessarie per raggiungere un risultato soddisfacente dato che l'argomento è molto ampio, di seguito se ne discute per capire come migliorare la tesi.

Quando si utilizza un modello BIM esportato in IFC è necessario imporre delle regole di base su come modellare il BIM, questo perché per estrarre le quantità corrette, su cui basare i tempi di ciascuna lavorazione, gli elementi devono corrispondere alla realtà. Si ritiene che le regole di corretta modellazione non siano una semplificazione del problema al fine di non complicarlo, ma piuttosto sono regole con il fine di rendere la pianificazione credibile. Per questo motivo si consiglia di mantenerle e ampliarle, non cercare un modo per eliminarle. Questo serve anche a creare un metodo di modellazione univoco da seguire per evitare problematiche anche nel dialogo tra più modelli BIM all'interno del software stesso.

Come detto, una pianificazione parte sempre da dei metodi di costruzione, l'obiettivo però sarebbe far sì che l'algoritmo funzioni in modo indipendente dai metodi di costruzione scelti per l'edificio. Inoltre, c'è anche il problema di definire i predecessori per passare dalla creazione di un singolo elemento del modello ad un altro. Per quanto riguarda la parte strutturale l'algoritmo funziona bene perché le strutture possono essere fatte solamente in ordine, dal basso verso l'alto, perciò una volta definito il metodo di costruzione ,che può essere struttura gettata in opera in CA, struttura prefabbricata in CA, struttura in legno, ecc.,

l'algoritmo funziona senza problemi. L'importante è creare uno script che distingua in base al tipo di struttura i metodi di costruzione, gli elementi poi saranno messi necessariamente in ordine dal basso verso l'alto nella pianificazione. Per quanto riguarda la parte architettonica il problema è più complesso perché non cambiano solo i metodi di costruzione ma al cambiare di quest'ultimi molto spesso cambia anche l'ordine delle lavorazioni all'interno del singolo piano, esempio se ho le tramezze interne in cartongesso posso decidere di farle prima o dopo il massetto. Quindi per la questione metodi di costruzione/predecessori, per la parte strutturale sarà necessario implementare l'algoritmo in modo che possa riconoscere più metodi, mentre per la parte architettonica oltre ad implementare l'algoritmo perché riconosca più metodi di costruzione sarà necessario anche trovare delle logiche comuni che possano mettere in ordine le lavorazioni indipendentemente dal metodo scelto, esempio si può dire che le tamponature esterne si devono fare sempre come prima operazione e poi definire le attività per creare la tamponatura in base al metodo. Importante è definire delle logiche chiare che vadano bene in generale in modo che l'algoritmo sia utilizzabile in varie casistiche.

Ragionando poi sulle tempistiche delle varie lavorazioni e della squadra tipo per ogni lavorazione si è utilizzato il dato ricavato dal prezziario della regione Veneto accompagnato da qualche stima per analogia. Il che va bene per questa tesi, ma la questione merita un approfondimento ulteriore in modo da capire come vengono fatte queste stime delle tempistiche e se c'è un metodo per migliorarle dato che la validità del cronoprogramma si basa necessariamente sulla tempistica della singola lavorazione.

Un'altra ipotesi utilizzata è che i materiali necessari per ogni lavorazione siano subito disponibili escludendo ritardi o attese. Questo nella realtà è abbastanza falso, nel senso che l'organizzazione dei depositi e delle forniture del materiale sono temi importanti all'interno di un cantiere, i quali possono far modificare, e non di poco, la durata dei lavori. Sarebbe quindi necessario integrare questo discorso alla pianificazione fatta in questa tesi. Si potrebbero utilizzare le quantità ricavate dal BIM e il cronoprogramma per prevedere in automatico la quantità di materiale necessarie per il determinato giorno in modo che siano disponibili in cantiere. Inoltre, poi c'è tutto il tema dell'organizzazione dei depositi dei materiali in cantiere, il quale però è un tema che va inizialmente sviluppato a parte per poi essere unito ai ragionamenti fatti in questa tesi.

Altro ragionamento è quello legato al fatto che per fare i piani architettonici in contemporanea c'è bisogno di un numero di risorse umane adeguate, e se queste risorse sono disponibili non ci sono problemi finché si parla di lavorazioni in contemporanea tra due piani diversi. Esempio se faccio le murature al piano terra e al piano primo in contemporanea non ci sono grossi problemi, una volta definiti i depositi del materiale e il movimento delle gru o della gru. I problemi nascono se si fanno più lavorazioni in contemporanea sullo stesso piano in quanto, all'interno di un singolo piano, le risorse applicate non possono essere infinite dato che i vari spazi hanno delle dimensioni e quindi c'è un limite di risorse per garantire la sicurezza. Quindi, anche qui andrebbe sviluppato un tema a parte per capire quante persone in

contemporanea posso mettere per velocizzare al massimo le lavorazioni senza che ci siano problemi di intralcio tra i lavoratori. Così da capire se fare in contemporanea più lavorazioni all'interno dello stesso piano può essere utile a velocizzare il progetto o magari può causare danni.

In questa tesi non si è mai discussa l'organizzazione ed il montaggio delle impalcature o dei dispositivi anti-caduta, esempio quando si passa dalla costruzione del piano terra strutturale a quella del piano primo è necessario montare delle impalcature. In questa tesi il montaggio delle impalcature non si è considerato in quanto si è ipotizzato che esse vengano montate nei tempi "morti" in cui si aspetta la maturazione degli elementi al piano terra, e quindi ininfluenti nel calcolo delle tempistiche. Anche questo tema però merita un approfondimento migliore in modo da capire se il montaggio dei ponteggi o simili possono generare ritardi o problematiche di sovrapposizioni con altre lavorazioni.

Oltre alle semplificazioni sopra discusse, si sono dovuti imporre dei vincoli per rendere il caso studio più coincidente alla realtà, essi sono:

- Settimana lavorativa di cinque giorni;
- Giornata lavorativa di otto ore.

Questi vincoli esulano dai discorsi precedenti, infatti essi sono vincoli esterni non dipendenti da particolari ragionamenti ma solo dagli usi del paese in cui si costruisce. Certo che se ce ne fosse la possibilità ed il bisogno si potrebbero modificare per ridurre il periodo di costruzione di un edificio, facendo più turni giornalieri o più giornate lavorative.

Anche se come discusso in questo paragrafo gli ulteriori temi da affrontare sono molti, il risultato di questa prima tesi risulta comunque molto soddisfacente in quanto si sono ottenuti gli obiettivi prefissati. Infatti, grazie alle semplificazioni discusse si è arrivati ad una pianificazione automatica dell'edifico, la quale spinge a continuare a lavorare su questa strada con sviluppi futuri. Ad esempio, un primo passo potrebbe essere fare un test su un edifico reale più complesso e capire quali sono gli eventuali ulteriori aspetti da migliorare oltre a quelli già discussi.

Infine, si dedica un'ultima parte ad IFC4 il quale è migliorabile sotto l'aspetto della distinzione degli spessori dei materiali di un singolo oggetto, ad esempio un muro. Provando ad utilizzare anche IFC2.3 si è visto che per arrivare al singolo strato il percorso da compiere era più diretto e meno complicato. Detto ciò, però sono anni che IFC4 è disponibile, perciò, si spera che a breve possa uscire una nuova versione, la quale magari semplifichi questo passaggio.

Capitolo Quinto

CONCLUSIONI E SVILUPPI FUTURI

In questo capitolo si riportano in sintesi le conclusioni relative a come i risultati offrono una soluzione alle criticità individuate o una capitalizzazione degli aspetti strategici. Si evidenziano inoltre gli sviluppi futuri che la tesi può attivare in relazione alla disciplina oggetto di studio.

Si riporta di seguito, in sintesi, quello che è stato fatto durante la tesi. Nel primo capitolo si è analizzata la procedura per ottenere un cronoprogramma manualmente, utilizzando il modello BIM come riferimento. Di seguito, sempre all'interno del primo capitolo, si sono analizzati il BIM, il formato IFC e la pianificazione 4D. Grazie a questi si è potuto poi comprendere più facilmente gli strumenti di pianificazione assistita proposti negli anni. Poi, nel capitolo secondo, si è effettuata l'analisi critica della letteratura evidenziando varie problematiche negli attuali sistemi. Per risolvere le problematiche attuali è necessario fare un passo avanti e pensare ad un nuovo sistema, di cui in questa tesi se ne sviluppa una prima parte che è la pianificazione automatizzata dei lavori edili. Successivamente, nel capitolo terzo, si è passati alla spiegazione dettagliata della metodologia, tramite la quale da un modello BIM di partenza si arriva ad un cronoprogramma automatico rappresentato tramite Project Libre. Nello stesso capitolo, si è applicata la metodologia ad un caso studio in modo da confermare il procedimento metodologico proposto. Infine, nel capitolo quarto, si è analizzato in modo critico il risultato ottenuto con un focus sulle parti migliorabili della metodologia.

