IndoorGML Reader, un plugin de QGIS.

Alfonso Mariscal Garcia
Grau d'Enginyeria Informàtica
Anna Muñoz Bollas
Gener de 2017

FITXA DEL TREBALL FINAL

Títol del treball:	*Implementació d'un plugin de QGIS: IndoorGML_ToolBox*
Nom de l'autor:	*Alfonso Mariscal Garcia*
Nom del consultor:	*Anna Muñoz Bollas*
Data de lliurament (mm/aaaa):	*01/2017*
Àrea del Treball Final:	*Sistemes d'Informació Geogràfica*
Titulació:	*Grau d'Enginyeria Informàtica*

Resum del Treball (màxim 250 paraules):

La irrupció massiva de dispositius personals amb capacitats navegacionals a l'exterior mitjançant receptors GPS ha impulsat també la necessitat de capacitar-los per fer el mateix en espais indoor. S'estableix així l'objectiu de dotar els usuaris d'aquests dispositius de transparència completa en l'experiència de navegació.

En aquest context, l'OGC ha oficialitzat l'estàndard IndoorGML com la referència en l'àmbit de representació espacial d'interiors amb propòsits navegacionals.

Aquest projecte contempla, en primer lloc, l'estudi profund i ampli de l'estàndard IndoorGML per tal d'incorporar els coneixements necessaris en l'àmbit concret d'aquest estàndard que faciliti el desenvolupament del treball i la consecució dels seus objectius.

L'aparició d'un nou model estàndard de representació implica la necessitat d'eines per la seva explotació. Prenent com a referència mecanismes d'exploració i lectura dels elements que composen el conjunt de dades i la seva representació gràfica, el treball presenta un inventari de les més rellevants i les seves capacitats, així com una conclusió clara; s'identifica la necessitat d'una eina versàtil, d'ús lliure i àgil per la lectura del format IndoorGML.

El mecanisme emprat per la resolució d'aquesta necessitat és la creació d'un plugin per al programari QGIS, una eina SIG de llicència lliure i d'àmplia implantació al mercat, amb capacitats de lectura del format IndoorGML. Aquest plugin s'implementa amb l'entorn de treball de Python i QGIS, que comprèn diverses llibreries i l'ús del programari Qt en el context de la definició de la interfície de treball.

El plugin ofereix una interfície pròpia on l'usuari podrà carregar l'arxiu amb el conjunt de dades, i el plugin generarà un arbre d'exploració dels elements que el conformen i en farà la representació gràfica.

Abstract (in English, 250 words or less):

The massive emergence of outer navigational capable personal mobile devices, working with GPS Technology, has triggered the necessity to give them abilities to offer the same services in indoor spaces. This fact establishes the goal to give users a completely transparent navigational experience.

In this context, OGC has appointed officially IndoorGML Standard as the reference in navigational purposes indoor spaces representation.

This project considers, in first term, a deep and wide study about IndoorGML standard in order to incorporate the required knowledge in this specific field which should make easier project development and its goals achievement.

A new representation standard apparition implies tools requirement in order to exploit it. Moving the focus to the exploration and representation, both in a structural and graphical view, of a concrete data set, this job offers an inventory of the most relevant and their capabilities. This triggers a clear conclusion; it's necessary a versatile, open source and agile tool in order to read IndoorGML format.

The chosen mechanism in order to solve this requirement is the creation of a QGIS application plugin which offers IndoorGML reading capabilities to it. This software is defined as an open source SIG tool, massively stablished in the market. The plugin is implemented in a Python and QGIS concrete environment, which comprise the incorporation of different libraries and the use of Qt software in the interface definition as well.

The plugin is offered though its own user interface, where user will be able to upload the file with the data set information, and the plugin will automatically create an exploration tree of the features read, and at the same time will perform the graphical representation.

Paraules clau (entre 4 i 8):

maps, indoor, navigation, positioning, gml, IndoorGML, QGIS, plugin

Índex

Llista de figures

1. Introducció

En aquest primer apartat del treball es presentarà el context i les premisses inicials que l'han fonamentat. En aquest sentit, també es tindrà oportunitat de fer inventari dels objectius que s'hi estableixen, els mecanismes previstos en la seva execució i una planificació temporal dels principals ítems a assolir.

1.1. Context i justificació del Treball

En els darrers anys la massiva adopció de dispositius mòbils amb capacitats de geolocalització ha fomentat l'aparició generalitzada d'aplicacions que basen la seva activitat en el posicionament de l'usuari. Fins i tot, aplicacions amb un focus d'activitat allunyat d'aquest (per exemple, les aplicacions de missatgeria) integren ara capacitats de geolocalització.

No obstant, actualment l'abast d'aquest tipus d'activitat es veu altament condicionada per la impossibilitat dels sistemes GPS per treballar en condicions normals en àmbits de navegació interior, com per exemple edificis. Aquesta situació justifica la necessitat d'oferir mecanismes que facilitin a l'usuari la transparència de fer transicions de navegació entre espais al aire lliure i espais interiors de forma desatesa.

En plena consciencia d'aquesta necessitat, l'Open Geospatial Consortium (OGC) ha oficialitzat recentment IndoorGML com l'estàndard de referència per la navegació espacial en interiors [7].

En l'àmbit de la descripció geomètrica, cartogràfica i espacial d'espais interiors ja es disposava d'altres estàndards, com el CityGML o el KML. Però, IndoorGML centra el seu focus en les capacitats navegacionals, amb objectiu de integrar aquestes capacitats entre espais interiors i exteriors.

1.2. Objectius del Treball

A continuació es farà revisió del llistat que contempla el conjunt d'objectius que és defineixen, i que poden ajudar a mesurar el grau de consecució del projecte.

1.2.1. Profunditzar en el coneixement de l'estàndard OGC IndoorGML.

Recentment l'Open Geospatial Consortium (OGC) ha designat de manera oficial el IndoorGML com l'estàndard de referència en la representació del model abstracte de dades per la informació espacial d'interiors, especialitzat aquest en els sistemes de navegació.

En tant que això, és objectiu d'aquest projecte assolir un grau de competència alt en el coneixement d'aquest estàndard.

1.2.2. Conèixer Estat de l'art actual del software per la lectura del format IndoorGML

Arran de la seva oficialització per part del OGC com a referència per la representació d'espais interiors en sistemes de navegació, l'estàndard IndoorGML ha agafat certa importància en l'àmbit de la representació SIG.

D'acord amb aquest supòsit, és precís fer un treball de recerca en diferents eines, tant de tipus open source com de pagament, que ofereixin capacitats de tractament, lectura, representació gràfica i operació d'aquest estàndard.

Aquesta recerca comprèn la necessitat d'oferir un inventari clar d'eines i conclusions clares respecte la seva usabilitat.

1.2.3. Implementar un plugin Python per QGIS per la lectura del format IndoorGML

QGIS és una eina d'Informació geogràfica gratuïta i de codi obert, disponible per diferents sistemes operatius i fruït d'un projecte dirigit per un col·lectiu voluntariat.

La utilització d'aquesta eina per part del col·lectiu d'usuaris d'eines de tipus SIG és amplia, a més el seu focus d'activitat és la col·laboració, per tant potenciant l'eina es capacita a la comunitat amb millors instruments de treball.

Així, és del tot adient fer el disseny d'un plugin que ajudi a incorporar la capacitat de lectura i visualització de dissenys basats en l'estàndard IndoorGML.

1.2.4. Publicar el plugin IndoorGML Reader al repositori oficial de QGIS

L'aplicació QGIS disposa d'un repositori oficial on els usuaris de la comunitat tenen oportunitat de publicar els plugins que desenvolupen, de manera que la resta de d'usuaris de la pròpia comunitat poden beneficiar-se de les funcionalitats de manera lliure.

Aquest objectiu queda condicionat i depenent, de manera inherent, a la consecució de l'objectiu per la implementació d'un plugin Python per QGIS per la lectura del format IndoorGML, ja que s'entén aquest com el producte d'entrada del procés per la seva publicació.

Per tal que la publicació sigui efectiva, aquest s'ha de sotmetre a un mecanisme d'aprovació, per part d'un usuari amb rol administrador a la comunitat, que preveu validar que aquest compleix un seguit de mínims establerts, aquests preveuen, entre d'altres i en termes genèrics:

- Ha de disposar de documentació.
- El seu funcionament ha de ser robust.
- Ofereix funcionalitats no cobertes per la pròpia eina.
- El contingut del codi és adequat.

1.3. Enfocament i mètode seguit

L'elecció de l'estàndard IndoorGML com a model de referència en la representació de dades navegacionals en espais interiors és recent, i aquest és el motiu pel qual el mercat d'eines SIG no l'ha incorporat com a estàndard de referència, així la capacitat de tractament és presumeix encara residual.

En aquest sentit, per tal d'afavorir la completa consecució dels objectius establerts amb anterioritat, és adient en primer lloc assumir la competència plena en la comprensió de l'estàndard, així com definir la situació de la capacitat de diferents eines amb propòsits SIG per a interpretar el nou format.

Donat que es preveu que la incorporació de mecanismes de lectura d'aquest format sigui immadura, s'entén que la creació d'un nou plugin per a l'eina QGIS, de massiva utilització per la comunitat SIG, oferirà un bon instrument a la comunitat per familiaritzar-se amb el nou estàndard, i que alhora això generi una bona base per al desenvolupament d'aquest.

1.4. Planificació del Treball

En aquest apartat es farà revisió de les fites més representatives del projecte, així com les dates clau en la consecució dels seus objectius intermedis i finals.

En aquest sentit, també seran presentades les activitats principals en què es segmentaren les tasques relacionades amb els diferents objectius.

1.4.1. Calendari de fites i dates clau

Es presenta a continuació l'inventari de fites i dates clau relacionades amb l'elaboració i consecució de les diferents tasques del projecte.

- Inici de l'assignatura
  Data: 21 de Setembre de 2016.

- Lliurament esborrany PAC1.
  Data: 1 d'Octubre de 2016.

- Lliurament PAC1 – Pla de Treball.
  Data: 4 d'Octubre de 2016.

- Lliurament esborrany PAC2.
  Data: 30 d'Octubre de 2016.

- Lliurament PAC2 – Fonaments teòrics i tècnics.
  Data: 8 de Novembre de 2016.

- Lliurament esborrany PAC3.
  Data: 29 de Novembre de 2016.

- Lliurament PAC3 – Implementació i publicació de plugin per a QGIS.
  Data: 6 de Desembre de 2016.

- Lliurament esborrany PAC4.
  Data: 2 de Gener de 2017.

- Lliurament PAC4 – Lliurament final.
  Data: 9 de Gener de 2017.

- Lliurament Informe d'Autoavaluació.
  Data: 9 de Gener de 2017.

- Inici del debat virtual.
  Data: 17 de Gener de 2017.

- Fi del debat virtual.
  Data: 19 de Gener de 2017.

1.4.2. Principals activitats

Per tal d'assolir els diferents objectius plantejats en el Treball de Final de Grau, es defineixen una sèrie d'activitats principals, la consecució de les quals es indispensable per aconseguir el resultat esperat.

Creació Pla de Treball			
Dedicació		Temporització	
Hores	Número de planes	Data inici	Data fi
24	20	26/09/16	04/10/16

Elaboració d'un document que enregistrarà la planificació del conjunt de tasques que comprenen la consecució del Treball de Final de Grau. S'establiran còmputs de dedicació, identificaran fites, dates clau, i s'exposaran els riscos identificats.

Entrega PAC1			
Dedicació		Temporització	
Hores	Número de planes	Data inici	Data fi
24	20	26/09/16	04/10/16

La PAC1 serà el primer lliurament oficial, que estarà composat principalment pel Pla de Treball del Treball de Final de Grau.

Estudi i comprensió de l'estàndard IndoorGML			
Dedicació		Temporització	
Hores	Número de planes	Data inici	Data fi
25	10	10/10/16	18/10/16

Un dels pilars sobre els que es fonamentarà el projecte serà la comprensió minuciosa de l'estàndard XML anomenat IndoorGML. Es defineix com la base integradora de les tasques de tota l'evolució del projecte, és imprescindible coneixen detalladament l'arquitectura.

