POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Edile
Tesi di Laurea Magistrale

BIM Validation per il controllo informativo con la programmazione visuale

Relatore

Anna Osello
Matteo Del Giudice

Candidato

Stefano Venturini

Anno Accademico 2020/2021

Indice

Abstract

La tesi presenta anzitutto lavoro di modellazione su sulla parte architettonica, strutturale e antincendio nella sezione degli Interrati della Torre della Regione Piemonte, mostrando le problematiche affrontate e le soluzioni adottate per la modellazione geometrica e per la rappresentazione grafica in modo da ottenere un lavoro coerente con il resto dell'intervento.
Si descrive il contenuto informativo e la classificazione degli oggetti all'interno del BIM model basata su una documentazione fornita dalla stazione appaltante.
Successivamente vengono descritte le caratteristiche della fase di BIM Validation, contestualizzandole all'interno del discorso più ampio riguardante il BIM Model Checking.
In ultimo, sono descritte le strategie di BIM Validation che sono state attuate operativamente all'interno del caso studio degli Interrati della Torre. Tali soluzioni hanno previsto l'utilizzo della programmazione visiva di Dynamo, la quale si è resa utile per creare script che, sulla base di Regular Expression (RegEx) e porte logiche, cercassero errori all'interno della compilazione dei parametri, per poi evidenziarli, contarli e fornirne un output grafico sotto forma di grafici a torta all'interno di una Dashboard.
Sono infine stati creati script per correggere e compilare automaticamente, grandi quantità di dati.

This thesis shows the work of modelling the architectural, structural and fire-safety part of underground part of Torre della Regione Piemonte, presenting the faced issues and used solutions in geometrical modelling and graphic rapresentation, in order to obtain an object consistent with the rest of project.
Then, it describes the information content and the classification of object within the BIM model, based on the documentation given by contracting authority.
Then are described the characteristics of BIM Validation, giving the wider discourse regarding BIM Model Checking.
After that, BIM Validation strategies adopted within the case study of Underground Part of Tower are presented.
These solutions have required the use of visual programming of Dynamo, which is useful to create scripts based on Regular Expressions (RegEx) and logic doors in order to find errors in compilation of parameters, to point out and count them and to give a graphic output in pie charts inside a general Dashboard.
Finally, scripts to correct and comply automatically large amounts of data are created.

Metodologia BIM: Vantaggi e svantaggi

Il BIM, nell'ambito delle costruzioni, è una **metodologia di lavoro che mira alla collaborazione ed allo scambio di informazioni**. Tale metodologia si basa su dei software attraverso i quali creare un modello virtuale dell'edificio, non solo grafico e spaziale ma anche informativo.
Un modello BIM permette una visualizzazione 3d dell'edificio (rispetto ad una tradizionalmente bidimensionale tipica si software CAD) in ogni sua parte, in un ambiente di lavoro predisposto all'architettura (in questo è evidente il vantaggio rispetto ad un qualsiasi software di modellazione 3d non dedicato).
Tale modello, poi consente l'archiviazione di un gran numero di informazioni, in maniera ordinata e logica; si può affermare anzi che un modello BIM è un modello geometrico al quale "appendere" informazioni, informazioni rappresentate proprio dalla lettera I nell'acronimo BIM, ovvero **Building Information Model**.
Le informazioni archiviate nel modello, potranno poi essere gestite in vari modi, sia all'interno del software nel quale sono state inserite, sia in altri programmi comunicanti, spesso attraverso il formato non proprietario **.ifc**. Si parla, in questo caso di interoperabilità tra software, ovvero di comunicazione tra programmi diversi nell'ottica della riduzione dei tempi di lavoro e di una maggiore qualità.
L'**interoperabilità** è una delle basi su cui si fonda la filosofia del BIM, ovvero la comunicazione tra software; un altro pilastro è poi la **comunicazione**, questa volta, tra professionisti diversi e spesso di diverse aree di competenza.
Un modello BIM si pone in questo senso, come un **aggregatore di progetti** in un unico modello in cui ogni soggetto ha la possibilità di avere un punto della situazione globale ed in tempo reale dell'edificio in via di sviluppo. Tale condivisione del lavoro può avvenire attraverso il metodo dei link oppure attraverso il metodo dei workset; si parlerà

Immagine 01: Logo del formato .ifc

.ifc è l'acronimo che sta per "**Industrial Foundation Class**

più avanti della differenza tra questi due metodi, ma il concetto generale è che il BIM porta ad un flusso di informazioni più ordinato, più snello e contemporaneamente più completo, capace di ridurre errori in fase di progettazione (e, conseguentemente, tempo e denaro).
Tra gli innumerevoli vantaggi del BIM vanno citati:

- Un ottimo controllo sulle varianti, in quanto è sufficiente modificare, ad esempio, la posizione di un oggetto sul modello per aggiornare automaticamente ed in una manciata di secondo, tutte le viste del progetto per tutti i progettisti che hanno accesso al modello.
- Una gestione del lavoro centralizzata, che permette di avere sott'occhio, se non tutti, molti degli elaborati redatti.
- La possibilità, in una fase di modellazione precedente a quella della costruzione, di avere anche una **programmazione cronologica**, la quale potrà poi manifestarsi in un cronoprogramma.
- **Verifica delle interferenze** già in fase progettuale
- **Controllo dei costi** prima dell'effettiva redazione di un computo metrico estimativo finale, controllo che avviene dinamicamente, aggiornandosi ad ogni aggiornamento del modello
- Diretto **conteggio di elementi** ed aree, anch'esso **dinamico**.
- L'**utilizzo di un solo modello durante tutta la vita dell'edificio**; un modello abbozzato utilizzato nelle fasi di progettazione preliminare può infatti essere affinato, riempito di dettagli, in modo da diventare adatto per un progetto esecutivo ed infine, utilizzato dal proprietario della costruzione (o dal facility manager) per la manutenzione, in un arco di tempo che può, in certi casi, arrivare oltre alla vita stessa dell'edificio.
  Essendo il BIM un concetto piuttosto recente, non si hanno in realtà molte testimonianze di modelli di edifici demoliti, ma è facile pensare un futuro dove archivi edilizi conterranno dati di costruzioni demolite, in formato BIM).

Volendo dunque sintetizzare in pochissime parole i principi di base di tale metodologia, li si può racchiudere nelle parole ***informazione***, ***comunicazione*** e ***coerenza***

Immagine 02: Confronto tra flussi di lavoro tradizionali e flussi basati sulla condivisione BIM

Svantaggi

Nonostante questi enormi vantaggi, la metodologia BIM presenta ancora alcuni problemi: in primo luogo, la **difficoltà di apprendimento** piuttosto elevata se paragonata a quella di sistemi più tradizionali. Tale difficoltà si declina nella necessità di spendere parecchie ore di formazione, ore che dovranno necessariamente essere sottratte all'attività lavorativa; da tale considerazione, si comprende l'importanza di formazione a questa metodologia all'interno di un percorso universitario.

La licenza annuale di AutoCAD costa circa 1800€.

Una licenza annuale di Revit costa invece circa 2600€

In secondo luogo, i **costi piuttosto alti di un software** come Revit rendono antieconomico il passaggio alla metodologia BIM, soprattutto per piccoli progetti e considerando che tale spesa, va solitamente affiancata ad una licenza di un software CAD.

In ultimo, sebbene il file .ifc sia stato creato proprio per fare dialogare tra loro diversi software BIM, esso non è ancora privo di difetti e può creare problemi; d'altro canto, non è presente sul mercato un software BIM molto più utilizzato rispetto alla concorrenza (come invece è AutoCAD per i software di Computer Aided Design) ed il rischio per uno studio di progettazione è quello di avere un modello che, sostanzialmente, dialoga con se stesso.

Va notato di come questi svantaggi, escluso forse il costo elevato, verranno superati dal cambio di paradigma che

sta avvenendo nel settore delle costruzioni. Lo stesso costo poi, andrebbe poi considerato come un investimento da fare per evitare successivi problemi in cantiere economicamente più onerosi.

Software BIM

BIM, come già spiegato, non è un software in particolare ma, piuttosto, una metodologia di lavoro che si basa su alcuni software, i più famosi dei quali sono Revit della Autodesk, ArchiCAD della Graphisoft e, soprattutto in Italia, Edificius della Acca Software, assieme a tutta la suite Acca. In questa tesi è stato usato unicamente Autodesk Revit 2019.

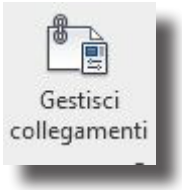


Immagine 03: Simbolo Gestisci Collegamenti

Link e Workset

Come si è detto in apertura, uno degli ambiti in cui si notano i vantaggi della metodologia BIM, è la gestione del lavoro svolto da diverse figure professionali. Diversi progettisti, infatti, possono creare parti del modello che dialogheranno con le restanti parti. Si immagini ad esempio un architetto modellare le partizioni in laterizio di un modello architettonico che ha come "scheletro", un modello strutturale modellato dall'ingegnere strutturista.
Su questi due modelli poi, un ingegnere impiantista potrà modellare, ad esempio, un sistema di spegnimento incendi automatico, sapendo in quali punti è possibile far passare tubazioni ed in quali no. La condivisione del lavoro avviene in tempo reale, ovvero, in ogni momento si possono confrontare tutti i modelli nella loro ultima versione.
Alla luce di queste considerazioni, il processo che porta un professionista a contattare un altro professionista per farsi inviare una versione, tra l'altro non definitiva, di una parte sola del lavoro, con il rischio di accumulare molti file disordinati con impaginazioni differenti, appare decisamente obsoleta.

Immagine 04: Logo di ArchiCAD

Immagine 05: Logo di ACCA Software

Revit permette la suddivisione del lavoro tramite 2 modalità, ovvero quella che fa uso dei file linkati e quella che invece utilizza i workset.
Per collaborazione tramite **Link** si intende la **creazione di modelli differenti**, sovrapposti tramite il posizionamento in uno spazio 3d secondo delle coordinate condivise.
L'approccio è lo stesso che si ha con i riferimenti esterni di Autocad.

Ogni professionista lavorerà dunque su un proprio file, sul quale avrà importato anche i file dei colleghi. Questi **file linkati possono essere visualizzati, interrogati** per avere le informazioni in essi contenuti, **ma non modificati**. Le operazioni di importazione e di aggiornamento, su Revit vengono svolte nel pannello "**Gestisci collegamenti**", lo stesso, tra l'altro, dal quale è possibile gestire file .dwg importati.

Con la metodologia dei **Workset** si prevede invece la creazione di un file centrale nel quale convergono diversi file locali, appartenenti a diversi componenti di un team. Su ogni file locale viene svolta una parte del lavoro per poi "riversare" le nuove modifiche sul modello centrale dal quale potranno poi essere viste dagli altri workset.
Molto sinteticamente, si può affermare che procedere per link equivale a creare modelli separati che vengono poi uniti mentre la strada dei workset equivale a prendere un modello centrale onnicomprensivo e a staccare varie parti da esso. L'approccio tramite link, inoltre è meno centralizzato: ogni modello dialoga con tutti gli altri separatamente e non fa invece riferimento ad un modello centrale unico, come avviene coi workset.
Non è detto che un approccio escluda l'altro ed infatti non è raro lavorare su progetti che fanno uso di entrambe le metodologie per dividere, ad esempio, con workset, gli oggetti di varie discipline all'interno di un edificio, edificio facente parte di un complesso al quale appartengono altri edifici linkati.

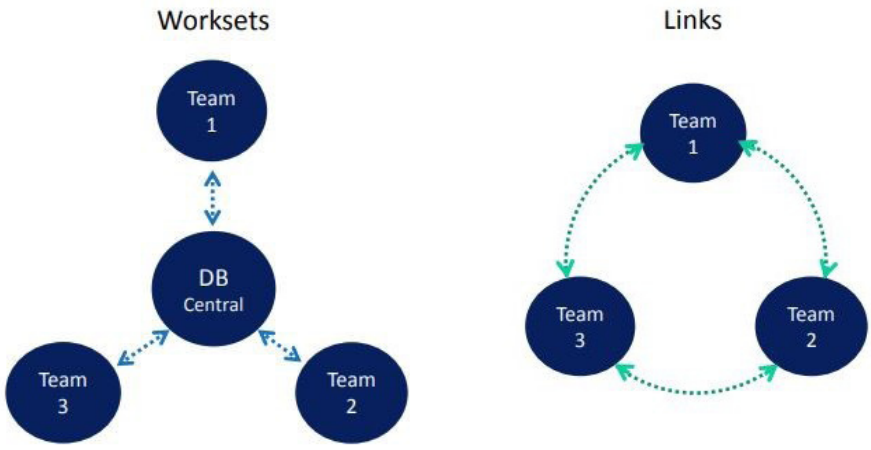


Immagine 06: Schema metodologia Workset confrontata con quella Link

Secondo esperienza personale di chi scrive, la metodologia workset permette una maggiore sinergia nel lavoro di un team e si rivolge soprattutto a studi che lavorano con rete locale, mentre la metodologia link si rivolge di più a studi differenti che devono cooperare a distanza.
La metodologia link, pagando lo scotto di creare modelli meno coesi (di fatto si tratta di modelli che si compenetrano senza interagire), ha il vantaggio di essere più "robusta": un modello linkato pieno di problemi proveniente da uno studio, non avrà ripercussioni sul lavoro che si sta svolgendo in un altro studio. A questo, si aggiungono eventuali problemi di connessione che potrebbero rendere impossibile lavorare per workset, a distanza, efficacemente.

Breve Storia del BIM

Per secoli, la progettazione architettonica si è basata su carta, squadre e matite. L'introduzione dei tecnigrafi ha sostanzialmente reso più comodo realizzare tavole di disegno tecnico senza tuttavia risolvere gli enormi problemi insiti nel disegno manuale.
Tralasciando alcuni episodi embrionali, fu negli anni 70 del 1900 che si diffusero le prime applicazioni commerciali del CAD, soprattutto in ambito meccanico, navale ed architettonico.
Nel **1982** esce sul mercato **Autodesk AutoCAD**, il primo software commerciale di CAD a funzionare su PC e non su mainframe o mini-computer. Questo ha portato ad un'ulteriore diffusione di questa metodologia, a mano a mano, in tutti gli studi di progettazione, diffusione che vede il proprio compimento soprattutto negli anni 90 del secolo scorso.
Parallelamente alla diffusione del CAD, nel **1975**, Charles M. Eastman pubblica un articolo chiamato "**The Use of Computers instead o Drawing in Design**". In tale articolo, Eastman ragiona su quali sarebbero stati i benefici di avere un modello unico dal quale derivare contemporaneamente tutte le viste, in modo che esse siano coerenti tra loro. Si tratta tuttavia ancora di uno stato germinale del BIM dove è ancora assente la componente dell'informazione.
Questo ulteriore passo lo si deve a **Robert Aish**: negli anni 80 fece parte dello sviluppo di **RUCAPS** (Riyadh University Computer-Aided Production System), la prima vera applicazione BIM. Fu il primo a teorizzarne i vantaggi anche in ambito informativo.
Il primo programma commerciale (ed oggi ancora ampiamente utilizzato, seppur con numerose modifiche ed implementazioni) a vedere la luce fu **ArchiCAD**, lanciato sul mercato dalla ungherese Graphisoft nel **1987**.
Revit seguì 10 anni dopo, sviluppato dalla Charles River Software, poi rinominata **Revit Technology Coporation** ed acquistata nel 2002 dalla Autodesk.
Nel **2004** fu introdotto il concetto di **collaborative working** di cui si è parlato nel primo capitolo e, negli anni seguenti, si è vista la progressiva trasformazione del processo BIM con il miglioramento nella gestione dei parametri e, ad

Immagine 07: Studio con tecnigrafi

Immagine 08: Primo logo di AutoCAD

Robert Aish: professore di Design Computation alla Barlett School di Architettura di Londra.
Oggi si occupa di Design Computation e di applicativi quali Dynamo Revit.

esempio, la possibilità di importare nuvole di punti ottenute

Immagine 09: Schema di avvio della prima versione di Revit

da rilievi con laser scanner, tanto che nel **2016** il **Regno Unito decide di rendere l'utilizzo del BIM obbligatorio per tutti i progetti pubblici**.

Da quell'anno, altri paesi hanno seguito il Regno Unito rendendo obbligatorio l'utilizzo di questa metodologia per certe tipologie di lavori pubblici o a partire da un certo importo; la Germania, ad esempio, lo ha fatto nell'ambito dei trasporti.

In Italia, l'introduzione del BIM per le opere pubbliche avverrà gradualmente nei prossimi anni, in base all'importo dei lavori. Ad oggi (2020) è previsto per opere superiori ai 50M di euro, ma tale limite si abbasserà ad 1M già nel 2023, per poi venir completamente eliminato nel 2025.

Si riporta di seguito l'articolo 6 del **Decreto Ministeriale n.560 del 1 dicembre 2017**

Art. 6

(Tempi di introduzione obbligatoria dei metodi e strumenti elettronici di modellazione per l'edilizia e le infrastrutture)

1. Le stazioni appaltanti richiedono, in via obbligatoria, l'uso dei metodi e degli strumenti elettronici di cui all'articolo 23, comma 1, lettera h), del codice dei contratti pubblici secondo la seguente tempistica:

a) per i lavori complessi relativi a opere di importo a base di gara pari o superiore a 100 milioni di euro, a decorrere dal 1° gennaio 2019;

b) per i lavori complessi relativi a opere di importo a base di gara pari o superiore a 50 milioni di euro a decorrere dal 1° gennaio 2020;

c) per i lavori complessi relativi a opere di importo a base di gara pari o superiore a 15 milioni di euro a decorrere dal 1° gennaio 2021;

d) per le opere di importo a base di gara pari o superiore alla soglia di cui all'articolo 35 del codice dei contratti pubblici, a decorrere dal 1° gennaio 2022;

e) per le opere di importo a base di gara pari o superiore a 1 milione di euro, a decorrere dal 1° gennaio 2023;

f) per le opere di importo a base di gara inferiore a 1 milione di euro, a decorrere dal 1° gennaio 2025.

Immagine 10: Art. 6 del DM 560/2017

Il cuore di Autodesk Revit

Sebbene questa tesi non si ponga come obiettivo quello di essere un manuale del software, è utile chiarire alcuni concetti alla base del funzionamento del programma.

Un modello Revit è sostanzialmente **un database**; tutti gli elementi, dagli oggetti architettonici a quelli di documentazione si chiamano **Famiglie**. Una famiglia viene definita come un "*gruppo di elementi con un insieme di proprietà comuni denominate parametri ed una rappresentazione grafica associata*".Si riporta di seguito un esempio, dal manuale Autodesk Revit per l'Architettura di Simone Pozzoli, Marco Bonazza e Werner Stefano Villa.
Si pensi ad un tavolo e si ponga che sia caratterizzato dalle dimensioni (larghezza-profondità-altezza), da un determinato materiale, un prezzo e cosi via.
Queste proprietà del tavolo potranno anche assumere valori differenti, per esempio in funzione del modello, ma l'oggetto, nella sostanza, rimarrà sempre un tavolo e non diventerà, per assurdo, una sedia solo perché ne si aumenta la lunghezza

Immagine 11: Revit 2019 per L'architettura: guida completa per la progettazione BIM

Le famiglie sono i pezzi con cui si costruisce un edificio e la relativa documentazione su Revit.

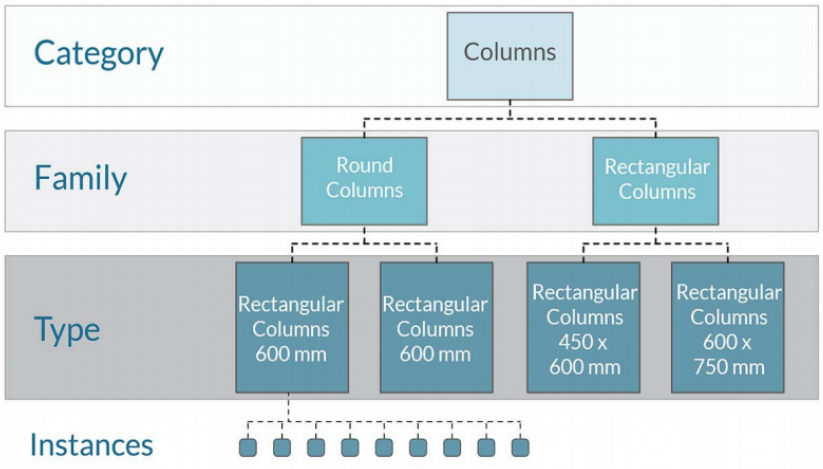


Immagine 12: Gerarchia degli elementi

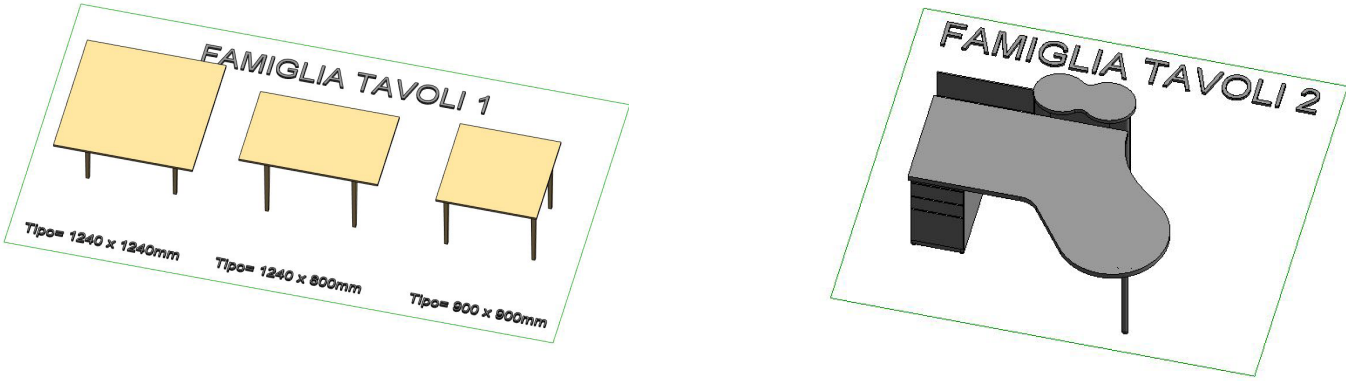


Immagine 13: Esempio della divisione tra famiglie e tipi

Famiglie di Sistema, Caricabili e Locali

Revit struttura l'organizzazione delle famiglie in tre tipologie principali.

- Famiglie di Sistema
- Famiglie Caricabili
- Famiglie Locali

Famiglie di sistema

Le famiglie di sistema sono quelle famiglie che compongono lo scheletro del modello. Si tratta in particolare dei **muri**, dei **tetti**, dei **pavimenti**, delle **scale** e **rampe** dei **condotti** e delle **tubazioni**. Queste famiglie sono sempre presenti all'interno del modello e non possono essere scaricate da esso, né è possibile caricarne di nuove. Un'altra importante particolarità delle famiglie di sistema è il fatto di possedere un **nome non modificabile dall'utente** (al più, ad essere modificato, sarà il nome del tipo).

Esempio:
Capita, in un progetto, di avere diversi tipi di muro corrispondenti a varie stratigrafie: si tratta appunto di diversi tipi della stessa famiglia "Muro" e non di diverse famiglie.

Famiglie caricabili

Le famiglie caricabili sono, banalmente, **tutte le famiglie non di sistema**. Si tratta di famiglie che è **possibile creare nell'editor famiglie di Revit** e importare nel modello. Presentano molti meno vincoli delle famiglie di sistema, tra cui il fatto di avere i **nomi delle famiglie modificabili** a piacimento. Tra le famiglie di sistema che avranno

maggiore peso all'interno della tesi, si citano le porte, gli estintori e le griglie di aerazione.

Famiglie locali

Si tratta di famiglie che vengono **create direttamente all'interno del modello** (dunque non caricate dopo averle modellate nell'editor di famiglie) e ad esso legate. Non è possibile posizionare una famiglia di sistema su più modelli separati.
Non sono presenti famiglie locali in nessuno dei modelli del complesso della Torre della Regione Piemonte; le si vuole citare in questo elenco solo per completezza.

Categorie

Tutte le famiglie di Revit, siano esse caricabili o di sistema, si raggruppano in categorie. **Ogni categoria raggruppa tutte le famiglie che possiedono una determinata funzione**. La categoria "porte", ad esempio, comprenderà tutte le porte caricate nel modello, siano esse metalliche REI 120 che semplici porte tamburate. E' possibile cambiare la categoria di appartenenza di una famiglia caricabile dall'editor delle famiglie; non è invece possibile fare lo stesso con le famiglie di sistema in quanto impossibili da aprire in tale editor.

Parametri Condivisi

Si è detto in precedenza che i parametri in Revit possono essere distinti in parametri di tipo ed in parametri di istanza. Esiste un'altra suddivisione, ovvero quella tra parametri di famiglia e parametri di progetto.

Un **parametro famiglia** è un parametro definito all'interno dell'editor famiglia ed è interno ad essa. E' possibile, dal modello, visualizzare e modificare tali parametri, ma non è possibile definirli. Spesso i parametri di famiglia sono utilizzati per la modellazione geometrica degli oggetti (ad esempio la larghezza di un armadio).

I parametri di famiglia **NON compaiono negli abachi di progetto**.

I **parametri di progetto** sono dei parametri definiti, appunto, nel progetto ed assegnati ad una determinata categoria.

Parametri famiglia di istanza	Parametri famiglia di tipo
Parametri di progetto di istanza	Parametri di progetto di tipo

Immagine 14: Schema delle possibili proprietà che può avere un parametro

Non compaiono nell'editor di famiglie e vanno dunque ridefiniti qualora si volesse utilizzare un parametro di progetto su vari modelli. I parametri di progetto spesso vengono utilizzati per la parte informativa del modello, spesso sono di tipo testo o numero ed hanno il grande vantaggio di essere **letti dagli abachi di progetto**.

Sia parametri di famiglia che parametri di progetto possono essere sia di istanza che di tipo.

Chiariti questi concetti, si possono introdurre i parametri condivisi.

Ad ogni singola famiglia sono stati assegnati 12 Parametri Condivisi.
Per una spiegazione più approfondita, si rimanda il lettore a "Parametri Condivisi del Progetto" a pagina 59

I **parametri condivisi** sono dei parametri di progetto derivanti da un **file .txt esterno al modello**. Diversi modelli possono ricavare parametri dalla stessa fonte .txt. Ciò permette la creazione di parametri che siano **riconosciuti da più modelli contemporaneamente**. Tale vantaggio è evidente soprattutto quando si parla della gestione di dati su modelli di coordinamento (si parlerà di modelli di coordinamento al capitolo"Controllo Parametri di Modelli Federati" a pagina 75). Per esempio, è possibile controllare i parametri condivisi di tutte le porte presenti all'interno del progetto della Torre Regione Piemonte (non solo la torre) in un unico abaco creato in un modello di coordinamento.

Complesso Torre della Regione Piemonte

Situato nell'area ex industriale di Nizza Millefonti, nelle vicinanze del Lingotto, a Torino, la Torre della Regione progettata da **Massimo Fuksas** è la **costruzione più alta del Piemonte**, nonché terzo grattacielo più alto d'Italia (secondo se non si considera la guglia della Torre Unicredit di Milano, in piazza Gae Aulenti, con i suoi 231m, di cui 79 proprio di guglia).

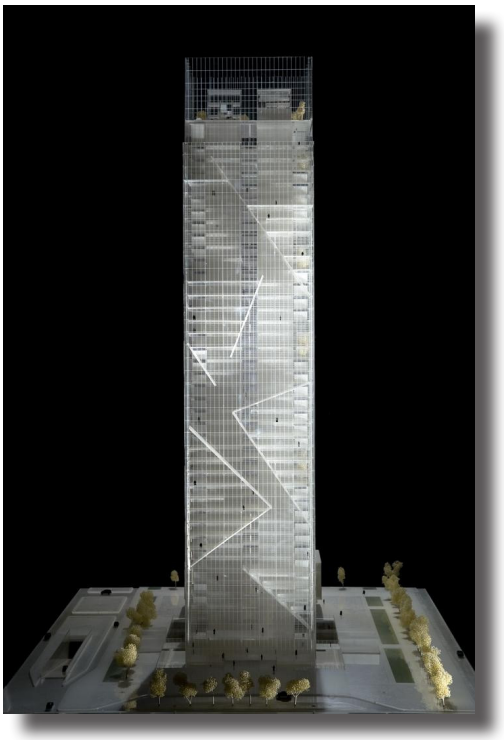


Immagine 15: Un render della Torre

A livello interrato, sotto la piazza è presente un **parcheggio interrato** mentre gli **interrati della Torre circondano la base della stessa** conterranno uffici ed archivi.

La Torre della Regione Piemonte si compone di **42 piani**, di cui **2 interrati**. L'ultimo piano ospiterà un bosco pensile adibito ad osservatorio aperto al pubblico.
La superficie complessiva è di circa 70 000 metri quadri mentre sulle facciate sono installati 1000 metri quadri di pannelli fotovoltaici.

La Torre **fa parte di un complesso generale** che comprende uno spazio vede nelle vicinanze, un centro servizi ed una promenade, ovvero una pista ciclabile sormontata da una tettoia coperta di pannelli solari.
Il progetto prevede anche la costruzione in futuro di una stazione della metropolitana.

La Torre non appoggia direttamente al piano terreno ma **presenta due piani sotto il piano campagna**. Attorno alla base della Torre, è presente un cortile che circonda l'impronta dell'edificio. Attorno a questo cortile si affaccia poi il complesso degli **Interrati Torre**.

A questi interrati si accede attraverso i **4 box scale presenti al piano terreno**, 2 situati nell'ala nord e 2 nell'area sud. Al cortile è invece possibile accedere attraverso una rampa carrabile che costeggia il lato sud.

Gli interrati si sviluppano per 2 piani sotto terra; sebbene dalle tavole risulti un piano -3, esso in realtà è situato a soli 50cm dal piano -2 e comprende un'area a nord ovest utilizzata per carico e scarico merci.

La parte ovest degli interrati si trova sotto al centro servizi e, sebbene il layout di questa sia ancora non definito, è comunque stabilito che ospiterà servizi dello stesso, tra i quali, probabilmente, la mensa. L'ala est è invece adibita ad archivi.