Si può dire che il risultato che si è ottenuto è soddisfacente in quanto era fondamentale, in questo primo approccio al problema, capire come estrarre le quantità dal modello BIM. Oltre a questo, si è fatto un passaggio ulteriore, cioè si è esportata la pianificazione in modo che fosse visibile e la si potesse valutare. Si è detto che questa tesi funge da primo passo per lo sviluppo di un sistema di pianificazione automatizzata che sia in grado di restituire poi le informazioni al BIM. Per questo sviluppo futuro si ipotizza che non sia necessario esportare la pianificazione in un ambiente di visualizzazione del cronoprogramma, questo perché ci si vuole allontanare dal concetto di "scheduling", in cui la pianificazione e una semplice rappresentazione del tempo. Ma come detto si vuole creare un modello informativo che contenga le attività, le risorse e le tempistiche come attributi di ciascun oggetto. Quindi, questo può essere semplicemente fatto all'interno di un ambiente computazionale. Dall'altro lato si riconosce il fatto dell'importanza della visione della pianificazione tramite un cronoprogramma, il quale sia anche possibilmente modificabile a mano e poi restituibile all'ambiente di computazione. Si aprono quindi questi due scenari da analizzare, di cui in questa tesi si è discusso quello in cui si esporta la pianificazione in modo che sia visualizzabile il cronoprogramma, ma non si scarta il secondo

scenario, il quale sicuramente si avvicina di più al concetto di pianificazione automatizzata.

Oltre a quanto appena discusso due altri importanti sviluppi futuri sono legati ai temi delle risorse e dei macchinari, esempio la gru. Partendo dal primo, in questa tesi si sono utilizzate le squadre "tipo" tratte dal prezziario della regione Veneto però volendo fare un passo ulteriore si può aumentare il numero di operai in modo da ridurre le tempistiche. Ovviamente se un operaio fa un muro in 100 giorni non posso pensare che 100 operai me lo facciano in un giorno, questo perché c'è un limite di risorse applicabili in un certo spazio che possiamo chiamare "affollamento critico". Sarebbe quindi interessante analizzare il concetto in modo da trovare dei dati o eventualmente effettuare delle simulazioni agent-based in modo da ricavare degli indici che mi dicono quanti operai possono lavorare contemporaneamente in un certo spazio. In questo modo aggiungendo questi dati all'algoritmo si può arrivare a stimare il numero di operai per ogni lavorazione, in base allo spazio disponibile, per far sì che il tempo totale della lavorazione sia il minimo possibile. Tutto questo avendo la massima efficienza ed evitando i problemi di sicurezza derivanti da un numero troppo elevato di operai, che paradossalmente andrebbe aumentare il tempo totale in quanto gli operai si intralciano tra di loro. Il secondo tema è quello legato ad esempio alla gru, in quanto una gru può essere al servizio di un certo numero di lavorazioni massime in una giornata lavorativa. Questo fa pensare che aumentando il numero delle gru magari si possano anche velocizzare le tempistiche di alcune a lavorazioni, potendone fare di più in contemporanea, se ci sono le risorse disponibili. Questo, ad esempio, vale anche se sono in dubbio se mettere una gru grande o due gru piccole, con le gru piccole avrei il vantaggio di fare più lavorazioni in contemporanea. Mettere più di una gru però, oltre ad aumentare i costi, apre il tema della disposizione delle gru in cantiere in modo che entrambe si possano muovere liberamente. Sarebbe quindi interessante analizzare anche questa tematica in futuro con l'ottica di migliorare la pianificazione ed arrivare ad una riduzione dei tempi delle lavorazioni.

1 APPENDICE 1 – TABELLE DI DEFINIZIONE DELLE SEQUENZE TIPO DEI METODI DI COSTRUZIONE

Scavo qualsiasi profondità con escavatore idraulico a 17 T.					
#	Attività	Risorse	Durata in ore (h)	Quantità unitaria	U.d.M.
1	Scavo	1	0,03	1	m^3

Tabella 1

Fondazione a platea in CA gettata in opera					
#	Attività	Risorse	Durata in ore (h)	Quantità unitaria	U.d.M
1	Casseratura magrone	2	0,25	1	m^2
2	Getto cls magrone	2	0,2	1	m^3
3	Maturazione magrone	\	32	\	\
4	Casseratura platea	2	0,25	1	m^2
5	Posa armatura platea	3	0,02	1	Kg
6	Getto cls platea	2	0,125	1	m^3
7	Maturazione platea	\	80	\	\

Tabella 2

Pilastri in CA gettati in opera					
#	Attività	Risorse	Durata in ore (h)	Quantità unitaria	U.d.M
1	Posa armatura pilastri	3	0,02	1	Kg
2	Casseratura pilastri	2	0,33	1	m^2
3	Getto cls pilastri	2	0,8	1	m^3
4	Maturazione pilastri	\	32	\	\

Tabella 3

Setti in CA gettati in opera					
#	Attività	Risorse	Durata in ore (h)	Quantità unitaria	U.d.M
1	Posa armatura setti	3	0,02	1	Kg
2	Casseratura setti	2	0,23	1	m^2
3	Getto cls setti	2	0,6	1	m^3
4	Maturazione setti	\	32	\	\

Tabella 4

Solaio a piastra in CA + Travi in CA gettati in opera					
#	Attività	Risorse	Durata in ore (h)	Quantità unitaria	U.d.M
1	Casseratura solaio + travi	2	0,3	1	Kg
2	Posa armatura solaio + travi	3	0,02	1	m^2
3	Getto cls solaio + travi	2	0,4	1	m^3
4	Maturazione solaio + travi	\	96	\	\

Tabella 5

Tamponatura in muratura					
#	Attività	Risorse	Durata in ore (h)	Quantità unitaria	U.d.M
1	Posa muratura	2	1,1	1	m^2

Tabella 6

Isolamento pareti esterne					
#	Attività	Risorse	Durata in ore (h)	Quantità unitaria	U.d.M
1	Posa isolante	2	0,12	1	m^2

Tabella 7

Massetto					
#	Attività	Risorse	Durata in ore (h)	Quantità unitaria	U.d.M
1	Getto massetto	2	0,125	1	m^2
2	Asciugatura massetto	\	16	\	\

Tabella 8

Isolamento pavimenti					
#	Attività	Risorse	Durata in ore (h)	Quantità unitaria	U.d.M
1	Posa isolante	2	0,08	1	m^2

Tabella 9

Sottofondo					
#	Attività	Risorse	Durata in ore (h)	Quantità unitaria	U.d.M
1	Getto sottofondo	2	0,125	1	m^2
2	Asciugatura sottofondo	\	16	\	\

Tabella 10

Finestre					
#	Attività	Risorse	Durata in ore (h)	Quantità unitaria	U.d.M
1	Montaggio finestra	2	1,5	1	m^2

Tabella 11

Scala assemblata in opera					
#	Attività	Risorse	Durata in ore (h)	Quantità unitaria	U.d.M
1	Montaggio scala	\	16	\	\

Tabella 12

Finitura pareti					
#	Attività	Risorse	Durata in ore (h)	Quantità unitaria	U.d.M
1	Finitura	2	0,28	1	m^2

Tabella 13

Finitura pavimenti					
#	Attività	Risorse	Durata in ore (h)	Quantità unitaria	U.d.M
1	Posa piastrelle	2	0,35	1	m^2