Recerca d'eines pel tractament de dades IndoorGML			
Dedicació		Temporització	
Hores	Número de planes	Data inici	Data fi
8	6	18/10/16	24/10/16

Arran de l'oficialització del model IndoorGML com a l'estàndard de referència per la representació de dades d'interior en l'àmbit de la navegació, queda palesa la necessitat d'eines amb capacitats de gestionar el model.

Aquest grup d'activitats tindrà com objectiu fer una recerca, anàlisi i avaluació de diferents eines amb aquestes capacitats.

Entrega PAC2			
Dedicació		Temporització	
Hores	Número de planes	Data inici	Data fi
24	20	26/09/16	04/10/16

En la segona entrega oficial de l'assignatura s'oferirà informació completa de l'anàlisi i la comprensió del model IndoorGML, així com investigació de l'estat de l'art de les diferents eines amb capacitat de interactuar-hi.

En el marc d'aquesta entrega es preveuen conclusions de la necessitat d'una eina per la lectura del model de dades IndoorGML.

Comprensió codi programació Python			
Dedicació		Temporització	
Hores	Número de planes	Data inici	Data fi
20	10	8/11/16	14/11/16

El codi de programació Python té unes condicions òptimes per l'aprenentatge àgil, i a més es presenta com una de les principals alternatives per tal de desenvolupar plugins per l'aplicació QGIS.

Desenvolupar plugin IndoorGML en Python per QGIS			
Dedicació		Temporització	
Hores	Número de planes	Data inici	Data fi
30	15	22/11/16	27/11/16

Un cop assolides les habilitats necessàries quant a la comprensió del codi de programació Python, i amb l'objectiu de satisfer la necessitat de generar un plugin per la lectura de l'estàndard IndoorGML, el desenvolupament del codi necessari per el plugin, així com la creació d'aquest, esdevenen una activitat essencial en el projecte.

Publicar plugin IndoorGML en Python per QGIS			
Dedicació		Temporització	
Hores	Número de planes	Data inici	Data fi
2	4	28/11/16	28/11/16

Tenint en compte la natura intrínseca de l'aplicació QGIS, de codi lliure, és apropiat que el plugin desenvolupat estigui disponible per la comunitat. Aquesta fase comprèn les activitats necessàries per tal de dur a terme aquesta publicació.

Entrega PAC3			
Dedicació		Temporització	
Hores	Número de planes	Data inici	Data fi
24	20	26/09/16	04/10/16

La tercera entrega oficial de l'assignatura preveu oferir la versió definitiva del plugin per a QGIS, així com la publicació del plugin al repositori oficial.

Muntar memòria del projecte			
Dedicació		Temporització	
Hores	Número de planes	Data inici	Data fi
50	90	06/12/16	16/12/16

Un cop desenvolupades totes les activitats previstes al Treball de Final de Grau, és precís donar forma a totes aquestes tasques complimentant i enllestint la versió final de la memòria del projecte.

Aquest document recollirà tota la documentació relacionada amb les diferents activitats que s'han dut a terme.

Enregistrar presentació virtual del TFG			
Dedicació		Temporització	
Hores	Número de planes	Data inici	Data fi
35	25	16/12/16	03/01/17

Una de les activitats més importants dins el Treball de Final de Grau és la creació d'una presentació virtual d'aquest, que ajudarà a oferir una visió completa del conjunt d'activitats i tasques realitzades, així com els diferents productes generats.

Entrega PAC4			
Dedicació		Temporització	
Hores	Número de planes	Data inici	Data fi
24	20	26/09/16	04/10/16

La darrera entrega oficial de l'assignatura, comprèn el lliurament de la versió definitiva de la memòria, així com la presentació virtual del projecte i el document oficial d'autoavaluació.

Debat virtual			
Dedicació		Temporització	
Hores	Número de planes	Data inici	Data fi
24	N/A	16/01/17	19/01/17

La darrera activitat de l'assignatura és la participació en el debat virtual on és generarà un espai comunicatiu entre l'estudiant i el tribunal d'avaluació al voltant de les tasques relacionades amb el propi Treball de Final de Grau.

1.5. Breu sumari de productes obtinguts

En base a totes les tasques relacionades amb la consecució del projecte, seran generats tot un segui de productes que ara exposem breument.

- **Pla de treball.**
  Document que inventaria les diferents àrees de treball del TFG, fent-ne un desglossament de tasques, i oferint-ne una previsió d'agenda temporal.

- **Memòria de treball.**
  Aquest document contindrà la documentació generada fruit de les diferents tasques, així com previsió temporal detallada, referències, etc. És un dels elements principals del projecte.

- **Plugin Python per QGIS.**
  El plugin de Python que es generarà per a l'aplicació QGIS serà entregat en un fitxer comprimit, acompanyat d'un conjunt breu de instruccions per la seva instal·lació i fitxers d'exemple en format IndoorGML per fer-ne ús.

- **Vídeo de presentació virtual.**
  Tot el conjunt de tasques realitzades es presentaran de forma sintetitzada en format multimèdia. L'objectiu és fer una presentació dels diferents aspectes que han envoltat totes les tasques del projecte.

1.6. Breu descripció dels altres capítols de la memòria

Aquest projecte, que s'emmarca en l'estudi del model de representació de dades de representació d'espais interior IndoorGML ha estat projectat en diferents capítols, que ofereixen cobertura a les diferents àrees de treball del projecte.

A continuació s'entregarà una breu descripció d'aquestes seccions:

- **Capítol 2. Estudi de l'estàndard OGC® IndoorGML**

L'estàndard IndoorGML ha estat anomenat recentment com la referència per la representació d'espais interiors amb objectius navegacionals.

En aquest capítol es porta a terme una revisió dels principals aspectes que ajuden a fer comprensió del funcionament de l'estàndard, així com la seva definició d'espais, de manera conceptual i fins i tot en l'àmbit de la composició de l'estructura del seu model de dades.

- **Capítol 3. Estat de l'art actual del software per la lectura del format IndoorGML**

La situació contemporània del grau d'adaptació de l'estàndard IndoorGML per les principals eines de lectura i gestió de continguts en l'àmbit de la representació d'espais és rellevant en el context d'aquest projecte.

La immaduresa que les principals eines, pel que fa a la compatibilitat amb l'estàndard de referència, justificaran la identificació de la necessitat d'oferir una alternativa de lectura i representació àgil i versàtil.

- **Capítol 4. Construcció d'un plugin per la lectura del format IndoorGML**

Aquest capítol ofereix la informació necessària per la comprensió de les tasques completades amb l'objectiu de definir un plugin per a l'eina de representació de informació geogràfica QGIS.

A més, els continguts pretenen també entregar informació adient per entendre els objectius funcionals del plugin, en tant que detallant el motiu en els principals àmbits de decisió.

- **Capítol 5. Exemple d'ús del plugin IndoorGML**

Tenint en compte que el principal lliurable d'aquest treball és oferir un plugin amb capacitats de lectura i representació de conjunts de dades en format IndoorGML, aquesta secció té com a objectiu fer presentació d'un exemple simple i breu del seu funcionament.

- **Capítol 6. Conclusions i línies futures de treball**

D'acord amb l'abast de les tasques i objectius d'aquest Treball de Final de grau, és interessant fer-ne una descripció del grau de consecució, revisant-ne desviacions i mesurant-ne satisfacció.

En segon terme, també es preveuen futurs àmbits de desenvolupament que s'estableixin fruits de les conclusions preses en aquest treball.

- **Capítol 7. Glossari**

En aquest capítol s'ofereix un breu inventari del conjunt de termes i acrònims entregats en el transcurs de la memòria amb intenció de facilitar-ne la comprensió.

- **Capítol 8. Bibliografia**

La bibliografia d'aquest document oferirà referències reals a la documentació consultada com a font d'informació en l'àmbit del treball, així com mecanismes de difusió de continguts que ajudin a comprendre millor alguns aspectes que podrien donar lloc a interpretacions errònies o confusió.

- **Annexos**

Determinats continguts que no aporten valor divulgatiu al treball, però sí s'entén que ofereixen bons mecanismes per completar alguna tasca o millorar la qualitat del conjunt del treball han estat incorporats a aquesta secció.

En concret, s'hi estableixen els procediments per la instal·lació de l'aplicació QGIS així com el plugin IndoorGML Reader.

2. Estudi de l'estàndard OGC® IndoorGML

El format de representació geoespacial d'espais interiors IndoorGML [1] neix de la necessitat urgent de representació de rutes interiors per oferir a sistemes control de protocols d'emergències en espais interiors, serveis per discapacitats, o fins i tot per Indoor LBS (Location Based Services). A la Figura 1 observem la tendència positiva d'adopció al mercat d'aquest tipus de serveis els propers anys.

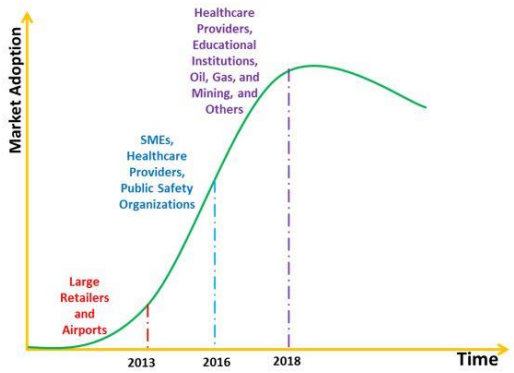


Figura 1. Adopció de sistemes LBS els propers anys. [2]

A efectes tècnics, els dissenyadors han fet un esforç per oferir només un mínim de dades de disseny i construcció geomètrica amb intenció de no assumir funcionalitats fora d'abast i que ofereixen altres estàndards, com IFC [3], KML [4] o CityGML [5]. En aquest sentit, IndoorGML està desenvolupat en el marc del esquema de GML (Geography Markup Language) en la seva versió 3.2.1.

En la línia de consecució del seu objectiu d'oferir informació dels espais interiors per capacitar-ne la navegació, es fan servir principalment dos mecanismes:

- Representació de les propietats de l'espai.
- Definició de referències de les característiques de l'espai.

2.1. Espai de cel·les

Aquest estàndard no està interessat en la representació arquitectònica dels espais, si no que la visió interessant és la conformació relacional que aquests acaben definint i com els elements tenen oportunitat de desplaçar-se en aquest espai i amb altres espais propers.

Aquest ajuda a introduir el concepte de cel·la com l'element estructural més petit que defineix un espai interior. De la mateixa manera, un espai de cel·les és un conjunt de cel·les que poden conformar la totalitat d'un edifici.

Per tal d'entendre millor la composició d'un espai de cel·les, podem dir que cadascuna de les cel·les ha de ser identificable en vers les altres, totes les cel·les han de tenir un espai fronterer amb d'altres, però sense que hi hagi superposició entre cap, i per últim la seva posició pot ser entregada amb l'identificador de cel.la o mitjançant les coordinades de la mateixa.

Aquestes cel·les poden tenir diferents representacions amb intenció que aquestes ajudin als propòsits de navegació, parlem de la semàntica, geomètrica i topològica.

2.1.1. Representació semàntica

Aquesta interpretació semàntica de l'espai permetrà fer diferents divisions cel·lulars en funció de la propietat que es vulgui representar. Un exemple realista i representatiu podria ser la necessitat de representar la divisió topogràfica de l'edifici – en tant que murs, finestres i portes – i alhora la de representar la cobertura WiFi – en funció de punts d'accés o fins i tot de degradació de la senyal – en diferents espais de cel·les.

Des del punt de vista de la navegació, la representació semàntica ofereix la capacitat de fer diferenciació entre cel·les navegables i aquelles que no ho són.

La integració de diferents espais semàntics es fa possible mitjançant la utilització de la representació d'espais multicapa, que seran analitzats més endavant en aquest mateix treball.

2.1.2. Representació geomètrica

La representació geomètrica és la definició pròpia de la forma de l'espai interior. Cal recordar que aquesta representació no és el focus de l'estàndard IndoorGML, ja que està plenament cobert per altres formats. No obstant això, és interessant puntualitzar en primer lloc que la definició d'aquesta estructura sempre vindrà condicionada per el sistema de coordenades de referència utilitzat (CRS) [6].