La struttura dell'edificio è in calcestruzzo armato e poggia su plinti e cordoli

Immagine 16: Torre della Regione Piemonte con Centro Servizi

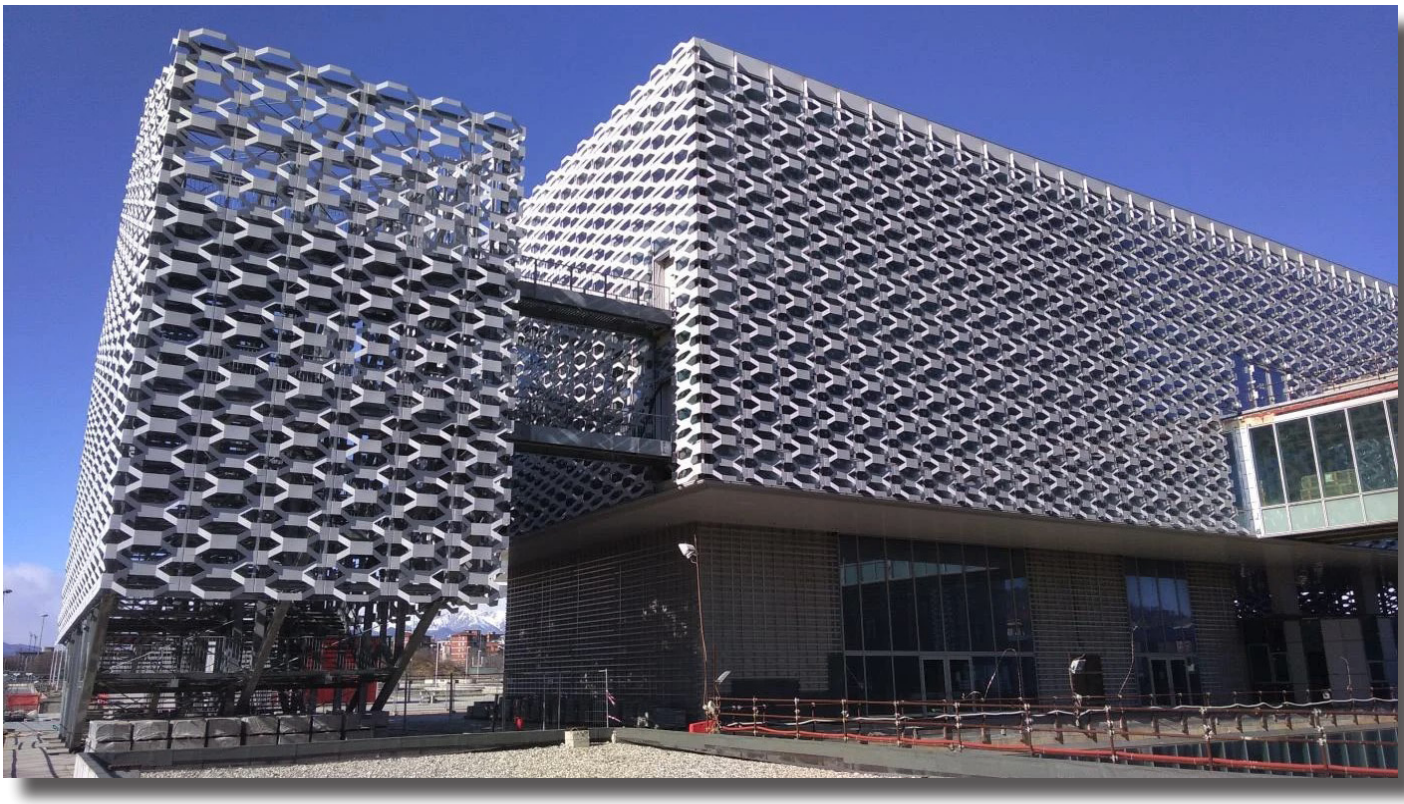


Immagine 17: Centro Servizi

Gestione dei file di Progetto

Tutto il complesso è stato modellato su Revit in modo da fornire agli Uffici della Regione Piemonte un modello BIM.

Il progetto in questione, come facilmente intuibile è piuttosto grande e, come spesso accade in questi casi, è stato necessario suddividerlo in più parti distinte, in modo da facilitare la gestione del lavoro. Tali parti sono collegate tra loro tramite **link uniti per coordinate condivise**. I modelli linkati suddividono il lavoro in base al **criterio spaziale** (ogni modello descrive una sola parte dell'edificio e non altre) e **disciplinare**.

Volendo essere più specifici, per quanto riguarda il **criterio spaziale**, tutto il progetto è stato diviso tra la torre, i suoi interrati, il parcheggio sotterraneo ad ovest, il centro servizi e la promenade.

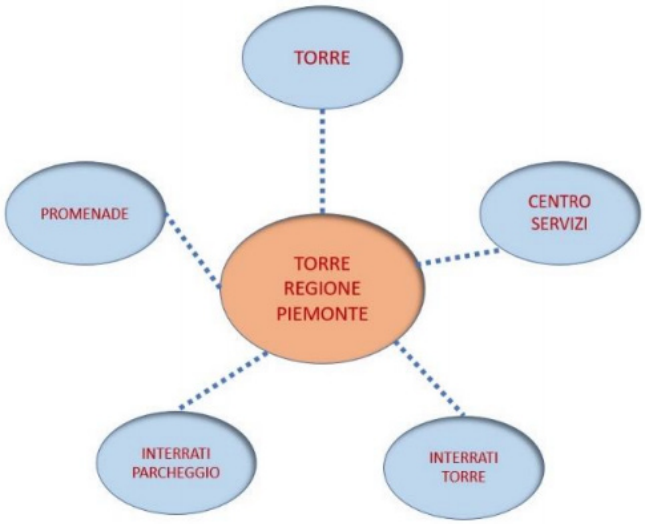


Immagine 18: Suddivisione secondo criterio spaziale

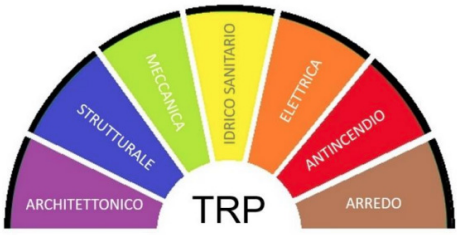


Immagine 19: Suddivisione secondo criterio disciplinare

Il **criterio della disciplina** divide invece il modello di ogni edificio per varie discipline. Si hanno così modelli Architettonico, Strutturale, Meccanico, Antincendio ed Elettrico, Idrico Sanitario ed Arredo.

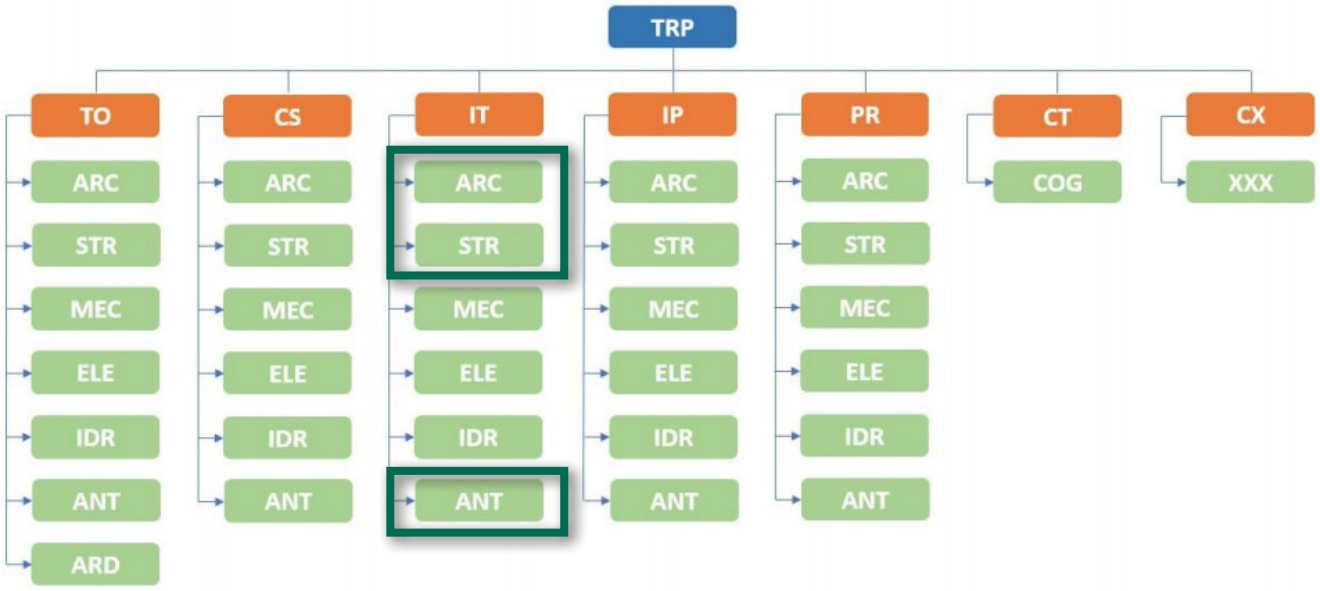


Immagine 20: Suddivisione basata sui due criteri.

Qui sopra uno schema che rappresenta lo schema della divisione in vari modelli, con la relativa nomenclatura dei file.

Oggetto del tirocinio presso il DrawinfTOtheFuture è stata la **modellazione del modello architettonico, strutturale ed impiantistico degli Interrati Torre IT.**

Modello Architettonico

I maggiori problemi riscontrati nella modellazione hanno riguardato i file di riferimento .pdf e .dwg. Sono infatti state trovate **numerose incongruenze tra file, soprattutto nelle sezioni**.
Tali problematiche sono probabilmente dovute al fatto che i vari disegni rappresentano il progetto in tempi diversi, alcune al netto ed altre prima di revisioni e cambiamenti.
Una situazione di questo tipo e la mancanza di elaborati dello stato di fatto dell'edificio hanno portato a dover decidere a quale file dare la priorità come riferimento. In questo caso, la **priorità** è stata data **alle piante architettoniche** dei piani -1 e -2. Ulteriori problemi sono poi sorti dagli **abachi**, soprattutto dei locali, laddove è facile trovare **informazioni mancanti** ed altre superflue.
Quanto appena scritto non vuole essere né una giustificazione per un lavoro incompleto né una critica ad altri professionisti, quanto piuttosto una **dimostrazione delle difficoltà che si hanno nel gestire file di grandi progetti senza ricorrere alla metodologia BIM**. Chi scrive ritiene infatti che il modo migliore per apprezzare i vantaggi dell'adozione di questa metodologia sia proprio scontrarsi con i problemi derivati dalla sua assenza. Tutte le incongruenze trovate non sarebbero infatti sorte se si fosse modellato tutto, fin dalla fase di progettazione, in ambiente BIM.
Tra le altre cose, infatti, BIM è coerenza.
Di seguito, vengono invece trattati gli problemi riscontrati.

Muri per le finiture

Una delle richieste della committenza è stata quella di **modellare separatamente le finiture e la struttura di ogni muro**, invece di definire questi strati tramite una stratigrafia di un muro singolo. Tale scelta va da ricercarsi nella volontà di conteggiare separatamente le due parti del muro, in modo da facilitare il computo metrico di una parte o dell'altra. Tale espediente ha presentato delle criticità: anzitutto, agendo in questo modo, si triplica il numero di

muri da modellare e, di conseguenza, anche il tempo necessario per completare il lavoro, nonché per controllare se vi siano problemi o dimenticanze nel modello.
Agire in questo modo, inoltre, rende meno elastico il modello a future modifiche in quanto, modificando ad esempio lo spessore di un tramezzo, la famiglia "muro" di finitura perderebbe l'aderenza con la struttura; per questo motivo è sconsigliabile utilizzare questo approccio in fase di progettazione e modifiche (non il caso oggetto del lavoro comunque).

Volendo fare una considerazione personale, sebbene l'utilizzo di famiglie muro per modellare le finiture possa sembrare una forzatura, essa in realtà è l'unico modo per computare, ad esempio nel caso di un bagno, i metri quadrati di piastrelle necessari per il rivestimento. Si pensi poi al caso in cui dopo anni, sia necessario ritinteggiare i muri di parte dei locali dell'edificio: in questo caso tornerebbe molto utile sapere in pochi secondi, i metri quadrati di vernice necessari.
A livello concettuale infine, per chi scrive, non è sensato che due muri completamente identici e che differiscono soltanto per il colore di finitura su un lato rappresentino, per il software, due tipi di muro differenti, con due codifiche differenti.
Tali problemi sarebbero superati con l'implementazione di una categoria di sistema "Finitura" che possa associarsi ai muri, spostandosi con essi qualora si spostino i muri pur venendo conteggiata come categoria differente.

Con il termine "hostato" si intende "ospitato all'interno. In questo caso, una porta è una famiglia che deve essere per forza associata ad un muro

Porte non forano i muri

Altro problema derivante dal suddividere muri e muri-finitura è stato il comportamento delle porte e delle griglie verticali, ovvero delle famiglie hostate su questi muri.
Non è possibile infatti immaginare una porta che non buchi una parete.
Avendo tre muri vicini al posto di uno solo pone il **problema** di come **creare il foro in tre muri contemporaneamente**;
la famiglia porta è infatti in grado di creare un'apertura solo su un muro, non su tre.
Questo problema si può risolvere ricorrendo a due stratagemmi.
Anzitutto, nell'editor delle famiglie porte è possibile definire

il foro nel muro in due modi:

Apertura

Si tratta del metodo più semplice, utilizzato di default dal programma: sostanzialmente si tratta di definire una **sagoma bidimensionale che indicherà i bordi del foro** del muro. In questo senso, assomiglia allo strumento "modifica profilo" riservato, quest'ultimo, alla famiglia muro.

Immagine 21: Simbolo dello Strumento Apertura

Vuoto

Alternativamente, è possibile creare un **solido**, in questo caso di estrusione, **tridimensionale e vuoto**, da andare poi a **sottrarre al muro di host, secondo una logica booleana**. Rispetto al metodo precedente, l'utilizzo di quest'ultimo approccio permette di creare nicchie nel muro e, in generale, geometrie tridimensionali più complesse.
Nel caso in questione è stato necessario utilizzare il secondo metodo, andando a creare un **solido di sottrazione rettangolare che eccedesse, in profondità, il muro host della porta**. Attraverso questo aggetto, verranno sottratte anche porzioni delle famiglie muro di finitura.

Immagine 22: Simbolo dello Strumento Vuoti

A questo punto, è stato necessario **unire i tre muri** (struttura e i due strati di finitura) **attraverso lo strumento "Unisci"**per ottenere il risultato ricercato.
L'utilizzo di questo approccio non risolve il problema di dover lavorare con un file strutturale linkato all'architettonico. **Vi sono** infatti **alcuni punti dove una porta è montata su un muro** in calcestruzzo, **muro che fa parte del file strutturale**, sebbene la porta sia parte del file architettonico.
In questo caso, è **possibile hostare la porta su un muro di finitura** (muro parte del file architettonico), **ma non è possibile creare automaticamente una nicchia nel muro strutturale**. In questi casi, l'unica soluzione è quella di creare, in corrispondenza della porta, manualmente una nicchia nel file strutturale, sapendo tuttavia che essa non si sposterà qualora dovesse essere modificata la posizione della porta. E' inoltre praticamente impossibile fare un lavoro di questo tipo senza andare incontro ad imprecisioni.

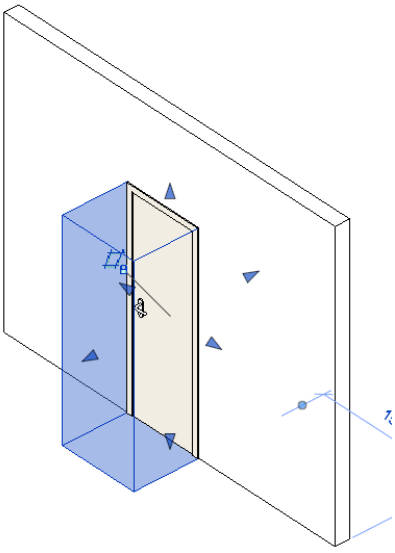


Immagine 23: Come un vuoto fora un muro

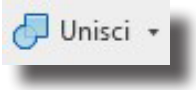


Immagine 24: Comando Unisci di Revit

Pareti in cartongesso

Nel progetto sono presenti alcune pareti in cartongesso. Una parete in cartongesso è formata, oltre che, eventualmente dalla finitura, (come tutte le altre pareti del resto), da due lastre in cartongesso, divise da uno strato per lo più vuoto, contenente la sottostruttura per il cartongesso.
Sono stati modellati separatamente i due strati di cartongesso come pareti separate invece di avere un unico muro che presentasse una stratigrafia di 3 strati (ovvero 2 pareti in calcestruzzo ed un'intercapedine di aria).
Questo approccio permette di modellare facilmente alcuni casi dove una lastra in cartongesso non è accoppiata ad un'altra lastra ma, ad esempio, ad un muro in blocchi Ytong.
Il **problema** sorge tuttavia **quando occorre unire le due pareti per creare il foro in tutte e tre** le pareti; due muri infatti non possono essere uniti se non aderiscono perfettamente tra loro.
Per risolvere questo problema, è stata creata un'**altra tipologia di muro chiamata "PA_Void**" , completamente trasparente e priva di qualsiasi proprietà fisica ma utile per unire le due lastre tra loro in modo da essere forate facilmente da una porta hostata.
Agire in questo modo comporta la creazione di muri che, non esistono nella realtà ma che vengono comunque contati come muri all'interno degli abachi a meno di non essere filtrati in qualche modo.
Si è comunque ritenuto questo approccio più coerente rispetto al creare un muro di tre strati utilizzato solo in corrispondenza delle porte.

Creazione delle Porte Metalliche

La modellazione degli interrati della Torre della Regione non ha richiesto la modellazione di molte nuove famiglie caricabili e, quando accaduto, si è trattato di oggetti abbastanza semplici.
Un caso di eccezione sono tuttavia state le **porte metalliche**.
Presenti in gran numero all'interno del progetto, si caratterizzano di numerose proprietà, molte delle quali saranno trattate nel capitolo relativo alla compilazione dell'abaco dedicato (vedi pagina 33).

Le famiglie di Porte Metalliche sono 3, ovvero:

- **PM** = Porta Singola
- **PD** = Porta Doppia
- **PS** = Porta Scorrevole

Le coppie di lettere maiuscole che precedono i nomi sono i rispettivi `Codice Famiglia` (vedi cap "Codice Famiglia" a pagina 60).

Per le prime due, **è possibile specificare** quale **tipo di maniglia** esse montano sul lato interno ovvero **maniglia, maniglione antipanico e touchbar**, senza che sia necessario creare una nuova famiglia.

Tale risultato è stato ottenuto attraverso il parametro **`testo etichetta`**, un particolare parametro che permette di scegliere quale, tra un gruppo scelto di sotto-famiglie hostate, verrà posizionata in un determinato punto scelto.

Nota

Nonostante il nome, il parametro `testo etichetta` non riguarda le etichette nelle viste.

Nota sul testo etichetta

Nonostante in linea teorica sia possibile hostare una famiglia all'interno di una famiglia a sua volta hostata in un'altra in una sorta di gioco di scatole cinesi, nella realtà, **se si vuole utilizzare il parametro testo etichetta, non è possibile avere più di due livelli di hosting**.

In questo specifico caso, ad essere intercambiabile con il testo modello, non è stata la maniglia (contenuta nella sotto famiglia battente, a sua volta contenuta nella famiglia porta) quanto direttamente il battente intero con la maniglia scelta.

Si riporta di seguito uno schema che rappresenta la nidificazione all'interno di una porta metallica PM.

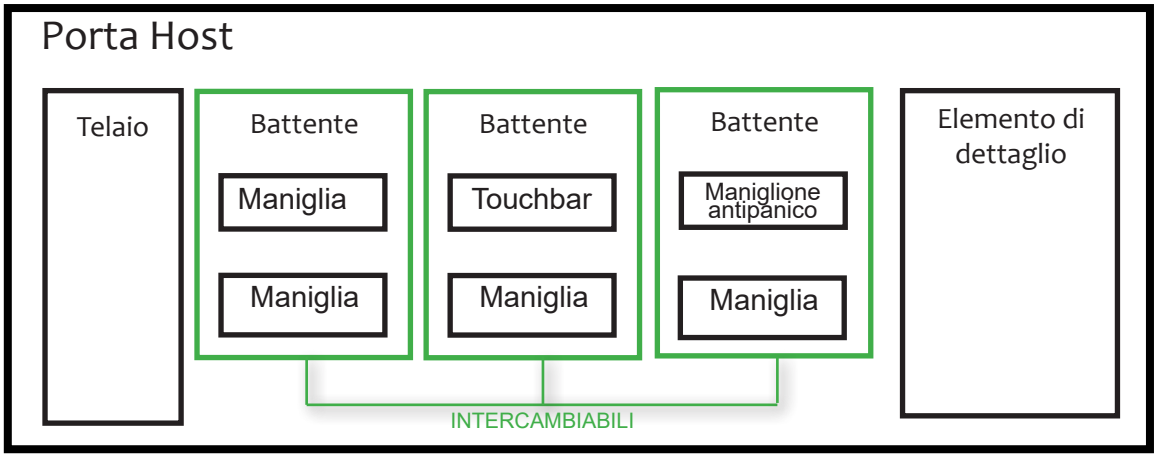


Immagine 25: Schema di nidificazione all'interno di una porta metallica PM.

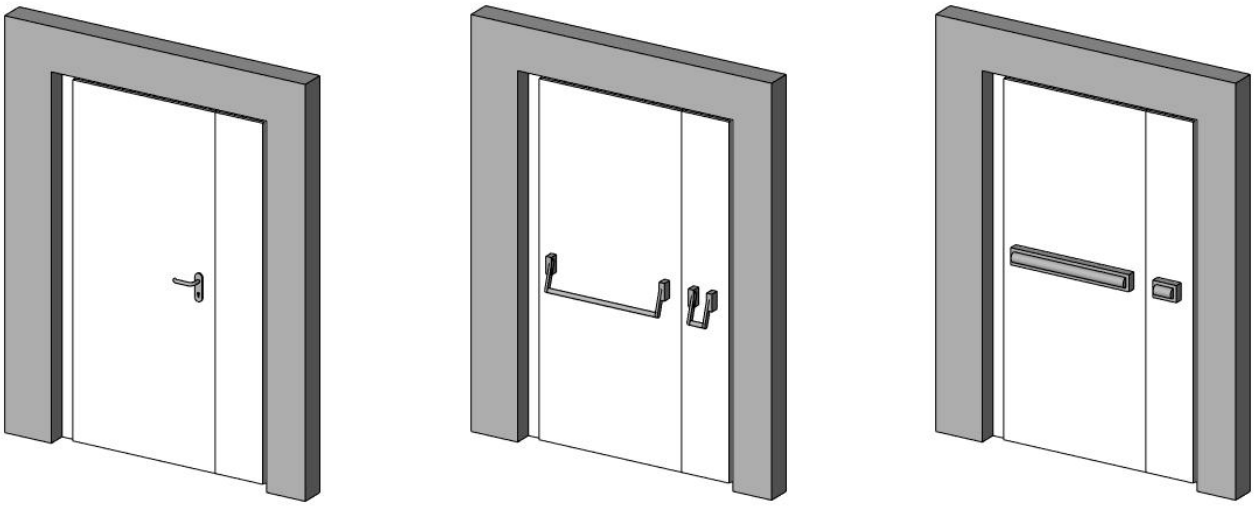
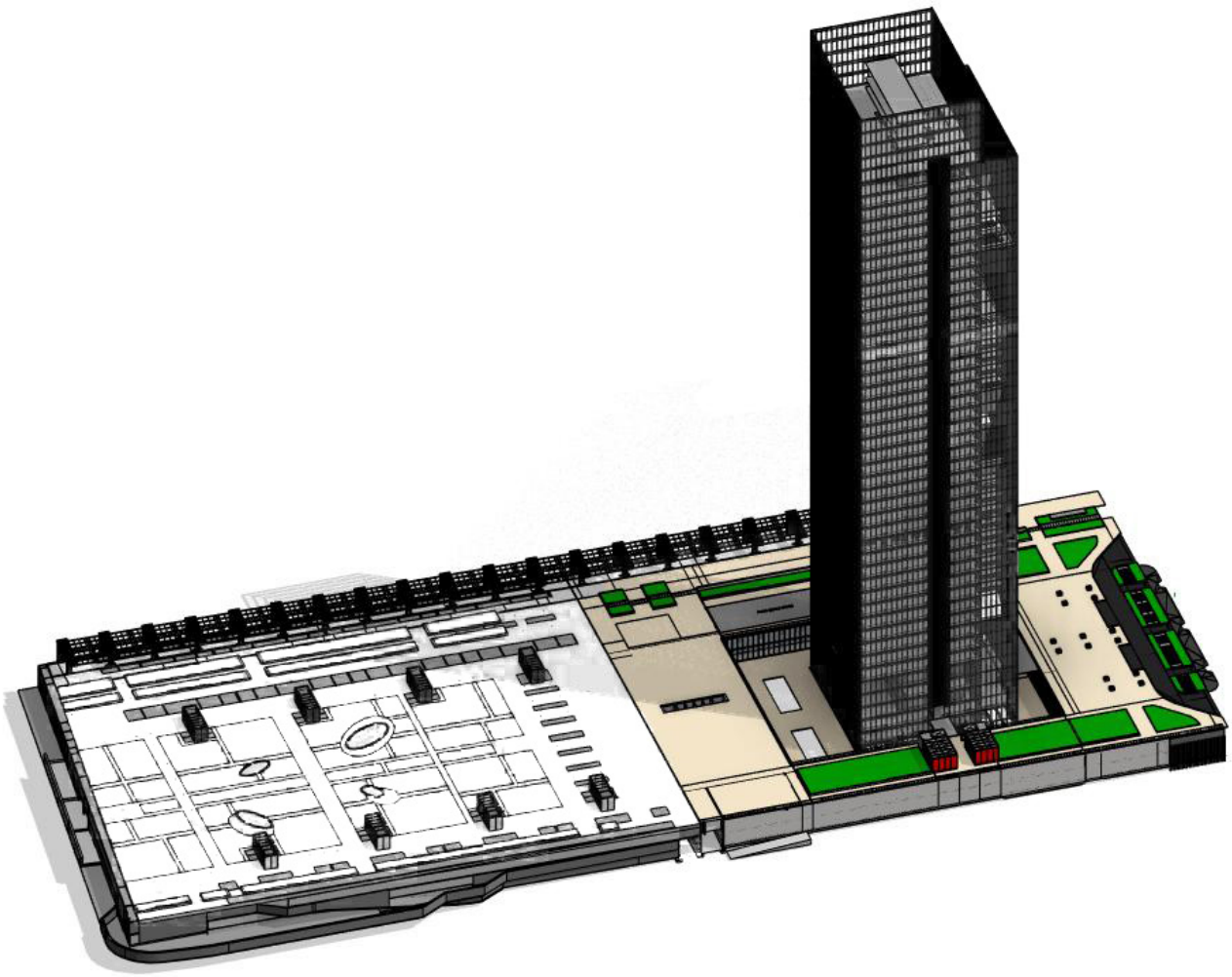


Immagine 26: Tre tipi di porta modellati con tre maniglie differenti



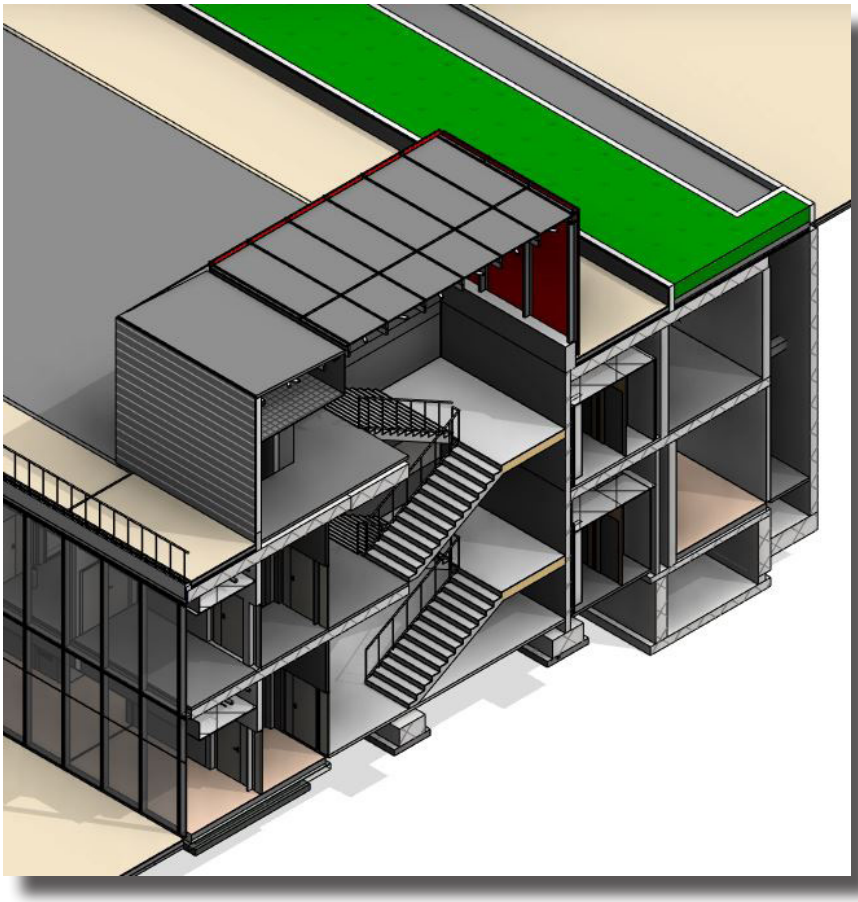


Immagine 27: (Sopra) Interrati Torre all'interno del progetto generale

Immagine 28: (Sinistra) Sezione Scale

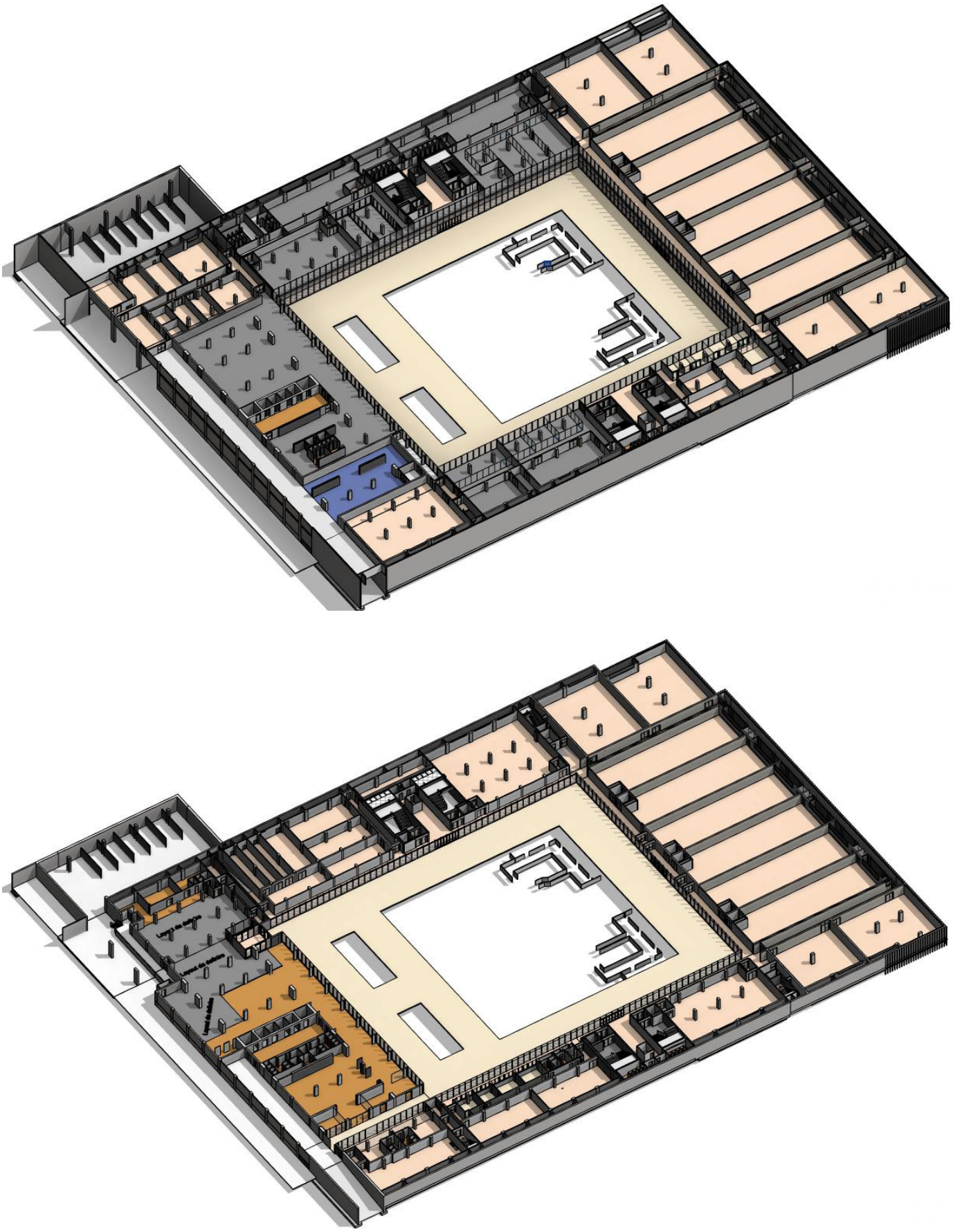


Immagine 29: Vista 3d degli interni architettonici degli Interrati Torre (Piani -1 e -2).