Tabella 14

2 APPENDICE 2 – ALGORITMO COMPLETO DEL CASO STUDIO

```
import ifcopenshell
ifc_file_str=ifcopenshell.open('C:/Users/Utente/Desktop/Crono piccolo
edificio/nuovo/modello strutturale.ifc')
ifc_file_arch=ifcopenshell.open('C:/Users/Utente/Desktop/Crono piccolo
edificio/nuovo/modello architettonico.ifc')
import xml.etree.ElementTree as ET
tree = ET.parse('c:/Users/Utente/Desktop/prova scrittura file XML/prova
scrittura XML.xml')
root = tree.getroot()
from math import modf

Task=ET.SubElement(root[54], 'ns0:Task')
Uid=ET.SubElement(Task, 'ns0:UID')
Uid.text='1'
Id=ET.SubElement(Task, 'ns0:ID')
Id.text='1'
Name=ET.SubElement(Task, 'ns0:Name')
Name.text='Allestimento_Cantiere'
Start=ET.SubElement(Task, 'ns0:Start')
Start.text='2023-07-24T8:00:00'
Duration=ET.SubElement(Task, 'ns0:Duration')
Duration.text='PT16H0M0S'

i_i=0
p=2
IfcbuildingStorey = ifc_file_str.by_type('IfcBuildingStorey')
while i_i<len(IfcbuildingStorey):
    IfcElements = IfcbuildingStorey[i_i].ContainsElements
    if IfcElements!=():
        IfcElements=IfcbuildingStorey[i_i].ContainsElements[0].RelatedElements

        beams=[]
        for element in IfcElements:
            if element.is_a('IfcBeam'):
                beams.append(element)

        i_3=0
        lista_vol_CA_travi=[]
        lista_sup_casseri_travi=[]
        lista_vol_armatura_travi=[]
        for beam in beams:
            while i_3<len(beams):
                beam_i=beams[i_3]
                j=0
                while j<len(beam_i.IsDefinedBy):
```

```
                    IfcElementQuantity=beam_i.IsDefinedBy[j].RelatingPropertyD
efinition
                    if IfcElementQuantity[2]==str('Qto_BeamBaseQuantities'):
                        k=0
                        Quantities_tot=IfcElementQuantity.Quantities
                        while k<len(Quantities_tot):
                            for quantity in Quantities_tot:
                                if quantity.is_a('IfcQuantityVolume'):
                                    if quantity[0]==str('GrossVolume'):
                                        volume_CA=quantity.VolumeValue
                                        k=k+1
                                        j=j+1
                                        i_3=i_3+1
                                    else:
                                        k=k+1
                                else:
                                    k=k+1
                        l=0
                        while l<len(Quantities_tot):
                            if Quantities_tot[l][0]==str('Length'):
                                lunghezza=Quantities_tot[l][3]
                                l=l+1
                            else:
                                l=l+1
                    else:
                        j=j+1
                j_1=0
                while
j_1<len(beam_i.IsTypedBy[0].RelatingType.HasPropertySets):
                    Ifcpropertyset=beam_i.IsTypedBy[0].RelatingType.HasPropert
ySets[j_1]
                    if Ifcpropertyset[2]==str('Quote'):
                        k_1=0
                        IfcPropertySingleValue=Ifcpropertyset.HasProperties
                        while k_1<len(IfcPropertySingleValue):
                            if IfcPropertySingleValue[k_1][0]==str('b'):
                                misura_b=IfcPropertySingleValue[k_1].NominalVa
lue[0]
                                k_1=k_1+1
                                j_1=j_1+1
                            else:
                                None
                            if IfcPropertySingleValue[k_1][0]==str('h'):
                                misura_h=IfcPropertySingleValue[k_1].NominalVa
lue[0]
                                k_1=k_1+1
                                j_1=j_1+1
                            else:
```

```
                                k_1=k_1+1
                        else:
                            j_1=j_1+1

                    j_a=0
                    while j_a<len(beam_i.IsDefinedBy):
                        IfcPropertySet=beam_i.IsDefinedBy[j_a].RelatingPropertyDef
inition
                        if IfcPropertySet.Name==str('Strutturale'):
                            k=0
                            Properties=IfcPropertySet.HasProperties
                            while k<len(Properties):
                                for property in Properties:
                                    if property.Name==str('Volume armatura
stimato'):
                                        Volume_armatura=property.NominalValue[0]
                                        j_a=j_a+1
                                        k=k+1
                                    else:
                                        k=k+1
                        else:
                            j_a=j_a+1

                    lista_vol_armatura_travi.append(Volume_armatura)

                    lista_vol_CA_travi.append(volume_CA)
                    superficie_casseri_travi=(lunghezza*misura_h)+(lunghezza*misur
a_b)
                    lista_sup_casseri_travi.append(superficie_casseri_travi)
                i_3=i_3+1

            slabs=[]
            for element in IfcElements:
                if element.is_a('IfcSlab'):
                    slabs.append(element)

            i_0=0
            lista_vol_CA_slabs=[]
            lista_sup_casseri_slabs=[]
            lista_vol_CA_magrone=[]
            lista_sup_casseri_magrone=[]
            lista_vol_armatura_slabs=[]
            if len(slabs)==1:
                slab_i=slabs[0]
                if slab_i[8]==str('FLOOR'):
                    j=0
                    while j<len(slab_i.IsDefinedBy):
```

```
                    IfcElementQuantity=slab_i.IsDefinedBy[j].RelatingPropertyD
efinition
                    if IfcElementQuantity[2]==str('Qto_SlabBaseQuantities'):
                        Volume_CA_magrone=IfcElementQuantity.Quantities[0].Vol
umeValue
                        j=j+1
                    else:
                        j=j+1

                l=0
                while l<len(slab_i.IsDefinedBy):
                    IfcPropertySet=slab_i.IsDefinedBy[l].RelatingPropertyDefin
ition
                    if IfcPropertySet[2]==str('Quote'):
                        k=0
                        Quantities=IfcPropertySet[4]
                        while k<len(Quantities):
                            if Quantities[k].Name==str('Perimetro'):
                                Perimetro_magrone=Quantities[k][2][0]
                                k=k+1
                            elif Quantities[k].Name==str('Area'):
                                Area_magrone=Quantities[k][2][0]
                                k=k+1
                            else:
                                k=k+1
                                l=l+1
                    else:
                        l=l+1

                profondità_livello_magrone=slab_i.ObjectPlacement.PlacementRel
To.PlacesObject[0].Elevation
                profondità_scavo=profondità_livello_magrone-
(Volume_CA_magrone/Area_magrone)
                volume_scavo=Area_magrone*(-profondità_scavo)

                superficie_casseri_magrone=(Volume_CA_magrone/Area_magrone)*Pe
rimetro_magrone
                lista_vol_CA_magrone.append(Volume_CA_magrone)
                lista_sup_casseri_magrone.append(superficie_casseri_magrone)

            elif slab_i[8]==str('BASESLAB'):
                j=0
                while j<len(slab_i.IsDefinedBy):
                    IfcElementQuantity=slab_i.IsDefinedBy[j].RelatingPropertyD
efinition
                    if IfcElementQuantity[2]==str('Qto_SlabBaseQuantities'):
                        k=0
                        Quantities_tot=IfcElementQuantity.Quantities
```

```
                        while k<len(Quantities_tot):
                            for quantity in Quantities_tot:
                                if quantity.is_a('IfcQuantityLength'):
                                        if quantity[0]==str('Depth'):
                                            spessore_CA=quantity.LengthValue
                                            k=k+1
                                            j=j+1
                                        else:
                                            k=k+1
                                else:
                                    k=k+1
                        l=0
                        while l<len(Quantities_tot):
                            if Quantities_tot[l][0]==str('GrossArea'):
                                area=Quantities_tot[l][3]
                                l=l+1
                            elif Quantities_tot[l][0]==str('Perimeter'):
                                Perimeter=Quantities_tot[l][3]
                                l=l+1
                            else:
                                l=l+1
                    else:
                        j=j+1

                j_a=0
                while j_a<len(slab_i.IsDefinedBy):
                    IfcPropertySet=slab_i.IsDefinedBy[j_a].RelatingPropertyDef
inition
                    if IfcPropertySet.Name==str('Strutturale'):
                        k=0
                        Properties=IfcPropertySet.HasProperties
                        while k<len(Properties):
                            for property in Properties:
                                if property.Name==str('Volume armatura
stimato'):
                                    Volume_armatura=property.NominalValue[0]
                                    j_a=j_a+1
                                    k=k+1
                                else:
                                    k=k+1
                    else:
                        j_a=j_a+1

                lista_vol_armatura_slabs.append(Volume_armatura)

                volume_CA_slabs=spessore_CA*area
                lista_vol_CA_slabs.append(volume_CA_slabs)
```

```
                superficie_casseri=(spessore_CA*Perimeter)
                lista_sup_casseri_slabs.append(superficie_casseri)
        else:
            for slab in slabs:
                while i_0<len(slabs):
                    slab_i=slabs[i_0]
                    j=0
                    while j<len(slab_i.IsDefinedBy):
                        IfcElementQuantity=slab_i.IsDefinedBy[j].RelatingPrope
rtyDefinition
                        if
IfcElementQuantity[2]==str('Qto_SlabBaseQuantities'):
                            k=0
                            Quantities_tot=IfcElementQuantity.Quantities
                            while k<len(Quantities_tot):