Pel que fa a l'estratègia de representació, tal i com es representa en la Figura 2, es proposen tres alternatives:

1. Referència externa: La informació geomètrica de les cel·les queda representada en forma d'enllaços simbòlics a altres conjunts de dades, en formats IFC o CityGML, per exemple.
2. Geometria pròpia: Dins el propi conjunt de dades s'ofereix informació geomètrica, en el format CRS definit, de l'espai de cel·les.
3. Sense informació geomètrica.

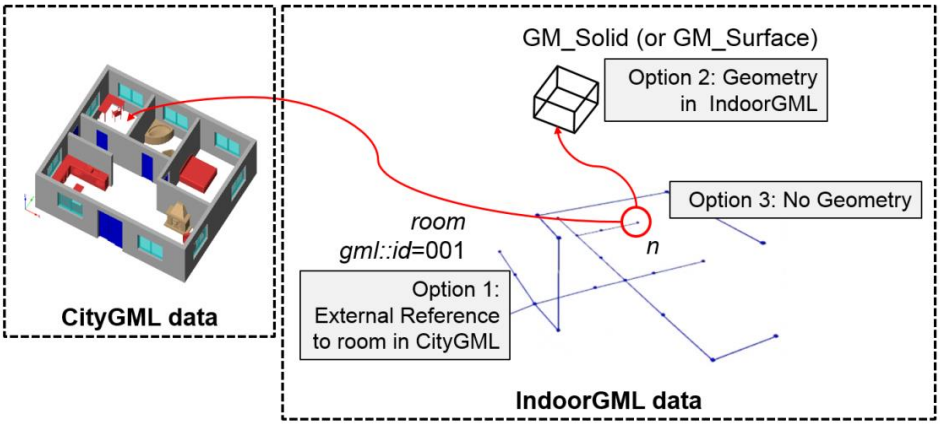


Figura 2. Alternatives per la representació geomètrica IndoorGML. [1]

2.1.3. Representació topològica

La informació topològica és essencial en el interès de definir un model que aporti capacitat per la navegació i, donat que no queda compresa en la representació geomètrica ni semàntica, s'ha fet un esforç important en oferir-la.

El mecanisme per la representació d'aquesta relació topològica entre diferents cel·les és el Graf de Relació entre Nodes (NRG) [1]. Aquest graf permet fer representacions de les relacions entre les diferents cel·les obviant la seva informació geomètrica, i representar-la només com si d'un punt es tractés. Aquest mecanisme, que es fonamenta en la dualitat de Poincaré [1] disposa de validesa tant per representacions geomètriques en tres dimensions com per aquelles que fan fent servir només dues. Així, en aquesta transformació, es defineix l'espai primitiu com aquell que queda definit per l'espai geomètric complet (que podria ser en dues o tres dimensions), i l'espai dual com aquell transformat en un NRG.

En primer lloc, la Figura 3 mostra una transformació des de l'espai primitiu de tres dimensions a l'espai dual:

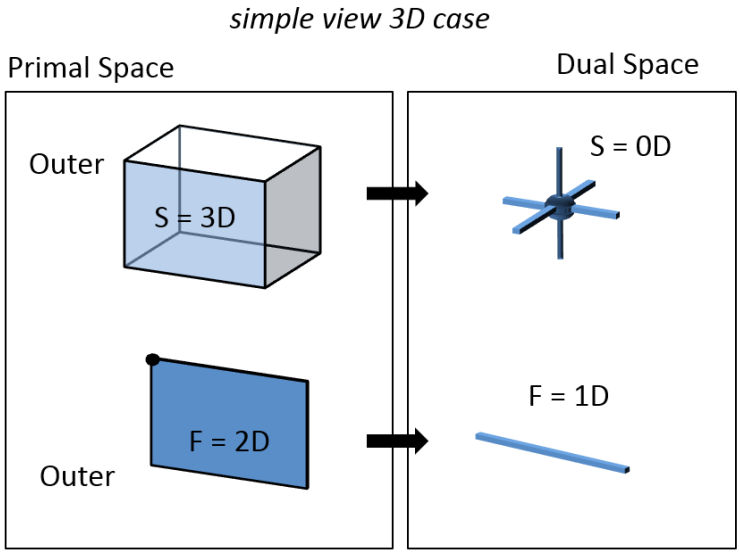


Figura 3. L'espai primitiu 3D i l'espai dual. [1]

De la mateixa manera en la Figura 4 es presenta un exemple gràfic de transformació d'un espai primitiu en dues dimensions a espai dual:

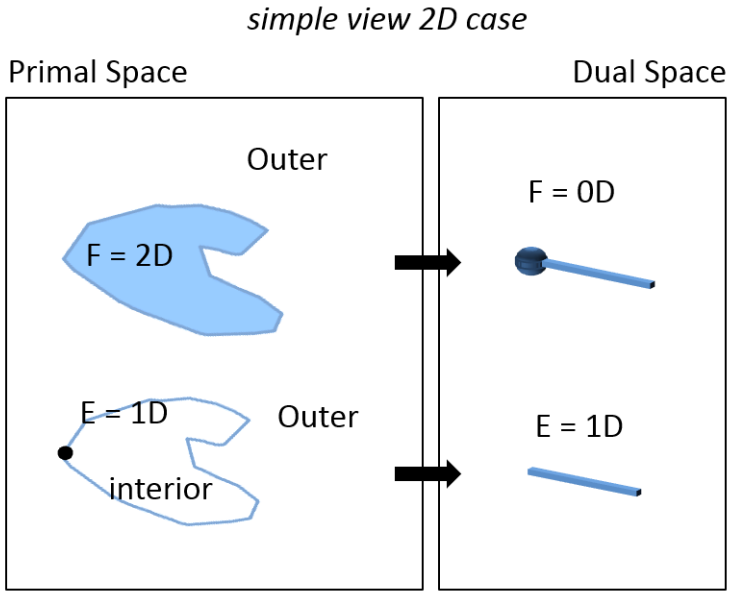


Figura 4. L'espai primitiu 2D i l'espai dual. [1]

La representació topològica, com s'esmentava amb anterioritat, s'estableix com un gran mecanisme per tal de representar relacions de diferent tipus entre les cel·les que conformen l'espai indoor.

- **Graf d'adjacències.**

Disposant d'una representació gràfica en dues dimensions d'un espai concret, mitjançant el mecanisme de transformació basat en la dualitat de Poincaré, és possible oferir la informació d'adjacència en forma de NRG, com es mostra en la Figura 5:

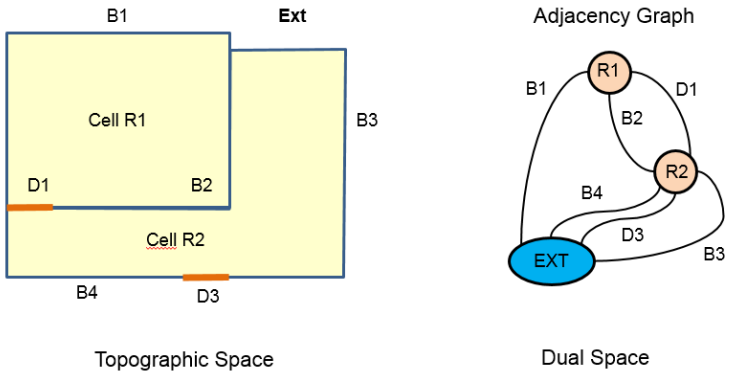


Figura 5. Espai topogràfic i graf d'adjacències. [1]

S'observa en aquest cas com la relació d'adjacència entre els diferents espais en el mapa topogràfic, s'estableix com relacions lineals entre els nodes del graf d'adjacències.

- **Graf de connectivitats.**

De la mateixa manera que s'ofereix el graf d'adjacències representat els punts de contacte entre els diferents espais d'un espai topogràfic, es poden representar les seves connectivitats, és a dir, incorporant informació lògica de quins espais ofereixen connectivitat entre ells:

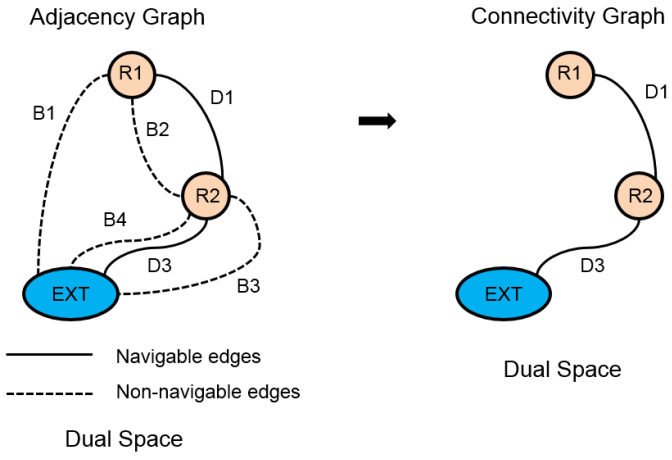


Figura 6. Graf d'adjacències i graf de connectivitat. [1]

En la Figura 6 és possible comprovar que el graf de connectivitats informa d'aquells espais que estan interconnectats, partint de la informació topogràfica que diu que existeix una porta entre un espai i l'altre, i entre el darrer i l'espai exterior.

2.1.4. Representació d'espai multicapa

En termes genèrics, qualsevol espai indoor precisa quedar definit en funció de diferents representacions semàntiques d'espais cel·lulars, en apartats anteriors s'han introduït les representacions d'espais de cobertura WIFI o topogràfics. Així, l'estàndard IndoorGML presenta de manera nativa la possibilitat de fer aquest tipus de representacions mitjançant la representació d'espai multicapa.

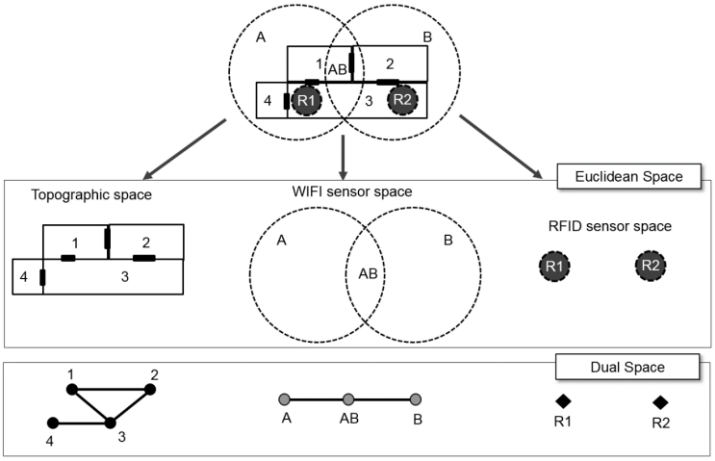


Figura 7. Espai multicapa amb els seus NRG. [1]

En l'exemple de la Figura 7, s'observa com el mateix espai es representat alhora en funció de l'espai topogràfic, la cobertura WIFI, i les àrees de cobertura RFID.

Aquest mecanisme no només permet presentar el mateix espai en diferents plans semàntics, sinó que a més l'espai dual quedarà també condicionat per propòsits de navegació, ja que no sempre la connectivitat de diferents espais serà la mateixa en l'espai dual que en el topogràfic.

2.2. Models espacials

L'estàndard IndoorGML es basa en la col·laboració de dos models espacials que satisfan el conjunt de necessitats establertes; el Model d'Espai Multicapa i el Model d'Espai Estructurat.

2.2.1. Model d'Espai Estructurat

Aquest model d'espai ofereix una distribució estàndard que defineix l'espai euclidià i topològic de l'espai primari, així com les seves respectives transformacions en l'espai dual.