Abaco porte metalliche

La compilazione dell'abaco delle porte metalliche è stata eseguita facendo riferimento all'abaco .pdf presente nelle tavole di progetto, al quale sono stati inseriti alcuni dati supplementari. Di seguito si riporta una presentazione di ogni colonna parametro

Edificio

Si tratta di un parametro che **rappresenta la parte dell'edificio in cui è posizionata la famiglia**. In questo caso, essendo il progetto degli Interrati Torre, per ognuna delle istanze è stato scelto il codice "**IT**".

Il parametro Edificio è uno dei parametri condivisi compilati per ogni istanza del modello: si rimanda il lettore a "Parametri Condivisi del Progetto" a pagina 59

Codice Livello

Si tratta di un parametro che, sostanzialmente, fornisce le **stesse informazioni del parametro `Livello`, ma** rese **più snelle** nell'annotazione (si ricorda che nel progetto, i livelli sono stati definiti con nomi che riportano anche la quota degli stessi, nonché l'informazione sul fatto che siano interrati o meno).
Un parametro più sintetico permette di creare più facilmente eventuali parametri combinati e abachi più puliti.

Codice Tipologia Porta

Definisce in prima battuta, s**e si tratti di una porta metallica** REI, di una **vetrata o** di una **tamburata**.

Tipolgia porta	Codice Tipologia Porta
Porte Metalliche	M
Porte Tamburate	H
Porte Vetrate	V

Immagine 30: Come viene compilato il Codice Tipologia Porta

Zona

Tutto il complesso della Torre della Regione viene diviso in zone. Queste zone **non sempre coincidono con la divisione** spaziale **fornita dai modelli** e, ad esempio, gli Interrati Torre comprendono 4 zone, corrispondenti alla porzione Nord, Sud, Est ed Ovest (Quella del Centro Servizi).

Sottozona

L'**intero progetto** degli Interrati Torre è poi **ulteriormente suddiviso in sottozone**. Tale parametro aiuta a localizzare una certa istanza all'interno del progetto. Le sottozone, si raggruppano in Zone, trattate nel paragrafo precedente.

Numerazione

All'interno di ciascuna sottozona e ad un dato livello, le porte sono numerate con un **numero progressivo**. Tale codice **permette di identificare univocamente ciascuna istanza** ed è utile soprattutto per formare il `Codice Progetto Esecutivo`.

Codice Progetto Esecutivo

Tale parametro è compilato solamente per le porte metalliche e per le porte vetrate.
Assieme alle tavole .dwg, come detto in precedenza, il progetto di riferimento si compone di abachi .pdf, tabelle che attribuiscono varie proprietà ad oggetti presenti nelle tavole CAD. Il **collegamento tra le informazioni date dalla tavola e quelle date dall'abaco** avviene tramite il `Codice Progetto Esecutivo`.

`Codice Progetto Esecutivo`, per le porte, combacia con il parametro condiviso `Codice Esistente`(vedi "Codice Esistente" a pagina 61).

Abaco dei Locali

La compilazione dell'abaco dei locali è stata eseguita facendo riferimento all'abaco presente nelle tavole di progetto, al quale sono stati inseriti alcuni dati supplementari. Di seguito si riporta una presentazione di ogni colonna parametro

Edificio

Valgono le stesse cose dette per l'abaco delle porte
Vedi "Edificio" a pagina 33

Zona

Valgono le stesse cose dette per l'abaco delle porte
Vedi "Zona" a pagina 33.

Sottozona

Valgono le stesse cose dette per l'abaco delle porte
Vedi "Sottozona" a pagina 34.

Codice Livello

Valgono le stesse cose dette per l'abaco delle porte
Vedi "Codice Livello" a pagina 33

Codice Zona

Si tratta di un **parametro combinato** che **unisce** tra loro i parametri **`Edificio`** e **`Sottozona`**, separandoli con un trattino. Tale parametro è stato creato per seguire la struttura dell'abaco dei locali fornito dalla Ragione Piemonte.

Codice Locale Costruttivo

Si tratta del **codice assegnato ai locali nelle tavole di progetto** .dwg ed è **univoco per ciascuno dei locali**. Sebbene non sia stato ritenuto utile suddividere effettivamente tale parametro in altri sottoparametri per poi combinarli in questo `Codice Locale Costruttivo`, è comunque facile notare che tale parametro è composto da più informazioni.

I **parametri combinati** sono parametri che riportano i valori di altri parametri, di solito suddividendoli tramite underscore, trattini o punti. Sono definiti all'interno del singolo modello Revit.

Consentono meno operazioni rispetto agli altri parametri ed hanno solo una funzione grafica all'interno degli abachi.

Il `Codice Locale Costruttivo` è compilato secondo regole diverse nell'abaco della Torre.

`Codice Locale Costruttivo`, per i Locali combacia con il Parametro Condiviso `Codice Esistente` (vedi "Codice Esistente" a pagina 61).

In particolare, esso fornisce informazioni sul livello di posizionamento e sulla zona, oltre che un codice identificativo progressivo.
Si di seguito, l'esempio del locale I2E-003 (Archivio), ovvero un locale posizionato al secondo piano interrato, nella Zona Est.

Note del modellatore

Durante l'operazione di trasposizione dei dati dall'abaco fornito dalla Regione a quello del file Revit, ci sono stati numerosi casi in cui si è reso necessario fare qualche annotazione, ad esempio per segnalare errori o per giustificare brevemente alcune compilazioni. Tali note rientrano nel parametro non condiviso `Note del modellatore`.

Categoria Locale

Individua la macro-area di appartenenza del locale in base alla destinazione d'uso dello stesso.

Sotto la `Categoria Locale` vengono raggruppate varie `Tipologia Locale`. Di seguito lo schema riportante i casi presenti negli interrati della Torre.

Categoria Locale	Tipologia Locale
Servizio all'edificio	Apparati telefonici
	Bagno
	Cabina MT/BT
	CED
	Control room
	Deposito
	Filtro
	Gestione manutenzione apparati telefonici
	Intercapedine di aerazione
	Locale di servizio
	Locale impianti speciali
	Locale QBGT
	Locale quadri elettrici
	Locale quadro QGBT
	Locale safety
	Locale security
	Locale tecnico
	Locale UPS
	Locale UTA
	Magazzino
	Magazzino economale
	Postazione di controllo
	Presidio medico
	Ripostiglio
	Sala autisti fumatori
	Sottocentrale
	Spogliatoio
	Vasca riserva idrica
	Zona contenitori area stoccaggio rifiuti
Area ausiliaria	Connettivo
	Corsello carrabile
	Disimpegno
Area di lavoro	Centro stampa
	Cucina
	Ufficio
Area di supporto	Archivio
	Reception
	Sala riunioni
	Sala taglio
Collegamenti verticali	Scale

Immagine 31: Tabella che fa corrispondere ad ogni Tipologia Locale, la relativa Categoria Locale

Tipologia Locale

Individua la **destinazione d'uso del locale**

Denominazione locale

Tale parametro riporta il **nome dei locali ricavabile dalle tavole**, nome che, talvolta, fornisce informazioni ulteriori rispetto al parametro `Tipologia`.
Il caso più evidente della differenza tra i parametri `Tipologia` e `Denominazione Locale` è quello dei bagni; laddove infatti il secondo va a distinguere i servizi maschili da quelli femminili, `Tipologia` riporta tutto come "bagno".

Modello Antincendio

La fase di modellazione ha visto anche la realizzazione dell'**impianto antincendio**, sempre **negli interrati della Torre**. Quando si parla di antincendio, entrano in gioco elementi che fanno parte del modello architettonico (si pensi alla compartimentazione REI di muri e porte, o ai percorsi di esodo) altri del modello elettrico (le insegne luminose che indicano le vie di esodo, o gli impianti di evacuazione di fumo e calore automatici) ed altri ancora che potrebbero rientrare in un modello idrico (l'impianto sprinkler).
A tal proposito, è stata definita una regola: nel **modello antincendio rientra tutto ciò che concorre allo spegnimento dell'incendio**.
In pratica, il modello ANT contiene l'impianto sprinkler, l'impianto di spegnimento water mist, l'impianto naspi e idranti, nonché i numerosi estintori portatili a CO_2 e polvere; non contiene invece le informazioni riguardanti compartimentazione, percorsi di esodo, dispositivi di allarme e smaltimento fumi.

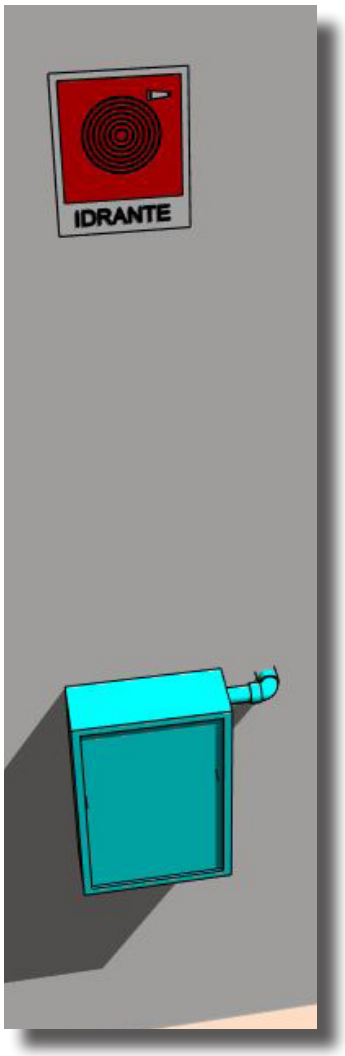


Immagine 32: Famiglia Idrante

Divisione in colori

Il modello antincendio, come già detto, si suddivide in 3 impianti, ovvero quello Sprinkler, quello Water Mist e quello Idranti. In una situazione di questo tipo, se non si ricorre ad un opportuno schema colori, è facile confondere un impianto per l'altro. Per evitare confusione è dunque necessario colorare gli impianti in modo differente.
Ci sono diversi metodi per giungere a tale risultato.

Colorare il sistema

E' possibile applicare un colore all'intero tipo di sistema. Un **sistema** è un insieme di famiglie avente una funzione (ad esempio, un sistema di mandata aeraulico). Sebbene non si tratti di famiglie vere e proprie, essi hanno comunque alcune analogie con le stesse, soprattutto nella possibilità di assegnare dei parametri di tipo. E' possibile assegnare ad un tipo di sistema, una sostituzione grafica, in modo da colorare tutto il sistema del medesimo colore
Tale via non è stata scelta in quanto non è possibile creare

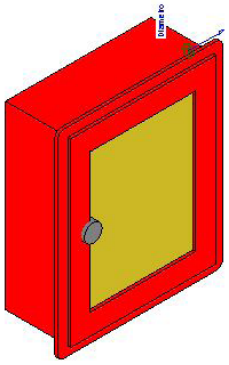


Immagine 33: Famiglia incassato a muro

nuovi tipi di sistema su Revit, oltre a quelli già impostati. Tale limite è un problema in quanto tutti e 3 i sistemi presenti nel modello sono di "protezione antincendio ad umido".

Assegnare diversi materiali alle tubazioni

E' possibile aggirare il problema **assegnando** a ciascuna tubazione ed a ciascun accessorio, **un materiale che abbia le proprietà grafiche desiderate**. Si avranno così, ad esempio, 3 materiali acciaio chiamati

- Acciaio SPK
- Acciaio WTM
- Acciaio IDR

da assegnare alle rispettive famiglie, identici in tutto e per tutto se non nelle impostazioni grafiche.

Il **problema** di un approccio di questo tipo è la **creazione di materiali fittizi** quando, in realtà, tale suddivisione non è rappresentativa del caso reale. Le operazioni sui materiali, inoltre, tendono ad essere abbastanza pesanti per il software ed è dunque meglio cercare di effettuarle solo quando necessario.

Creazione di un nuovo parametro e filtro

Quest'ultimo metodo è quello che è stato **effettivamente utilizzato**.
Si tratta di **assegnare ad ogni singola istanza del modello, un parametro di progetto, chiamato `Tipo di Impianto`, composto di 3 cifre, che appunto identifica se la famiglia è appartenente ad uno o all'altro impianto**.
Compilato tale parametro per ogni istanza (tubazioni, estintori, raccordi, tubazioni flessibili, collocazione tubazione), è possibile isolare tali elementi con un **filtro**, ed impostare una **sostituzione grafica**.
L'unico problema di tale approccio è il fatto di dover essere **svolto per ogni istanza**, il che rende la compilazione una fase abbastanza lunga, e soggetta a dimenticanze.
Il grande vantaggio di questo metodo invece, oltre alla snellezza e trasparenza, è la **suddivisione di tutto il progetto in 4 parti** (la quarta sono gli estintori portatili); questa suddivisione può essere utile per isolare in pochissimi clic una tipologia di impianto ed, eventualmente, nasconderla da

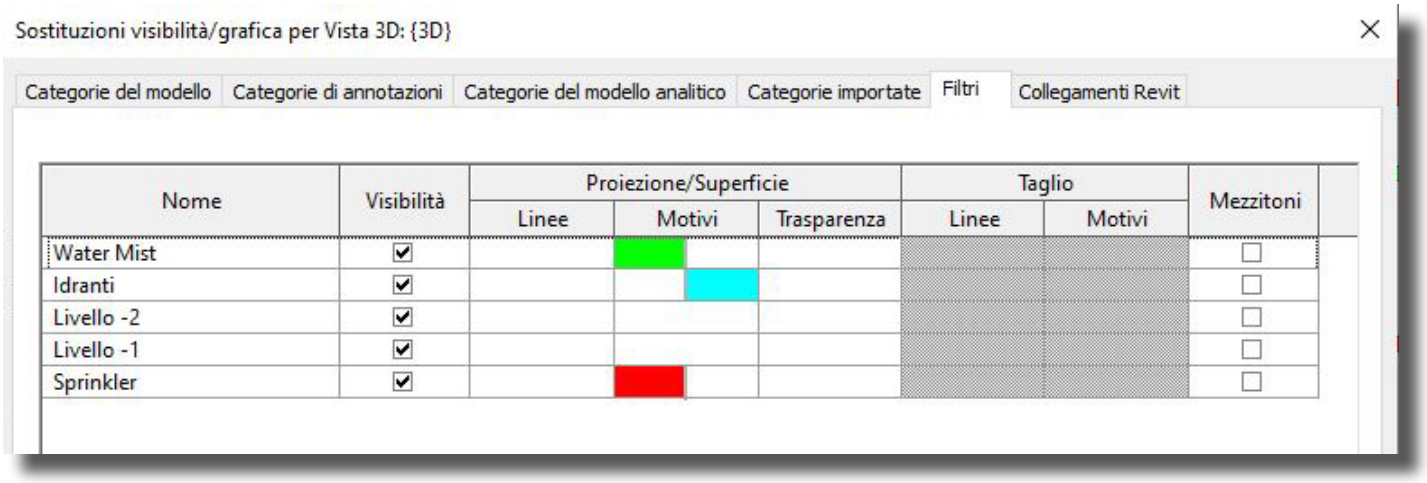


Immagine 34: Applicazione dei filtri per la colorazione corretta degli impianti

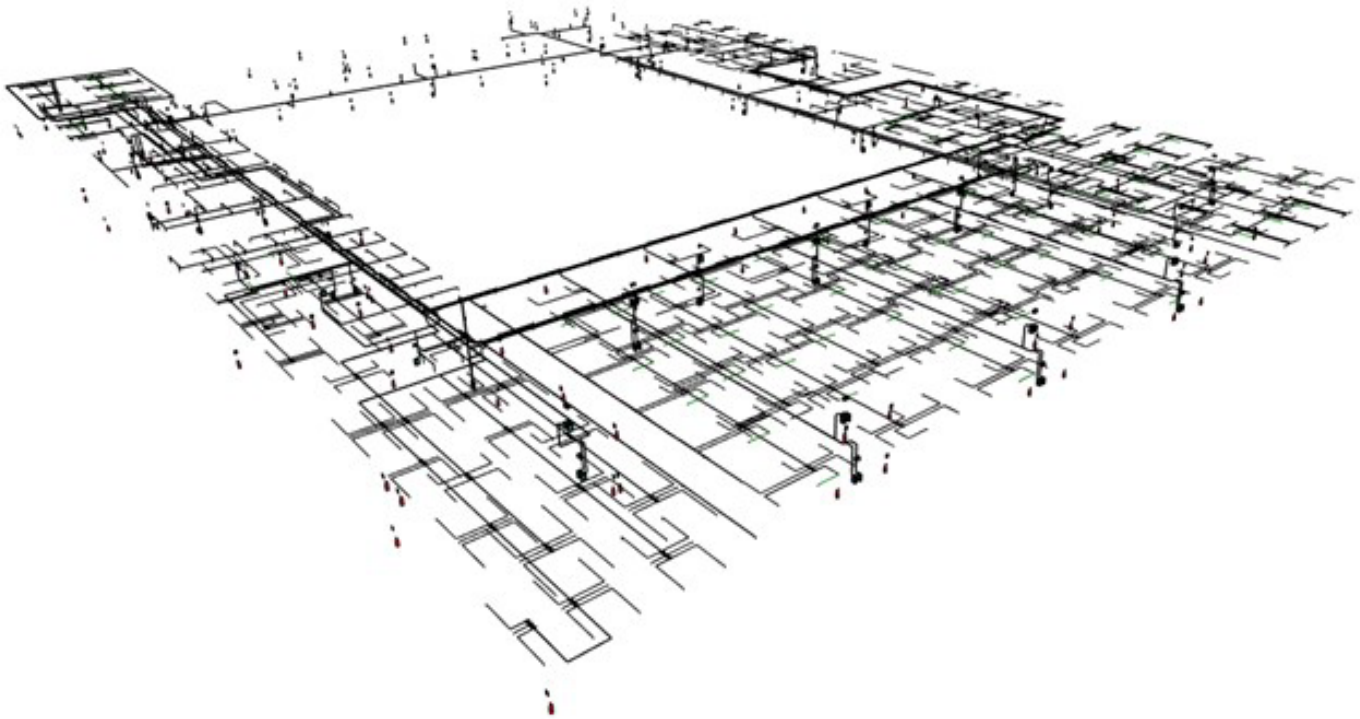


Immagine 35: Vista 3d del solo impianto antincendio del piano -2.

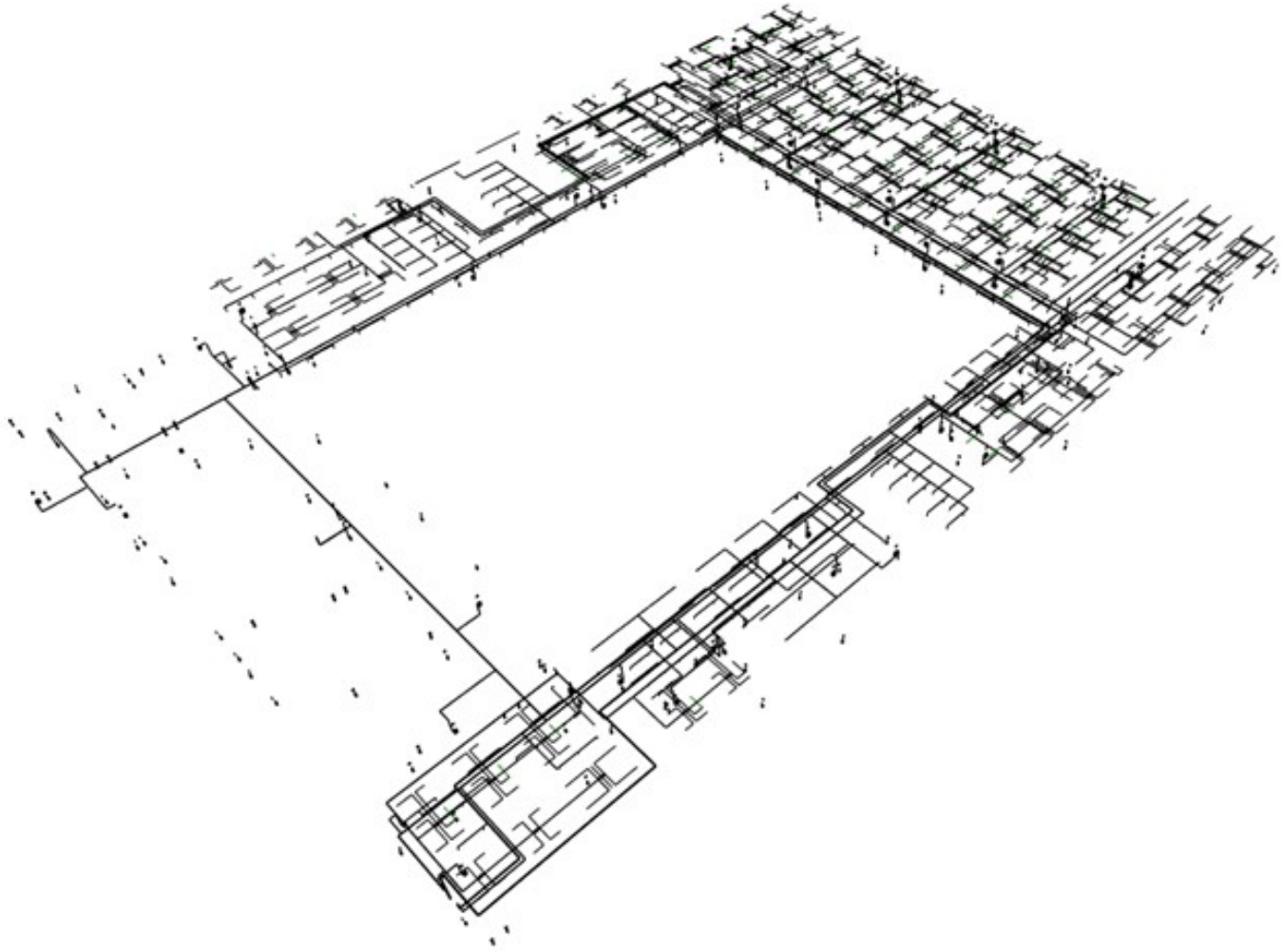


Immagine 36: (Sopra) Vista 3d del solo impianto antincendio del piano -1.

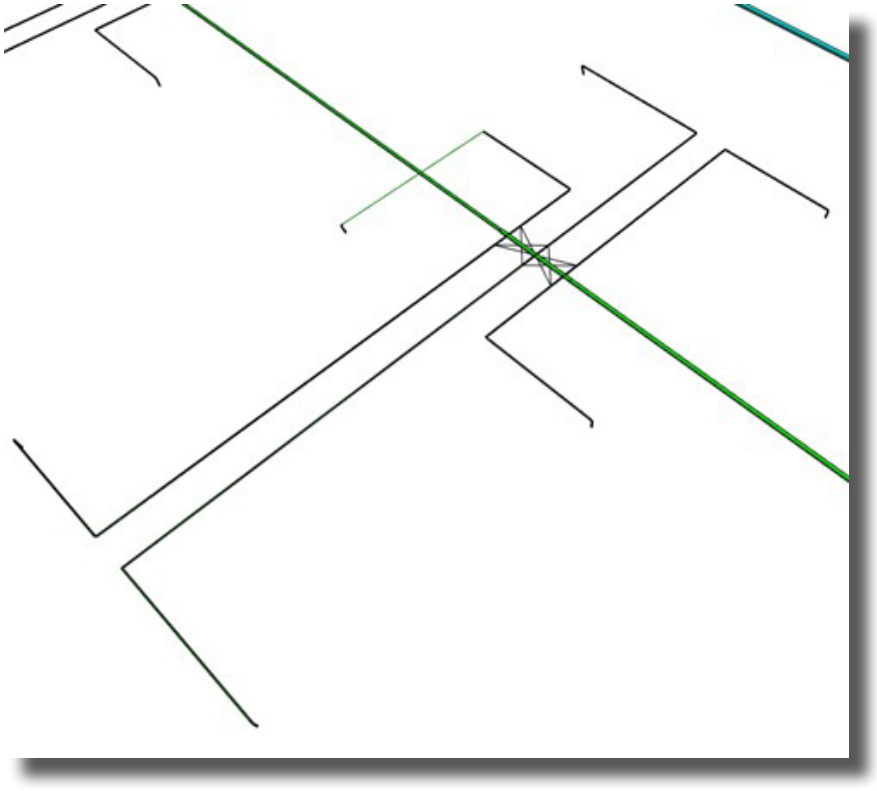


Immagine 37: (Destra) Vista 3d di un gruppo di Water-mist basato su raccordo.

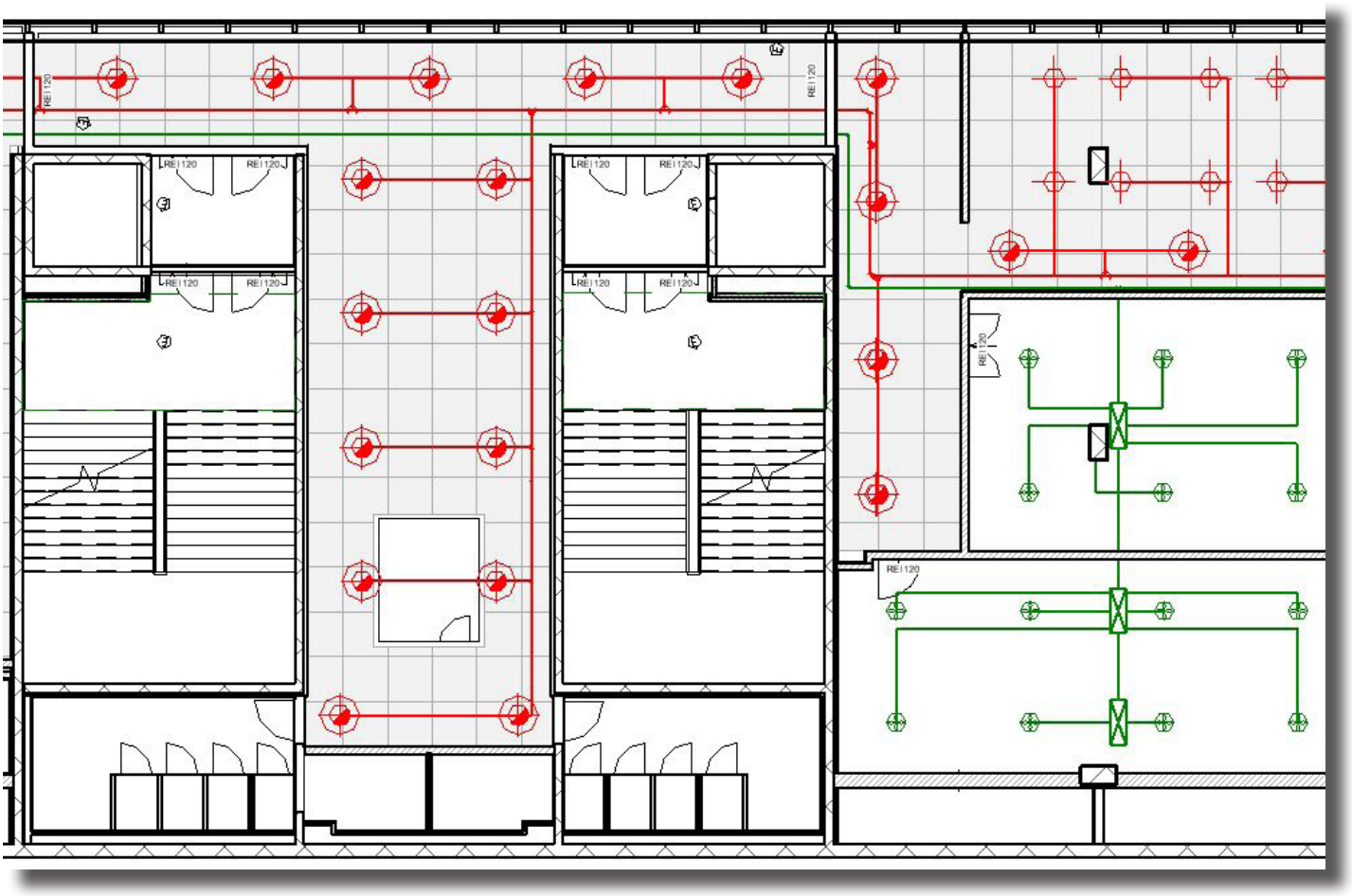


Immagine 38: Stralcio di pianta del livello -1 che mostra gli impianti Sprinkler e Water-mist

Famiglie estintori: Sprinkler

Immagine 39: Coppia di Sprinkler UP/PD

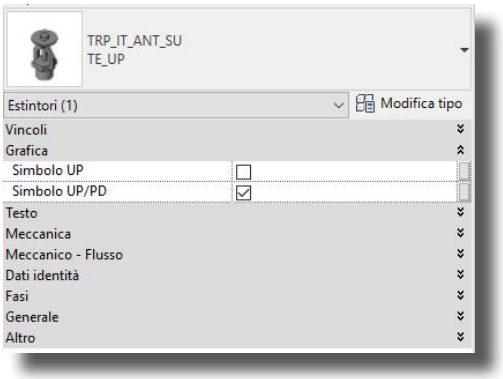


Immagine 40: Parametri per accendere e spegnere i Simboli degli Sprinkler.

In questo paragrafo, per estintori si intendono tutte le famiglie presenti nella categoria estintori di Revit, ovvero gli estintori portatili ma anche sprinkler, idranti e ugelli water-mist.

Nelle piante presenti sulle tavole, gli estintori non vengono rappresentati come proiezione ortogonale del loro modello geometrico, ma compaiono invece come simboli.

I **simboli**, su Revit sono **famiglie di annotazione** e come tali, sono **visibili unicamente da viste ortogonali** (sezioni e piante) e non da viste 3d. La particolarità dei simboli rispetto ad altre famiglie di annotazione come ad esempio l'elemento di dettaglio, è la loro **capacità di rimanere sempre della stessa dimensione** anche al variare della scala della vista dove sono inseriti. In questo caso, **i simboli** sono stati creati e **hostati dentro le famiglie degli estintori**.

Nel caso degli sprinkler, sulla tavola .dwg di partenza sono stati utilizzati simboli diversi a seconda del fatto che si tratti di **sprinkler UP** (rivolti verso l'alto), **PD**(pendenti, rivolti verso il basso) o **UP/PD** (due sprinkler diversi ma nella stessa posizione: è il caso rappresentato nell'immagine)

E' ovvio che alle famiglie di sprinkler UP sia stata assegnato il simbolo 2d dello sprinkler UP; lo stesso è per la famiglia sprinkler PD. Entrambe le famiglie poi caricano un secondo simbolo, ovvero quello dello sprinkler UP/PD. Ne consegue che entrambe le famiglie caricano 2 simboli; questi possono essere nascosti grazie ad un parametro famiglia dedicato e, nel caso di una coppia di sprinkler UP/PD, solo uno dei due ugelli mostra il simbolo della coppia UP/PD.