                                for quantity in Quantities_tot:
                                    if quantity.is_a('IfcQuantityLength'):
                                        if quantity[0]==str('Depth'):
                                            spessore_CA=quantity.LengthValue
                                            k=k+1
                                            j=j+1
                                            i_0=i_0+1
                                        else:
                                            k=k+1
                                    else:
                                        k=k+1
                            l=0
                            while l<len(Quantities_tot):
                                if Quantities_tot[l][0]==str('GrossArea'):
                                    area=Quantities_tot[l][3]
                                    l=l+1
                                else:
                                    l=l+1
                        else:
                            j=j+1

                    j_a=0
                    while j_a<len(slab_i.IsDefinedBy):
                        IfcPropertySet=slab_i.IsDefinedBy[j_a].RelatingPropert
yDefinition
                        if IfcPropertySet.Name==str('Strutturale'):
                            k=0
                            Properties=IfcPropertySet.HasProperties
                            while k<len(Properties):
                                for property in Properties:
                                    if property.Name==str('Volume armatura
stimato'):
```

```
                                        Volume_armatura=property.NominalValue[
0]
                                        j_a=j_a+1
                                        k=k+1
                                    else:
                                        k=k+1
                            else:
                                j_a=j_a+1

                        lista_vol_armatura_slabs.append(Volume_armatura)

                        volume_CA_slabs=spessore_CA*area
                        lista_vol_CA_slabs.append(volume_CA_slabs)
                        superficie_casseri=area
                        lista_sup_casseri_slabs.append(superficie_casseri)
                    i_0=i_0+1

            somma_volume_CA_magrone=sum(lista_vol_CA_magrone)

            if somma_volume_CA_magrone>0:

                volume_scavo_ampliato=(volume_scavo*1.2)
                tempo_scavo=round(volume_scavo_ampliato*0.03, 1)
                somma_superficie_casseri_magrone=sum(lista_sup_casseri_magrone)
                tempo_casseratura=round(superficie_casseri_magrone*0.25, 1)
                tempo_getto=round(somma_volume_CA_magrone*0.2, 2)
                tempo_maturazione=32

                Task=ET.SubElement(root[54], 'ns0:Task')
                Uid=ET.SubElement(Task, 'ns0:UID')
                Uid_p=str(p)
                Uid.text=Uid_p
                Id=ET.SubElement(Task, 'ns0:ID')
                Id.text=Uid_p
                Name=ET.SubElement(Task, 'ns0:Name')
                Name.text='Scavo(con escavatore)'
                Duration=ET.SubElement(Task, 'ns0:Duration')
                frazionaria,intera = modf(tempo_scavo)
                Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
                Duration.text=Duration_str
                PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
                PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
                PredecessorUID_str=str(p-1)
                PredecessorUID.text=PredecessorUID_str
                p=p+1
```

```
            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Casseratura_Magrone'
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_casseratura)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Getto_Magrone'
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_getto)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
```

```
            Name.text='Maturazione_Magrone'
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_maturazione)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1


        somma_vol_CA_travi=sum(lista_vol_CA_travi)
        somma_vol_CA_slabs=sum(lista_vol_CA_slabs)
        if somma_vol_CA_slabs>0 and somma_vol_CA_travi==0:

            somma_sup_casseri_slabs=sum(lista_sup_casseri_slabs)
            somma_volume_armatura_fondazione=sum(lista_vol_armatura_slabs)
            armatura_in_kg=somma_volume_armatura_fondazione*7850

            tempo_posa_armatura=round(armatura_in_kg*0.02, 1)
            tempo_casseratura=round(somma_sup_casseri_slabs*0.25, 1)
            tempo_getto=round(somma_vol_CA_slabs*0.125, 2)
            tempo_maturazione=80

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Casseratura_Fondazioni'
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_casseratura)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1
```

```
            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Posa_Armatura_Fondazioni'
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_posa_armatura)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Getto_Fondazioni'
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_getto)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Maturazione_Fondazioni'
```

```
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_maturazione)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1

        elif somma_vol_CA_slabs>0 and somma_vol_CA_travi>0:

            somma_sup_casseri_slabs=sum(lista_sup_casseri_slabs)
            somma_sup_casseri_travi=sum(lista_sup_casseri_travi)
            somma_volume_armatura_solaio=sum(lista_vol_armatura_slabs)
            somma_volume_armatura_travi=sum(lista_vol_armatura_travi)
            somma_vol_arm_travi_solaio=somma_volume_armatura_travi+somma_volum
e_armatura_solaio

            armatura_in_kg=somma_vol_arm_travi_solaio*7850
            tempo_posa_armatura=round(armatura_in_kg*0.02, 1)
            tempo_casseratura=round((somma_sup_casseri_slabs+somma_sup_casseri
_travi)*0.3, 1)
            tempo_getto=round((somma_vol_CA_slabs+somma_vol_CA_travi)*0.4, 2)
            tempo_maturazione=96

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Casseratura_Solaio + Travi' + ' liv. ' +
str(IfcbuildingStorey[i_i][2])
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_casseratura)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-5)
            PredecessorUID.text=PredecessorUID_str
```

```
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Posa_armatura_Solaio + Travi' + ' liv. ' +
str(IfcbuildingStorey[i_i][2])
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_posa_armatura)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Getto_Solaio + Travi' + ' liv. ' +
str(IfcbuildingStorey[i_i][2])
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_getto)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1
```

```
            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Maturazione_Solaio + Travi' + ' liv. ' +
str(IfcbuildingStorey[i_i][2])
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_maturazione)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1
        else:
            None

        walls=[]
        for element in IfcElements:
            if element.is_a('Ifcwall'):
                walls.append(element)

        i_1=0
        lista_vol_CA_muri=[]
        lista_sup_casseri_muri=[]
        lista_vol_armatura_muri=[]
        for wall in walls:
            while i_1<len(walls):
                wall_i=walls[i_1]
                j=0
                while j<len(wall_i.IsDefinedBy):
                    IfcElementQuantity=wall_i.IsDefinedBy[j].RelatingPropertyD
efinition
                    if IfcElementQuantity[2]==str('Qto_WallBaseQuantities'):
                        k=0
                        Quantities_tot=IfcElementQuantity.Quantities
                        while k<len(Quantities_tot):
                            for quantity in Quantities_tot:
                                if quantity.is_a('IfcQuantityLength'):
                                    if quantity[0]==str('Width'):
```

```
                                    spessore_CA=quantity.LengthValue
                                    k=k+1
                                    j=j+1
                                    i_1=i_1+1
                                else:
                                    k=k+1
                            else:
                                k=k+1
                    l=0
                    while l<len(Quantities_tot):
                        if Quantities_tot[l][0]==str('GrossSideArea'):
                            area=Quantities_tot[l][3]
                            l=l+1
                        elif Quantities_tot[l][0]==str('Height'):
                            height=Quantities_tot[l][3]
                            l=l+1
                        else:
                            l=l+1
                else:
                    j=j+1

            j_a=0
            while j_a<len(wall_i.IsDefinedBy):
                IfcPropertySet=wall_i.IsDefinedBy[j_a].RelatingPropertyDef
inition
                if IfcPropertySet.Name==str('Strutturale'):
                    k=0
                    Properties=IfcPropertySet.HasProperties
                    while k<len(Properties):
                        for property in Properties:
                            if property.Name==str('Volume armatura
stimato'):
                                Volume_armatura=property.NominalValue[0]
                                j_a=j_a+1
                                k=k+1
                            else:
                                k=k+1
                else:
                    j_a=j_a+1

            lista_vol_armatura_muri.append(Volume_armatura)

            volume_CA=spessore_CA*area
            lista_vol_CA_muri.append(volume_CA)
            superficie_casseri=(area*2)+(spessore_CA*height*2)
            lista_sup_casseri_muri.append(superficie_casseri)
        i_1=i_1+1
```

```
        somma_vol_CA=sum(lista_vol_CA_muri)
        if somma_vol_CA>0:

            somma_sup_casseri=sum(lista_sup_casseri_muri)
            somma_volume_armatura_muri=sum(lista_vol_armatura_muri)
            armatura_in_kg=somma_volume_armatura_muri*7850

            tempo_posa_armatura=round(armatura_in_kg*0.02, 1)
            tempo_casseratura=round(somma_sup_casseri*0.23, 1)
            tempo_getto=round(somma_vol_CA*0.6, 2)
            tempo_maturazione=32

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Posa_Armatura_Muri_CA' + ' liv. ' +