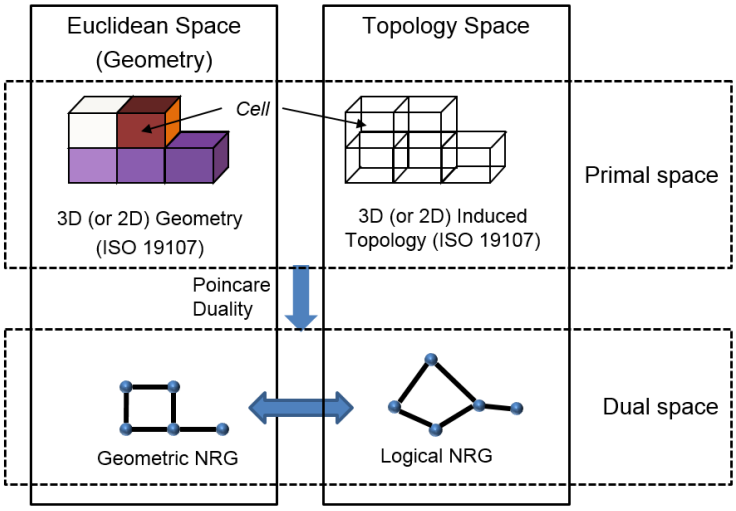


Figura 8. Model d'espai estructurat. [1]

D'aquesta manera, per a cada un dels espais representats dintre el conjunt de dades, es disposa de tota la informació geomètrica i topològica.

S'observa en la Figura 8 la representació en format NRG d'aquests dos espais, en el que els nodes, en aquest context, seran anomenats estats, mentre que els marges seran nomenats transicions. És interessant recordar que els dos NRG representen la connectivitat en termes de navegació en l'aspecte geomètric el primer, i topològic el segon.

2.2.2. Model d'Espai Multicapa

En el context navegacional, un mateix espai topogràfic pot requerir d'una representació diferent en funció del servei o necessitat que sigui precís satisfer, novament posant en context l'exemple de l'espai topogràfic i la cobertura WIFI. Tenint això en compte, es defineixen diferents composicions espacials completes que representaran aquests diferents àmbits.

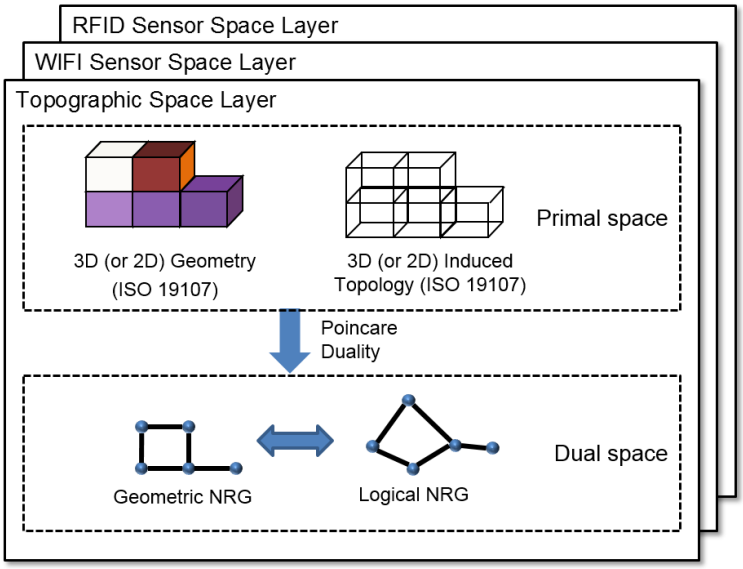


Figura 9. Model d'espai multicapa. [1]

En la Figura 9 és possible identificar diferents conjunts de dades del model d'espai estructurat, una per a cada capa definida, en el cas de l'exemple; la topogràfica, la de WIFI i per últim la de RFID.

2.2.3. Relació entre capes

Les capes dels diferents espais del model poden relacionar-se mitjançant connexions entre capes. Donat que en la realitat diferents capes, amb els seus espais de cel·les i grafs NRG, coexisteixen en el mateix espai físic, un element pot residir en el mateix instant en diferents estats o transicions, un en cada capa de les que hagin estat definides en el conjunt de dades.

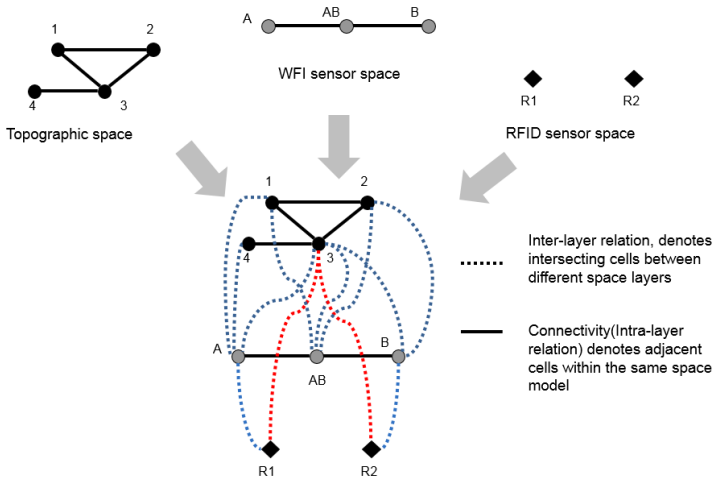


Figura 10. Relació entre diferents capes. [1]

En l'exemple de la Figura 10 es representa la definició del NRG de tres capes diferents; la topogràfica, la WIFI i la RFID. Així, la posició real d'un determinat objecte queda definit per la combinació del seu estat en les diferents capes.

Evidentment, només alguns estats de diferents capes són compatibles entre ells, això és fruït inherent de la necessitat que comparteixin cel.la topogràfica en l'espai primari.

La següent Figura 11 ajudar a comprendré com pot definir-se aquesta relació entre les diferents capes, les característiques i els espais de cel·les.

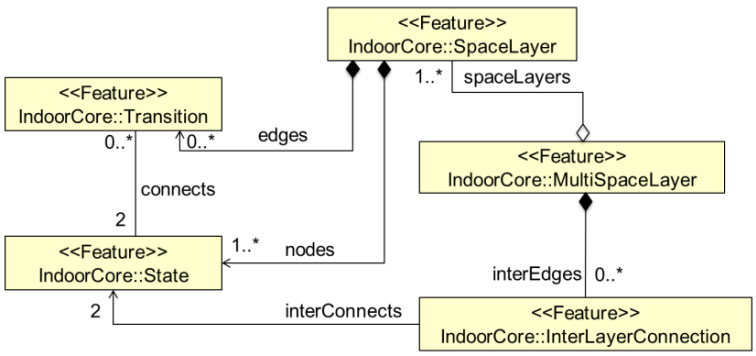


Figura 11. Representació UML relació entre capes. [1]

2.3. Referències externes

Donat que IndoorGML està focalitzat en la representació topològica de la informació, és molt probable que no disposi d'informació relativa a la geometria real dels espais. Com ja hem esmentat en la secció 2.1.2 d'aquest mateix treball, es proposa un mecanisme pel qual aquesta informació podria referenciar-se directament d'altres conjunts de dades, com per exemple CityGML.

Aquesta possibilitat de renunciar a fer una definició geomètrica dels espais dintre el propi conjunt de dades dota a aquest estàndard d'una bona capacitat de versatilitat i flexibilitat, donat que aquesta manera de referenciar facilitaria, per exemple, un eventual canvi en la disposició geomètrica dels espais.

2.4. Connexió amb espais exteriors

La iniciativa per la creació de l'estàndard IndoorGML, al menys de manera parcial, sorgeix de la necessitat de poder reproduir les capacitats de navegació tradicional en espais oberts. En aquest sentit, es defineix un marc de treball per tal de definir en quines condicions s'establirà la connexió dels mapes d'espais oberts amb els conjunts de dades per la navegació en espais interiors de IndoorGML.

Cada espai interior disposa al menys d'un punt de connexió real amb l'espai exterior, cadascun d'aquests elements seran representats en el model IndoorGML com un node especial de la definició topològica de l'espai, que serà anomenat node àncora. Aquest node serà diferent d'altres nodes, en tant que estarà dotat d'informació per tal de convertir, si cal, la informació CRS [6] de l'espai interior, amb la codificació CRS de l'exterior.

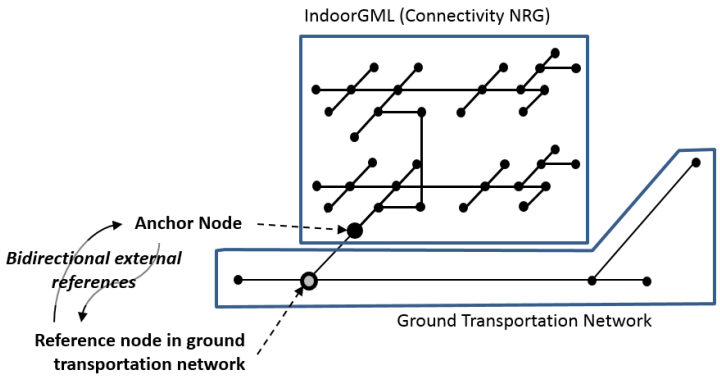


Figura 12. Exemple de node àncora. [1]

En l'exemple de la Figura 12, es veu com el node àncora del NRG IndoorGML incorpora informació de la seva codificació CRS que intercanviarà amb el sistema de navegació d'espais exteriors per tal de poder fer una connexió transparent entre ambdós mapes.

2.5. Models de dades

Seguint les directrius de l'OGC, la especificació del model de dades de IndoorGML queda segmentat en especificacions funcionals, en forma de mòduls. El principal serà el mòdul anomenat core, que entrega la definició bàsica i conceptual per al model, i també es presenten els mòduls d'extensió, que tenen com a objectiu oferir una ampliació de cobertura en diferents camps, com per exemple la navegació.

La Figura 13 mostra la definició dels mòduls IndoorGML core i IndoorGML navigation, dels que se'n fa revisió en aquests materials.

Module Name	IndoorGML core
XML Namespace Identifier	http://www.opengis.net/indoorgml/1.0/core
XML Schema File Name	indoorgmlcore.xsd
Namespace Prefix	IndoorCore
Module Description	The IndoorGML core module defines the basic components of IndoorGML data model. It includes the schema definitions of basic classes for cells, dual spaces and multi-layered space model. It is an application schema of GML 3.2.1.

Module Name	IndoorGML navigation
XML Namespace Identifier	http://www.opengis.net/indoorgml/1.0/navigation
XML Schema File Name	indoorgmlnavi.xsd
Namespace Prefix	IndoorNavi
Module Description	The IndoorGML navigation module defines the semantic extension of IndoorGML core module for indoor navigation. It defines the schema definitions of the classes for indoor navigation.

Figura 13. Mòduls IndoorGML core i navigation.

2.5.1. Mòdul IndoorGML core

El mòdul core de IndoorGML estableix els conceptes i components principals del model de dades per l'estàndard. En la Figura 14 es presenta el seu model de dades basat en el model de multi capes, i que té com a objectiu definir les classes i relacions necessàries per fer representació geomètrica i topològica de cada capa tant en l'espai dual com en el primitiu.