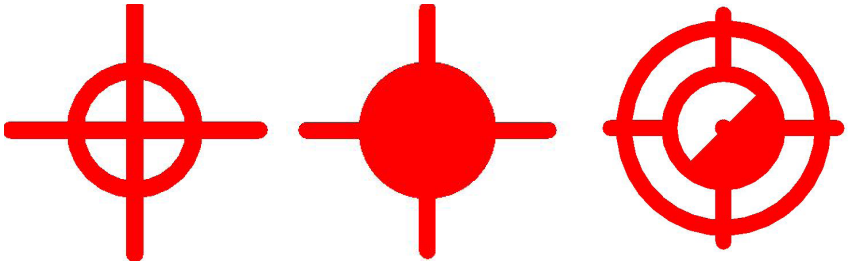


Immagine 41: Simbolo Sprinkler PD, UP e UP/PD

Famiglie estintori: Water Mist

Per quanto riguarda gli ugelli Water Mist, il simbolo associato è molto diverso da quello presente nelle tavole .dwg. Il motivo di ciò è l'impossibilità con Revit di creare linee troppo corte (cosa invece possibile su Autocad), linee che comunque sarebbero non visibili con certi spessori.

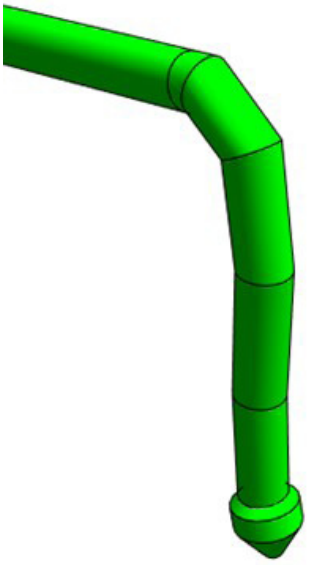


Immagine 42: Ugello Water Mist

Immagine 43: Simbolo ugelli Water Mist

Non è possibile su Revit creare linee più corte di 0,8mm

Famiglie estintori: Idranti

Vi sono due famiglie di idranti: quella avente la cassetta inserita all'interno di un muro e quella avente la cassetta esterna.
Le famiglie idranti presentano, hostata, anche la **famiglia del cartello**. Essa è stata creata semplicemente andando a modellare **forme molto sottili e di colori diversi da sovrapporre per creare l'effetto di un disegno**. Sebbene tale metodo possa sembrare una forzatura, esso è molto più diretto rispetto ad esempio, all'importare una serigrafia (molto pesanti per il software). La **famiglia segnale** può essere eventualmente nascosta o semplificata andando a cambiare il livello di dettaglio della vista; è inoltre possibile indicare l'altezza del posizionamento.
Le **famiglie idranti sono state create a partire da "modello generico"** semplice, non basato su muro, nonostante a prima vista, quest'ultima via possa sembrare la via migliore. Sebbene infatti nel modello antincendio siano visibili i muri, essi sono in realtà parte del modello architettonico linkato; **non è possibile hostare un oggetto su una famiglia linkata** nel modello (come è il muro in questo caso).
L'utilizzo del modello generico permette di avere due modelli, architettonico ed antincendio, perfettamente indipendenti.

Famiglie estintori: Povere e CO_2

Gli estintori presentano una geometria appena accennata: un livello di dettaglio maggiore sarebbe stato non necessario ed anzi, avrebbe inutilmente appesantito il modello.
Il simbolo dell'estintore mostra la scritta CO_2 qualora il tipo dell'estintore sia effettivamente ad anidride carbonica.
Gli estintori presentano infine una famiglia hostata del cartello, analogamente a quanto visto per gli idranti.

Tubazioni Diametri

Le tubazioni su Revit sono famiglie di sistema. Non è possibile creare nuove famiglie tubazione; è però possibile definire diversi tipi della famiglia tubazione, i quali avranno caratteristiche differenti tra cui il materiale e i raccordi che utilizzerà.
Tra queste caratteristiche, tuttavia, non vi è il diametro.
Nel modello antincendio, tutti i **tubi sono suddivisi in 3 tipi (ARC,WTM e IDR)** e lo stesso tipo, ad esempio ARC, comprende tubi di diametro differente.
Ogni tipo di tubo incorpora una tabella formata da 3 colonne ed un numero variabile di righe; le 3 colonne contengono i 3 diametri che caratterizzano un tubo.
Dato un tipo di tubazione, è possibile assegnare una delle possibili terne di diametri andandola a scegliere da un elenco in cui è visibile soltanto il diametro nominale.
Il file .dwg di partenza, sul quale è stato modellato l'impianto sprinkler, presenta indicazioni riguardo al diametro delle tubazioni, espresse in pollici e probabilmente riferite al diametro nominale; non presenta però alcuna informazione relativamente a diametro esterno e interno. Tali informazioni sono state ricavate dalla tabella Schedule 40 della SCH. Essa fornisce una lista di diametri nominali espressi in pollici e relativi diametri esterni, spessori della parete del tubo e peso al metro (basandosi sul peso specifico dell'acciaio).
Esistono diverse schedule oltre alla 40; è stata scelta la seguente per mancanza di informazioni e per il fatto di essere molto comune.

Il **Diametro Interno**: è utile per calcoli fluidodinamici
Il **Diametro Esterno**: ha una funzione grafica e geometrica, utile per il controllo delle interferenze
Il **Diametro Nominale**: è un diametro fittizio utilizzato per identificare commercialmente il diametro del tubo.

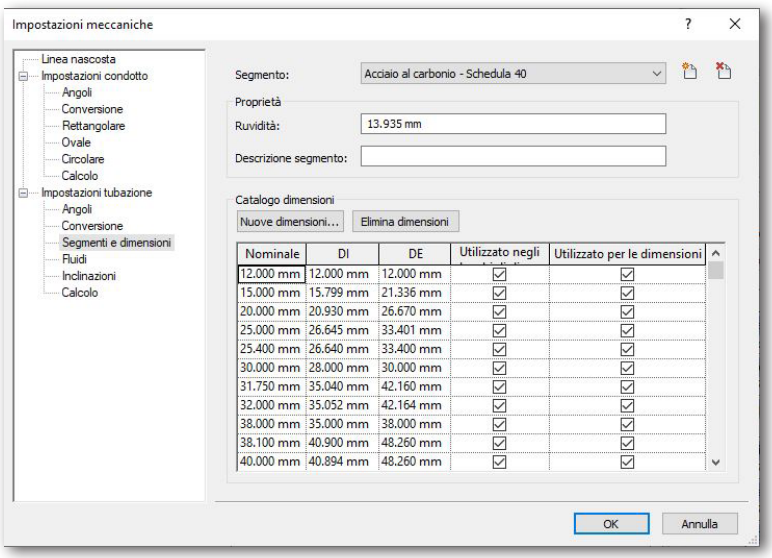


Immagine 44: Schedula 40 sulle impostazioni meccaniche di Revit

Il BIM Model Checking (BMC)

Quadro Normativo UNI 11337-5

La norma che in Italia si occupa dell'argomento BIM è la norma UNI 11337,norma che, attualmente, si divide in 10 parti, ovvero:

- Parte 1: Modelli Elaborati ed Oggetti
- Parte 2: Denominazione e Classificazione
- Parte 3: (Schede Informative) LOI e LOG
- Parte 4: LOD e Oggetti
- **Parte 5: Gestione Modelli ed Elaborati**
- Parte 6: Esempio Capitolato Informativo
- Parte 7: Qualificazione Figure
- Parte 8: PM / BIM-M
- Parte 9: Fascicolo del Costruito
- Parte 10: Verifica Amministrativa

Ai fini della tesi, assume particolare importanza la **parte 5** della norma: essa anzitutto descrive le fasi del processo che inizia con la definizione di requisiti da parte del committente e termina con la consegna di un file verificato e validato da parte di un soggetto affidatario.
Più precisamente, il committente è chiamato a redigere un documento chiamato **Capitolato Informativo** nel quale includere tutte le richieste a base di gara.
Sebbene il Capitolato Informativo non sia legato ad avere una struttura particolare, è buona prassi che esso contenga almeno la descrizione della divisione del progetto in vari modelli federati, le regole per gestire le interferenze (Clash Detection) e per correggere le incoerenze informative e normative (si parlerà meglio di questi punti più avanti nella tesi).
Sulla base del capitolato informativo, diversi affidatari si impegnano a presentare ciascuno una **oGI (Offerta per la Gestione Informativa)** nella quale viene documentata l'offerta per il soddisfacimento delle esigente della

committenza formulate nel Capitolato Informativo.
Il Committente decreterà così il vincitore tra i diversi soggetti affidatari, vincitore che sarà poi chiamato a redigere un **pGI (Piano per la Gestione Informativa).**
Il pGI è sostanzialmente lo step successivo all'oGI ed in esso vengono descritte meglio le modalità con cui verrà creato il modello, i contenuti dell'offerta ed eventuali sub-affidatari.
La norma definisce 3 livelli di coordinamento, ovvero LC1,LC2 ed LC3.

- **LC1** : Si tratta del **coordinamento fra le informazioni di un singolo modello BIM**
- **LC2**: E' il coordinamento di **modelli BIM differenti**
- **LC3**: E' il coordinamento svolto **tra modelli BIM ed altri elaborati non grafici**, come ad esempio relazioni tecniche, grafici cad od abachi .pdf.

Di seguito si riporta un flow-chart preso direttamente dalla normativa che rappresenta le fasi che riguardano l'LC2 e l'LC3.

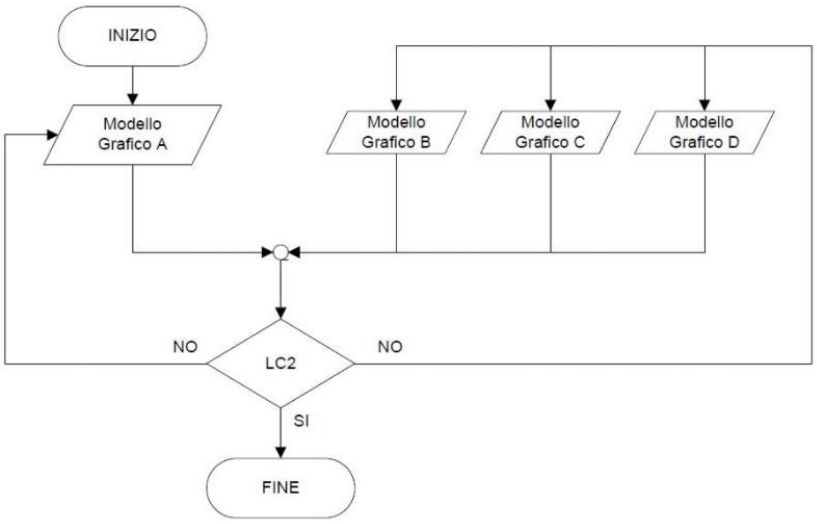


Immagine 45: Flusso di coordinamento livello 2 (diagramma da UNI 11337-5)

Alla luce di quanto detto, si può affermare che i **controlli effettuati all'interno di questa tesi rientrano nei primi due livelli di coordinamento**; per quanto ci si sia accertati più volte della congruenza dei dati all'interno dei modelli con documenti esterni,

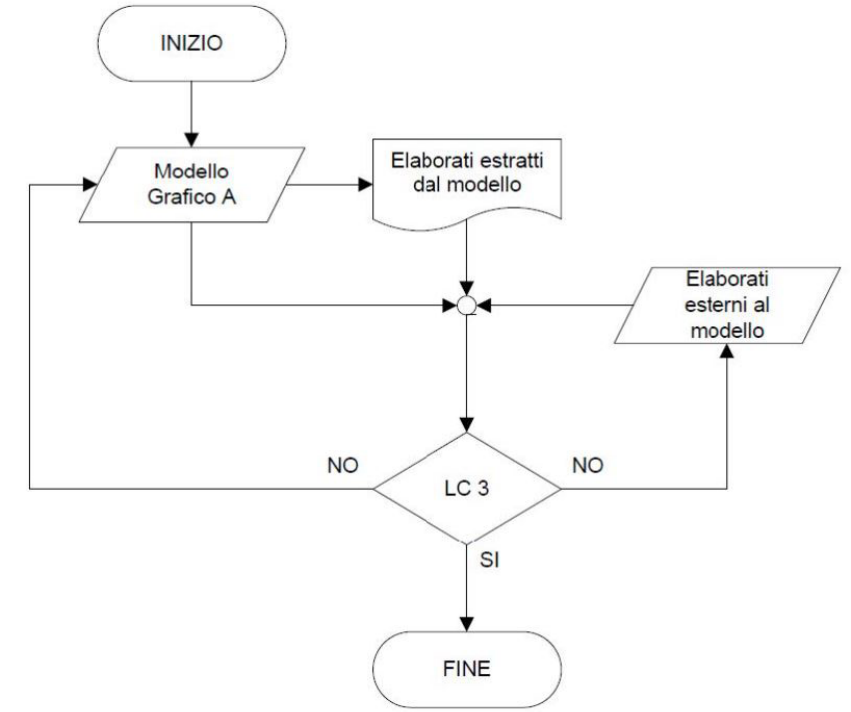


Immagine 46: Flusso di coordinamento livello 3 (tabella da UNI 11337-5)

Per quanto riguarda invece le **verifiche dei dati e della loro correttezza**, essa avviene attraverso tre livelli di verifica:

- **LV1**: E' una verifica della corretta **modalità di creazione, consegna e gestione delle informazioni all'interno dei vari modelli federati**. Tale verifica si basa su quanto indicato all'interno del Capitolato Informativo e nel pGI. Si tratta di una verifica interna e formale.
- **LV2**: è sempre una verifica interna ma, questa volta, di tipo sostanziale; essa ha l'obiettivo di accertare la **leggibilità**, la **tracciabilità** e la **coerenza delle informazioni contenuto all'interno dei vari modelli**.
  Si occupa inoltre di verificare che sia stata raggiunta una adeguata evoluzione informativa dei modelli, sempre in base a quanto scritto nel capitolato informativo e nel pGI.
- **LV3**: è la **verifica di cui si occupa il committente** per accertarsi della validità del materiale ricevuto

Si riporta di seguito uno schema che rappresenta il percorso di un modello A attraverso i tre livelli di verifica, tra cui, il controllo LV2 che vede la presa in esame di un altro modello A.

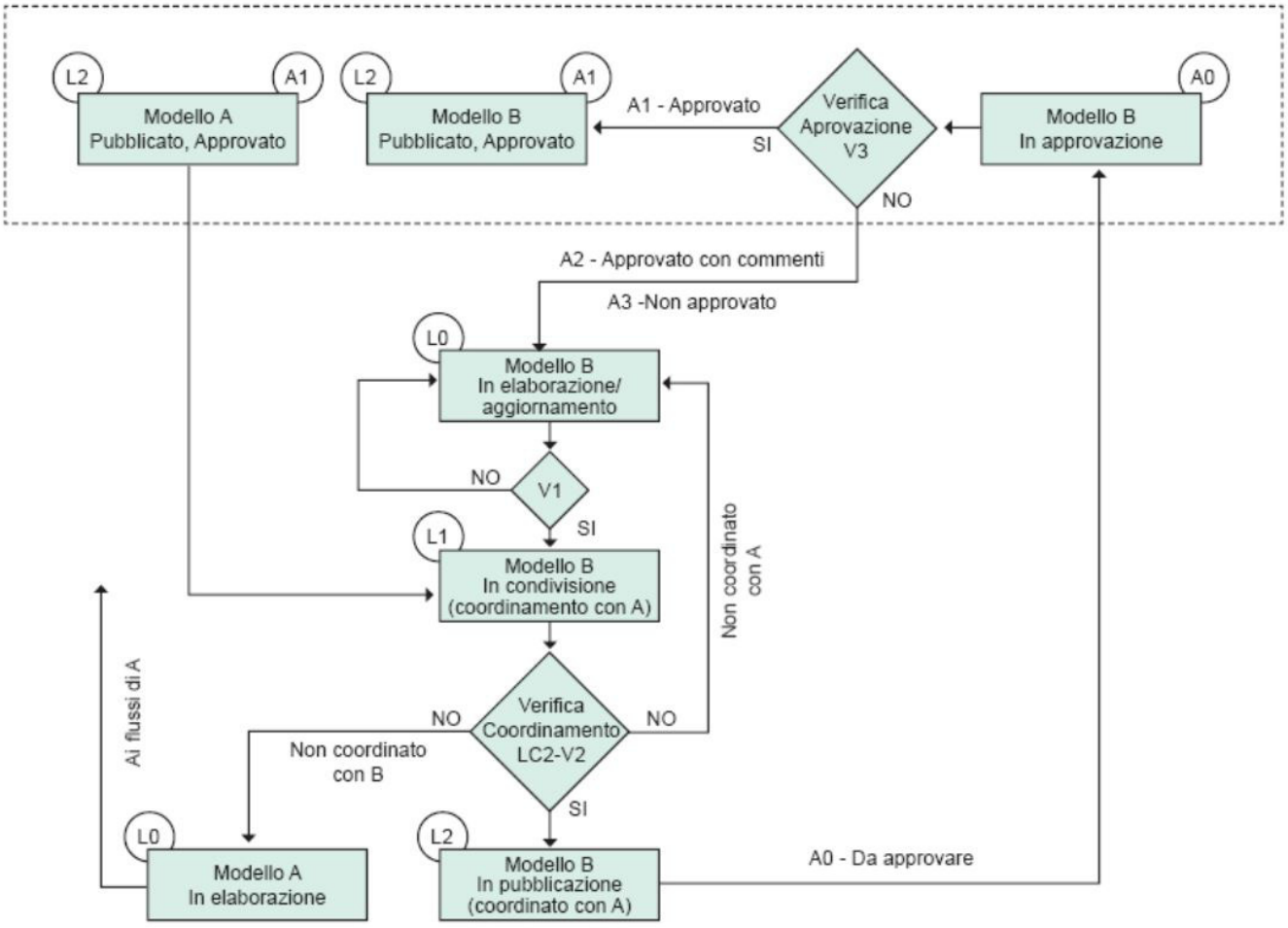


Immagine 47: Flusso di coordinamento, pubblicazione, verifica e approvazione.

Quadro Normativo UNI 11337-7

La parte 7 nella Norma UNI 11337 definisce le **varie figure professionali all'interno del processo BIM** ed i relativi compiti.
Esse, in particolare, sono il BIM Specialist, il BIM Coordinator ed il BIM Manager.

- Il **BIM Specialist** è, di solito, un tecnico specializzato in un singolo ambito disciplinare e opera a livello di una singola commessa. Sulla base dei documenti contrattuali redatti dal BIM Manager, modella appunto i modelli e le famiglie in esso contenute. Lavora dunque in prima persona sui file, rappresentando la **base del processo BIM.**
- Il **BIM Manager** è la figura professionale che opera al livello più alto e si occupa della **gestione generale del processo**. Coordina e supervisiona le commesse, si occupa degli aspetti contrattuali tra cui la redazione dell'oGI e pGI, definisce ed aggiorna le linee guida ed analizza i dati.
- Il **BIM Coordinator** è una figura a metà tra le precedenti. Opera a livello della singola commessa ed è chiamato a **coordinare il lavoro dei BIM Specialist** . Gestisce in prima persona l'organizzazione dei progetti sulla base di quanto deciso dal BIM Manager e si occupa di eventuali interferenze geometriche ed incoerenze. Il BIM Manager è chiamato a controllare la correttezza dei modelli e del loro contenuto informativo e, nel farlo, può ricorrere a controlli automatici e programmazione.

La tesi ha come argomento la BIM Validation ed il controllo di dati tramite programmazione visiva e ciò è parte di un processo più generale chiamato BIM Model Checking demandato, il più delle volte, al BIM Coordinator.

Il BIM Model Checking

Per Model Checking si intende tutta una serie di **operazioni**, spesso a carico del BIM Coodinator **volte a verificare che il materiale modellato soddisfi determinati requisiti imposti e che dunque sia "di buona qualità"** sia in termini geometrico-spaziali che nell'ottica di verifiche normative.
La mancanza di un'attività di Model Checking rischia di portare ad avere modelli non aderenti alla situazione reale, inaffidabili, eccessivamente pesanti in termini di memoria su disco e carichi di parametri mal compilati ed inutilizzabili.
In breve, si può affermare che il BIM Model Checking sia quell'attività che permette ad un modello BIM di sfruttare il vantaggio che ha rispetto ad un file CAD o ad un modello 3d generico.
L'attività di Model Checking è molto varia ma è possibile schematizzarla in 3 momenti differenti, ovvero:

- La BIM Validation
- La Clash Detection
- Il Code Checking

BIM Validation

Prima di procedere alle successive fasi del BIM Model Checking e ad altre analisi avanzate, è necessario effettuare un pre-check del contenuto informativo del modello attraverso un set di regole che ne validi la correttezza.
La BIM Validation, attraverso la gestione di un **opportuno set di regole parametriche** e sulla base di analisi logiche e semantiche, analizza e **determina il livello di qualità e coerenza interna di un BIM Model** garantendo l'estrazione di risultati affidabili per successive fasi di analisi; **controlla che tutti gli elementi siano stati nominati e classificati correttamente**. **Assicura** poi che il modello contenga **tutte le informazioni necessarie per un controllo avanzato**, inclusi tutti quegli attributi alfanumerici che, in un processo di Information Management correttamente strutturato, rientrano tra i requisiti individuati in fase di redazione del BIM Execution Plan (BEP) e che sono fondamentali per lo scambio informativo tra progettisti differenti, nonché per la completezza ed affidabilità dei documenti estratti dal Building Information Model.

La BIM Validation ha poi il compito di **garantire la comunicazione tra software di modellazione BIM e BIM tool di Model Checking**: le regole di controllo e il modello devono infatti contenere la medesima semantica, il che significa che gli oggetti parametrici contenuti nel modello devono necessariamente poter essere riconosciuti e mappati dallo strumento di Model Checking attraverso alcune proprietà.
Le criticità rilevabili nella fase di BIM Validation riguardano il o contenuto alfanumerico, o l'aspetto meramente geometrico dello stesso. Si possono dunque rilevare due tipologie di errore**: errore di modellazione ed errore di progettazione**.
E' indispensabile controllare anche la correttezza della modellazione tridimensionale; per esempio, l'erronea modellazione di due pavimenti sovrapposti si ripercuoterebbe in un doppio conteggio dei materiali costituenti il relativo pacchetto stratigrafico durante la fase di computazione.
Altre criticità rilevabili sono attribuibili ad errori progettuali e vengono individuate tramite le potenzialità di analisi logica del funzionamento dei componenti edilizi delle quali dispongono alcuni degli strumenti di Model Checking. Questo permette di controllare un'eventuale mancanza di coerenza progettuale come, a titolo esemplificativo, il corretto dimensionamento di un infisso rispetto alla quota di un controsoffitto.
Infine, la BIM Validation consente di **analizzare l'interezza del contenuto informativo associato ad un singolo oggetto** parametrico e quindi, di validarne il relativo **Level Of Development (LOD)** in funzione di quanto specificato nel BIM Execution Plan.
A ogni LOD corrispondono infatti diversi attributi che devono essere necessariamente definiti e compilati per ogni oggetto. Si consideri, ad esempio, un elemento Porta per il quale, a un determinato LOD, corrispondono degli attributi quali "Resistenza al Fuoco"o "Tipo di Apertura".
La validazione del contenuto informativo verifica l'effettiva presenza e corretta compilazione di tali parametri al fine di un confronto tra quanto dichiarato e quanto effettivamente modellato, ponendosi a supporto di un corretto flusso informativo tra le parti interessate.

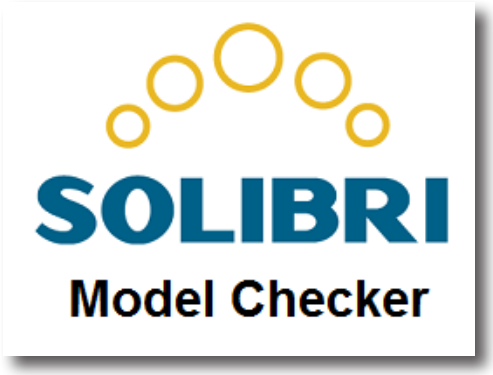


Immagine 48: Logo Solibri Model Checker

Immagine 49: Icona degli avvisi su Revit, utile per una prima Clash Detection preliminare

Clash Detection

Per Clash Detection si intende il **controllo delle interferenze geometriche e spaziali** all'interno del modello ed assume particolare importanza per modelli federati; essa ha, in prima battuta, l'obiettivo di evitare compenetrazioni tra oggetti, problemi che se non preventivamente identificati, si ripercuoterebbero pesantemente sui costi di costruzione in cantiere.
Una **Clash Detection più raffinata** permette poi di garantire, in una seconda fase, le dovute **tolleranze di posa**, necessarie per le operazioni in cantiere e i dovuti **spazi funzionali**: si pensi ad esempio alla verifica del raggio di rotazione di una sedia a rotelle all'interno di un ascensore.
Revit al suo interno possiede un modulo per controllare interferenze geometriche inserendole all'interno degli "avvisi". Esistono però altri software compatibili con altri software BIM che permettono di effettuare la Clash Detection. Tra i più importanti, si cita Soliri Model Checker.

Code Checking

Il Code Checking valida il progetto andando ad **elaborare dati contenuti nel modello per confrontarli con normative e codici di riferimento**. E' il caso, ad esempio, della verifica del rapporto aeroilluminante nelle stanze, verifica che con una metodologia BIM può essere svolta automaticamente per ogni stanza, evitando dunque un controllo a campione inevitabilmente destinato ad essere meno preciso.
Altri esempi di code Checking potrebbero essere la verifica del copriferro nelle armature dei pilastri, la verifica delle vie di esodo all'interno di una stanza, o dei requisiti normativi in ambito energetico.

Eilf Hjelset è un professore associato della Oslo and Akershus University College of Applied Sciences, in Norvegia

Tassonomia dell'errore

Eilf Hjelset, in una pubblicazione intitolata "*Classification of BIM-based Model checking concepts*" fa una **categorizzazione** differente di tutte le modalità in cui può avvenire appunto, il **Model Checking**, indipendentemente dai contenuti dello stesso ma piuttosto basata sulla tipologia di controllo effettuabile e sui risultati attesi.
Il risultato di questa suddivisione è riportato nella seguente tabella che segue

Concept group	Concept type	Purpose of checking	Outcome	Examples
Compliance checking	Validation checking	Validation	pass/fail	clash detections code compliance
	Model content checking	Content of information	a filtered list	relevant information for exchange
Design solution checking	Smart object checking	Integration (adaptation)	a modified model	size of building parts (objects) related to the building (model)
	Design option checking	Guidance	options and advice	knowledge system for selecting relevant solutions

Immagine 50: Classificazione del Model Checking secondo Eilf Hjelset

Hjelset divide i vari "concept type" in due categorie, ovvero il **Compliance Checking** e il **Design Option Cheking**; a loro volta, queste due gruppi concettuali si dividono in Validation Checking e Model Content Checking per la prima e Smart Object Checking e Design Option Checking per la seconda

Validation Checking

Il principio base del Validation Checking è il **confronto tra un dato ed un set di regole**. I risultati possibili di un controllo di questo tipo sono essenzialmente tre e, per rappresentarli, torna utile riportare i diagrammi della pubblicazione di Hjelset:

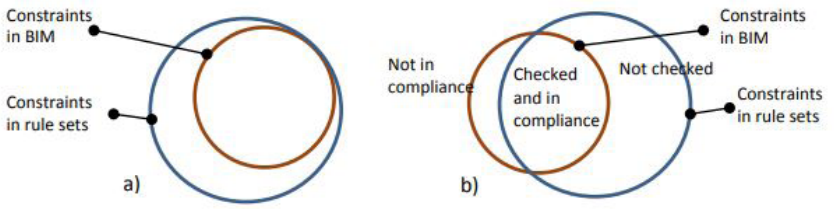


Immagine 51: I casi possibili nel Validation Checking

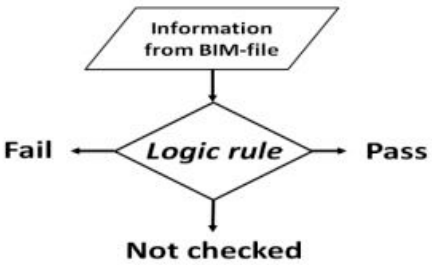


Immagine 52: I risultati possibili nel Validation Checking

nelle rappresentazioni, i cerchi rossi si riferiscono all'insieme delle Costraints in BIM, ovvero, in questo caso, dei contenuti dei parametri da verificare, mentre le circonferenze blu rappresentano i casi accettati dal set di regole impostato.
Il caso a rappresenta una situazione in cui tutti i parametri sono stati testati e tutti quanti rientrano all'interno delle regole stabilite.
Il caso b rappresenta invece un caso in cui sono presenti tutte e tre i casi di output possibili di un Compliance Checking, ovvero il caso di un parametro che rispetta le regole, di uno che non le rispetta ed infine, di uno non testato.

Esempio:
Se si volesse testare il soddisfacimento di requisiti REI degli elementi di chiusura di una compartimentazione antincendio, il software dovrebbe confrontare il parametro `REI` di ogni elemento, definendo se il numero di minuti di resistenza al fuoco garantiti dall'oggetto sia sufficiente per i requisiti normativi. Se però non dovesse trovare tale parametro `REI`, necessariamente il software si troverebbe costretto a segnare come "Not Checked" l'elemento in quanto impossibile da testare.
Questo esempio rappresenta un tipico caso di Code Compliance ma, se fossero entrati in gioco parametri geometrici, si sarebbe potuto parlare di Clash Detection.

Model Content Checking

Il Model Content Checking controlla che il modello contenga il giusto numero di informazioni per un uso specifico. Si tratta di solito di un'operazione che non richiede l'utilizzo di software o metodologie particolari ma viene invece svolta manualmente durante tutto il processo di lavoro sul modello e talvolta, richiede molto tempo. Essa è utile soprattutto nella consegna del modello da un professionista all'altro o prima di altri tipi di controllo, come la verifica della compilazione dei parametri.
Il principio logico del model Content Checking è molto semplice: il **contenuto BIM** viene **confrontato con una lista definita di informazioni rilevanti**, il che porta a due opzioni: **identificato** o **non identificato**. A quel punto, è possibile svolgere un report dei quali informazioni sono state trovate e di quali no e decidere, sulla base della lista

di informazioni cosa aggiungere (e cosa togliere, in caso di modelli contenenti troppe informazioni superflue per un determinato utilizzo).

Design Solution Cheking

Si basa sul testare il buon funzionamento di Smart Objects e sul Design Option Checking. Entrambi sono concetti ampiamente esplorati in certi settori industriali, come aviazione o ingegneria meccanica ma, ad oggi, trovano **riscontro limitato nel settore delle costruzioni.**
Per **Smart Object**, si intende, in ambiente di modellazione 3d parametrica (non solo in ambiente BIM, ma anche in CAM) **un oggetto che si adatta automaticamente in base alle condizioni dell'ambiente circostante**. Volendo fare un esempio, si potrebbe pensare ad una finestra settata per regolare automaticamente le proprie dimensioni per fare rispettare il rapporto aeroilluminante dell'ambiente che serve.
La base del **design option checking** è basata su una regola che **identifica una situazione predeterminata e**, sulla base di questa, **presenta al progettista una rosa di possibili soluzioni** da prendere in considerazione. Il concetto base è lo stesso che utilizzano diversi shop online nella sezione "consigliati". Di questa pratica vi sono ancora meno esempi rispetto alla precedente, nell'ambito delle costruzioni.