str(IfcbuildingStorey[i_i][2])
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_posa_armatura)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Casseratura_Muri_CA' + ' liv. ' +
str(IfcbuildingStorey[i_i][2])
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_casseratura)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
```

```
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Getto_Muri_CA' + ' liv. ' +
str(IfcbuildingStorey[i_i][2])
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_getto)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Maturazione_Muri_CA' + ' liv. ' +
str(IfcbuildingStorey[i_i][2])
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_maturazione)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1
```

```
        #print()

        columns=[]
        for element in IfcElements:
            if element.is_a('IfcColumn'):
                columns.append(element)

        i_2=0
        lista_vol_CA_pilastri=[]
        lista_sup_casseri_pilastri=[]
        lista_vol_armatura_pilastri=[]
        for column in columns:
            while i_2<len(columns):
                column_i=columns[i_2]
                j=0
                while j<len(column_i.IsDefinedBy):
                    IfcElementQuantity=column_i.IsDefinedBy[j].RelatingPropert
yDefinition
                    if IfcElementQuantity[2]==str('Qto_ColumnBaseQuantities'):
                        k=0
                        Quantities_tot=IfcElementQuantity.Quantities
                        while k<len(Quantities_tot):
                            for quantity in Quantities_tot:
                                if quantity.is_a('IfcQuantityVolume'):
                                    if quantity[0]==str('GrossVolume'):
                                        volume_CA=quantity.VolumeValue
                                        k=k+1
                                        j=j+1
                                        i_2=i_2+1
                                    else:
                                        k=k+1
                                else:
                                    k=k+1
                        l=0
                        while l<len(Quantities_tot):
                            if Quantities_tot[l][0]==str('OuterSurfaceArea'):
                                somma_aree_laterali=Quantities_tot[l][3]
                                l=l+1
                            else:
                                l=l+1
                    else:
                        j=j+1

                j_a=0
                while j_a<len(column_i.IsDefinedBy):
```

```
                    IfcPropertySet=column_i.IsDefinedBy[j_a].RelatingPropertyD
efinition
                    if IfcPropertySet.Name==str('Strutturale'):
                        k=0
                        Properties=IfcPropertySet.HasProperties
                        while k<len(Properties):
                            for property in Properties:
                                if property.Name==str('Volume armatura
stimato'):
                                    Volume_armatura=property.NominalValue[0]
                                    j_a=j_a+1
                                    k=k+1
                                else:
                                    k=k+1
                    else:
                        j_a=j_a+1

                lista_vol_armatura_pilastri.append(Volume_armatura)

                lista_vol_CA_pilastri.append(volume_CA)
                superficie_casseri=somma_aree_laterali
                lista_sup_casseri_pilastri.append(superficie_casseri)
            i_2=i_2+1

        somma_vol_CA=sum(lista_vol_CA_pilastri)
        if somma_vol_CA>0:

            somma_sup_casseri=sum(lista_sup_casseri_pilastri)
            somma_volume_armatura_pilastri=sum(lista_vol_armatura_pilastri)
            armatura_in_kg=somma_volume_armatura_pilastri*7850

            tempo_posa_armatura=round(armatura_in_kg*0.02, 1)
            tempo_getto=round(somma_vol_CA*0.8, 2)
            tempo_casseratura=round(somma_sup_casseri*0.33, 1)
            tempo_maturazione=32

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Posa_Armatura_Pilastri' + ' liv. ' +
str(IfcbuildingStorey[i_i][2])
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_posa_armatura)
```

```
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-5)
            PredecessorUID.text=PredecessorUID_str
            #ConstraintType=ET.SubElement(Task, 'ns0:ConstraintType')
            #ConstraintType.text='1'
            p=p+1

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Casseratura_Pilastri' + ' liv. ' +
str(IfcbuildingStorey[i_i][2])
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_casseratura)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Getto_Pilastri' + ' liv. ' +
str(IfcbuildingStorey[i_i][2])
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_getto)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
```

```
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Maturazione_Pilastri' + ' liv. ' +
str(IfcbuildingStorey[i_i][2])
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_maturazione)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1

        i_i=i_i+1
    else:
        i_i=i_i+1

i_a=0
h=1
t=1
IfcbuildingStorey = ifc_file_arch.by_type('IfcBuildingStorey')
while i_a<len(IfcbuildingStorey):
    IfcElements = IfcbuildingStorey[i_a].ContainsElements
    if IfcElements!=():
        IfcElements=IfcbuildingStorey[i_a].ContainsElements[0].RelatedElements

        walls=[]
        for element in IfcElements:
            if element.is_a('Ifcwall'):
                walls.append(element)

        i_1=0
        lista_sup_muratura=[]
```

```
        for wall in walls:
            while i_1<len(walls):
                wall_i=walls[i_1]
                j=0
                while j<len(wall_i.IsDefinedBy):
                    IfcElementQuantity=wall_i.IsDefinedBy[j].RelatingPropertyD
efinition
                    if IfcElementQuantity[2]==str('Qto_WallBaseQuantities'):
                        k=0
                        Quantities_tot=IfcElementQuantity.Quantities
                        while k<len(Quantities_tot):
                            if
Quantities_tot[k].is_a('IfcPhysicalComplexQuantity'):
                                if Quantities_tot[k].Name==str('Mattone,
Forato'):
                                    l_m=0
                                    while l_m<len(Quantities_tot):
                                        if
Quantities_tot[l_m][0]==str('NetSideArea'):
                                            area=Quantities_tot[l_m][3]
                                            lista_sup_muratura.append(area)
                                            k=k+1
                                            j=j+1
                                            l_m=l_m+1
                                        else:
                                            k=k+1
                                            l_m=l_m+1
                                else:
                                    k=k+1
                            else:
                                k=k+1
                        j=j+1
                        i_1=i_1+1
                    else:
                        j=j+1


        somma_sup_muratura=sum(lista_sup_muratura)

        if somma_sup_muratura>0:

            tempo_costr_muratura=round(somma_sup_muratura*1.1, 1)

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
```

```
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name_str='Posa_Muratura' + ' liv. ' +
str(IfcbuildingStorey[i_a][2])
            Name.text=Name_str
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_costr_muratura)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-h)
            PredecessorUID.text=PredecessorUID_str
            h=h+1
            p=p+1


        i_2=0
        lista_sup_isolante_pareti=[]
        for wall in walls:
            while i_2<len(walls):
                wall_i=walls[i_2]
                j=0
                while j<len(wall_i.IsDefinedBy):
                    IfcElementQuantity=wall_i.IsDefinedBy[j].RelatingPropertyD
efinition
                    if IfcElementQuantity[2]==str('Qto_WallBaseQuantities'):
                        k=0
                        Quantities_tot=IfcElementQuantity.Quantities
                        while k<len(Quantities_tot):
                            if
Quantities_tot[k].is_a('IfcPhysicalComplexQuantity'):
                                if Quantities_tot[k].Name==str('Isolamento
rigido'):
                                    l_m=0
                                    while l_m<len(Quantities_tot):
                                        if
Quantities_tot[l_m][0]==str('NetSideArea'):
                                            area=Quantities_tot[l_m][3]
                                            lista_sup_isolante_pareti.append(a
rea)
                                            k=k+1
                                            j=j+1
                                            l_m=l_m+1
                                        else:
```

```
                                        k=k+1
                                        l_m=l_m+1
                                else:
                                    k=k+1
                            else:
                                k=k+1
                        j=j+1
                        i_2=i_2+1
                    else:
                        j=j+1

        somma_sup_isolante_pareti=sum(lista_sup_isolante_pareti)

        if somma_sup_isolante_pareti>0:

            tempo_posa_isolante=round(somma_sup_isolante_pareti*0.12, 1)

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name_str='Posa_isolante_pareti_esterne' + ' liv. ' +
str(IfcbuildingStorey[i_a][2])
            Name.text=Name_str
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_posa_isolante)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1
            h=h+1


        slabs=[]
        for element in IfcElements:
            if element.is_a('IfcSlab'):
                slabs.append(element)

        lista_sup_massetto=[]
        if len(slabs)==1:
```

```
            slab_i=slabs[0]
            j=0
            while j<len(slab_i.IsDefinedBy):
                IfcElementQuantity=slab_i.IsDefinedBy[j].RelatingPropertyDefin
ition
                if IfcElementQuantity[2]==str('Qto_SlabBaseQuantities'):
                    k=0
                    Quantities_tot=IfcElementQuantity.Quantities
                    while k<len(Quantities_tot):