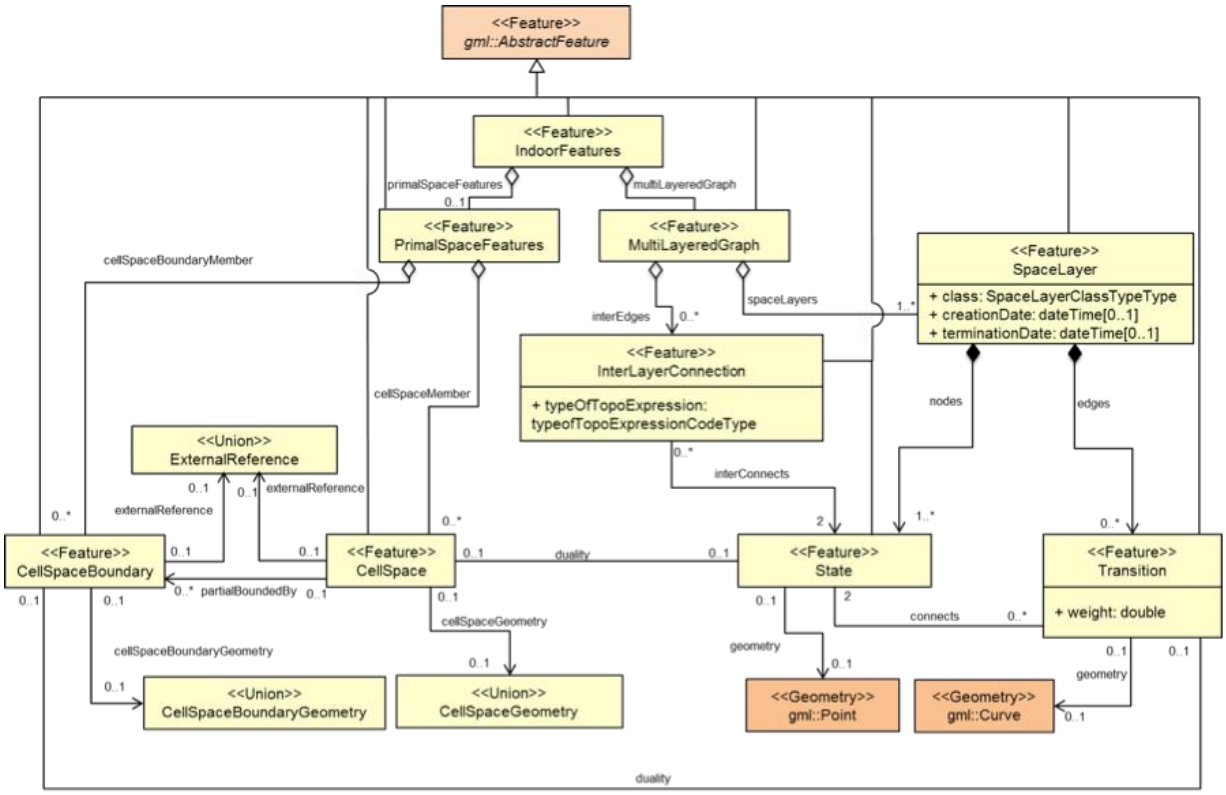


Figura 14. Diagrama UML per al model de dades IndoorGML core.

A continuació és defineix un breu inventari dels diferents elements que conformen l'esquema XML del model de dades core de IndoorGML:

- **State**

Cada node que es defineix a l'espai dual es representa mitjançant un State, pot disposar de connexió amb un altre State o estar aïllat, i associat a un espai dintre de l'espai primari – i.e. una habitació o un menjador, per exemple –.

Els diferents State disposen d'una relació de dualitat amb un CellSpace, i una relació de connexió amb un element de tipus Transition, que necessàriament enllaça dos o més State.

- **Transition**

Dintre de l'estructura de cada capa dual, l'element Transition estableix la unió entre diferents nodes – que reben el nom de State – i que alhora representen espais en l'espai primari.

Per a cada element del tipus Transition es disposarà d'una relació de connectivitat amb diferents State, i una de dualitat que representa la afiliació a un CellSpaceBounday a l'espai primari.

- **CellSpace**

El CellSpace és la classe que representarà l'espai interior en el seu aspecte purament geomètric. Aquesta relació geomètrica queda definida amb l'associació a un objecte geomètric.

Pel que fa a les relacions amb l'espai euclidià, aquestes es defineixen com a xlink, mentre que la referència al possible conjunt de dades extern s'informa amb l'atribut externalReference.

- **CellSpaceBoundary**

Cada element CellSpaceBoundary descriu els límits geomètrics de cada espai interior. De manera anàloga a la descrita per al CellSpace, l'atribut externalReference ofereix informació del paquet de dades extern al que correspon l'element.

- **SpaceLayer**

La classe que permetrà fer la representació de l'espai de dades al model és l'anomenada SpaceLayer, que s'encarrega d'oferir cobertura semàntica a l'espai. Aquesta disposa d'elements de tipus State i Transition de manera que capacita la presentació de l'espai dual.

D'altra banda, el mateix element SpaceLayer també identifica elements de classes CellSpace i CellSpaceBoundary per tal de poder fer relació amb els espais a l'espai primari.

- **InterLayerConnection**

Els elements de la classe InterLayerConnection presenten dos State, cadascun d'ells definits a un diferent SpaceLayer, i s'encarrega de definir la relació existent entre dos espais que pertanyen a diferents capes espacials. Aquesta relació entre State de diferents capes pot ser de pertinència, solapament o equivalència.

- **MultiLayeredGraph**

La classe MultiLayeredGraph te com a objectiu representar el model d'espai en capes mitjançant la conjugació d'elements de les classes SpaceLayer i InterLayerConnection.

L'objectiu en aquest cas és servir de punt de trobada entre diferents espais semàntics, en el que es preveu que el MultiLayeredGraph disposi de tots els State de cada SpaceLayer, així com la connexió entre aquestes que, com s'explicava amb anterioritat, capacita la classe InterLayerConnection.

- **PrimalSpaceFeatures**

Les característiques de l'espai de dades primari queden representades a la classe PrimalSpaceFeatures, contenint elements de les classes CellSpace i CellSpaceBoundary.

- **IndoorFeatures**

El model de dades core de IndoorGML presenta com arrel la classe IndoorFeatures, que alhora es segmenta entre les classes PrimalSpaceFeatures i MultiLayeredGraph, els detalls dels quals ja han estat exposats amb anterioritat.

2.5.2. Mòdul IndoorGML navigation

El mòdul navigation del model de dades de IndoorGML estableix mecanismes per el suport de les capacitats de navegació en espais interiors.

A la Figura 15 s'observa la definició conceptual en format UML del model estructurat de dades navigation.

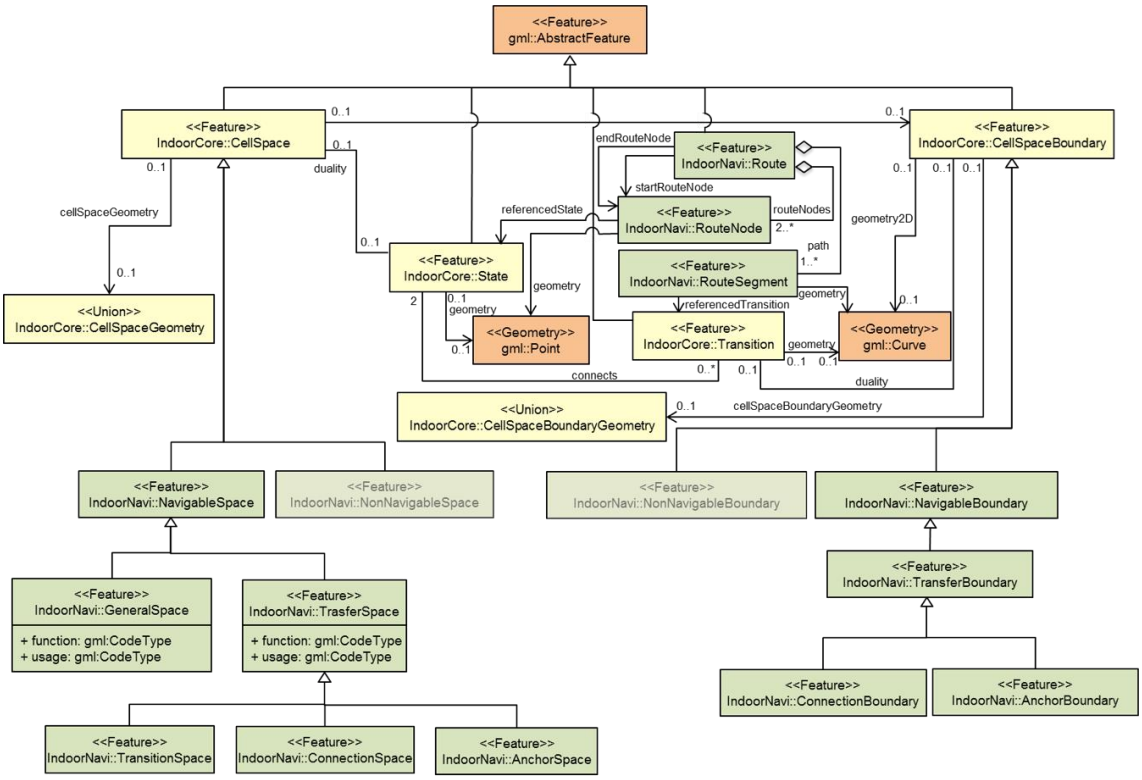


Figura 15. Diagrama UML per al model de dades IndoorGML navigation.

A continuació es presenta un breu resum de les principals classes que conformen el mòdul de dades:

- **NavigableSpace**

La classe NavigableSpace defineix l'espai que s'estableix com disponible per al moviment. Queda especificada per les subclasses GeneralSpace i TransferSpace.

La definició geomètrica quedarà representada per gml:Solid en geometries de tres dimensions i gml:Surface en aquelles de només dues dimensions.

- **NonNavigableSpace**

La representació dels espais que contenen obstacles i defineixen espais no navegables és farà mitjançant la classe NonNavigableSpace.

- **ConnectionSpace**

Els espais oberts que estableixen passadissos entre diferents espais interiors es defineixen a partir de la classe ConnectionSpace.

- **AnchorSpace**

De la mateixa manera que ConnectionSpace establia el passadís entre diferents espais interiors, la classe AnchorSpace defineix l'espai que interconnecta un espai interior amb un exterior. Acostuma a establir-se aquest element com el AnchorNode.

- **TransitionSpace**

L'espai que s'estableix entre dos elements diferents de tipus ConnectionSpace, i que genera l'espai real per la navegació entre ells, queda establert mitjançant la classe TransitionSpace.

- **NavigableBoundary**

La classe NavigableBoundary és defineix com el límit entre l'espai que permet la navegació i el que no.

S'estableixen dues subclasses; la ConnectionBoundary i la AnchorBoundary.

3. Estat de l'art actual del software per la lectura del format IndoorGML

L'acceptació del format IndoorGML ha estat oficialitzat recentment – el 20 de Gener de 2015 – per l'OGC (Open Geospatial Consortium) com a estàndard de codificació per a la informació de navegació a interior [7]. Aquest especifica un model abstracte i obert, així com un esquema XML especialment focalitzat en la navegació.

Fruit del curt espai de temps transcorregut d'ençà aquesta elecció, s'observa que l'acceptació d'aquest format no és generalitzat, i s'intueix una manca d'oferta de solucions software per a la seva lectura.

Així, en aquest apartat es farà revisió de les diferents alternatives, tant de llicencia lliure com de pagament, identificades en un anàlisi exhaustiu de l'oferta existent de solucions que es preveia que podien ser compatibles amb la seva lectura.

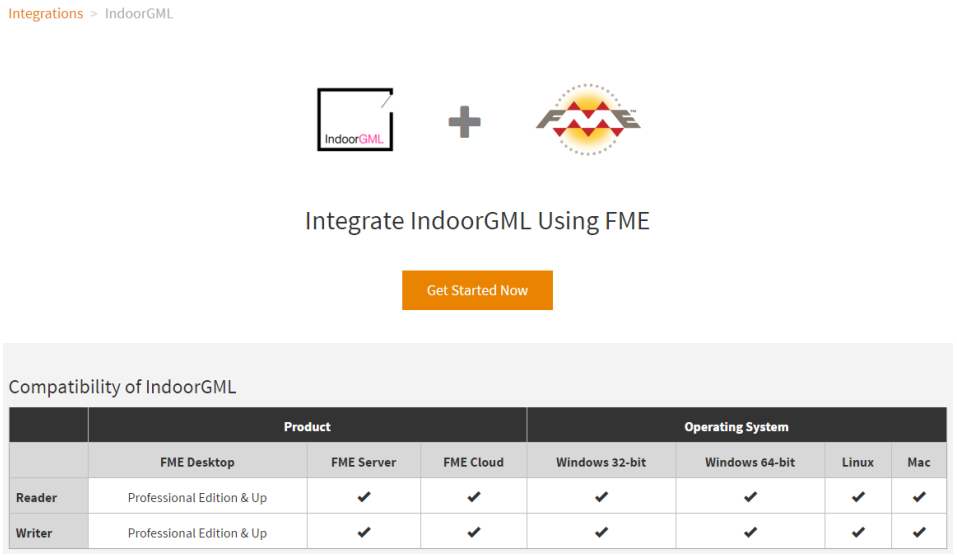
3.1. FME Desktop 2016

En l'àmbit comercial d'aplicacions per la lectura i transformació de conjunts de dades en diferents formats – sobretot els de localització espacial –, el suite d'aplicacions FME Desktop es posiciona com referència absoluta en el mercat. I ho és donat que ofereix compatibilitat amb la gran majoria de models disponibles.