Il lavoro svolto per questa tesi è basato sull'applicazione del concetto di **Model Validation** ed, in particolare, sul controllo della correttezza formale dei dati e della nomenclatura e organizzazione degli oggetti.

CAM (Computer-aided Manifacturing) è un ambiente di lavoro 3d che associa ad informazioni geometriche, altre numeriche nell'ottica di gestione ed analisi dei progetti di macchinari.

In parole povere, si tratta del corrispettivo BIM calato nell'ambiente dell'ingegneria meccanica. Il software Autodesk dedicato a tal scopo è Inventor.

Immagine 53: Logo Inventor

Nomenclatura delle famiglie

Tutto il lavoro in BIM sul complesso della Torre della Regione è stato impostato in modo che ogni oggetto abbia una codifica ben precisa. In particolare, **ogni famiglia del modello è nominata secondo un codice che riporta varie informazioni.**
Di seguito viene riportato, a titolo d'esempio, la nomenclatura di una famiglia già incontrata nella tesi: quella delle porte metalliche ad un battente.

TRP_IT_ARC_PM

Come ben evidente, si tratta di 4 campi composti da lettere maiuscole e divisi da underscore.

1° Campo = Si tratta di un codice di 3 lettere maiuscole che identificano il **progetto** nel quale è caricata la famiglia. In questo caso, come del resto in tutte le famiglie del progetto, il codice **TRP** indica la **Torre della Regione Piemonte**.

2° Campo = Indica in quale parte del progetto è caricata la famiglia. Si tratta, sostanzialmente del parametro **`Edificio`** che viene compilato per ogni famiglia.

Come si vedrà nei capitoli successivi, i primi due campi non sono altro che i due parametri condivisi "Progetto" ed "Edificio".
Per ulteriori informazioni, si rimanda il lettore a "Parametri Condivisi del Progetto" a pagina 59

3° Campo = Indica la **disciplina di appartenenza della famiglia**, cui corrisponde anche un diverso modello. Si può anzi affermare che, con questo campo e con il precedente si riesce a capire in quale file/modello del progetto è possibile trovare la famiglia in questione.
In questo caso, **ARC** sta per ***Architettonico***.

Come si vedrà nei capitoli successivi, l'ultimo campo formato da due lettere non è altro che il parametro condiviso `Codice Famiglia`.
Tale campo si compila solamente per le famiglie caricabili in quanto, per le famiglie di sistema, il nome della famiglia non è modificabile. Si pensi ad esempio al caso dei muri semplici

4° Campo = Si compone di due lettere maiuscole che identificano la **famiglia**. In questo caso, PM starebbe per Porte Metalliche mentre, citando un caso simile, PD sarebbe l'abbreviazione di Porte Doppie Metalliche.

Nome dei tipi

Il nome del tipo segue una definizione molto meno restrittiva e prevede un campo di **2 lettere maiuscole** che definiscono la funzione del tipo di famiglia, **seguite da un underscore** e da altre informazioni generali che variano di caso in caso.
A titolo d'esempio, si prendano le porte metalliche:

PO_03_d

E' il nome di un tipo di porta, in cui "PO" rappresenta la funzione, mentre 03_d identificano la tipologia di porta all'interno dell'abaco.

Parametri Condivisi del Progetto

Le linee guida del progetto prevedono che ogni istanza abbia 12 parametri condivisi definiti e compilati.
Di seguito verranno analizzati e spiegati tutti questi parametri.

Per una spiegazione sul cosa siano i Parametri Condivisi, si rimanda il lettore a pagina"Parametri Condivisi" a pagina 17

Progetto

Parametro di istanza

E' un codice di 3 lettere maiuscole **rappresentativo del progetto in oggetto**. In questo caso, esso è sempre "TRP" trattandosi del progetto della Torre della Regione Piemonte.

Edificio

Parametro di istanza

Si tratta di un codice di 2 lettere maiuscole che indicano la **porzione del progetto** in cui è stata modellata la famiglia.
Si riporta la tabella con tutte le porzioni e le relative sigle.
Oggetto di tesi è stata solo la modellazione degli I**nterrati Torre** e dunque è sempre stato compilato con "**IT**".

Codice Categoria

Parametro di tipo

E' un codice di 2 lettere maiuscole utilizzato per i**dentificare la categoria di una famiglia**.
A lato si riporta la tabella estratta dalle linee guida del progetto che accoppia ad ogni categoria il relativo codice.

Categoria	Codice Categoria
Accessori per condotti	AC
Accessori per tubazioni	AT
Apparecchi Elettrici	AE
Apparecchi idraulici	AI
Apparecchi per illuminazione	AL
Armatura strutturale	AS
Arredi	AR
Arredi fissi	AB
Attrezzatura elettrica	AF
Attrezzatura meccanica	AM
Bocchettoni	BO
Collocazioni condotto	CA
Collocazioni tubazione	CB
Condotto	CN
Condotto flessibile	CF
Contesto	CC
Controsoffitti	CD
Dispositivi allarme incendio	DA
Dispositivi dati	DT
Dispositivi di comunicazione	DC
Dispositivi di illuminazione	DI
Dispositivi di sicurezza	DS
Finestre	FI
Fondazioni strutturali	FS
Isolamenti tubazioni	IT
Massa	MA
Modelli generici	MG
Montanti della facciata continua	MO
Muri	MU
Pannelli della facciata continua	PA
Pavimenti	PV
Pilastri	PL
Porte	PO
Raccordi condotto	RC
Raccordi tubazione	RT
Scale	SC
Sistemi di arredo	SA
Sistemi di facciata continua	SF
Sistemi di travi strutturali	ST
Telaio strutturale	TS
Tubazione	TU
Tubazione flessibile	TF
Verde	VE

Immagine 54: Codici Categoria

Codice Famiglia

Parametro di tipo

E' un codice di due lettere maiuscole che **identifica** univocamente **una famiglia nel progetto**. Viene definito volta per volta per le famiglie caricabili e compare nel nome delle stesse.
Per le famiglie di sistema, esso coincide con il `Codice Categoria`.

Classi di unità Tecnologiche, Unità Tecnologiche, Classi di Elementi Tecnici

Parametri di tipo

Definiscono la funzione della famiglia. La compilazione di questi parametri è fatta sulla base della **norma UNI 8290**.

Codice MasterFormat, Titolo MasterFormat

Parametri di tipo

Si tratta di parametri derivati dalla codifica associata a **codice MasterFormat per elementi funzionali definita da CSI CODE.**
Ad ogni definizione (il Titolo) è assegnata una sequenza di numeri (Codice) che la individua univocamente.

Identificativo

Parametro di istanza

Si tratta di un **parametro che identifica univocamente ogni singola istanza del progetto**.
Il parametro identificativo si forma unendo più "pezzi", separati da underscore.
Si compone del parametro `Progetto`, `Edificio`, di tre lettere maiuscole indicanti la **disciplina**, del `Codice Famiglia`, del **nome del tipo**, del **livello,** della **quota del livello di riferimento** e di un **numero progressivo** formato di 5 cifre.
Il livello, in particolare, è un campo formato di 4 caratteri di

Revit, automaticamente, assegna ad ogni singola istanza un **codice ID**. Il codice ID è utile in varie situazioni, ad esempio per individuare nel modello i segnali di errore inviati dal software, ma ha il problema di essere modificato più volte dal programma in fase di modellazione.
L'ID, inoltre, identifica univocamente ogni istanza di un solo modello, ma è assolutamente **possibile avere due ID coincidenti su due modelli differenti ma linkati**, il che annullerebbe l'univocità, ovvero la caratteristica chiave che deve avere il parametro identificativo.

cui i primi due sono o LF (livello fuori terra) o LI (Livello interrato) e gli ultimi 2 rappresentano il livello.
Nel caso degli interrati, si è sempre lavorato con 3 livelli, ovvero LI02, LI01 e LF00.

Codice Esistente

Parametro di Istanza

Si tratta di un parametro **utilizzato per mantenere le informazioni da CAD a BIM**.

Esempio
Tutte le porte sul file .dwg sono caratterizzate da un'etichetta che le individua univocamente e che consente di cercare le relative informazioni sull'abaco .pdf.
Esso non ha una struttura uguale per tutte le famiglie, né è compilato per tutte.
Il `Codice Esistente`, nel caso delle porte, riporta gli stessi valori del `Codice Progetto Esecutivo` e, nel caso dei Locali, i valori del `Codice Locale Costruttivo`.

Vedi:
"Codice Progetto Esecutivo" a pagina 34
"Codice Locale Costruttivo" a pagina 35

Affidabilità

Parametro di istanza

Rappresenta l'affidabilità di una certa istanza modellata.
Esso può assumere 3 valori numerici

- 1: misure in sito
- 2: misure da .dwg
- 3: ipotizzato

Codice Padre

Parametro di istanza

Si tratta del **codice identificativo dell'oggetto da cui dipende l'istanza in questione**. Esso viene **compilato soltanto per componenti MEP**.

Tutti questi parametri condivisi, assieme a `Famiglia` e `Tipo`, compongono gli **Abachi CPR** (ve ne è uno per ogni categoria)

Controllo nomenclatura di Famiglia e Tipo

Può capitare che un BIM Manager, o **BIM Coordinator**, o comunque, una figura professionale preposta al controllo dell'uniformità del lavoro svolto da diversi professionisti su diverse parti, **voglia controllare che le famiglie** dei vari modelli **seguano la codifica stabilita in fase preliminare**, presentata poco fa.
Un modo per svolgere questo controllo sarebbe quelli di andare sul browser di famiglia per avere un elenco di tutte le famiglie e relativi tipi, suddivise per categoria e modificare da lì gli errori.
Sebbene si tratti, tutto sommato, di un'operazione che richiederebbe pochi minuti, ciononostante potrebbe diventare problematica nel momento in cui si volessero controllare molti modelli più volte. Si pensi, ad esempio, al caso in cui questo controllo debba essere fatto quotidianamente per ogni modello su cui si sta lavorando. In una situazione di questo tipo, potrebbe tornare utile un algoritmo che controlli velocemente ed automaticamente tutti i nomi di tutte le famiglie e che indichi, in modo chiaro, in quali punti è necessario correggere un errore.
In questo modo, si riuscirebbe a ridurre da pochi minuti a pochi secondi, il controllo quotidiano di un modello, con ovvi vantaggi dal punto di vista delle tempistiche e della precisione.
Il cuore della tesi è proprio la creazione di uno script di questo tipo in ambiente Dynamo.
Seguiranno poi altri script che si occuperanno di controllare la compilazione dei parametri condivisi, andando eventualmente a correggerli.

Dynamo

Immagine 55: Logo Dynamo Revit

Si tratta di un software open source e quindi gratuito che permette la **programmazione visuale**.

Programmazione Visuale (VPL): Vantaggi e svantaggi

Per programmazione visuale, si intende la **creazione di algoritmi** senza l'utilizzo di codici alfanumerici quanto piuttosto **attraverso** il collegamento di **nodi** che rappresentano funzioni.

Il vantaggio di un approccio di questo tipo è dato dalla **maggiore semplicità di utilizzo** e chiarezza nella comprensione della gestione dei dati. È infatti più facile, con un colpo d'occhio, rendersi conto del percorso di un dato che entra in una funzione per uscirne trasformato piuttosto che vedere la stessa trasformazione attraverso le righe di un codice.

L'aspetto negativo di un approccio visuale sono invece le dimensioni grafiche che può assumere un programma particolarmente complesso, assieme alla confusione che potrebbe creare un elevato numero di collegamenti tra un nodo e l'altro.

Si pensi ad esempio al caso in cui un'unica variabile è usata come input di decine di nodi; ciò porterebbe ad avere un ambiente di lavoro carico di collegamenti che possono creare confusione. Quest'ultimo aspetto, tuttavia, non è così importante per quello che è l'utilizzo che ne viene fatto su Revit; si tratta infatti di lavorare quasi sempre con programmi relativamente semplici. Va inoltre detto che nel caso fosse necessaria una programmazione testuale, è sempre possibile utilizzare applicazioni basate su Python o, addirittura, API scritte in linguaggio C#.

Questa tesi non tratterà questi ultimi due casi.

VPL è un acronimo che sta per **Visual Programming Language**

Dynamo per Revit

Nelle ultime versioni di Revit, Dynamo è integrato come **plug-in interno al programma**: in questo modo, questo potente strumento risulta funzionale per un'infinità di mansioni che possono prendere come input, i dati di un file Revit e hanno effetto sempre sul medesimo file BIM.
È, ad esempio, possibile definire in Dynamo, solidi 3d parametrici che possono essere importati direttamente in ambiente BIM. La sperimentazione di tale approccio ha portato al nascere della cosiddetta architettura parametrica, caratterizzata dall'assenza di angoli retti e da figure molto dinamiche ed avvenieristiche. Senza arrivare a casi estremi, la modellazione geometrica fatta su Dynamo può poi essere di grande aiuto in quei casi in cui si abbia da modellare un grande numero di oggetti identici a distanza costante, il che capita di sovente nelle infrastrutture o in edifici di grandi dimensioni.
L'**utilizzo più frequente che si fa con Dynamo Revit è però editare velocemente grandi moli di dati in pochi secondi.**
In questo caso, l'obiettivo del lavoro è stato quello di creare un programma con Dynamo che facesse automaticamente il controllo dell'esattezza della compilazione.

Immagine 56: Heydar Aliyev Centre a Baku, di Zaha Hadid è uno degli esempi più celebri di architettura parametrica.

archi+lab

Immagine 57: Logo del Pacchetto Archi-lab.net per Dynamo

clockwork for dynamo

Immagine 58: Logo del Pacchetto Clockwork per Dynamo

bimorph DIGITAL ENGINEERING

Immagine 59: Logo del Pacchetto BIMorph per Dynamo

Reg[ular] Ex[pression]

Immagine 60: Logo del Pacchetto RegEx per Dynamo

Pacchetti

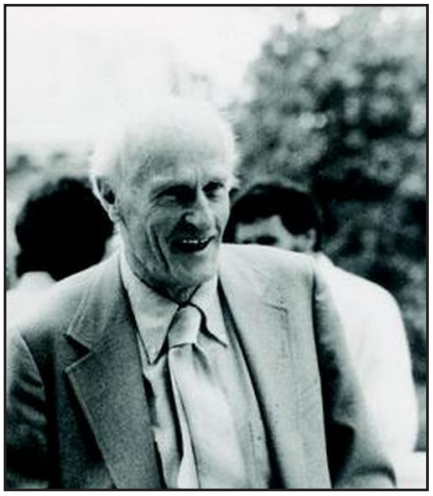
Di default, Dynamo presenta un certo numero di blocchi, insufficienti per creare la maggior parte degli script più complessi. E' però possibile scaricare da internet altri **pacchetti di nodi**. Il concetto è lo stesso delle cosiddette librerie utilizzate dai linguaggi di programmazione.
I pacchetti utilizzati in questa tesi (nonché tra i pacchetti più scaricati) vi sono:

- archi-lab.net
- Clockwork
- BIMorph
- RegEx

RegEx

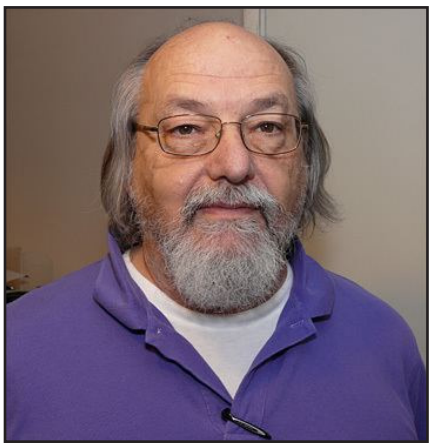
Prima di introdurre i vari script di Dynamo che sono stati creati, è necessario introdurre il concetto di RegEx.

Con il nome "**RegEx**", si intende "**Regular Expressions**", ovvero una funzione che serve a filtrare, identificare e confrontare stringhe di codice.
Sono state definite per la prima volta da **Stephen Kleene** nel **1950** e furono implementate per la prima volta da **Ken Thompson** nell'editor QED, nel **1966**. Ancora oggi sono utilizzate in svariati ambiti e trasversalmente in molti linguaggi di programmazione (non si tratta dunque di costruzioni esclusive di Dynamo Revit).



Stephen Cole Kleene (1909-1994) fu un matematico Americano, tra i fondatori della teoria computazionale, la quale aiutò lo sviluppo della moderna informatica.

Per capire di cosa si tratta, può tornare utile fare degli esempi.
Nella vita di tutti i giorni, si riesce quasi sempre a capire se una serie di numeri è un numero telefonico; la maggioranza dei numeri telefonici, infatti, si compongono di 10 o 9 cifre; essi poi cominciano con 3 o 0 e, a volte, presentano uno spazio dopo i primi 3 numeri (o 4, se il primo numero è 0). Tale spazio separa il prefisso dal numero vero e proprio.
Si è in grado di riconoscere un indirizzo mail dalla particolare disposizione di caratteri alfanumerici seguiti da una @, da un'altra stringa (per esempio "gmail") ed ancora da un punto e 2/3 lettere (ad esempio ".it").
Si riconosce un codice fiscale dal numero di caratteri alfanumerici, e dal modo in cui sono disposte lettere maiuscole e cifre.



Kenneth Lane Thompson (1943-) fu un pioniere delle scienze informatiche. E' conosciuto soprattutto per aver sviluppato il sistema operativo **Unix**. Inventò il linguaggio di programmazione **B**, diretto predecessore del più famoso linguaggio C (base di tutti i moderni linguaggi di programmazione). Diede inoltre un grande contributo nella creazione di **QED**, un rudimentale editor di testo.

Una RegEx è un codice che un computer può interpretare come istruzioni per capire se una stringa di caratteri soddisfa determinate regole definite.
Si comprende di come l'utilizzo delle stesse potrà essere il punto chiave del controllo di nomenclature e parametri nel modello.

Allo stesso modo di un linguaggio di programmazione, le RegEx seguono una loro grammatica. Di seguito si darà una breve spiegazione di tutte le costruzioni che sono state utilizzate

La lettera b, in questo caso, sta per "boundary", ovvero "confine/limite"

\b = rappresenta i **confini della della parola** e vengono applicati all'inizio ed alla fine della RegEx

Esempio

\bTO\b seleziona solo e soltanto una stringa che contiene le due lettere maiuscole T ed O, in questo ordine. Non selezionerebbe invece la parola "TORPEDINE", o "MATTONE".
\bTO selezionerebbe sia "TO" che "TORPEDINE" in quanto non sono poste condizioni per ciò che segue "TO".
TO, senza gli identificatori di confine, selezionerebbe sia "TORPEDINE", che "MATTONE" che "TO" in quanto tutte e tre le parole contengono la coppia di lettere, indipendentemente dalla posizione in cui sono.

\A e **\z** = Rappresentano rispettivamente l'**inizio e la fine della stringa**. Come il caso precedente, si applicano rispettivamente all'inizio ed alla fine della RegEx

[A-Z] = **Seleziona qualsiasi lettera**, dalla A alla Z, **maiuscola**. Non si attiva invece con lettere minuscole o caratteri numerici.

[A-Za-z0-9] = Tale stringa **seleziona** anche **stringhe formate da caratteri minuscoli e cifre**

{2} = specifica di **quanti caratteri deve essere lunga la stringa**.

ad esempio ***[A-Z]{2}*** accetta soltanto coppie di lettere maiuscole

| = tale carattere rappresenta l'**operatore logico OR** (**ovvero "o"**). Le alternative vanno racchiuse tra parentesi tonde.

Esempio
\b(1|2|3)\b accetta solo e soltanto o un numero 1, o un numero 2, o un numero 3, ma non un numero 5 e nemmeno un numero 32.

\ = la barra al contrario serve **per fare interpretare alla RegEx il carattere che segue non come un** carattere speciale (o **metacarattere**), ma piuttosto come un carattere normale "\".

Di seguito si fornisce la RexEx che controlla la codifica del parametro `Identificativo`, spiegando tutte le istruzioni. Aver compreso il funzionamento di questo codice è sufficiente per capire come funzionano tutti gli altri.

\ATRP_IT_ARC_[A-Z]{2}_[A-Z]{2}_([A-Za-z0-9]|-|_|\"|&|'|%|*|:|;|\]|\[|\(|\)|\\|\.)+_L(I|F)[0-9]{2}_[0-9]{3}\.[0-9]{2}_[0-9]{5}\z

La stringa deve cominciare necessariamente con i caratteri TRP_IT_ARC_.
A seguire, due lettere maiuscole, un underscore ed altre due lettere maiuscole seguite da un altro underscore.
Segue ancora una stringa di n caratteri o di lettere (maiuscole o minuscole) o di numeri, o di caratteri tra quelli compresi all'interno della parentesi. Si ricorda che il simbolo \ serve a rendere il simbolo che segue un non-metacarattere.
Segue poi una L preceduta da un underscore, una lettera maiuscola tra I o F e ancora due numeri.
Segue ancora, compreso tra due undescore, due numeri, rispettivamente di 3 e 3 cifre, separati tra loro da un punto (l'altezza del livello).
Infine, un numero necessariamente di 5 cifre (per esprimere "10" si dovrà dunque scrivere "00010").

Parametro	RegEx	Tipo/Istanza
Nome Famiglia	\bTRP_IT_ARC_[A-Z]{2}\b	Tipo
Nome Tipo	\b[A-Z]{2}_	Tipo
Progetto	\bTRP\b	Istanza
Edificio	\bIT\b	Istanza
Codice Categoria	\b(AC|AT|AE|AI|AL|AS|AR|AB|AF|AM|AN|BO|CA|CB|CN|CF|CC|CD|DA|DT|DC|DI|DS|EA|FI|FS|IT|MA|MG|MO|MU|PA|PV|PL|PS|PO|RC|RT|SC|SA|SF|ST|TS|TU|TF|VE|TT)\b	Tipo
Codice Famiglia	\b[A-Z]{2}\b	Tipo
Classi di Unità Tecnologiche	\A(([0-9]{1}|*)|NA|ND)\z	Tipo
Unità Tecnologiche	\A(([0-9]{1}|*)\.([0-9]{1}|*)|NA|ND)\z	Tipo
Classi di Elementi Tecnici	\A(([0-9]{1}|*)\.([0-9]{1}|*)\.([0-9]{1}|*)|NA|ND)\z	Tipo
Codice Master Format	\b([0-9]{2} [0-9]{2} [0-9]{2}|NA|ND)	Tipo
Titolo Master Format	\b([A-Za-z]{3,}|NA|ND)	Tipo
Codice Esistente	Nessuna RegEx	Istanza
Affidabilità	\b(1|2|3)\b	Istanza
Identificativo	\ATRP_IT_ARC_[A-Z]{2}_[A-Z]{2}_([A-Za-z0-9]|-|_|\"|&|'|%|*|:|;|\]|\[|\(|\)|\\\|\.)+_L(I|F)[0-9]{2}_[0-9]{3}\.[0-9]{2}_[0-9]{5}\z	Istanza
Codice Esistente (Porte)	\A(IT-(1|2|3) (N|S|C|E)(1|2|3|4|5|6|S) [0-9]{2}|PV[0-9]{1}\.[0-9]{1}|NA)\z	Istanza

Immagine 61: Le RegEx utilizzate nel corso del lavoro.

La tabella sopra riportata mostra le RegEx che saranno utilizzate in questa tesi per quanto riguarda il file Architettonico degli Interrati Torre.
I vari parametri sono trattati più approfonditamente a pagina 59.

Test su nomi di Famiglia e Tipo

Funzione

Il seguente script **individua tutti gli errori presenti nella nomenclatura delle famiglie e ne fornisce un report su Excel**.
Fornisce inoltre un **resoconto quantitativo dello stato di compilazione dei vari parametri condivisi**.
Sebbene tale script sia molto semplice e sebbene le funzioni che svolge saranno meglio eseguite dagli script successivi, esso è stato il primo ad essere stato creato, nonché la base di partenza per tutti gli altri

Selezionare e catalogare tutte le famiglie

Come prima cosa, è opportuno selezionare tutte le famiglie che dovranno passare al controllo del programma.
Non è possibile rendere automatica tale operazione in quanto non tutte le categorie di famiglie su Revit necessitano di un controllo di questo tipo; si pensi ad esempio alle Viste, o ai Cartigli. A causa di ciò, è stato necessario **selezionare manualmente tutte le categorie** oggetto di verifica.
Un secondo passo è la suddivisione tra famiglie di sistema e famiglie caricabili; le famiglie di sistema, infatti, non permettono di cambiare il nome della famiglia. Per intenderci, la famiglia "Muro" non potrà mai chiamarsi in un altro modo; a cambiare, al limite, sarà il nome del tipo.
Del resto, anche nel progetto preso in esame, ogni tipologia di muro corrisponde ad un diverso tipo. Ne consegue che non abbia senso verificare il nome della famiglia per le famiglie di sistema.
Si rende dunque necessario creare due elenchi: un **elenco** delle sole **famiglie caricabili** ed un altro delle sole f**amiglie di sistema**. Entrambi gli elenchi devono essere una selezione delle sole categorie i cui elementi devono essere soggetti a verifica della nomenclatura.
Tale problema è stato risolto, banalmente, creando due elenchi di nodi **category**, uno con le categorie di sistema e l'altro con le categorie caricabili.

Talvolta è comodo immaginare una lista a due livelli come una matrice, una tabella che organizza delle colonne di valori all'interno di righe

Di seguito si riporta l'elenco delle famiglie raccolte dallo script

Famiglie Caricabili

Accessori per condotti
Accessori per tubazioni
Apparecchi elettrici
Apparecchi idraulici
Apparecchi per illuminazione
Arredi
Arredi fissi
Attrezzatura elettrica
Attrezzatura meccanica
Attrezzature speciali
Dispositivi allarme antincendio
Dispositivi chiamata infermeria
Dispositivi di comunicazione
Dispositivi dati
Dispositivi di sicurezza
Dispositivi telefonici
Dispositivi di illuminazione
Pilastri
Pilastri strutturali
Telaio strutturale
Finestre
Porte
Bocchettoni
Estintori
Raccordi condotto
Raccordi tubazione
Verde

Famiglie di sistema

Condotto
Condotto flessibile
Collocazione condotto
Tubazione
Tubazioni flessibili
Collocazione tubazione
Controsoffitti
Pavimenti
Muri
Tetti

Entrambi gli elenchi vengono poi fatti convergere in un'ulteriore lista complessiva su due livelli (ovvero una lista di due voci le quali, a loro volta, contengono rispettivamente tutte le categorie si sistema e caricabili. E' possibile selezionare da questo elenco solo le categorie di famiglie caricabili andando a filtrare con un Codeblock.

Procedura per ottenere il nome delle famiglie

Per quanto a prima vista possa sembrare un'operazione semplice, ricavare una stringa con il nome di una famiglia (in questo caso, un elenco di famiglie), non è affatto scontato con i nodi presenti di default su Dynamo.
Per fare ciò, è necessario utilizzare 4 **nodi disposti in successione**

FamilyType.Name = Fornisce il nome del tipo delle famiglie sotto forma di string.

FamilyType.ByName = Ricava l'oggetto "tipo" di famiglia sulla base della stringa. In pratica, fa corrispondere ai nomi trovati con il nodo precedente), i veri e propri oggetti tipo.

FamilyType.Family = Indica le famiglie nelle quali sono allocati i tipi trovati con il nodo precedente

Family.Name = Ricava la stringa indicante il nome della famiglia trovata con il nodo precedente.

Ci si potrebbe chiedere se non sia possibile, dato un elenco di istanze, individuare direttamente il nome sotto forma di stringa delle famiglie con il nodo Family.Name.
In realtà, questo approccio non è possibile in quanto tale nodo fornisce, se utilizzato in questo modo, semplicemente il nome dei tipi, esattamente come FamilyType.Name.
Risulta a questo punto banale ricavare i nomi dei tipi

Test con RegEx

Una volta ottenuta una lista di stringhe, occorre testarle attraverso un nodo chiamato RegEx.IsMatch.

Il nodo RegEx.IsMatch necessita come input una stringa (o una lista di stringhe, come il caso in questione) e l'output di un nodo, sempre del pacchetto aggiuntivo RegEx, chiamato RegEx.ByPattern, a sua volta dipendente da una stringa contenente la RegEx utilizzata.
L'output del nodo RegEx.IsMatch è una serie binaria di true e false a seconda del fatto che la compilazione sia stata "giudicata" esatta o inesatta.
Questa lista di true e false viene utilizzata per creare una maschera di selezione attraverso il nodo Filter.ByBoolMask.
L'output di quest'ultimo sono proprio due elenchi, uno dei nomi di famiglia (sotto forma di stringa) che hanno passato il test della RegEx e l'altro dei nomi compilati in modo sbagliato.
Questa procedura viene applicata sia per i nomi delle famiglie (caricabili) che per i nomi dei tipi.

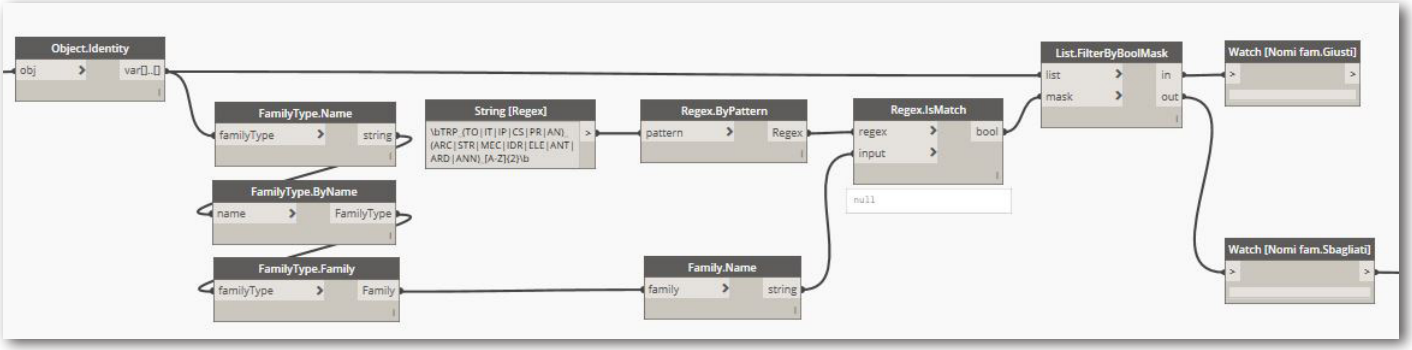


Immagine 62: Blocchi per il controllo dei nomi famiglia via RegEx.

Output: isolatore errori

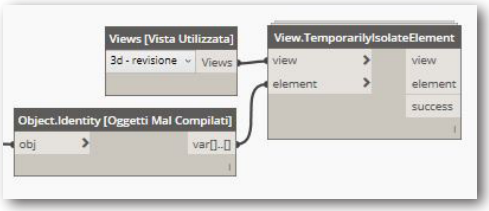


Immagine 63: Blocchi per isolare temporaneamente

Filtrati gli elementi compilati non correttamente, è **possibile** selezionarli e dire a Dynamo di **isolarli temporaneamente** in una determinata vista tramite il nodo View.TemporarilyIsolateElement.
Nonostante questo **approccio** sia il più **diretto** in assoluto per capire dove si trovano gli errori, esso è **tuttavia** il più **sconsigliabile**.
Nascondere un numero di istanze così elevato* non è un'operazione banale in termini computazionali; il rischio non è molto quello di aspettare troppo tempo, quanto piuttosto di bloccare Revit, cosa che può causare perdita di lavoro non salvato.
Questo tipo di **output** inoltre **va perduto tutte le volte che viene chiuso e riaperto Revit.**

* Tale numero è quello di tutte le istanze ben compilate che, verosimilmente, saranno la maggioranza, soprattutto nelle ultime fasi del lavoro, fasi in cui si suppone che uno script come questo sia più utile

Output: compilatore di parametro

E' possibile definire un parametro di tipo chiamato `Stato Revisione Nomenclatura` che riporta se la famiglia ha un nome che risponde alle regole imposte oppure meno.
Lo **svantaggio** principale di questo approccio è la necessità di dover **creare un nuovo parametro da zero** utilizzato solo per questo scopo.
Il **vantaggio** del parametro compilato è la possibilità di gestire gli errori tramite abachi e di creare filtri che consentono di **filtrare errori in una vista** (ottenendo così un risultato molto simile all'isolatore) o in un abaco per avere un abaco degli errori per categoria.