                        if
Quantities_tot[k].is_a('IfcPhysicalComplexQuantity'):
                            if Quantities_tot[k].Name==str('Calcestruzzo -
Leggero'):
                                l_m=0
                                while l_m<len(Quantities_tot):
                                    if
Quantities_tot[l_m][0]==str('GrossArea'):
                                        area=Quantities_tot[l_m][3]
                                        lista_sup_massetto.append(area)
                                        k=k+1
                                        j=j+1
                                        l_m=l_m+1
                                    else:
                                        k=k+1
                                        l_m=l_m+1
                            else:
                                k=k+1
                        else:
                            k=k+1
                    j=j+1
                else:
                    j=j+1

        somma_sup_massetto=sum(lista_sup_massetto)

        if somma_sup_massetto>0:

            tempo_getto_massetto=round(somma_sup_massetto*0.125, 1)
            tempo_maturazione=16

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
```

```
            Name_str='Getto_Massetto' + ' liv. ' +
str(IfcbuildingStorey[i_a][2])
            Name.text=Name_str
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_getto_massetto)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-2)
            PredecessorUID.text=PredecessorUID_str
            p=p+1
            h=h+1

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Asciugatura_massetto' + ' liv. ' +
str(IfcbuildingStorey[i_a][2])
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_maturazione)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1
            h=h+1


        lista_sup_isolante_pavimento=[]
        if len(slabs)==1:
            slab_i=slabs[0]
            j=0
            while j<len(slab_i.IsDefinedBy):
                IfcElementQuantity=slab_i.IsDefinedBy[j].RelatingPropertyDefin
ition
                if IfcElementQuantity[2]==str('Qto_SlabBaseQuantities'):
                    k=0
```

```
                    Quantities_tot=IfcElementQuantity.Quantities
                    while k<len(Quantities_tot):
                        if
Quantities_tot[k].is_a('IfcPhysicalComplexQuantity'):
                            if Quantities_tot[k].Name==str('Isolamento
rigido'):
                                l_m=0
                                while l_m<len(Quantities_tot):
                                    if
Quantities_tot[l_m][0]==str('GrossArea'):
                                        area=Quantities_tot[l_m][3]
                                        lista_sup_isolante_pavimento.append(ar
ea)
                                        k=k+1
                                        j=j+1
                                        l_m=l_m+1
                                    else:
                                        k=k+1
                                        l_m=l_m+1
                            else:
                                k=k+1
                        else:
                            k=k+1
                    j=j+1
                else:
                    j=j+1

        somma_sup_iso_pavimento=sum(lista_sup_isolante_pavimento)

        if somma_sup_iso_pavimento>0:

            tempo_posa_isolante=round(somma_sup_iso_pavimento*0.08, 1)

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name_str='Posa_isolante_pavimento' + ' liv. ' +
str(IfcbuildingStorey[i_a][2])
            Name.text=Name_str
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_posa_isolante)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
```

```
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1
            h=h+1
            t=t+0.5


        roofs=[]
        for element in IfcElements:
            if element.is_a('IfcRoof'):
                roofs.append(element)

        lista_sup_isolante_copertura=[]
        if len(roofs)==1:
            roof_i=roofs[0]
            j=0
            while j<len(roof_i.IsDefinedBy):
                Ifcpropertyset=roof_i.IsDefinedBy[j].RelatingPropertyDefinitio
n
                if Ifcpropertyset[2]==str('Quote'):
                    k=0
                    IfcPropertySingleValue=Ifcpropertyset.HasProperties
                    while k<len(IfcPropertySingleValue):
                        if IfcPropertySingleValue[k].Name==str('Area'):
                            Area_Tetto=IfcPropertySingleValue[k][2][0]
                            lista_sup_isolante_copertura.append(Area_Tetto)
                            k=k+1
                            j=j+1
                        else:
                            k=k+1
                else:
                    j=j+1

        somma_sup_isolante_copertura=sum(lista_sup_isolante_copertura)

        if somma_sup_isolante_copertura>0:

            tempo_posa_isolante=round(somma_sup_isolante_copertura*0.08, 1)

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
```

```
            Name=ET.SubElement(Task, 'ns0:Name')
            Name_str='Posa_isolante_copertura' + ' liv. ' +
str(IfcbuildingStorey[i_a][2])
            Name.text=Name_str
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_posa_isolante)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-round(t))
            PredecessorUID.text=PredecessorUID_str
            p=p+1
            h=h+1


        lista_sup_sottofondo=[]
        if len(slabs)==1:
            slab_i=slabs[0]
            j=0
            while j<len(slab_i.IsDefinedBy):
                IfcElementQuantity=slab_i.IsDefinedBy[j].RelatingPropertyDefin
ition
                if IfcElementQuantity[2]==str('Qto_SlabBaseQuantities'):
                    k=0
                    Quantities_tot=IfcElementQuantity.Quantities
                    while k<len(Quantities_tot):
                        if
Quantities_tot[k].is_a('IfcPhysicalComplexQuantity'):
                            if Quantities_tot[k].Name==str('Solaio,
sottofondo'):
                                l_m=0
                                while l_m<len(Quantities_tot):
                                    if
Quantities_tot[l_m][0]==str('GrossArea'):
                                        area=Quantities_tot[l_m][3]
                                        lista_sup_sottofondo.append(area)
                                        k=k+1
                                        j=j+1
                                        l_m=l_m+1
                                    else:
                                        k=k+1
                                        l_m=l_m+1
                            else:
                                k=k+1
                        else:
```

```
                            k=k+1
                    j=j+1
                else:
                    j=j+1

        somma_sup_sottofondo=sum(lista_sup_sottofondo)

        if somma_sup_sottofondo>0:

            tempo_getto_sottofondo=round(somma_sup_sottofondo*0.125, 1)
            tempo_maturazione=16

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name_str='Getto_Sottofondo' + ' liv. ' +
str(IfcbuildingStorey[i_a][2])
            Name.text=Name_str
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_getto_sottofondo)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1
            h=h+1

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Asciugatura_Sottofondo' + ' liv. ' +
str(IfcbuildingStorey[i_a][2])
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_maturazione)
```

```
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1
            h=h+1
            t=t+1


        lista_sup_sottofondo_copertura=[]
        if len(roofs)==1:
            roof_i=roofs[0]
            j=0
            while j<len(roof_i.IsDefinedBy):
                Ifcpropertyset=roof_i.IsDefinedBy[j].RelatingPropertyDefinitio
n
                if Ifcpropertyset[2]==str('Quote'):
                    k=0
                    IfcPropertySingleValue=Ifcpropertyset.HasProperties
                    while k<len(IfcPropertySingleValue):
                        if IfcPropertySingleValue[k].Name==str('Area'):
                            Area_Tetto=IfcPropertySingleValue[k][2][0]
                            lista_sup_sottofondo_copertura.append(Area_Tetto)
                            k=k+1
                            j=j+1
                        else:
                            k=k+1
                else:
                    j=j+1

        somma_sup_sottofondo_copertura=sum(lista_sup_sottofondo_copertura)

        if somma_sup_sottofondo_copertura>0:

            tempo_getto_sottofondo=round(somma_sup_sottofondo_copertura*0.125,
1)
            tempo_maturazione=16

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
```

```
            Name=ET.SubElement(Task, 'ns0:Name')
            Name_str='Getto_Sottofondo' + ' liv. ' +
str(IfcbuildingStorey[i_a][2])
            Name.text=Name_str
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_getto_sottofondo)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1
            h=h+1

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name.text='Asciugatura_Sottofondo' + ' liv. ' +
str(IfcbuildingStorey[i_a][2])
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_maturazione)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1
            h=h+1


        windows=[]
        for element in IfcElements:
            if element.is_a('IfcWindow'):
                windows.append(element)

        i_3=0
        lista_sup_finestre=[]
        for window in windows:
```

```
            while i_3<len(windows):
                window_i=windows[i_3]
                j=0
                while j<len(window_i.IsDefinedBy):
                    IfcElementQuantity=window_i.IsDefinedBy[j].RelatingPropert
yDefinition
                    if IfcElementQuantity[2]==str('Qto_WindowBaseQuantities'):
                        Area_finestra=IfcElementQuantity.Quantities[0].AreaVal
ue
                        lista_sup_finestre.append(Area_finestra)
                        j=j+1
                        i_3=i_3+1
                    else:
                        j=j+1

        somma_sup_finestre=sum(lista_sup_finestre)

        if somma_sup_finestre>0:

            tempo_montaggio_finestre=round(somma_sup_finestre*1.5, 2)

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            i_a_p1=i_a+1
            Name_str='Montaggio_finestre' + ' liv. ' +
str(IfcbuildingStorey[i_a][2])
            Name.text=Name_str