Aquesta aplicació es desenvolupada per la companyia SAFE Software, la primera versió data de l'any 1993 a Vancouver, i sorgeix amb la intenció de donar resposta a la necessitat de compartir i gestionar mapes amb les autoritats governamentals [8].

3.1.1. Capacitat de lectura del format IndoorGML

A continuació, a la Figura 16, s'ofereix una taula extreta de la seva plana web, en la que es mostren les seves capacitats de lectura i/o escriptura amb el format IndoorGML vers les diferents versions, en funció del sistema operatiu suportat o tipus de producte:



	Product			Operating System			
	FME Desktop	FME Server	FME Cloud	Windows 32-bit	Windows 64-bit	Linux	Mac
Reader	Professional Edition & Up	✔	✔	✔	✔	✔	✔
Writer	Professional Edition & Up	✔	✔	✔	✔	✔	✔

Figura 16. Capacitats de lectura IndoorGML FME. [8]

Així, podem constatar que el producte FME Desktop s'ofereix com a totalment capacitat per fer lectura i edició del format IndoorGML.

3.1.2. Modalitats de llicenciament

Pel que fa al llicenciament i versions del producte, com es detallava a la taula de la secció anterior, la manipulació completa del format IndoorGML està suportat en totes les seves edicions – i.e. Professional, Esri, Database o Smallworld –. No obstant això, és important remarcar que existeix la possibilitat d'adquirir llicencies fixes o flotants, en modalitat Desktop, Server o SaaS. Aquestes presentaran diferencies en d'altres aspectes purament funcionals o fins i tot pel que fa a la compatibilitat amb d'altres formats.

El preu de la versió més econòmica en modalitat Desktop és, per una única llicència, de 2.000 $, mentre que si s'opta per la solució al núvol, sempre en funció de les capacitats de computació del maquinari, el preu seria de 0.90€ per hora en la seva versió més assequible. Els detalls son a la Figura 17.

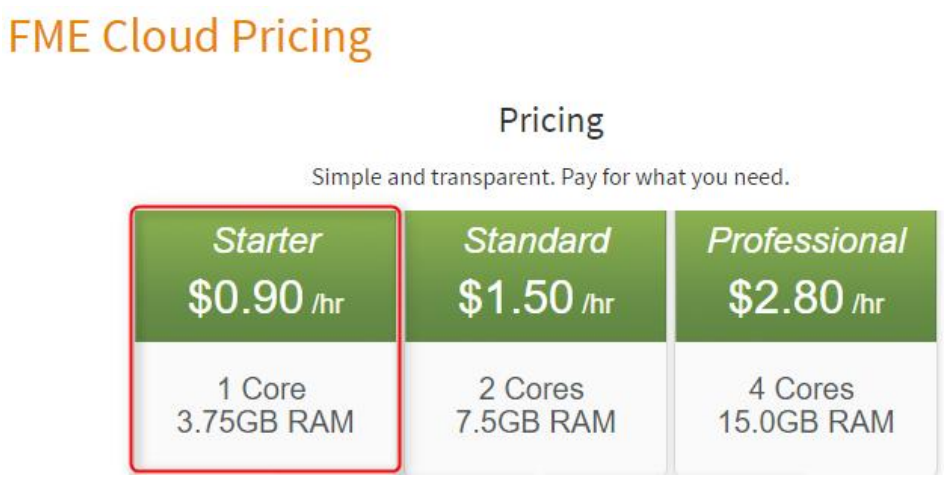


Figura 17. Modalitats de llicenciament FME. [8]

També és interessant remarcar que s'ofereixen condicions especials per a estudiants en actiu, recentment graduats o personal docent. En aquest sentit, aquesta possibilitat s'ha explotat per al desenvolupament d'aquest Treball de Final de Grau.

3.1.3. Conjunt de proves

Per tal de dur a terme la bateria de proves s'ha instal·lat la versió FME Desktop 2016 amb una llicencia per a estudiant que hem gestionat directament amb el departament de ventes de SAFE Software. En aquest sentit, en les proves s'ha fet servir un arxiu públic que la pròpia organització OGC ofereix en la seva plana WEB [9], amb l'aplicació del suite anomenada FME Data Inspector 2016.

- FJK-Haus_IndoorGML_withEXR.xml [9]

Un cop obert l'arxiu, s'observa en primer lloc que hi ha disponible una finestra anomenada Display Control, on es mostren totes les diferents entitats que comprèn el fitxer, a la Figura 18 es mostra un exemple, que són establerts en la definició formal del model de dades de l'estàndard IndoorGML [1]:

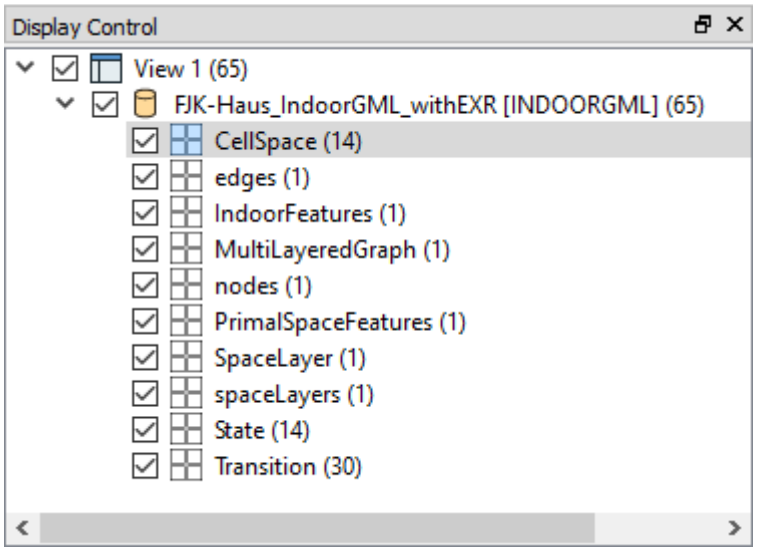


Figura 18. Captura de la finestra Display Control de FME.

En l'exemple d'aquest fitxer, s'entén aquesta finestra de l'aplicació com el mecanisme per tal de seleccionar quins tipus d'objectes dintre el fitxer volem que es mostri a la finestra View, com podem veure en la Figura 19:

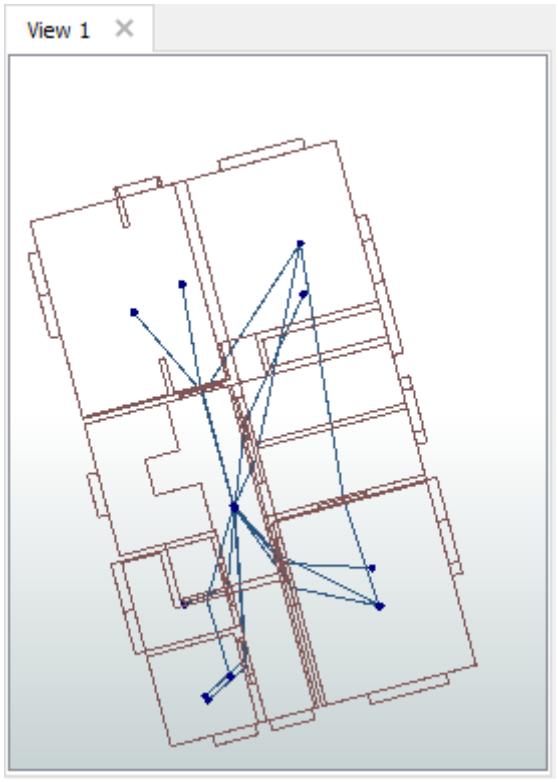


Figura 19. Captura de la finestra View de FME.

En aquest sentit, en la selecció d'aquest exemple també es mostra el llistat del conjunt d'objectes del tipus CellSpace a la secció Table View, a la que es pot arribar també en fer la selecció de la característica al propi panell View, es presenta un exemple a la Figura 20:

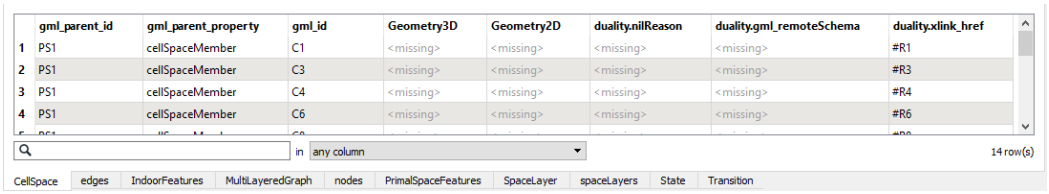


Figura 20. Captura de la secció Table View de FME.

Si bé en aquesta taula ja és possible fer una primera inspecció de la característica, en seleccionar un dels elements del llistat que es mostra a Table View, de la mateixa manera que en l'exemple de la Figura 21, s'observa com s'obre un conjunt de propietats definides per a l'element a la finestra Feature Information:

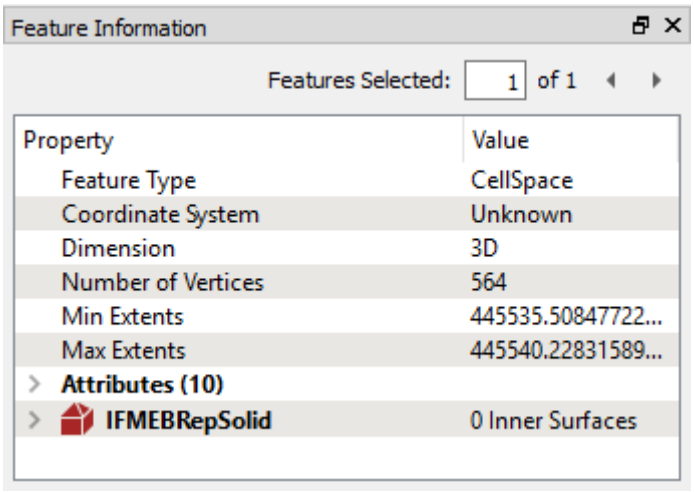


Figura 21. Captura de Feature Information de FME.

Dintre aquesta secció de l'aplicació es poden fer exploracions completes de la característica en la que l'usuari es troba, és a dir; analitzar els atributs, propietats, o característiques espacials.

3.1.4. Conclusions de l'eina

Aquesta és, amb tota probabilitat, l'aplicació que a efectes operacionals millor resol la lectura, edició i fins i tot visualització del format IndoorGML. La capacitat de seleccionar els elements visibles, navegar dins el propi disseny, o explorar detalladament la composició de cadascuna de les característiques entregant a l'usuari la possibilitat de moure's per les diferents seccions, acaba per establir un escenari amb una experiència d'usuari molt ben resolta.

D'altra banda, la instal·lació i posada en marxa de la eina és senzilla, àgil, i tenint en compte que es tracta d'un software de llicència de pagament, existeixen garanties de desenvolupament de noves característiques i correcció d'errades que satisfaran sens dubte la millora i gestió del canvi de l'eina.

No obstant això, el preu de les llicències és força elevat, i tenint en compte que els propòsits de la lectura de fitxers en aquest format no sempre poden estar relacionats purament amb l'àmbit empresarial, l'amortització d'aquest cost pot convertir-se en un problema infranquejable.

3.2. WebGL IndoorGML Viewer

S'analitza en aquest cas una eina desenvolupada íntegrament per membres de la Pusan National University [10], es tracta de Soojin Kim <soojin.kim@pnu.edu> i Hyung-Gyu Ryoo <soojin.kim@pnu.edu>. Tots dos formen part del STEM (Spatio-TEMporal data base Lab.) del Departament de Ciència de la Computació i Enginyeria de la mateixa universitat [11].