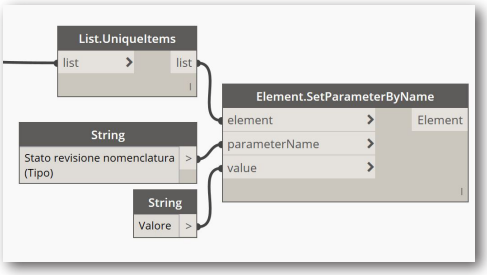


Immagine 64: Blocchi impostare il valore di un parametro.

Output: contatore di errori

Lo script fa un **conteggio degli errori nei nomi delle famiglie e di tipo**. Si hanno così due dati: quello delle istanze mal nominate e quello dei tipi mal nominati.
Questo tipo di output fornisce un indicatore dello stato di salute del modello, è **molto immediato ma non aiuta alla correzione** in quanto gli errori non sono evidenziati ma appunto solo contati.

In realtà gli errori nel nome del tipo e nel nome di famiglie vengono contati separatamente per poi venire sommati alla fine.
Un tipo di famiglia con nome errato e nome famiglia errato verrà contato due volte.

Output: elenco errori

Il programma raccoglie tutte le famiglie mal nominate e ne ricava il nome della categoria, della famiglia e di tipo. Viene successivamente effettuato uno "sfoltimento", in modo che vi sia una sola terna di voce per tipo errato (si ricorda che correggere il nome di un'istanza, automaticamente corregge quello di tutte le istanze dello stesso tipo). Questo output può tornare utile nel caso in cui si volessero correggere gli errori direttamente dal browser di sistema (forse il modo più rapido).

Scrittura del file Excel

Gli output trovati, vengono riportati su un file Excel attraverso il nodo Excel.WriteToFile. **Tale nodo necessita come input una stringa contenente il filepath del file Excel** che si vuole compilare. In caso esista già un file in quella posizione con quel nome, Dynamo sovrascriverà il file.

In caso contrario, ne verrà creato un altro.
Segue la richiesta del nome del foglio del documento che si vuole modificare e in quali coordinate sul foglio si vuole scrivere.
In ultimo, occorre inserire i dati da riportare.
Per quanto riguarda il file path, si nota di come sia lo stesso per tutti i nodi Excel.WriteToFile: è possibile definire tutte le volte la stessa stringa di input. Se però si vuole rendere più veloce l'operazione di cambio del path, magari per cambiare la posizione dove si vuole salvare la tabella dei risultati, è opportuno che tutti i nodi Excel.WriteToFile prendano tutti l'input del path dallo stesso nodo String, a meno che non si voglia cambiare lo stesso campo un numero n di volte, rischiando di andare incontro ad errori.
Sebbene questa operazione sia davvero molto banale, essa comporta necessariamente la creazione di numerosi "fili"tra nodi che renderebbero poco leggibile la struttura dello script.
Per ovviare a questo problema, si è **definita una variabile stringa globale**; si è cioè creato una sorta di parametro, con un nome e vi si è "inserita" dentro una variabile non numerica, ma di caratteri ASCII. Il valore della variabile è poi richiamabile dal programma in vari punti andando a specificare il nome della variabile.
Per ottenere questo risultato, si è ricorso a nodi Codeblock attraverso i quali definire delle semplici funzioni.
In programmazione, con il termine "funzione", si intende una serie di comandi definita dall'utente e richiamabile dallo stesso più volte.
La sintassi del Codeblock per definire una funzione prevede la scrittura di `def` seguito dal nome che si vuole dare alla funzione; quest'ultimo è terminato da due parentesi tonde. Segue, all'interno di parentesi graffe, il contenuto della funzione. In particolare, `return` indica l'output della funzione. Deve essere seguito dal simbolo dell'uguale, seguito a sua volta dalle istruzioni volute. In questo caso, la funzione voluta ha il semplice compito di restituire una stringa, la quale viene indicata tra le virgolette.

Si mostra di seguito un esempio di risultato riportato tramite tabella Excel.

Per ottenere lo stesso risultato senza ricorrere a grammatiche di programmazione particolari, si è ricorso a due blocchi del pacchetto Prorubim DS Common Kit chiamati Get.Var e Set.Var, i quali promettono di fare proprio questo.
Il problema di questi nodi è il fatto che vengono riconfigurati ad ogni avvio di Dynamo in modo da renderli, sostanzialmente, inutilizzabili.

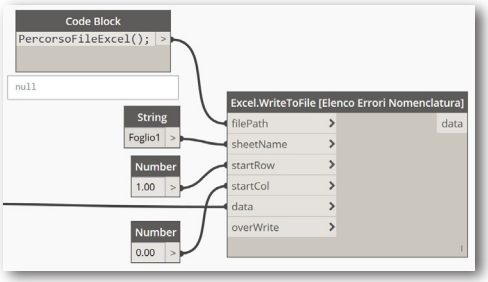


Immagine 65: Blocchi per scrivere su un file Excel.

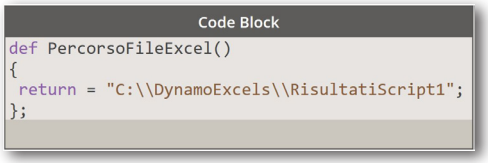


Immagine 66: Codeblock per la definizione della cartella dive scrivere il file Excel.

PS
In modo non dissimile da quanto visto per le RegEx, è necessario utilizzare il carattere backlash (\) per far considerare al programma il carattere che segue non come un metacarattere. Per richiamare la funzione è sufficiente aprire un Codeblock, riscrivervi dentro il nome della funzione seguito dalle due parentesi (terminando la stringa con un punto e virgola).
Si mostra di seguito un esempio di risultato riportato tramite tabella Excel.

	A	B	C	D	E	F	G
1	Categoria	Famiglia	Tipo		Numero di istanze totali	5326	
2	Pavimenti	Pavimento	Pc_S08.1r		Numero di istanze ben compilate	5242	
3	Muri	Muro di base	Generico - 20 cm		Numero di istanze mal compilate	84	
4	Muri	Muro di base	Generico - 45		Numero di tipi totale	5326	
5	Muri	Muro di base	Isolante solai coorte		Numero di tipi ben compilati	5318	
6	Muri	Muro di base	Controsoffitto verticale		Numeor di tipi mal compilato	8	
7	Muri	Facciata continua	Generico				
8	Muri	Muro di base	Vuoto				
9	Muri	Muro di base	Vuoto (1cm) (host porta)				
10							

Immagine 67: Output di Dynamo su Excel, grezzo.

Le tabelle Excel riportate sono molto spartane: si tratta infatti del file appena compilato da Dynamo, senza alcun intervento grafico se non il cambiamento della larghezza delle celle. E' possibile preparare a monte, un file Excel formattato con determinati stili di casella e grafici pre-impostati in modo che Dynamo possa andare ad inserirvi i dati.

Categoria	Famiglia	Tipo
Pavimenti	Pavimento	Pc_S08.1r
Muri	Muro di base	Generico - 20 cm
Muri	Muro di base	Generico - 45
Muri	Muro di base	Isolante solai coorte
Muri	Muro di base	Controsoffitto verticale
Muri	Facciata continua	Generico
Muri	Muro di base	Vuoto
Muri	Muro di base	Vuoto (1cm) (host porta)

Numero di istanze totali	**5326**
Numero di istanze ben compilate	**5242**
Numero di istanze mal compilate	**84**
Numero di tipi totale	**5326**
Numero di tipi ben compilati	**5318**
Numeor di tipi mal compilato	**8**

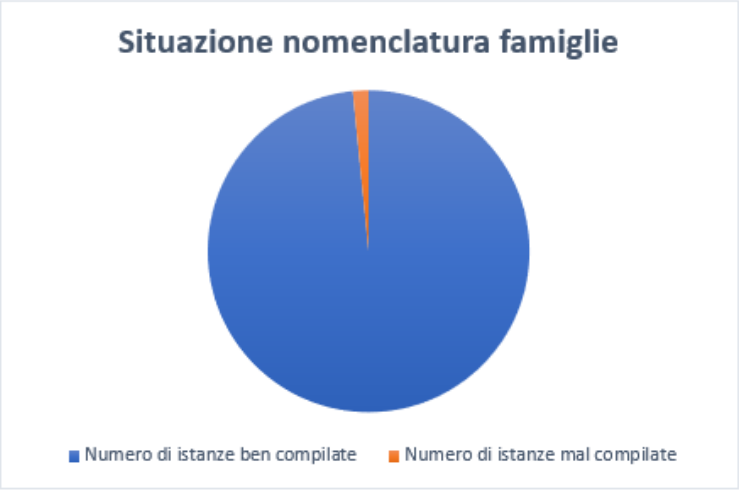


Immagine 68: Output di Dynamo sistemato su Excel

Controllo Parametri di Modelli Federati

Funzione

Il seguente script si propone di **testare la percentuale di completamento per quanto riguarda la compilazione dei parametri condivisi** fornendo degli indicatori di salute non soltanto di un modello singolo quanto piuttosto **di un modello federato**. La situazione che si immagina è quella di un BIM Coordinator che abbia la necessità di controllare in pochi minuti la correttezza del lavoro svolto da un team di progettisti senza dover aprire ogni singolo file e senza individuare la posizione degli errori.

Modello di Coordinamento

Per modello di coordinamento si intende un modello Revit sostanzialmente vuoto se non per la **presenza di tutti gli altri modelli linkati ad esso**. Un modello di coordinamento è dunque un insieme di tutti i modelli federati.
I modelli che ne fanno parte non sono modificabili, né a livello di geometria, né a livello di compilazione di parametri. E' inoltre sconsigliabile andare ad inserire nuove famiglie nel modello di coordinamento.
Dal modello di coordinamento è possibile visualizzare, oltre alle proprietà geometriche nei modelli linkati, in una vista grafica (come piante o viste 3d) anche tutti i dati compilati tramite abachi generali.
Ai fini del lavoro presentato in questa tesi, è stato creato un modello di coordinamento che comprendesse soltanto i tre modelli degli interrati della Torre, modellati dal sottoscritto, ovvero l'Architettonico, lo Strutturale e l'Antincendio.
Non sono stati compresi altri modelli per non appesantire troppo il lavoro. Va però detto che uno script di questo tipo servirebbe proprio in un caso in cui si abbiano tutti e quanti i modelli, proprio per fornire una fotografia dello stato di salute del modello nel suo insieme. Di fatto, allargare il campo di operatività dello script è una cosa piuttosto semplice da fare, in caso servisse, sarà sufficiente copiare dei nodi in blocco e modificare l'input.

Struttura dello script

Il processo di creazione di un elenco di tutte le famiglie è esattamente lo stesso visto nello script precedente.
L'elenco di tutte le famiglie passa poi attraverso una batteria di gruppi di nodi. Ogni gruppo si occupa di testare un determinato parametro fornendo al contempo parte del risultato complessivo.

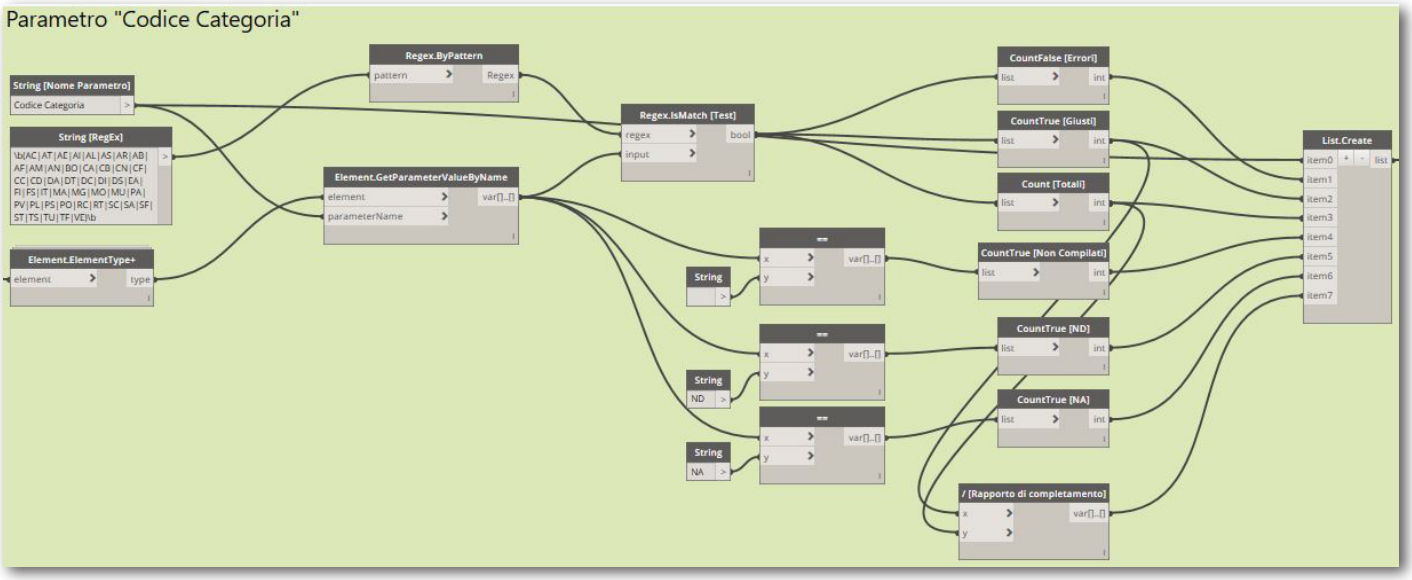


Immagine 69: Batteria di nodi per monitorare la compilazione del parametro `Codice Categoria`.

Sopra è stato riportato lo schema del gruppo del parametro `Codice Categoria`.
In alto a sinistra vengono definito il nome del parametro (in questo caso, appunto `Codice Categoria` e la Regular Expression che testerà le famiglie, in modo simile a quanto visto nello script precedente.
Dal nodo **Regex.IsMatch** vengono contati i `false`, ovvero gli errori di compilazione, i `true`, ovvero le istanze ben compilate e il numero di voci totali.
Altri tre risultati, questa volta ricavati tramite semplici operatori logici (non RegEx), contano il numero di istanze non ancora compilate, compilate con ND o con NA.
Viene infine ricavato un **rapporto di completamento** andando a dividere il numero di istanze ben compilate per il numero di istanze totali.
Tutti questi output vengono organizzati in liste tramite un nodo **List.Create**.
Il gruppo di nodi che si è presentato è valido solo per il parametro `Codice Categoria` ma è comunque molto

simile a quelli riferiti ad alti parametri (al cambiare, al limite è un blocco **Elements.Type** davanti ai parametri di tipo). Tutte le liste ottenute dai vari gruppi di parametri confluiscono a loro volta in una seconda lista formando così una matrice a 2 dimensioni che verrà stampata tramite il nodo **Excel.WriteToFile** già visto nello script precedente.

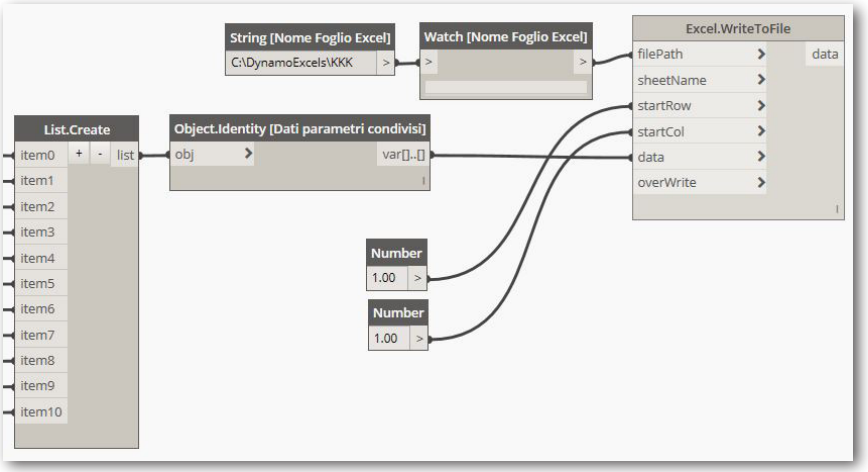


Immagine 70: Nodi per stampare l'output su Excel.

Identificatore errori nei Parametri Condivisi

Funzione

Lo script visto in precedenza conta il numero di errori ma ha la limitazione di non indicarli, non fornisce cioè un output che consenta di individuarli all'interno del modello per correggerli. Si pensi al caso in cui, dopo aver eseguito il primo script, risulti esserci un errore: il preposto al controllo del modello dovrebbe comunque cercare in tutto il modello quell'unica istanza scorretta, andando a perdere talvolta anche molto tempo.

Struttura

Il nuovo script comincia come tutti gli altri, andando a fare un **elenco di categorie** e ricavando da esso tutte le istanze. Queste istanze poi vengono passate ad un **controllo RegEx su ogni parametro**, controllo che fornisce semplicemente un output true o false (una mancata compilazione ricade direttamente nel caso false).
La lista true false cosi ottenuta viene poi trasformata.
Ogni voce `false`, corrispondente ad un parametro mal compilato, viene **sostituita con il nome del parametro** controllato stesso, mentre in caso di buona compilazione, il false viene sostituito da un semplice "()".
Questo processo viene ripetuto per n parametri ottenendo cosi n liste di Nomi parametro e "().
Le **n liste** vengono infine **combinate** in modo da ottenerne una sola che verrà poi scritta in un parametro Revit precedentemente creato appositamente.

Vantaggi

Il grande vantaggio di questo script è il **condensare in un unico parametro un report facilmente leggibile** della situazione della compilazione di tutti i parametri condivisi, di tutte le istanze del modello. Quell'unico parametro consente poi di creare **filtri vista** direttamente su Revit ed abachi che estrapolano esattamente quello che serve.

Si pensi ad esempio alla situazione in cui si voglia evidenziare in un abaco tutti gli estintori che presentano l'`Identificativo` e il `Codice MasterFormat` mal compilati: basterà creare un abaco degli estintori e filtrare tutti gli elementi che "contengono" nel parametro creato, le parole "Identificativo" e "Codice MasterFormat".

Svantaggi

Il grande limite di questo script è l'**enorme carico computazionale** che richiede ai computer. Si è testato tale script su un PC portatile montante un processore Intel i7 di ottava generazione con il risultato di bloccare Revit.
Si è poi testato lo stesso script su un computer fisso di maggiori prestazioni ed ha comunque impiegato circa 5 minuti per essere completato.
Non si riescono a capire i motivi di una tale pesantezza, probabilmente si tratta dei vari nodi list.combine.
Per mitigare tale problema, sono stati esclusi da questa compilazione i parametri "`Progetto`" ed "`Edificio`" in quanto di facile controllo ma pure con questo stratagemma, lo script si è rilevato inutilizzabile se non per una categoria per volta.

Identificatore errori nei Parametri Condivisi 2

Funzione

Questo nuovo script nasce come correzione dello script numero 3, (vedi "Identificatore Errori Parametri Condivisi" a pagina 79) quello che contava il numero di parametri mal compilati ma,al posto di contare gli errori di compilazione per ogni parametro condiviso, esso **sostituisce l'errore con la parola "ERRORE".** In questo modo, ogni voce da correggere è facilmente individuabile da abaco non solo tramite filtri (cosa che permetteva anche lo script precedente), ma anche con un semplice colpo d'occhio.

Vantaggi

Non dovendo ricorrere a nodi list.combine, questo **script** è **molto più veloce** del precedente e permette dunque di analizzare migliaia di istanze del modello in pochi secondi. **Non è poi necessario definire un nuovo parametro nel progetto**.
Una sola voce per tutti gli errori permette di **creare facilmente filtri** per isolare nella vista le istanze mal nominate e, soprattutto per **colorare** (ad esempio, di rosso), le **celle di un abaco tramite la formattazione condizionale** dell'abaco. Proprio quest'ultima opzione è in assoluto il modo migliore per evidenziare un errore di compilazione.

Svantaggi

Rispetto allo script precedente, che si occupava di individuare errori e fornire un semplice report, il nuovo script sovrascrive il valore individuato come errore; **il pericolo è quello di cancellare delle informazioni, magari concettualmente esatte ma errate nella forma**, anche solo per un piccolo segno di punteggiatura. Ne deriva la **necessità di prestare particolare attenzione alla definizione delle RegEx** prima di eseguire lo script.

Struttura

Lo script è praticamente identico al precedente se non per il fatto di compilare parametri al posto di creare una matrice da stampare su Excel come output. Nodi **Count** forniscono il numero di errori trovati.

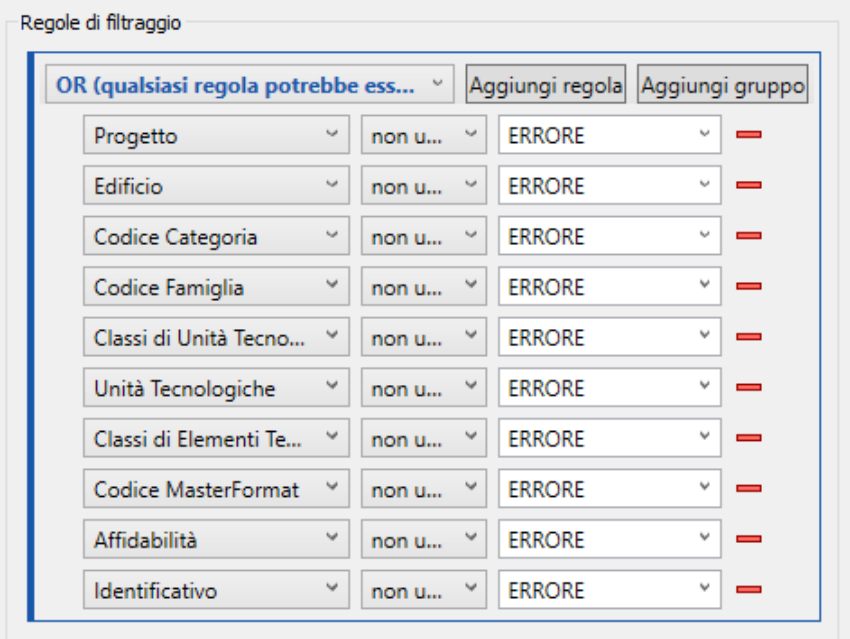


Immagine 71: Regole di filtraggio errori per ottenere una vista che visualizza solo e soltanto oggetti contenenti almeno un errore di compilazione

Interruttore compilazione

Come anticipato, lo script ha il problema di sovrascrivere dati presenti con il rischio di eliminare informazioni magari giuste ma non capaci di passare attraverso un RegEx test, anche solo per un piccolo errore nella compilazione dei codici. Per questo motivo è stato creato una sorta di **interruttore**, una funzione costruita tramite **Codeblock** che permette a monte, di "spegnere" tutti i blocchi **Element.SetParameterByName**, prima che essi possano effettivamente scrivere. In questo modo, **è possibile far girare lo script e avere una previsione di quali sostituzioni Dynamo è in procinto di fare** e controllare se vi siano problemi in questo senso.

Alla prima stesura dei codici, questa funzione era ottenuta andando a congelare i blocchi di scrittura. Ciò è possibile

farlo cliccando con il tasto destro sul blocco e selezionando, appunto, "congela". Tale metodo è più diretto ma deve essere eseguito per ogni blocco manualmente, andando a perdere di vista l'obiettivo di uno script il quanto più possibile automatizzato ed universale.
Tramite i caratteri /* e */ , all'interno di un `Codeblock` è possibile definire una stringa note, in modo da creare un commento utile all'utilizzatore dello script.
Come è visibile dalle istruzioni direttamente all'interno del Codeblock, **per "accendere" l'interruttore, è sufficiente andare a modificare la scritta "null" all'interno del codice con la scritta "input".**

L'interruttore, una volta definito, è stato copiato e riutilizzato all'interno di altri script.

Code Block [Interruttore Blocchi di Sovrascrittura Parametro]

```
/*


ISTRRUZIONI
Per attivare la sovra-scrittura dei parametri, sostituire "null"
con la parola "input"(senza virgolette).
Viceversa se la si vuole disattivare


*/


def Interruttore(input)
{
 return = null;
};
```

Immagine 72: Codeblock per interrompere la compilazione dei parametri.

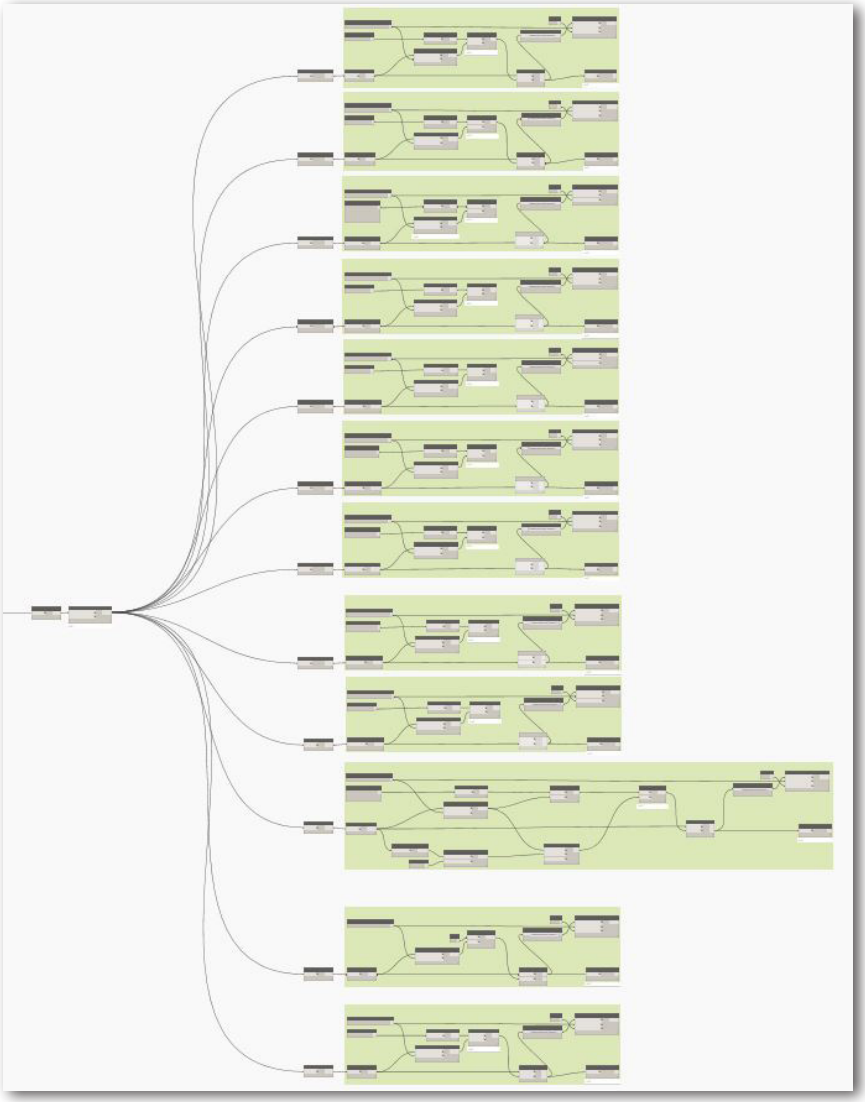


Immagine 73: Panoramica dello script.

Compilatore Identificativo

Tra tutti i parametri, l'identificativo è ovviamente il più complesso nella compilazione. Il seguente script si occupa di compilare il parametro automaticamente per tutte le categorie selezionate.

Struttura dello script

La struttura si basa , ancora, su una **batteria di gruppi ognuno dei quali funziona per una sola categoria**. Per estendere la compilazione a tutte le categorie è necessario copiare il gruppo e cambiare la categoria di input.
Si considera a questo punto nota la metodologia per compilare i primi 5 sottoparametri.

Per la struttura del parametro "Identificativo si rimanda il lettore a "Identificativo" a pagina 60.

Counter

L'ultima parte dell'identificativo, è un **numero progressivo di 5 cifre** che possiamo chiamare "**counter**". Caratteristica di questo counter è quella di **ricominciare il conteggio per ogni tipo di famiglia**; per questo motivo è stato necessario ricorrere al blocco GroupByFunction per organizzare l'ordine delle istanze.
Il blocco GroupByFunction permette di dividere una lista di istanze in più sotto-liste sulla base di una caratteristica, in questo caso il tipo.

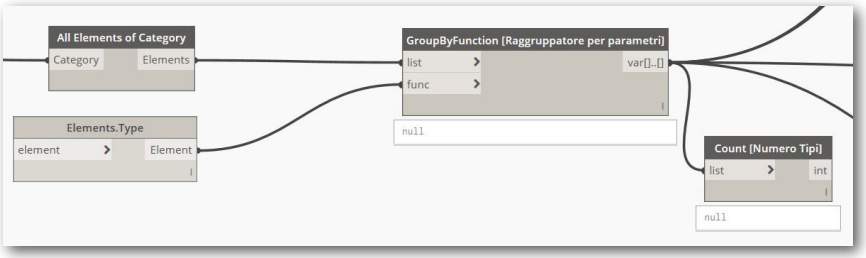


Immagine 74: Nodi per raggruppare istanze di famiglia in sotto gruppi basati sul tipo.

L'**output** di tale nodo sarà dunque una **lista a due livelli**; ogni sotto-lista (o lista di secondo livello) contiene tutte le istanze appartenenti ad un determinato tipo.
Un counter si crea sempre andando a definire all'interno di

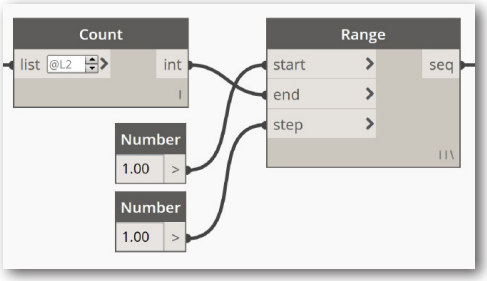


Immagine 75: Nodi per la creazione di un counter

un nodo Range, una sequenza di numeri avente un inizio, un valore di fine ed uno step, ovvero la differenza costante che intercorre tra un valore e l'altro. Prendere come input del valore di fine l'output di un nodo count delle istanze raggruppate, fornirebbe semplicemente il numero dei tipi di una determinata categoria. Per avere un counter che riparte per ogni tipo, è opportuno **riferire il nodo count al secondo livello**.

Il counter dell'identificativo deve però essere numero di 5 cifre, indipendentemente dal valore. Il nodo String.PadLeft permette di avere, in un elenco di stringhe, sempre lo stesso numero di caratteri (o cifre) andando a colmare con degli zero gli spazi rimanenti, a sinistra.