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_montaggio_finestre)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1
            h=h+1
            t=t+0.5

        i_4=0
        lista_sup_finitura_pareti_ext=[]
```

```
        for wall in walls:
            while i_4<len(walls):
                wall_i=walls[i_4]
                j=0
                while j<len(wall_i.IsDefinedBy):
                    IfcElementQuantity=wall_i.IsDefinedBy[j].RelatingPropertyD
efinition
                    if IfcElementQuantity[2]==str('Qto_WallBaseQuantities'):
                        k=0
                        Quantities_tot=IfcElementQuantity.Quantities
                        while k<len(Quantities_tot):
                            if
Quantities_tot[k].is_a('IfcPhysicalComplexQuantity'):
                                if Quantities_tot[k].Name==str('Finitura -
Gialla'):
                                    l_m=0
                                    while l_m<len(Quantities_tot):
                                        if
Quantities_tot[l_m][0]==str('NetSideArea'):
                                            area=Quantities_tot[l_m][3]
                                            lista_sup_finitura_pareti_ext.appe
nd(area)
                                            k=k+1
                                            j=j+1
                                            l_m=l_m+1
                                        else:
                                            k=k+1
                                            l_m=l_m+1
                                else:
                                    k=k+1
                            else:
                                k=k+1
                        j=j+1
                        i_4=i_4+1
                    else:
                        j=j+1

        somma_sup_finitura_pareti_ext=sum(lista_sup_finitura_pareti_ext)

        if somma_sup_finitura_pareti_ext>0:

            tempo_finitura_pareti_ext=round(somma_sup_finitura_pareti_ext*0.28
, 1)

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
```

```
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name_str='Finitura_pareti_esterne' + ' liv. ' +
str(IfcbuildingStorey[i_a][2])
            Name.text=Name_str
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_finitura_pareti_ext)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1
            h=h+1
            t=t+0.5

        i_5=0
        lista_sup_finitura_pareti_int=[]
        for wall in walls:
            while i_5<len(walls):
                wall_i=walls[i_5]
                j=0
                while j<len(wall_i.IsDefinedBy):
                    IfcElementQuantity=wall_i.IsDefinedBy[j].RelatingPropertyD
efinition
                    if IfcElementQuantity[2]==str('Qto_WallBaseQuantities'):
                        k=0
                        Quantities_tot=IfcElementQuantity.Quantities
                        while k<len(Quantities_tot):
                            if
Quantities_tot[k].is_a('IfcPhysicalComplexQuantity'):
                                if Quantities_tot[k].Name==str('Finitura -
Bianca'):
                                    l_m=0
                                    while l_m<len(Quantities_tot):
                                        if
Quantities_tot[l_m][0]==str('NetSideArea'):
                                            area=Quantities_tot[l_m][3]
                                            lista_sup_finitura_pareti_int.appe
nd(area)
                                            k=k+1
                                            j=j+1
                                            l_m=l_m+1
```

```
                                        else:
                                            k=k+1
                                            l_m=l_m+1
                                    else:
                                        k=k+1
                                else:
                                    k=k+1
                            j=j+1
                            i_5=i_5+1
                        else:
                            j=j+1

        somma_sup_finitura_pareti_int=sum(lista_sup_finitura_pareti_int)

        if somma_sup_finitura_pareti_int>0:

            tempo_finitura_pareti_int=round(somma_sup_finitura_pareti_int*0.28
, 1)

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name_str='Finitura_pareti_interne' + ' liv. ' +
str(IfcbuildingStorey[i_a][2])
            Name.text=Name_str
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_finitura_pareti_int)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-2)
            PredecessorUID.text=PredecessorUID_str
            p=p+1
            h=h+1
            t=t+0.5

        lista_sup_finitura_pavimento=[]
        if len(slabs)==1:
            slab_i=slabs[0]
            j=0
```

```
            while j<len(slab_i.IsDefinedBy):
                IfcElementQuantity=slab_i.IsDefinedBy[j].RelatingPropertyDefin
ition
                if IfcElementQuantity[2]==str('Qto_SlabBaseQuantities'):
                    k=0
                    Quantities_tot=IfcElementQuantity.Quantities
                    while k<len(Quantities_tot):
                        if
Quantities_tot[k].is_a('IfcPhysicalComplexQuantity'):
                            if Quantities_tot[k].Name==str('Piastrella di
ceramica'):
                                l_m=0
                                while l_m<len(Quantities_tot):
                                    if
Quantities_tot[l_m][0]==str('GrossArea'):
                                        area=Quantities_tot[l_m][3]
                                        lista_sup_finitura_pavimento.append(ar
ea)
                                        k=k+1
                                        j=j+1
                                        l_m=l_m+1
                                    else:
                                        k=k+1
                                        l_m=l_m+1
                            else:
                                k=k+1
                        else:
                            k=k+1
                    j=j+1
                else:
                    j=j+1

        somma_sup_finitura_pavimento=sum(lista_sup_finitura_pavimento)

        if somma_sup_finitura_pavimento>0:

            tempo_finitura_pavimento=round(somma_sup_finitura_pavimento*0.35,
1)

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name_str='Finitura_pavimento' + ' liv. ' +
str(IfcbuildingStorey[i_a][2])
```

```
            Name.text=Name_str
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_finitura_pavimento)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1
            h=h+1
            t=t+0.5


        doors=[]
        for element in IfcElements:
            if element.is_a('IfcDoor'):
                doors.append(element)

        lista_sup_porte=[]
        if len(doors)==1:
            door_i=doors[0]
            j=0
            while j<len(door_i.IsDefinedBy):
                IfcElementQuantity=door_i.IsDefinedBy[j].RelatingPropertyDefin
ition
                if IfcElementQuantity[2]==str('Qto_DoorBaseQuantities'):
                    Area_porta=IfcElementQuantity.Quantities[0].AreaValue
                    lista_sup_porte.append(Area_porta)
                    j=j+1
                else:
                    j=j+1

        somma_sup_porte=sum(lista_sup_porte)

        if somma_sup_porte>0:

            tempo_montaggio_porte=round(somma_sup_porte*1.5, 2)

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
```

```
            i_a_p1=i_a+1
            Name_str='Montaggio_porta' + ' liv. ' +
str(IfcbuildingStorey[i_a][2])
            Name.text=Name_str
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_montaggio_porte)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-4)
            PredecessorUID.text=PredecessorUID_str
            p=p+1
            h=h+1


        stairs=[]
        for element in IfcElements:
            if element.is_a('IfcStair'):
                stairs.append(element)

        if stairs!=[]:

            tempo_montaggio_scala=16

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            i_a_p1=i_a+1
            Name_str='Montaggio_Scala' + ' liv. ' +
str(IfcbuildingStorey[i_a][2]) + '/' + str(IfcbuildingStorey[i_a_p1][2])
            Name.text=Name_str
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_montaggio_scala)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-6)
            PredecessorUID.text=PredecessorUID_str
```

```
            p=p+1
            h=h+1

        lista_sup_finitura_copertura=[]
        if len(roofs)==1:
            roof_i=roofs[0]
            j=0
            while j<len(roof_i.IsDefinedBy):
                Ifcpropertyset=roof_i.IsDefinedBy[j].RelatingPropertyDefinitio
n
                if Ifcpropertyset[2]==str('Quote'):
                    k=0
                    IfcPropertySingleValue=Ifcpropertyset.HasProperties
                    while k<len(IfcPropertySingleValue):
                        if IfcPropertySingleValue[k].Name==str('Area'):
                            Area_Tetto=IfcPropertySingleValue[k][2][0]
                            lista_sup_finitura_copertura.append(Area_Tetto)
                            k=k+1
                            j=j+1
                        else:
                            k=k+1
                else:
                    j=j+1

        somma_sup_finitura_copertura=sum(lista_sup_finitura_copertura)

        if somma_sup_finitura_copertura>0:

            tempo_posa_finitura=round(somma_sup_finitura_copertura*0.25, 1)

            Task=ET.SubElement(root[54], 'ns0:Task')
            Uid=ET.SubElement(Task, 'ns0:UID')
            Uid_p=str(p)
            Uid.text=Uid_p
            Id=ET.SubElement(Task, 'ns0:ID')
            Id.text=Uid_p
            Name=ET.SubElement(Task, 'ns0:Name')
            Name_str='Posa_finitura' + ' liv. ' +
str(IfcbuildingStorey[i_a][2])
            Name.text=Name_str
            Duration=ET.SubElement(Task, 'ns0:Duration')
            frazionaria,intera = modf(tempo_posa_finitura)
            Duration_str='PT'+str(round(intera))+'H'+str(round((round(fraziona
ria, 2)*60)))+'M0S'
            Duration.text=Duration_str
            PredecessorLink=ET.SubElement(Task, 'ns0:PredecessorLink')
```

```
            PredecessorUID=ET.SubElement(PredecessorLink,
'ns0:PredecessorUID')
            PredecessorUID_str=str(p-1)
            PredecessorUID.text=PredecessorUID_str
            p=p+1
            h=h+1


        i_a=i_a+1
    else:
        i_a=i_a+1


tree.write('cronoprogramma.xml')
```