Aquesta universitat està referenciada en diverses classificacions universitàries, no només a Korea del Sud, sinó a nivell mundial. Els seus camps de referència son el científic-tècnic i tecnològic.

3.2.1. Capacitat de lectura del format IndoorGML

Les capacitats de transformació del format IndoorGML d'aquesta eina és centren en la visualització i lectura de l'arxiu, no queden contemplades funcionalitats d'edició ni modificació.

L'eina ha estat desenvolupada en llenguatge JavaScript, fent servir les llibreries Three.js [12], browserify [13] i jsonix [14].

L'aplicació ha estat desenvolupada amb l'objectiu d'executar-se al navegador d'Internet Google Chrome mitjançant una instancia de màquina virtual Java i executant-ne localment el codi JavaScript. No obstant això, a les proves ha estat possible constatar que l'aplicació també és totalment funcional amb Mozilla Firefox, mentre que no és compatible en absolut amb Microsoft Internet Explorer ni amb Microsoft Edge.

3.2.2. Modalitats de llicenciament

El WebGL IndoorGML Viewer està llicenciat sota el tipus de llicenciament MIT [15].

Aquestes llicencies són de tipus lliure de cost, i que atorga permisos de copia, edició, modificació, distribució i nou llicenciament sense cap tipus de restricció, únicament sotmesos a descàrrega de responsabilitat dels autors del desenvolupament.

Així, es pot classificar com una aplicació d'ús lliure i il·limitat.

3.2.3. Conjunt de proves

Amb l'objectiu de dur a terme les diferents proves previstes per testejar el visualitzador de format IndoorGML, s'ha fet una copia local, a l'entorn computacional de proves, del codi que s'ofereix lliurement al repositori GitHub de l'equip de treball STEMLab. Aquest ha estat descomprimit i executat, només amb la necessitat d'obrir un arxiu en format .html, en el navegador d'Internet (en aquesta prova en concret, s'ha fet servir Mozilla Firefox 49.0.2). En aquest mateix sentit, també és interessant apuntar que s'ha escollit novament com a document de referència en format IndoorGML per tant de fer les proves l'arxiu públic que la pròpia organització OGC ofereix en la seva plana WEB [9].

- FJK-Haus_IndoorGML_withEXR.xml [9]

En primer lloc, s'observa en obrir l'aplicació un conjunt de menús força pobre, fins i tot amb enllaços trencats i funcionalitats no desenvolupades.

En obrir l'arxiu, en primer lloc s'observa com és genera automàticament una vista del conjunt de dades en format tridimensional on podem observar la totalitat de cel·les, estats i transicions definides a l'arxiu importat, podem revisar el resultat a la Figura 22.

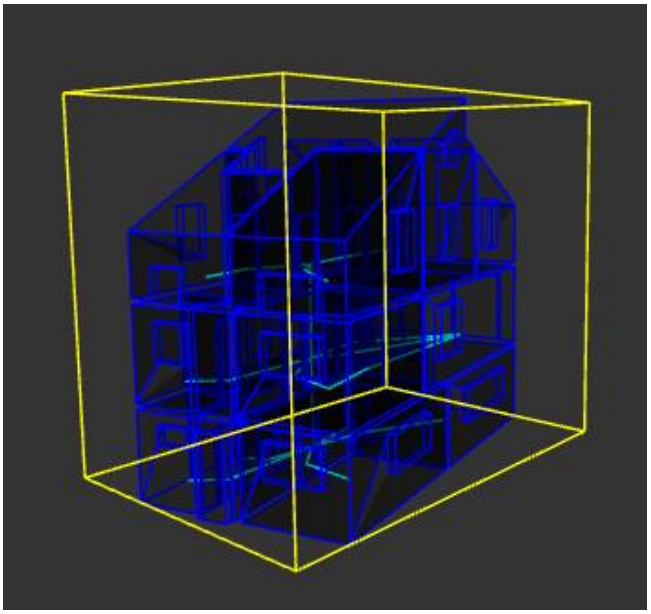


Figura 22. Vista de WebGL viewer.

De la mateixa manera, a la part dreta de la finestra s'hi defineixen tres seccions addicionals, la part SCENE i les PROPERTIES i VIEW que comparteixen espai i que s'intercanvien clicant en una o altre pestanya.

En la secció d'SCENE es troba tot el conjunt de dades de l'arxiu IndoorGML en forma d'arbre, es possible anar navegant còmodament explorant cel·les, estats, transicions, etc. Es presenta, a la Figura 23, un exemple del seu funcionament.

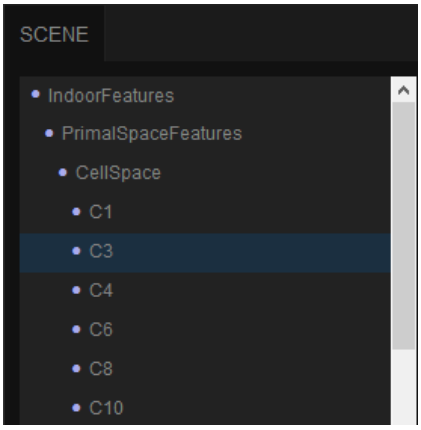


Figura 23. Secció SCENE del WebGL viewer.

D'aquesta manera, mentre es navega per les diferents característiques de l'arxiu, aquestes es van seleccionant a la vista tridimensional de l'arxiu, com s'observa a la Figura 24.

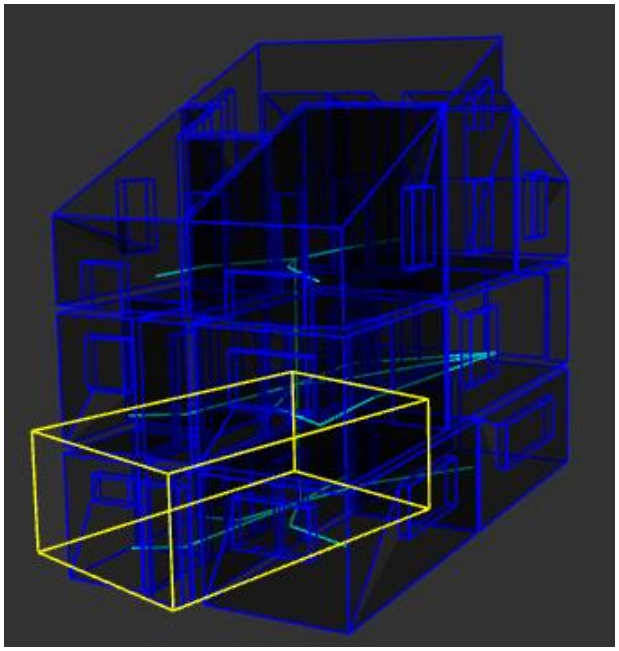


Figura 24. Detall de la selecció d'un determinat espai.

Alhora, aquestes també van mostrant-se a la secció PROPERTIES, es veu a l'exemple a la Figura 25.

Figura 25. Menú PROPERTIES del WebGL viewer.

I finalment, és possible controlar-ne la visibilitat amb la pestanya VIEW, amb detall a la Figura 26.

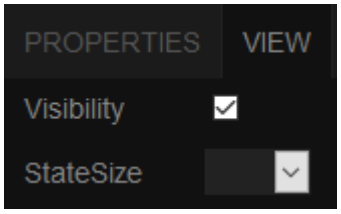


Figura 26. Pestanya VIEW del WebGL viewer.

En aquest sentit, és especialment interessant com l'atribut Duality mostra quina característica és correspon en l'altre capa d'espai, és a dir, informa quin estat es correspon amb el seu espai de cel.la. Un altre relació interessant mostrada és la que hi hauria entre els diferents nodes i transicions, de manera que es possible identificar quins estats estan interconnectats, com es pot veure a la Figura 27, o quines transicions els interconnecten, el cas de la Figura 28.

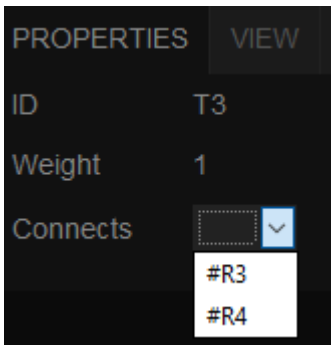


Figura 27. La transició T3 interconnecta els estats R3 i R4.

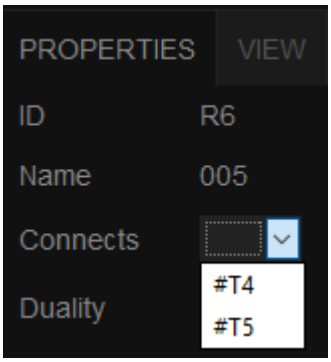


Figura 28. L'estat R06 forma part de les transicions T4 i T5.

3.2.4. Conclusions de l'eina

Malgrat que a nivell funcional i estètic el IndoorGML Viewer no arribi a l'estat de completesa de les eines de FME Desktop, si és possible entendre que, tenint en compte l'abast de la necessitat que es defineix, aquest visor satisfà les necessitats de lectura del format IndoorGML que s'havien establert.

És una notòria avantatge que aquesta eina sigui capaç d'oferir la lectura del format de manera gratuïta, fruit de la seva llicencia MIT, però te clares mancances quant a la fiabilitat del producte – una eventual actualització del navegador d'Internet podria deixar-la inutilitzada –, la versatilitat de les seves funcionalitats (tot i escapar a l'abast, no disposa d'eines d'edició) o les condicions en les que aquesta s'ha d'explotar.

3.3. ArcGIS Pro amb extensió d'Interoperabilitat.

Una bon mecanisme per introduir ArcGIS és parlar de l'Environmental Systems Research Institute, o el que és el mateix, l'ESRI [16]. Aquesta entitat va ser fundada l'any 1969 i en l'actualitat compta amb prop del 40 per cent de la quota de mercat de software de tipus SIG, que és tradueix en distribuïdors i col·laboradors comercials en tot el món.

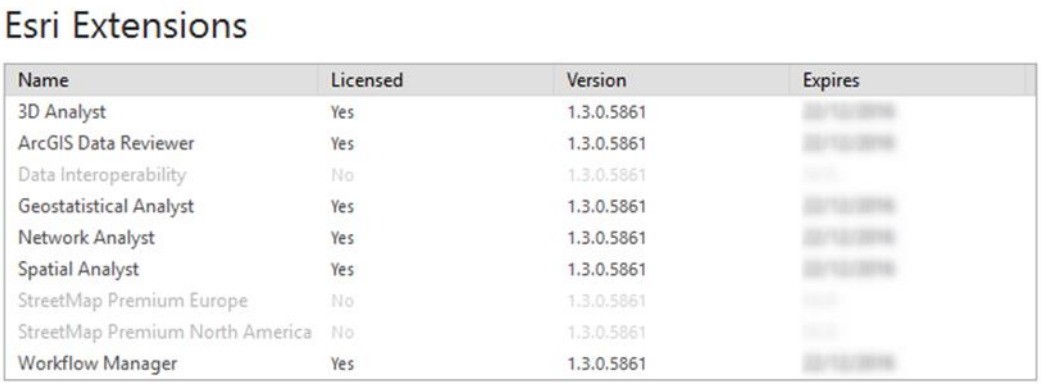
S'entén que la solució ArcGIS com un conjunt d'eines de sistemes d'informació geoespacial, on el ArcGIS Desktop havia estat històricament el sistema per la lectura i edició de mapes. En l'actualitat el seu nom comercial, acompanyat d'un canvi en la modalitat de funcionament, ha estat substituït per el d'ArcGIS Pro.

3.3.1. Capacitat de lectura del format IndoorGML

De manera nativa, aquesta eina no suporta la lectura i edició del format IndoorGML. Tot i això, es tracta d'una eina versàtil els responsables de la qual han entès que necessiten apropar-se a aquells fabricants que li poden oferir allò que requereixen i pel que no estan disposats a fer desenvolupaments propis i a mida.

En aquest marc de col·laboració han nascut diferents eines en forma d'extensió que complementen les seves funcionalitats, se n'ofereix una taula en forma d'inventari a la Figura 29, tot i que, en la majoria de casos, requerint d'un llicenciament especial.