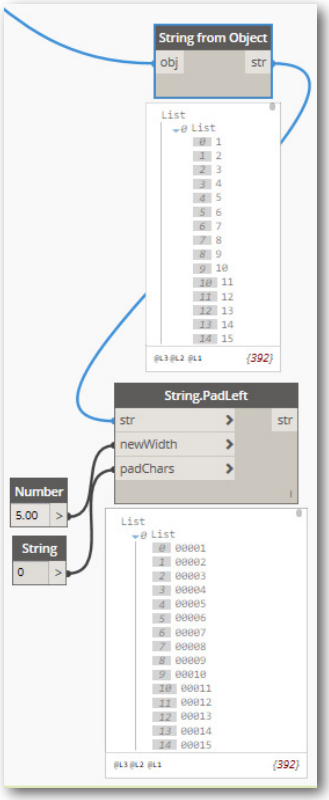


Immagine 76: Nodi per ottenere numeri da 5 cifre.

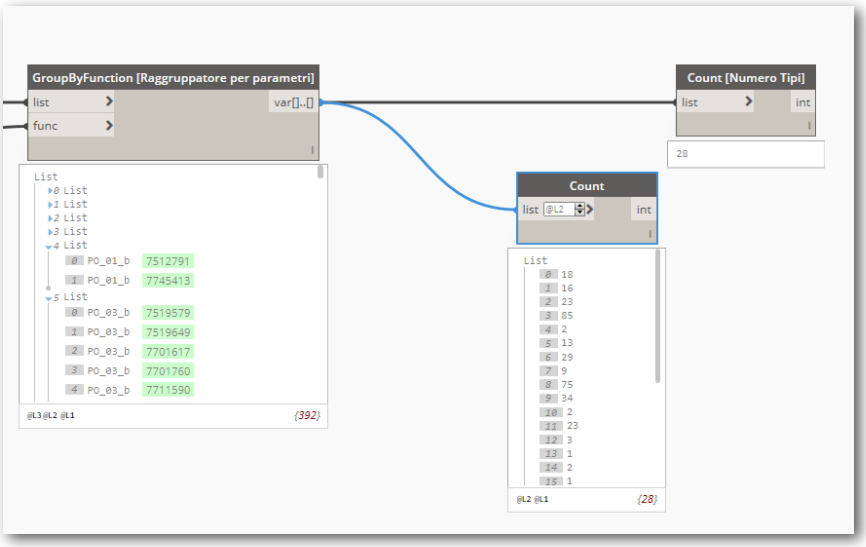


Immagine 77: Nodi per contare il numero di tipi ed il numero di istanze all'interno

Nome del livello

Il nodo Element.Level fornisce una **denominazione dei livelli molto più completa rispetto a quella che servirebbe per l'identificativo** (ovvero le 2 lettere e 2 cifre). Il codice cercato è tuttavia presente, sempre nella stessa posizione all'interno della macro stringa estratta dal nodo; è necessario dunque estrarre questa sotto stringa da una macro-stringa fornita dal nodo Element.Level.
Per farlo si è utilizzato di nuovo il comando String.PadRight per avere tutte stringe della medesima lunghezza fissata arbitrariamente a 50 caratteri). Da queste poi sono stati tagliati via prima tutti i caratteri successivi alla sotto stringa

ed in seguito, tutti i caratteri precedenti. Ne risulta dunque il codice cercato senza dover ricorrere a nodi string replace (i quali sarebbero stati adatti per gli Interrati Torre ma non per una torre di oltre 40 piani).

```
Level(Name=LI01_004.27, Elevation=-4,27)XXXXXXXXXX
```

Vi sono delle categorie, (in particolare, quella delle tubazioni), le cui famiglie, interrogate dal nodo **Element.Level** non forniscono il valore voluto. Ciò porta lo script a manipolare delle liste di elementi `null`, privi di significato, di fatto fornendo un risultato errato. Tale problema non è direttamente legato alla presenza, all'interno della famiglia, del parametro `Livello` in quanto la categoria muri, nonostante abbia un parametro chiamato `Vincolo di base` a farne le veci, non da errori.
Per ovviare a questo problema, si è inserito un controllo all'interno di ciascun blocco, controllo che appunto controlla l'esistenza del parametro `Livello di riferimento` (compilato di default nelle famiglie tubazione) e, nel caso affermativo, ricava da esso il nome del livello.
Per quanto riguarda la categoria montanti, essi non possiedono, tra i parametri, quello relativo al livello. Tale problema è stato aggirato andando a creare un parametro di progetto (non necessariamente condiviso) chiamato `Livello di riferimento` (come quello presente nelle tubazioni) ed andando a compilarlo manualmente.
Il programma infine unisce tramite un **Codeblock**, le varie informazioni (sotto forma di stringa) trovate, ovvero

- TRP_IT_ARC_ (definito all'inizio dello script)
- Il Codice Famiglia
- Il nome del tipo
- Il livello estratto
- Il counter

I campi sono suddivisi da underscore.

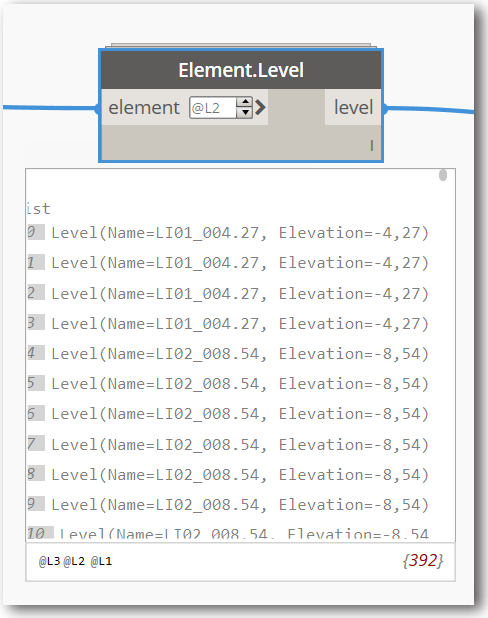


Immagine 78: Nomi dei livelli trovati con il nodo **Element.Level**.

NB

Affinché la sezione relativa al livello sia compilata adeguatamente, è opportuno che il **nome del livello** sia esattamente **lungo 11 caratteri**, meglio se formattati come visto nell'esempio e cioè 4 caratteri seguiti da underscore seguiti ancora da un numero di 5 cifre, di cui 2 decimali.

E' poi necessario che l'**unità di misura del modello** sia impostata su **metri [m]**.

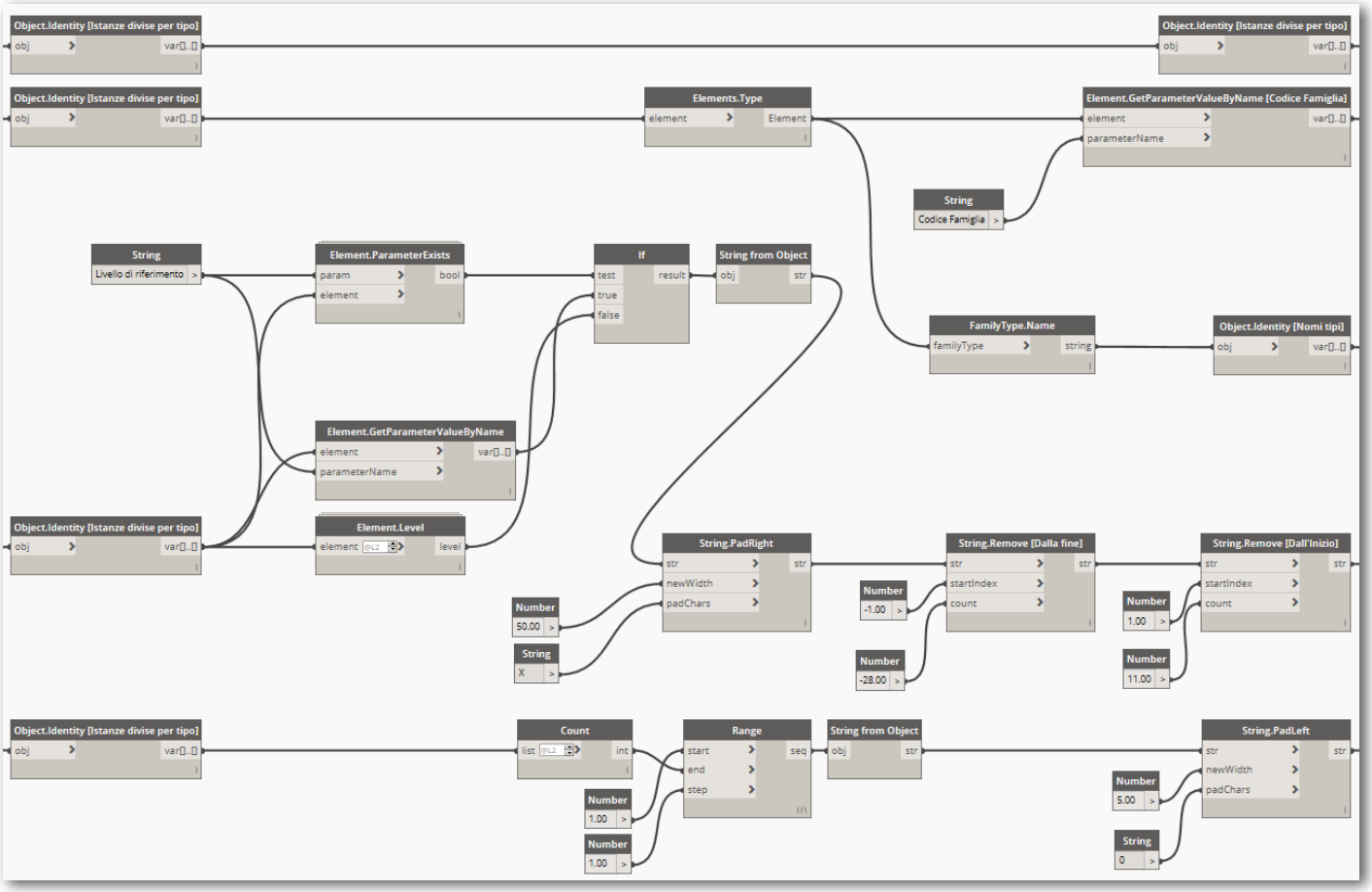


Immagine 79: Nodi per la compilazione dell'`identificativo`

La Pesatura degli Errori

Gli script di controllo visti fino ad ora forniscono il numero delle istanze ed eventualmente dei tipi che presentano un errore in un determinato parametro. Questa informazione, sebbene sia comunque un dato importante, tuttavia non è sufficiente per rappresentare quanto lavoro è necessario fare , all'atto pratico, per avere un modello compilato correttamente in ogni sua parte.

Si pensi ad esempio ad una situazione in cui 1000 istanze estintore presentano tutte e quante un errore nel parametro `Progetto` e solo 50 di esse, suddivise in 4 tipi, presentano un errore nel Codice MasterFormat: è chiaro che correggere questo secondo parametro porterà via più tempo rispetto all'assegnare il giusto valore di `Progetto` a tutte le istanze in cui serve, nonostante il numero di queste ultime sia enormemente maggiore.

Si rende dunque utile un'interpretazione critica dei risultati ottenuti che sappia fornire un significato pratico alle informazioni ricavate.

Per fare ciò, si è seguito un approccio ereditato dalle analisi multi criteri e basato sulla **pesatura degli errori** .

La logica che sta dietro allo script che si andrà a presentare successivamente, si basa sul **concetto di creare diverse schede**, ovvero delle **check-list, una per ogni parametro e di dimensioni diverse**. Ognuna di queste schede verrà compilata sulla base di quanti errori sono stati trovati nel parametro corrispondente. Unire le schede consentirà di avere un rapporto generale tra la situazione reale e l'obiettivo da raggiungere, anche esprimibile tramite una percentuale di completamento.

Le schede riferite ad ogni parametro hanno **diversa lunghezza** e ciò rappresenta il **diverso peso** che hanno errori in diversi parametri.

A seguito verrà utilizzato un esempio per spiegare in modo più comprensibile, il processo sviluppato e il modo in cui sono stati considerati tutti i diversi parametri.

Situazione esempio

Si immagini di voler indagare lo stato di avanzamento della compilazione degli errori di una serie di Y istanze (ad esempio 15 Porte) raggruppate in X tipi (esempio 5), a loro volta raggruppate in Z famiglie (ad esempio 2).

Z	X	Y	Progetto	Edificio	Codice Categoria	Codice Famiglia	Classi Unità Tecnologiche	Unità Tecnologiche	Classi di Elementi Tecnici	Codice MasterFormat	Titolo Masterformat	Identificativo	Affidabilità	Nome Famiglia	Nome Tipo	Codice Esistente
Famiglia 1	Tipo 1	Istanza 1														
		Istanza 2														
		Istanza 3														
	Tipo 2	Istanza 4														
		Istanza 5														
		Istanza 6														
	Tipo 3	Istanza 7														
		Istanza 8														
		Istanza 9														
Famiglia 2	Tipo 4	Istanza 10														
		Istanza 11														
		Istanza 12														
	Tipo 5	Istanza 13														
		Istanza 14														

Immagine 80: Tabella rappresentante la correttezza nella compilazione di un caso di esempio ipotetico. Caselle rosse corrispondono ad errori nel parametro corrispondente

La tabella presentata la situazione descritta e viene anche indicato, a titolo di esempio, il risultato di una eventuale verifica di correttezza dei parametri: **in verde chiaro sono indicati parametri ben compilati ed in verde scuro gli errori**.

Progetto

Come già anticipato, correggere il parametro `Progetto` è un'operazione semplice e relativamente veloce. E' inoltre trascurabile il numero di istanze da correggere in quanto ricorrendo ad Export/Import Excel, a correggere una voce o un migliaio di voci ci si mette letteralmente lo stesso tempo. Ne consegue che la **scheda `Progetto` conterrà una sola casella**. Questa casella viene riempita solamente quando tutte le istanze di una relativa categoria sono compilate correttamente. Un solo errore porterà la casella ad essere verde scuro.

Progetto

Edificio

Valgono le stesse considerazioni fatte per il parametro precedente.

Edificio

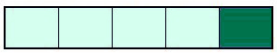
Codice Categoria

Codice Categoria

Il `Codice Categoria` ho lo stesso peso dei primi due in quanto, all'interno di una categoria, tutte le istanze (o meglio, tutti i tipi) hanno il medesimo valore facilmente compilabile.

Codice Famiglia

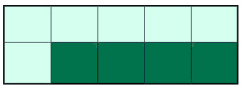
Codice Famiglia



Tale parametro presenta una lista di **X caselle**, **ognuna corrispondente ad un tipo di famiglia**.
Ogni tipo avente il Codice Famiglia compilato correttamente andrà a riempire una casella

Classi di Unità Tecnologiche, Unità Tecnologiche, Classi di Elementi Tecnici

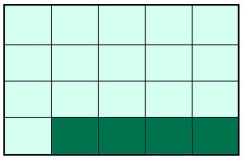
Classi di Unità Tecnologiche
Unità Tecnologiche
Classi di Elementi Tecnici



Questi tre parametri sono considerati in un **unico blocco** in quanto strettamente legati. Si parte considerando una sezione di X caselle, quanti sono i tipi: così facendo è come se si stesse considerando i tre parametri come uno unico aggregato e in effetti, è impensabile fornire un valore a `Classi di Elementi Tecnici` senza aver compilato anche `Classi di Unità Tecnologiche` e `Unità Tecnologiche`. E' altresì vero che riempire una terna di questo gruppo richiede più tempo di, ad esempio, dare un `Codice Categoria`: si è optato dunque per una via di mezzo, andando considerare i 2/3 di X (tale numero sarà sempre intero).

Codice e Titolo MasterFormat

Codice MasterFormat
Titolo MasterFormat



Nuovamente è necessario trattare i due parametri in coppia in quanto è difficile pensare che la compilazione di uno non vada pari passo con la compilazione dell'altro. Trattandosi di un'operazione relativamente complessa si sono considerati i parametri come separati ed anzi, sono state raddoppiate le celle della lista in modo da raddoppiare il peso di queste correzioni.
Tale scheda presenta dunque **[(X*2)*2] = 4X caselle**.

Identificativo

Identificativo

Immaginando di non poter utilizzare lo script elaborato in questa tesi, compilare l'`Identificativo` comporta l'esportare dei dati su Excel e riempire i campi trascinando una regola "grammaticale" lungo l'elenco di istanze, fermandosi ad ogni tipo per azzerare il contatore progressivo.

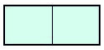
Il numero di pause è dunque identico al numero di tipi e dunque, il **numero di caselle della scheda è X, ovvero uguale al numero di tipi**.

Affidabilità

Affidabilità

A parte casi particolari, tale scheda è sempre uguale a **3 caselle**. Per i casi in cui ciò non sia vero, ci si aspetta che il modellatore abbia indicato subito dopo la posa dell'istanza, il corretto valore di affidabilità (si tratta infatti di un'informazione che se non prontamente inserita, potrebbe venir dimenticata facilmente). Per queste ragioni, si è considerato fisso a 3 il numero di caselle riferite al parametro `Affidabilità`. Tali caselle, come nel caso dei primi parametri, risultano vuote alla presenza anche solo di un singolo errore.
Il numero 3 è stato scelto arbitrariamente ma si ritiene che ben rappresenti l'impegno per la sua compilazione, soprattutto con riferimento, i parametri `Progetto` ed `Edificio`.

Nome Famiglia



Nome Famiglia

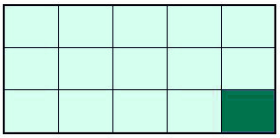
Non si tratta di un parametro quanto piuttosto della nomenclatura della famiglia. La scheda dedicata presenta **Z caselle**, ovvero il numero di famiglie considerato.

Nome Tipo

Nome Tipo

Caso analogo al precedente ma con **X caselle**.

Codice Esistente



Codice Esistente

I Codice Esistente non viene compilato per tutte le categorie. Se Porte e Locali, come si è visto nel capitolo dedicato, hanno un codice che le identifica univocamente anche sul disegno .dwg, per altre famiglie, come ad esempio i muri, tale parametro è compilato semplicemente con "NA". Si tratta di **due situazioni** completamente **differenti**. Per quest'ultimo caso, in particolare, si possono fare le stesse considerazioni viste per il parametro `Progetto`. Per quanto riguarda il caso di categorie che richiedono tale compilazione, si è scelto di creare una scheda di **Y caselle, ovvero una per ogni istanza**.

Si potrebbe pensare che tale scheda possa diventare troppo grande (e dunque pesante) all'interno della scheda aggregata; inserire questi parametri è però un lavoro

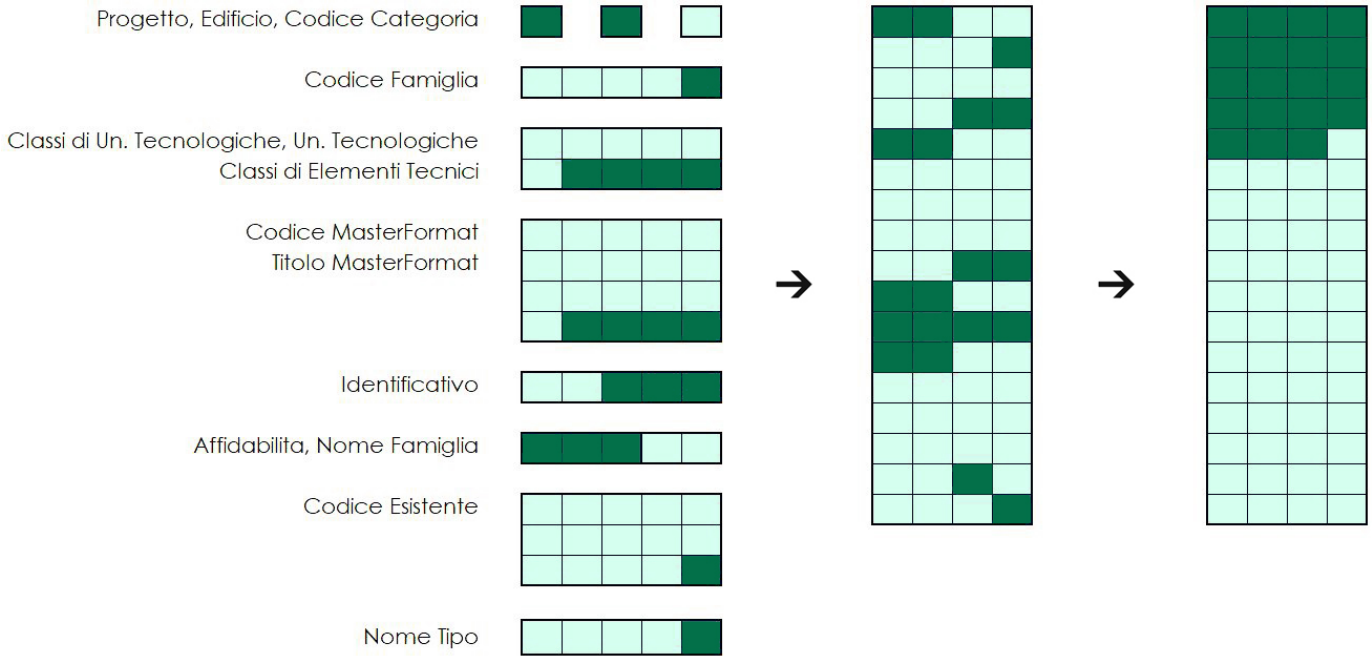


Immagine 81: Schema rappresentante il trattamento dei dati ricavati dalla pesatura, dalla raccolta all'estrapolazione di una percentuale di completamento.

abbastanza lungo. Va poi considerato il fatto che si tratta forse del parametro più importante tra tutti, indispensabile per riportare informazioni da abachi .pdf.

Stando a quanto visto fino ad ora, si riporta un elenco riassuntivo di tutte le schede e degli errori all'interno di essi, visti come caselle verde scuro. Tutte le schede vengono raggruppate in un'unica Macro-Scheda ed infine, vengono contate le celle errore, le celle verde chiaro e le celle totali. Da questi tre dati si ricava facilmente una **percentuale di completamento** basata sul tempo necessario alla correzione.

Nel caso esempio, sono stati individuati 19 errori su 68, ovvero il 28%.

Ci si potrebbe chiedere ora a **cosa corrisponda, realmente, una casella scura**, in termini di tempo. Rispondere a questa domanda è molto difficile in quanto **basata su aspetti**

davvero troppo soggettivi per essere significativi. La velocità di lavoro cambia molto individualmente e può dipendere molto da fattori esterni. Non esistono inoltre tempari dedicati, come quelli utilizzati per la stesura di cronoprogrammi di cantiere.
Volendo tuttavia dare un'identità ad un risultato che altrimenti sarebbe solo numerico, si potrebbe considerare **una casella scura** corrispondente a **30 secondi di lavoro di correzione necessari.**
19 errori corrisponderebbero dunque a circa a 10 minuti, un risultato che, esperienza alla mano, può considerarsi verosimile, a meno di situazioni particolari (comunque abbastanza frequenti).

Finder e Pesatore Errori

Il seguente script sebbene erediti molti concetti di quelli precedenti, non raccoglie automaticamente tutte le categorie all'inizio del programma.
Viene invece presa **una sola categoria alla volta**; i parametri di questa categoria vengono fatti passare attraverso una batteria di nodi custom oltre alla quale si ha come informazione, il numero di celle della scheda corrispondente a ciascun parametro e il numero di quelle bene e mal compilate.

Il risultato finale viene poi stampato su un foglio Excel.

Di seguito viene presentato il caso della categoria Arredi Fissi, ma il gruppo di nodi risulta essere sempre lo stesso, al netto di alcune differenze che andranno evidenziate.

Per capire cosa si intende per "celle", si rimanda al capitolo precedente.

Custom Nodes

I nodi Custom sono dei gruppi di nodi che possono essere appunto raggruppati all'interno di un solo nodo definito dall'utente, il quale potrà poi utilizzarli all'interno dei propri script senza dover ridefinire tutta la costruzione dall'inizio. Essi sono utili laddove la stessa costruzione debba essere usata ripetutamente all'interno dello script e/o per alleggerire, almeno graficamente, la struttura dello script; in tal modo si ottiene anche un alleggerimento del carico sulla Scheda Video.

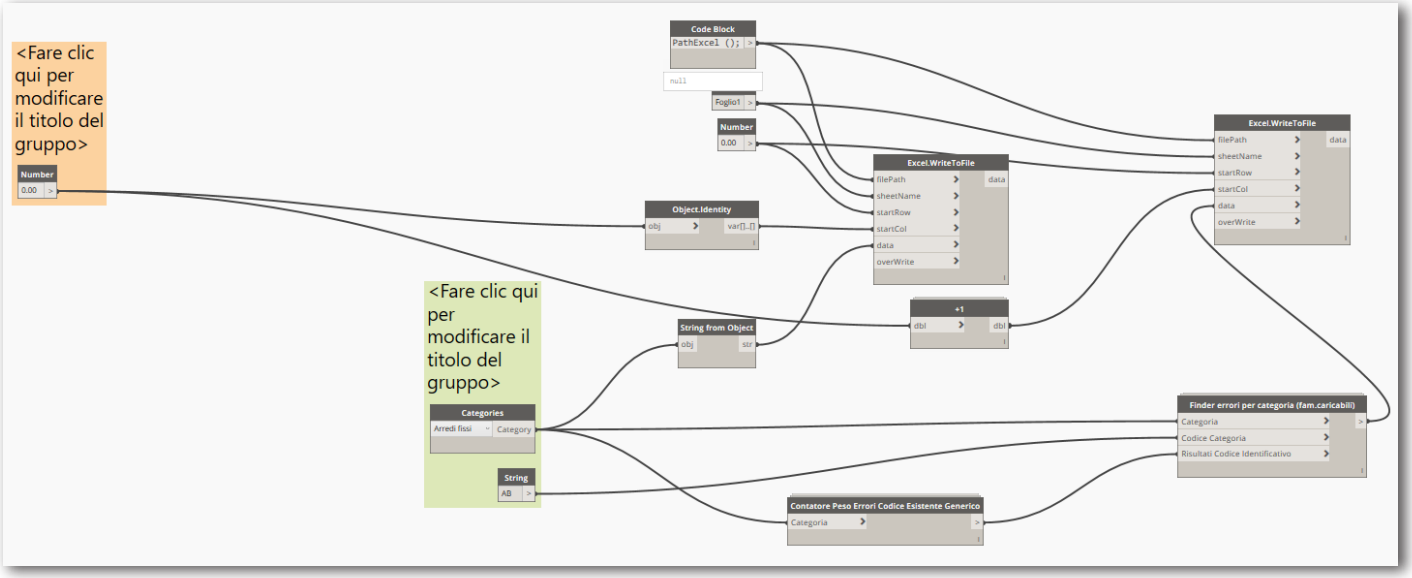


Immagine 82: Esempio di nodi per il controllo di una categoria

Alcuni gruppi (anche formati da un solo nodo) sono stati evidenziati in modo da richiamare l'attenzione dell'utente in quanto necessitano di essere impostati in base alle circostanze.
Come si nota, è necessario specificare la famiglia della quale si vuole effettuare il controllo seguito dal relativo `Codice Categoria.` E' poi necessario andare a

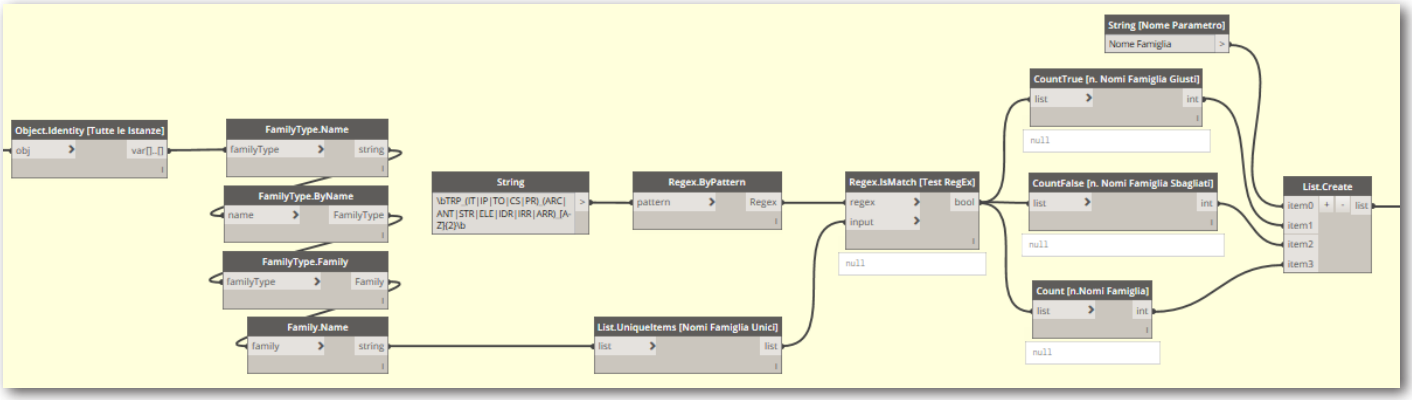


Immagine 83: Gruppo di nodi che controllano la correttezza dei nomi famiglia.

Lo sfondo degli script compare di colore giallino in quanto si tratta dell'interno di un Custom Node.

specificare in quale colonna del file Excel si vuole scrivere la tabella relativa ai risultati di quella Categoria. **Le tabelle devono distare l'una dall'altra almeno 5 colonne**.
Al nodo Categoria sono **collegati 2 coustom node**, uno contenente il controllo sul parametro `Codice Esistente` e l'altro, su tutti gli altri. Tale divisione è stata necessaria in quanto **non è possibile definire una RegEx unica per tutte le categorie** (in realtà, le porte sono state l'unico caso in cui è servito questo controllo).
E' stato detto che uno dei nodi controlla tutti i parametri a parte uno; esistono in realtà 2 nodi di quest'ultimo tipo: uno è utilizzato per le famiglie di sistema e l'altro per quelle caricabili.
I 2 nodi sono praticamente identici se non in alcuni punti dove è necessario ricavare il nome della famiglia.
All'interno di questi nodi sono presenti a loro volta dei gruppi monitorano la situazione di compilazione dei vari parametri.
A seguito verranno descritti i vari gruppi.
E' riportata la parte dello script che si occupa di testare la correttezza della **nomenclatura della famiglia**.

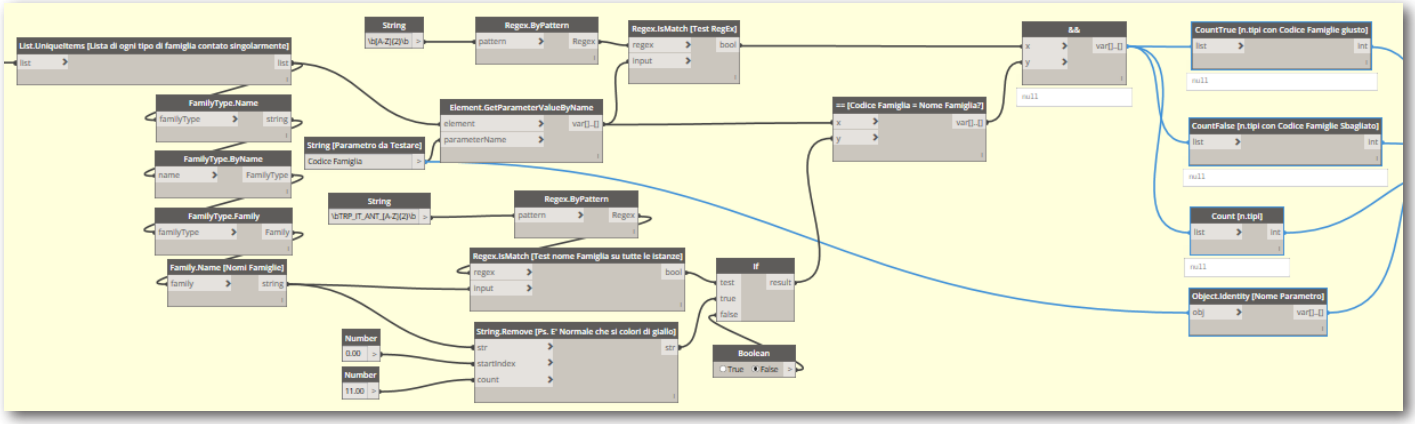


Immagine 84: Gruppo di nodi che controllano la correttezza del parametri `Codice Famiglia`.