BIBLIOGRAFIA

Adhox. (2022). *Il formato IFC: cos'è? A cosa serve nel metodo BIM?* Adhox.It. https://adhox.it/formato-ifc/

Amer, F., & Golparvar-Fard, M. (2021). Modeling dynamic construction work template from existing scheduling records via sequential machine learning. *Advanced Engineering Informatics*, *47*. https://doi.org/10.1016/j.aei.2020.101198

Amer, F., Koh, H. Y., & Golparvar-Fard, M. (2021). Automated Methods and Systems for Construction Planning and Scheduling: Critical Review of Three Decades of Research. *Journal of Construction Engineering and Management*, *147*(7). https://doi.org/10.1061/(asce)co.1943-7862.0002093

Angelotti Andrea. (2021). *SVILUPPO DI UNA PROCEDURA DI PROGRAMMAZIONE E VISUALIZZAZIONE DEI LAVORI DI COSTRUZIONE CON ANALISI DEGLI SCOSTAMENTI NELLA FASE DI ESECUZIONE*. UNIVERSITA' POLITECNICA DELLE MARCHE.

Berchet. (2016). *Cos'è un modello BIM*. Ccberchet.It. https://www.ccberchet.it/cos-e-un-modello-bim/

BibLus-BIM. (2021). *BIM 4D, cos'è e a cosa serve*. Bim.Acca.It. https://bim.acca.it/modello-bim-4d-cose-e-a-cosa-serve/

BibLus-net. (2022). *Obbligo BIM appalti pubblici: dal 2022 al 2025* . Bim.Acca.It. https://bim.acca.it/obbligo-bim-appalti-pubblici-dal-2022-al-2025/

buildingSMART. (2017). *Industry Foundation Classes 4.0.2.1 Version 4.0 - Addendum 2 - Technical Corrigendum 1*. Standards.Buildingsmart.Org. https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/

Desgagné-Lebeuf, A., Lehoux, N., Beauregard, R., & Desgagné-Lebeuf, G. (2019). Computer-assisted scheduling tools in the construction industry: A systematic literature review. *IFAC-PapersOnLine*, *52*(13), 1843–1848. https://doi.org/10.1016/j.ifacol.2019.11.470

Downloads - IfcOpenShell . (n.d.). Retrieved June 23, 2023, from https://ifcopenshell.org/downloads.html

Faghihi, V., Reinschmidt, K. F., & Kang, J. H. (2014). Construction scheduling using Genetic Algorithm based on Building Information Model. *Expert Systems with Applications*, *41*(16), 7565–7578. https://doi.org/10.1016/j.eswa.2014.05.047

Feng, C. W., Chen, Y. J., & Huang, J. R. (2010). Using the MD CAD model to develop the time-cost integrated schedule for construction projects. *Automation in Construction*, *19*(3), 347–356. https://doi.org/10.1016/j.autcon.2009.12.009

Fuccelli, A., & Fabio, M. (2017). *PIANIFICAZIONE OPERATIVA 4D DEL CANTIERE DELLO STADIO SAN PAOLO MEDIANTE METODOLOGIA BIM*. https://webthesis.biblio.polito.it/secure/9152/1/tesi.pdf

Furcolo Nicola. (2016). *IFC, cos'è: caratteristiche, vantaggi ed importanza della certificazione IFC*. BibLus.Acca.It. https://biblus.acca.it/focus/ifc-cose-e-quali-sono-i-vantaggi/

Hatori, F., Satou, K., Onodera, J., & Yashiro, Y. (2021). Development of BIM-Based 4D Simulation System for Construction Schedule Planning. In *Lecture Notes in Civil Engineering* (Vol. 98, pp. 547–560). Springer. https://doi.org/10.1007/978-3-030-51295-8_39

IBIMI. (n.d.). *IFC: cos'è? e come è fatto?* Ibimi.It. Retrieved May 24, 2023, from https://www.ibimi.it/ifc-cose-e-come-e-fatto/

Ivson, P., Nascimento, D., Celes, W., & Barbosa, S. D. J. (2018). CasCADe: A Novel 4D Visualization System for Virtual Construction Planning. *IEEE Transactions on Visualization and Computer Graphics*, *24*(1), 687–697. https://doi.org/10.1109/TVCG.2017.2745105

Kim, H., Anderson, K., Lee, S., & Hildreth, J. (2013). Generating construction schedules through automatic data extraction using open BIM (building information modeling) technology. *Automation in Construction*, *35*, 285–295. https://doi.org/10.1016/j.autcon.2013.05.020

Kim, K., & Cho, Y. K. (2015). Construction-specific spatial information reasoning in Building Information Models. *Advanced Engineering Informatics*, *29*(4), 1013–1027. https://doi.org/10.1016/j.aei.2015.08.004

König, M., Koch, C., Habenicht, I., & Spieckermann, S. (2012). Intelligent BIM-based construction scheduling using discrete event simulation. *Proceedings - Winter Simulation Conference*. https://doi.org/10.1109/WSC.2012.6465232

Liu, H., Al-Hussein, M., & Lu, M. (2015). BIM-based integrated approach for detailed construction scheduling under resource constraints. *Automation in Construction*, *53*, 29–43. https://doi.org/10.1016/j.autcon.2015.03.008

Luigi Ottaiano. (2016). PROJECT MANAGEMENT - Cronoprogramma dei lavori. In *CeSAF* . www.maestridellavoro.it

Mecos. (2020). *Armatura metallica e disarmo delle casseforme del calcestruzzo armato (cemento armato) - Mecos s.r.l. - Impresa edile e di servizi a Palermo*. https://www.mecosimmobiliare.it/2020/03/23/armatura-metallica-e-disarmo-delle-casseforme-del-calcestruzzo-armato-cemento-armato/

Namirial. (2022). *Che cos'è il BIM e chi dovrebbe usarlo*. Focus.Namirial.It. https://focus.namirial.it/bim/

Paolo Borin, & Carlo Zanchetta. (2020). *IFC - Processi e modelli digitali openBIM per l'ambiente costruito* (Maggioli Editore, Ed.).

Pedago. (2021). *Revit: Cos'è e a cosa serve il famoso software di Autodesk*. Pedago.It.

https://www.pedago.it/blog/revit-cos-e-cosa-serve.htm

Prezzario Regione Veneto. (2023). Regione Veneto. https://prezziario.regione.veneto.it/

Python. (n.d.). Retrieved June 23, 2023, from www.python.it

Song, S., Yang, J., & Kim, N. (2012). Development of a BIM-based structural framework optimization and simulation system for building construction. *Computers in Industry*, *63*(9), 895–912. https://doi.org/10.1016/j.compind.2012.08.013

Wang, W. C., Weng, S. W., Wang, S. H., & Chen, C. Y. (2014). Integrating building information models with construction process simulations for project scheduling support. *Automation in Construction*, *37*, 68–80. https://doi.org/10.1016/j.autcon.2013.10.009

Wikipedia. (2023). *Visual Studio Code*. It.Wikipedia.Org. https://it.wikipedia.org/wiki/Visual_Studio_Code

INDICE DELLE TABELLE

INDICE ICONOGRAFICO