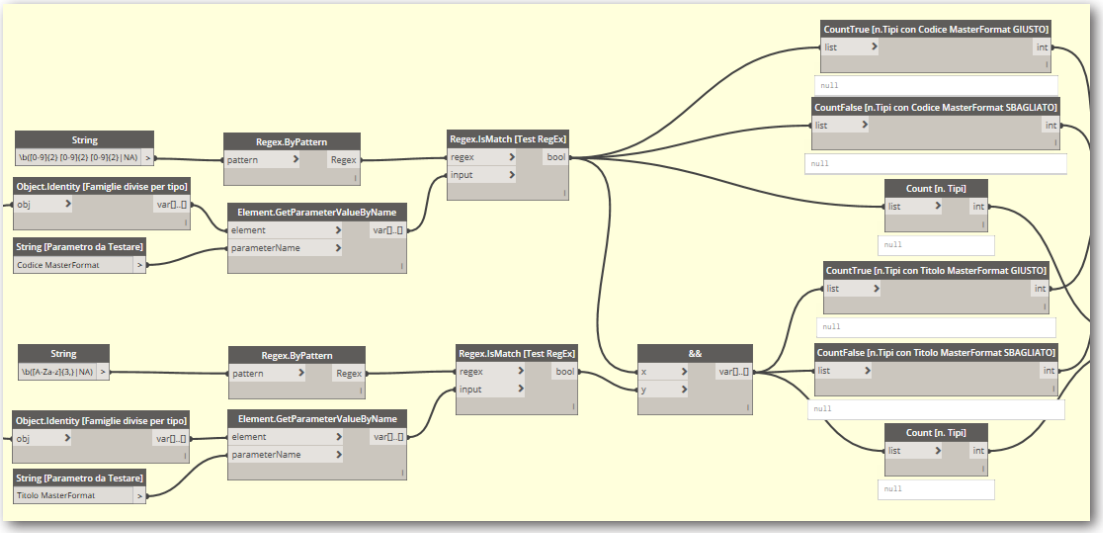


Immagine 85: Gruppo di nodi che controllano la correttezza dei parametri `Codice MasterFormat` e `Titolo MasterFormat`.

Si noti il nodo **List.UniqueItem** che **permette di definire una cella della scheda per ogni famiglia**. Mostrata questa parte di script, è possibile dare per nota anche quella che controlla il nome del tipo.
La struttura presentata nell'immagine dopo, si riferisce invece alla parte di script che **controlla la correttezza del parametro `Codice Famiglia`**; nella parte alta è presente una normale struttura di test basata sulle RegEx. Contemporaneamente, il `Codice Famiglia` viene confrontato con le ultime lettere del nome della famiglia le quali, come visto in precedenza nella tesi, dovrebbero essere proprio coincidenti con il codice famiglia.
Affinché il codice famiglia sia dunque considerato valido per un determinato tipo, è necessario che entrambe le verifiche siano state passate.
Questa seconda verifica basata sul nome della famiglia è presente soltanto per le categorie caricabili.
Per quanto riguarda la verifica dei parametri MasterFormat,

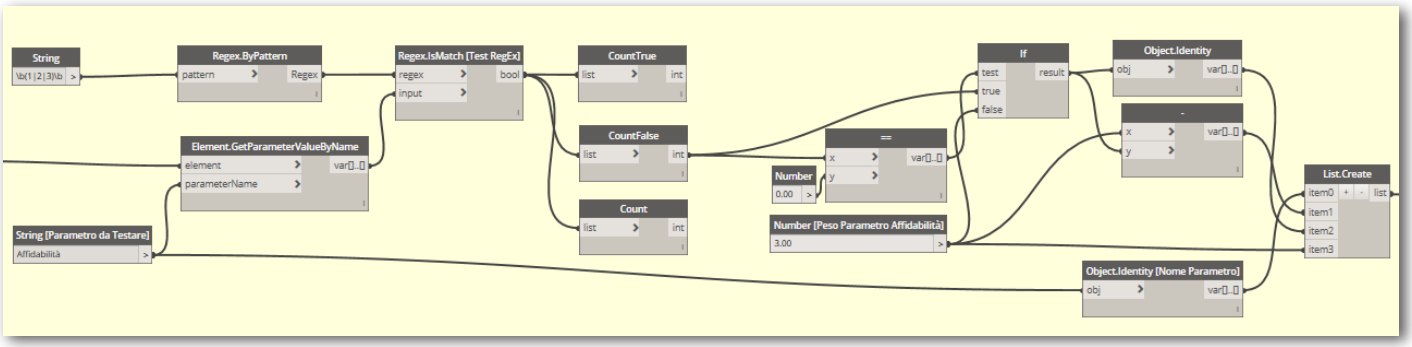


Immagine 86: Gruppo di nodi che controllano la correttezza del parametro `Affidabilità`.

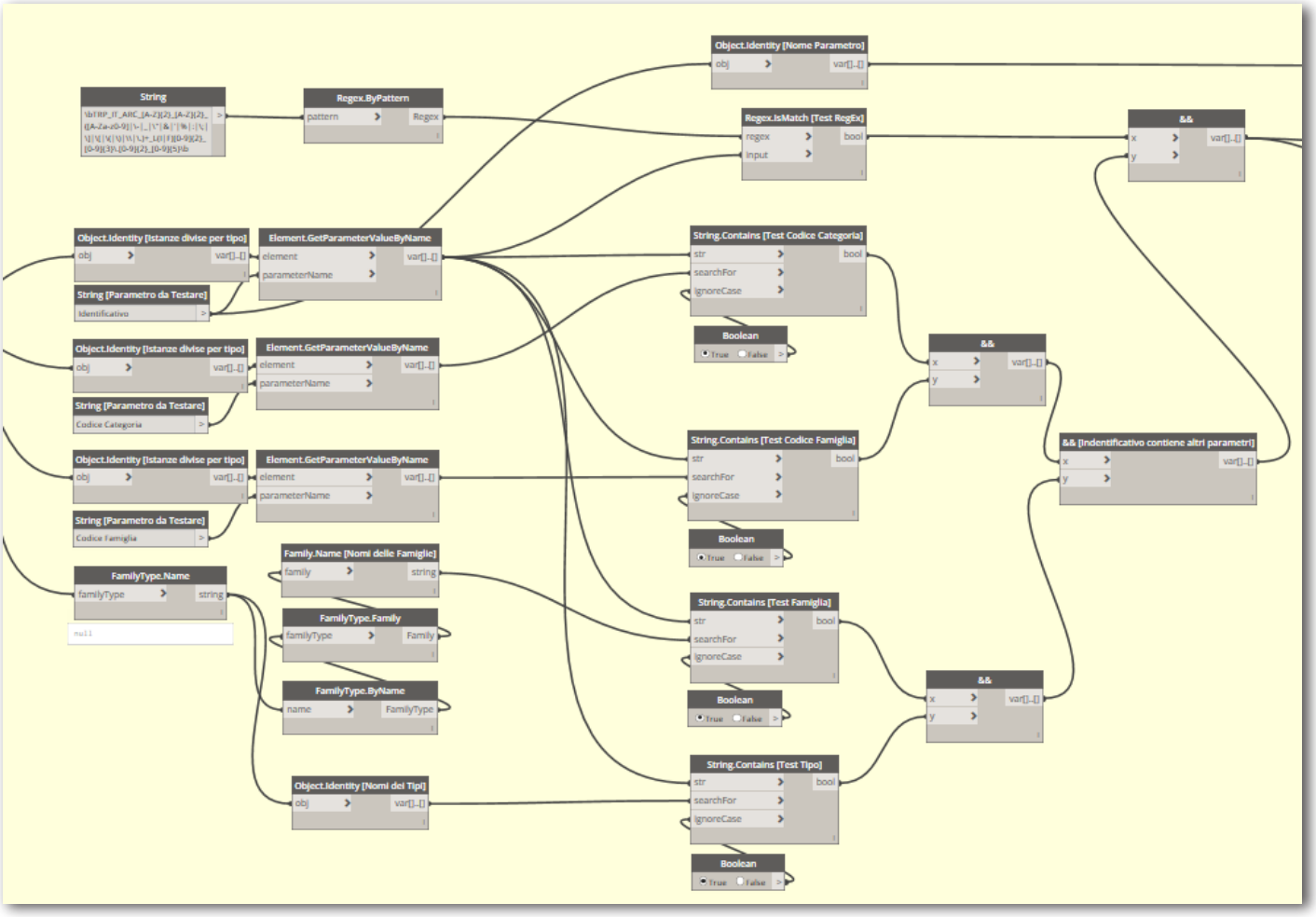


Immagine 87: Gruppo di nodi che controllano la correttezza del parametro `Identificativo`

essa viene svolta in modo non dissimile a quanto visto per altri parametri. Lo script effettua la verifica separatamente sui due parametri andando però a considerare errore una situazione in cui il titolo MasterFormat, pur essendo formalmente giusto, corrisponde ad un Codice MasterFormat formalmente sbagliato.

Si è poi riportato il **test per l'`Affidabilità`**, con il sistema che considera tutte le celle della scheda sbagliata qualora

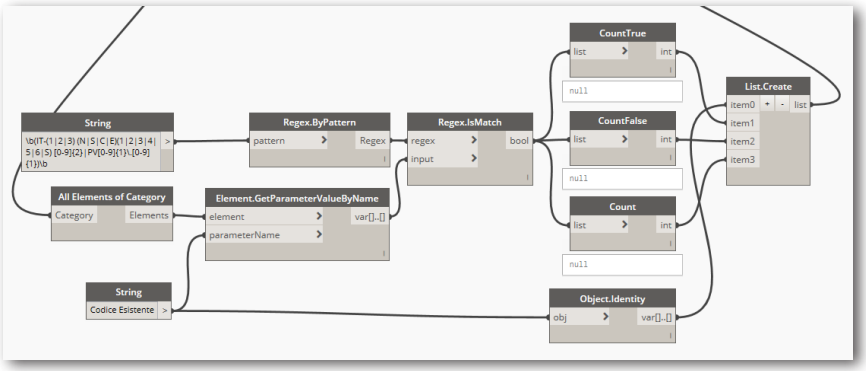


Immagine 88: Gruppo di nodi che controllano la correttezza del parametro `Codice Esistente` nella categoria Porte.

anche solo un caso non superi il test della RegEx.

Infine il **controllo sull'`Identificativo`** che controlla che oltre a rispettare la relativa RegEx, esso contenga il `Codice Categoria`, il `Codice Famiglia`, ed il nome del Tipo.

Attraverso l'utilizzo di porte logiche "AND" (ovvero il nodo **&&**) è possibile filtrare in modo da segnalare con `true` solo i codici che hanno superato tutte le verifiche.

Si considera oramai intuibile il contenuto del Custom Node che agisce sul `Codice Esistente`.
Si riportano a seguito, i nodi che sostituiscono il Custom Node nel caso della categoria Porte:

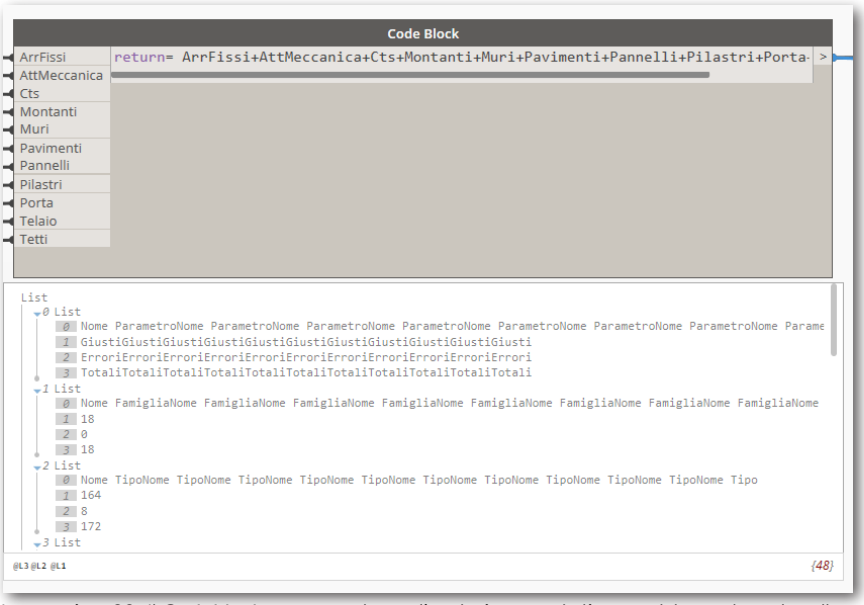


Immagine 89: Il **Codeblock** aggregatore di valori con relativo problema legato alla concatenazione di stringhe.

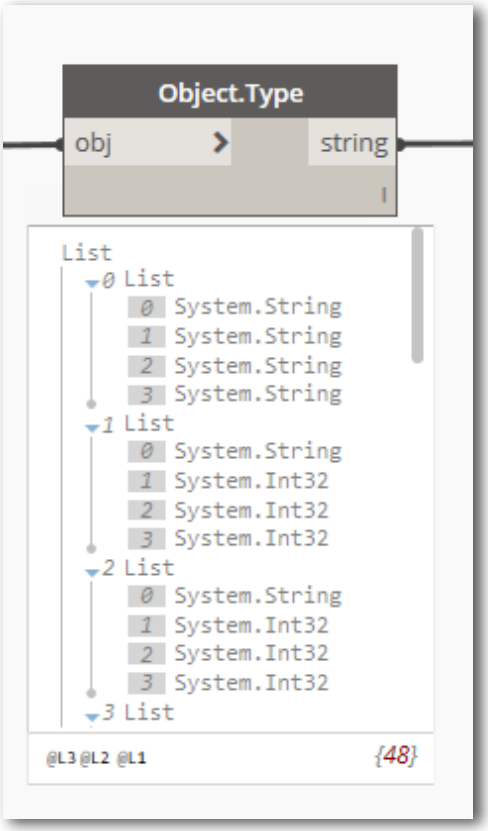


Immagine 90: Il nodo **Object.Type**

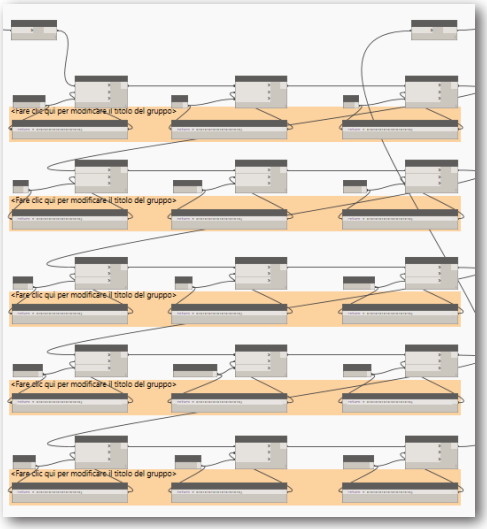


Immagine 91: Cluster di nodi String.Replace

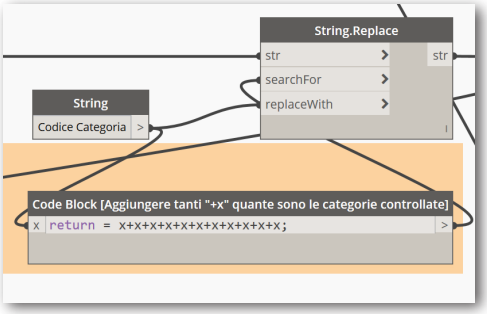


Immagine 92: Uno dei nodi String.Replace

Arredi fissi	Nome Parametro	Giusti	Errori	Totali
	Nome Famiglia	1	0	1
	Nome Tipo	1	0	1
	Progetto	1	0	1
	Edificio	0	1	1
	Codice Categoria	0	1	1
	Codice Famiglia	0	1	1
	Terna Unità Tecnologiche	2	1	3
	MasterFormat	4	0	4
	Affidabilità	3	0	3
	Identificativo	1	0	1

Immagine 93: Esempio di una delle n tabelle stampate su Excel per una categoria. Le celle sono colorate in relazione alla tipologia di dato che contengono. In beige dati numerici interi, in azzurro caratteri stringa.

A questo punto, ciò che si ottiene è una serie di liste a due livelli. Ciascuna lista, riferendosi ad una sola categoria riporta la situazione di completamento della scheda per ciascun parametro (o gruppo di parametri nel caso ad esempio dei parametri MasterFormat).
Questi dati verranno stampati individualmente come matrici su un foglio Excel. Lo script si occupa però anche di fornire una situazione globale di tutte le categorie contemporaneamente.
Per fare ciò, si avvale di un **Codeblock con il compito di sommare n matrici, dove n è il numero delle categorie esaminate.**
Un problema riscontrato a questo punto è stato avere come **input** del Codeblock **matrici contenenti dati di tipologia differente**, ovvero numeri interi e caratteri alfanumerici (stringhe). Tale nodo somma senza problemi i numeri mentre per le stringhe, concatena un numero n (in questo caso 11) di ripetizioni della stessa stringa.
In una situazione come questa, è utile il nodo Object.Type, il quale fornisce, sotto forma di Stringa, il nome del tipo di oggetto che si è dato in input. Sapere se una cella della matrice ha un contenuto di tipo stringa permette di filtrare le celle dove verrà effettuata una sostituzione, senza andare a toccare le celle contenenti numeri.
La sostituzione avviene facendo passare i dati attraverso dei nodi `Sting.Replace` posizionati in serie. Sotto ogni nodo di questo tipo, vi è un Codeblock che prede come input la stringa che si vuole inserire e la moltiplica per n volte in modo da avere la stringa errata che dovrà cercare il nodo Sring.Replace.

Purtroppo è necessario aggiornare il numero di "+x" in ogni Codeblock per farlo combaciare con n. Non è infatti possibile definire un moltiplicatore univoco per ciascun caso.
La tabella corretta può in questo modo essere esportata su Excel.

Output

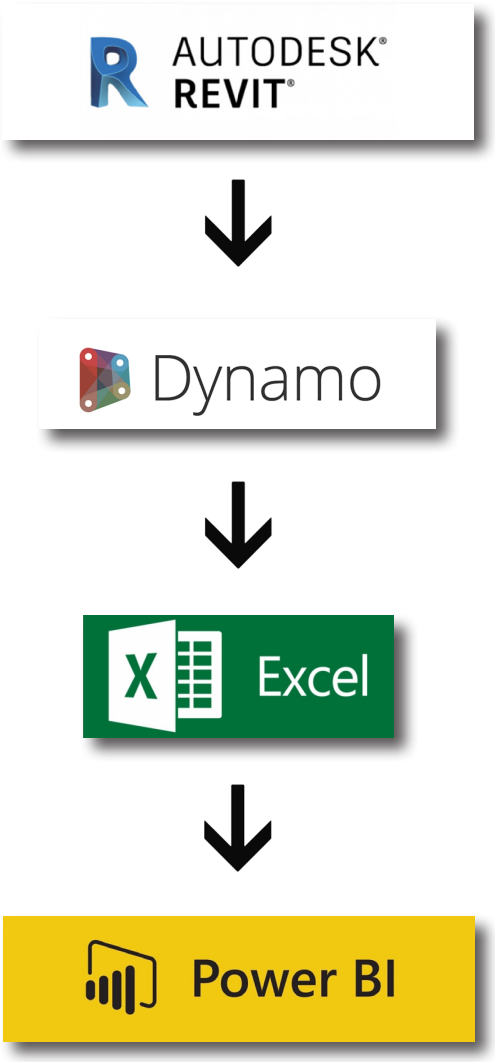
L'output dello script, come anche alcuni dei precedenti, si presenta come una lista di n tabelle riferite ad n categorie. E' possibile agire come visto nello script 1 e creare dei grafici a torta direttamente su Excel.
Avendo tuttavia molti grafici da estrapolare, in questo caso torna più utile appoggiarsi ad un tool esterno chiamato Power BI.
Power BI è un software gestito da Microsoft che **ha come scopo la creazione di dashboard dinamiche**, utili per la visualizzazione di dati. Esso è pensato proprio per visualizzazione di grafici e, nel farlo, risulta essere più completo e intuitivo di un programma non dedicato come Excel.
Power BI, in apertura, richiede l'**importazione di un file sorgente**, in questo caso, una tabella Excel. Un editor integrato permette di gestire i dati importati mentre la scheda principale è il vero e proprio editor della Dashboard e permette di creare, con i dati importati e visibili nella sezione "Dati", i grafici.
Come detto, Power BI fa riferimento a un file Excel avendone memorizzato la posizione. **In caso di modifica del file Excel di partenza**, ad esempio, a seguito di un ulteriore comando "Esegui" di Dynamo, **è possibile aggiornare i dati su Power BI** andando a cliccare l'icona "Aggiorna" presente sulla scheda Query.

Immagine 94: Logo di Power BI

Immagine 95: Icone delle sezioni "Report" "Dati" e "Modello" in PowerBI

Arredi fissi	Nome Parametro	Giusti	Errori	Totali	Attrezzatura m	Nome Parametro	Giusti	Errori	Totali	Controsoffitti	Nome Parametro	Giusti	Errori	Totali
	Nome Famiglia	1	0	1		Nome Famiglia	3	0	3		Nome Famiglia	0	0	0
	Nome Tipo	1	0	1		Nome Tipo	21	0	21		Nome Tipo	7	0	7
	Progetto	1	0	1		Progetto	1	0	1		Progetto	1	0	1
	Edificio	0	1	1		Edificio	0	1	1		Edificio	0	1	1
	Codice Categoria	0	1	1		Codice Categoria	0	1	1		Codice Categoria	0	1	1
	Codice Famiglia	0	1	1		Codice Famiglia	0	21	21		Codice Famiglia	7	0	7
	Terna Unità Tecnologiche	2	1	3		Terna Unità Tecnologiche	42	21	63		Terna Unità Tecnologiche	21	0	21
	MasterFormat	0	4	4		MasterFormat	84	0	84		MasterFormat	28	0	28
	Affidabilità	3	0	3		Affidabilità	3	0	3		Affidabilità	3	0	3
	Identificativo	1	0	1		Identificativo	19	2	21		Identificativo	4	3	7
	Codice Esistente	0	1	1		Codice Esistente	0	1	1		Codice Esistente	0	1	1

Immagine 96: Esempio di Output grafico grezzo su Excel



La successione di loghi presente nella parte sinistra di questa pagina rappresenta in maniera schematica i software che sono stati utilizzati e il percorso che i dati compiono, partendo da un abaco Revit, fino ad arrivare ad una Dashboard dinamica su Power BI.
Si tratta di un esempio di quello che si intende con in termine **interoperabilità**, ovvero la comunicazione tra programmi differenti.
Va detto che in casi come questo in cui ad essere trattati sono dati alfanumerici e non delle geometrie complesse, la **probabilità di errori è relativamente bassa**.
I 4 software, inoltre, provengono da sole due software house, ovvero Autodesk e Microsoft e dunque ci si aspettava che nascessero **pochi problemi nel passaggio di informazioni da una piattaforma all'altra**.

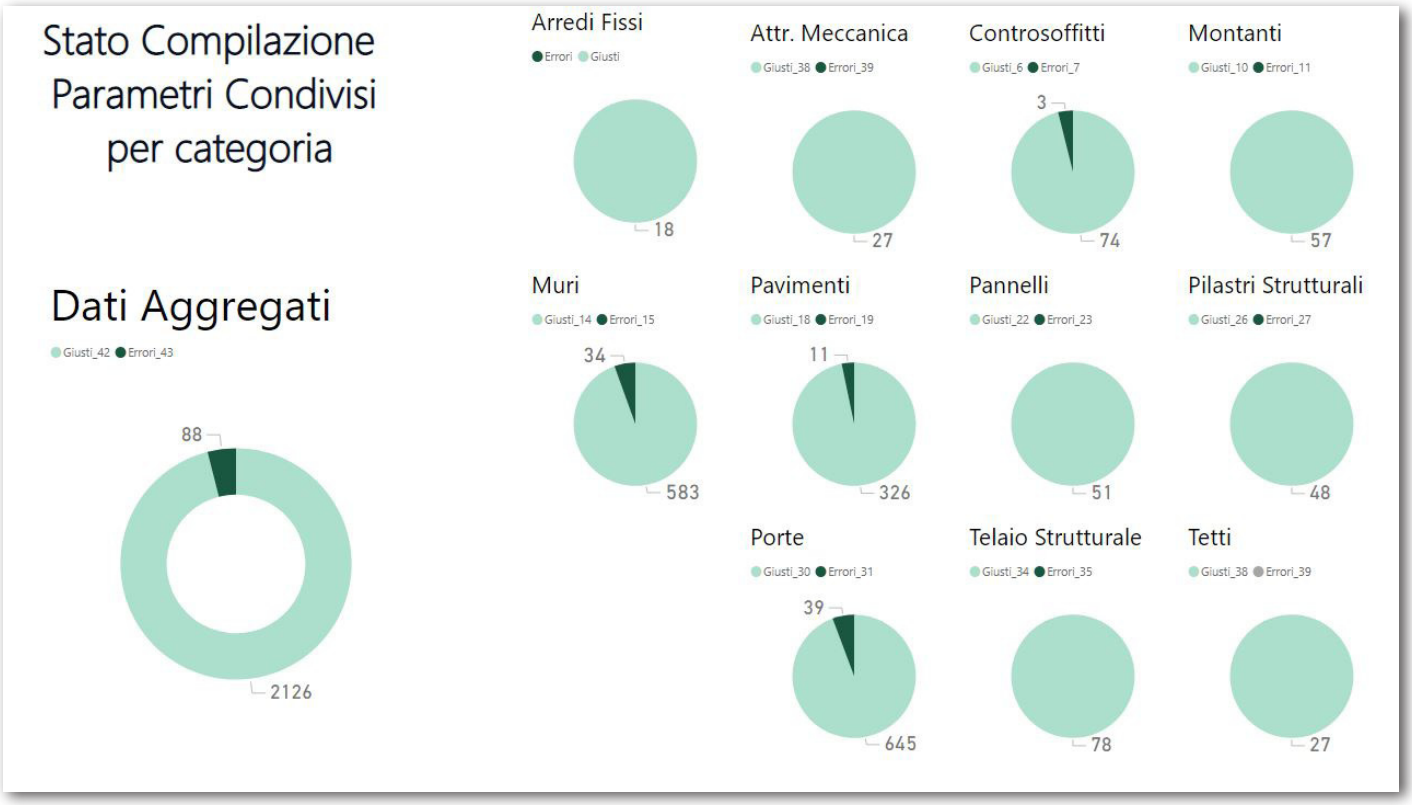


Immagine 97: Dashboard del modello Architettonico ottenuta con Power BI

Considerazioni e Possibili Sviluppi Futuri

Gli script si sono rivelati molto utili e veloci per gli obiettivi che si erano posti. Sono stati creati nell'ottica di essere il più possibile universali e fruibili anche da chi non possiede competenze di programmazione; ciò nonostante, richiedono ugualmente l'intervento dell'utilizzatore, di volta in volta, per andare a definire alcuni aspetti, come ad esempio adeguare le RegEx a seconda del fatto che ci si trovi in un modello Architettonico o Strutturale.
La **cosa più impegnativa da gestire su Dynamo in modo automatico è stata la gestione delle categorie.** Vi sono script (vedi in particolare lo script 6) in cui un blocco di nodi deve essere copiato e incollato per ogni categoria, rendendo dunque necessario stabilire a priori su quali categorie dovrà funzionare. Sempre parlando dello script 6, anche la stessa Dashboard con Power BI deve essere costruita ed ordinata sulla base del numero di categorie trattate definito preventivamente.
L'automatizzazione è dunque consentita solo a posteriori di un intervento di colui che ha creato lo script per adeguare quest'ultimo al modello oggetto di verifica o correzione.
Per rendere tale operazione più veloce e facile, si è deciso di creare dei **gruppi fittizi,** anche di un solo nodo, all'interno dello script, gruppi che **hanno il solo scopo di evidenziare i blocchi dove è richiesta l'attenzione dell'utente**.
Un altro problema riscontrato è stato quello di fare dialogare tra loro diverse versioni di Dynamo. Tutti gli script sono stati creati con la **versione 1.3.4.6666** e, per usarli con delle versioni successive, è stato necessario aggiornarli. Questo comporta il fatto di non essere compatibili con le versioni precedenti in quanto **Dynamo non è retro-compatibile**. Alla luce di ciò, è buona norma specificare all'interno del nome del singolo file di script, tra parentesi, la versione utilizzata.

Come già anticipato parlando del Code Checking, **Solibri Model Checker**, sviluppato da Graphisoft, (la stessa di ArchiCAD)è un software che permette di svolgere Code Checking e Clash Detection su un file .ifc.

Parlando invece di programmazione, il passo successivo all'utilizzo di Dynamo è rappresentato dalle cosiddette **Macro**.

Con la parola Macro, in informatica, si intende un blocco di azioni organizzate in modo da essere richiamate e svolte più volte, risparmiando cosi tempo ed errori.

Nonostante Dynamo sia installato automaticamente quando si installa Revit e nonostante sia legato ad esso, anche grazie a nodi che si occupano di estrarre informazioni dal BIM model, esso (Dynamo) è comunque un programma differente, tanto da poter essere all'occorrenza disinstallato individualmente.

Le macro, al contrario, vengono definite all'interno del software Revit e si basano sui linguaggi di programmazione Python e C#.

Rispetto a Dynamo, l'approccio tramite Macro risulta essere più complesso e meno chiaro, soprattutto agli occhi di un utente diverso da quello che le ha create. Di contro, esse permettono una libertà di azione ed una velocità di esecuzione degli script ancora maggiori di quanto visto con la programmazione visiva.

Fonti

Sitografia

- https://it.wikipedia.org/wiki/CAD
- https://www.evemilano.com/come-funzionano-le-espressioni-regolari-regex/
- https://it.wikipedia.org/wiki/Ken_Thompson
- https://en.wikipedia.org/wiki/Stephen_Cole_Kleene
- https://www.01building.it/bim/bep-documento-chiave-processo-bim/
- https://www.designcomputation.org/robert-aish
- https://www.ucl.ac.uk/bartlett/architecture/
- https://it.wikipedia.org/wiki/Grattacielo_della_Regione_Piemonte
- https://www.skyscrapercity.com/threads/torino-grattacielo-regione-piemonte-fuksas-209m-42p-t-o.640987/page-366
- https://www.agendadigitale.eu/infrastrutture/building-information-modeling-bim-cose-stato-di-adozione-in-italia-e-nel-mondo/
- https://people.unica.it/emanuelaquaquero/files/2020/05/Laboratorio-di-progettazione-integrata-Lezione-4.pdf
- https://biblus.acca.it/la-uni-11337-5-la-quinta-parte-della-normativa-tecnica-italiana-sul-bim/

Bibliografia

- "*La validazione del contenuto informativo è la chiave del successo di un processo BIM-based*" di Angelo Luigi Camillo Ciribini, Silvia Mastrolembo Ventura, Marzia Bolpagni
- "*Classification of BIM-based Model checking concepts*" di Eilif Hjelseth
- "Revit 2019 per l'Architettura - Guida completa per la progettazione BIM" di Simone Pozzoli, Marco Bonazza e Werner Stefano Villa.
- Norma UNI 11337