En el context d'anàlisi d'eines amb capacitat de lectura del format IndoorGML, és la extensió anomenada Data Interoperability la que ofereix prestacions de lectura i escriptura al ArcGIS Pro.


Esri Extensions

Name	Licensed	Version	Expires
3D Analyst	Yes	1.3.0.5861	
ArcGIS Data Reviewer	Yes	1.3.0.5861	
Data Interoperability	No	1.3.0.5861	
Geostatistical Analyst	Yes	1.3.0.5861	
Network Analyst	Yes	1.3.0.5861	
Spatial Analyst	Yes	1.3.0.5861	
StreetMap Premium Europe	No	1.3.0.5861	
StreetMap Premium North America	No	1.3.0.5861	
Workflow Manager	Yes	1.3.0.5861	

Figura 29. Detall d'extensions disponibles a ArcGIS Pro.

És interessant subratllar que, en aquest cas, la funcionalitat neix fruit de la col·laboració entre les eines FME de Safe – analitzades en apartats anteriors – i ESRI. Així, aquesta extensió no deixa de ser un conjunt de funcionalitats que operen en una instància lleugera d'FME cridada des del propi software d'ArcGIS.

3.3.2. Modalitats de llicenciament

El model de llicenciament que ESRI ofereix està condicionat principalment a dos factors; la durada de la llicència i el nombre d'extensions requerides.

Així, en l'àmbit de treball requerit, la llicència mínima necessària per tal d'obtenir la funcionalitat de lectura del format IndoorGML seria la de ArcGIS Pro standalone amb l'extensió de Data Interoperability, mentre que a nivell funcional no seria rellevant que es tractés d'una llicència perpetua o una de subscripció anual.

Quant als costos, aquests poden estar condicionats de manera granular per diferents aspectes més enllà dels principals que, com s'explicava amb anterioritat, són la durada i el nombre d'extensions. En aquest sentit, escapa a l'abast d'aquest treball fer-ne esmena detallada. No obstant, sí es interessant saber que seria possible disposar del producte llicenciat, en la versió "ArcGis for Desktop Basic Single Use Unkeyed License 6+" per 1.102$, mentre que l'extensió de Data Interoperability tindria un cost de 1.836$ per la mateixa versió. [17]

3.3.3. Conjunt de proves

Ha estat possible instal·lar la darrera versió d'ArcGIS Pro amb una llicència de tipus temporal, però no ho ha estat acoblar-hi l'extensió de Data Interoperability, motiu pel qual les proves no han estat complertes.

Malgrat això, donat que aquesta extensió principalment explota una interfície de FME Desktop, podem entendre que la funcionalitat i efectivitat de l'eina serà molt propera a l'experimentada en les proves de la solució FME, més enllà de petits detalls estètics o operatius.

3.3.4. Conclusions de l'eina

La fórmula escollida per ESRI per tal d'oferir la millor eina d'edició geoespacial del mercat, o si més no aquella que té una posició més dominant, és la d'oferir un conjunt de funcionalitats en les que ha posat el focus del seu desenvolupament de manera nativa, i cobrir les funcionalitats que escapen al seu abast a col·laboradors que ofereixen garanties – evidentment, exigint un desemborsament afegit –. Potser un dels millors exemples és el complement de Data Interoperability fruit de la relació amb FME.

No ha estat possible dur a terme la bateria de proves que sí s'ha completat amb altres eines en la recerca, però en el context exclusiu de la lectura del format IndoorGML, no és arriscat confiar en un nivell de excel·lència proper a l'observat amb la eina de SAFE.

3.4. QGIS amb plugin Complex GML Info

Mentre que en el espai de mercat de software de pagament per l'edició i lectura de sistemes d'informació geoespacial, ArcGIS Pro es troba en una situació de dominància clara, l'eina que ocupa aquest espai en les eines d'edició de codi lliure és el QGIS [18].

L'eina en la que és basa el software actual era el Quantum GIS, després de molts anys de treball, i de l'adopció per part de l'Open Source Geospatial Foundation com la seva eina de referencia, va fer que l'any 2009 fos alliberada la primera versió.

A nivell tècnic, aquesta aplicació ofereix grans possibilitats d'integració amb plugins desenvolupats en llenguatges C++ i Python. En l'àmbit concret d'aquest projecte, un dels plugins en concret, anomenat Complex GML Info [19], sembla que ha de ser capaç de llegir aquest format.

3.4.1. Capacitat de lectura del format IndoorGML

L'eina de QGIS, de manera nativa, és capaç de llegir i analitzar arxius en format GML. Tot i això, el GML és un estàndard obert que pot arribar a nivells de complexitat molt elevat. Fruit d'aquest fet neix el plugin Complex GML Info que, tot i no esmentar de manera explícita la lectura del format IndoorGML, si que declara esser capaç d'oferir funcionalitats de lectura en versions GML 2.0, 3.1 i 3.2.

La secció de proves mostrarà quines són les conclusions reals, a efectes operatius, d'aquesta capacitat de lectura.

3.4.2. Modalitats de llicenciament

QGIS és una eina de codi lliure i, en tant que això, no és necessari fer-ne un llicenciament. De la mateixa manera, tots els desenvolupaments a mida que la comunitat en fa també són de lliure disposició.

3.4.3. Conjunt de proves

En les proves que s'han efectuat en el laboratori, no s'ha aconseguit que QGIS aconsegueixi obrir cap fitxer en format IndoorGML. No obstant això, sí que tenia sentit posar a prova les capacitats del plugin Complex GML Info, donat que tenint en compte les funcionalitats que cobreix, era possible que sigues capaç de generar un panell de lectura del format IndoorGML.

Però, els resultats de totes aquestes proves, amb diferents arxius en format IndoorGML han estat negatives; es pot assegurar que QGIS, amb el plugin Complex GML Info, no disposa de capacitats de lectura del format IndoorGML, la Figura 30 mostra un dels missatges d'error mostrats per l'aplicació.

Figura 30. Missatge d'error de QGIS al intentar operar el plugin.

3.4.4. Conclusions de l'eina

En primer lloc, les proves executades amb intenció de llegir el format IndoorGML no han donat els resultats esperats amb la combinació de QGIS i el plugin Complex GML Info.

Tenint en compte la capacitat que ofereix QGIS d'acollir plugins dissenyats en formats Python i C++, és especialment interessant que malgrat les limitacions d'aquest plugin en concret, sí es podria desenvolupar un plugin que oferís la lectura del format IndoorGML en concret. Això permetria incorporar més versatilitat a la potent eina de QGIS mitjançant l'ús d'aquest plugin.

3.5. Resolució de l'anàlisi

Han quedat exposades i analitzades les principals alternatives, tant de ús lliure com de pagament, per la lectura del nou model estàndard de dades per la navegació d'espais interiors IndoorGML.

A continuació es mostra una taula on s'exposen sintetitzats els conjunt d'aspectes que, en diferents àmbits, s'han considerat interessants en l'avaluació de les diferents eines a l'estudi:

	Llicencia	Lectura	Edició	Usabilitat	Completesa
FME Desktop	Pagament	Sí	Sí	Alta	Alta
WebGL Viewer	Lliure	Sí	No	Mitja	Baixa
ArcGIS Pro Interop.	Pagament	Sí	Sí	Alta	Alta
Complex GML Info	Lliure	No	No	N/D	N/D

Tot i no ser rellevant per aquest treball, el primer aspecte que queda palès és que, com a conseqüència de la recent adopció d'aquest format per l'OGC, encara no s'ha despertat el interès necessari per part de desenvolupadors i fabricants per tal de fer compatibles les seves solucions amb la lectura d'aquest nou model de dades.

En aquest sentit, s'ha comprovat que les solucions software de mercat que l'incorporen al seu portfolio d'arxius compatibles utilitzen sempre el desenvolupament propi de FME Desktop, ha quedat evidenciat en aquesta mateixa aplicació, o fins i tot en ArcGIS Pro i la seva extensió de Interoperabilitat.

Quant a les plataformes de codi lliure, el WebGL IndoorGML viewer sí resol amb força solvència les necessitats de lectura, és una solució de nínxol, en fase beta i que no disposarà del suport necessari per convertir-se en una eina confiable. En canvi, dins l'espai de les solucions de codi lliure, observem una gran oportunitat per satisfer la necessitat de lectura de l'estàndard IndoorGML amb la solució QGIS. S'ha comprovat que no ofereix una compatibilitat nativa, però tenint en compte que el seu disseny convida a la integració de diferents plugins, que poden ser desenvolupats en codi Python o C++, s'entén que el disseny d'un nou plugin capaç de llegir els conjunts de dades integrable a QGIS podria satisfer de manera molt satisfactòria aquesta demanda.

4. Construcció d'un plugin per la lectura del format IndoorGML

Tenint en compte l'anàlisi realitzat en apartats anteriors, s'ha arribat a la resolució que és adient construir un plugin per l'aplicació QGIS que la capaciti per la lectura de l'estàndard de informació geoespacial IndoorGML.

Els principal objectius funcionals a satisfer per part del plugin de QGIS girarien en torn a dos aspectes fonamentals:

- La necessitat de visualitzar l'estructura de l'arxiu en quant a les seves característiques, així com oferir detalls de cadascuna d'elles i informació sobre com es relacionen entre elles i amb altres elements que poden residir fora del propi conjunt de dades.
- L'objectiu de mostrar gràficament la informació geospacial continguda dins el fitxer carregat.

A continuació es farà revisió dels detalls que han envoltat la seva creació, identificant els recursos utilitzats, detallant les principals activitats exercides i informant de les decisions preses.

4.1. Recursos emprats

Per tal de dur a terme la construcció del nostre plugin IndoorGML Reader s'han emprat tot un seguit de recursos. Aquests seran agrupats en programari, llibreries Python, y plugins de QGIS.

4.1.1. Programari

En aquesta secció es farà un breu inventari de les eines software emprades per tal de dur a terme la construcció del plugin IndoorGML Reader.

- **Qt**

Qt és una eina de programar lliure, desenvolupada per Qt Company [20], la primera versió de la qual va arribar al mercat fa vint-i-dos anys.

Una de les seves principals característiques és que, donat la seva amplíssima compatibilitat, permet fer dissenys de interfícies d'usuari que permeten interoperabilitat amb una amplia quantitat de llenguatges.

En el cas concret del projecte, el lligam que uneix Qt amb QGIS és d'absoluta dependència, ja que aquest últim està desenvolupat, entre d'altres eines, amb Qt Software. En aquest sentit, és remarcable que també sigui l'eina per excel·lència per fer desenvolupament de interfícies de plugins per a l'eina, ja siguin per Python com per C++.

- **Notepad++**

La pròpia instància d'instal·lació de QGIS ja incorpora una consola de Python que, en la majoria de casos, facilita el desenvolupament de codi i ajuda a fer depuració del mateix. No obstant això, per tal de fer més còmode el treball, s'ha fet servir un software de codi lliure anomenat Notepad++ [21], que tot i no incorporar totes les funcionalitats que en altres contextos podrien ser-li exigides a un IDE (Integrated Development Environment) tradicional, en l'escenari de treball plantejat incorpora tot allò necessari.

4.1.2. Llibreries Python

El llenguatge de programació Python ha estat l'escollit per tal de satisfer les necessitats del plugin. Així mateix, és necessari que aquest carregui diferents llibreries que és precisaran per tal d'obtenir els mecanismes de interoperabilitat necessaris en l'entorn de treball.

- **QGIS API**

Per tal de permetre que el codi Python que constitueix el plugin pugui interactuar amb la instància de QGIS, és estrictament necessari fer servir aquesta llibreria [22].

Més endavant es detallarà com aquesta permet la maniobrabilitat amb diferents elements de l'aplicació.

- **PyQt4 (QtCore, QtGui)**

El IndoorGML Reader disposa d'una interfície d'usuari que facilita l'entrega de totes les necessitats funcionals del plugin. No obstant això, per tal que el codi Python tingui capacitats per tal de interoperar amb aquesta interfície és necessari carregar aquesta llibreria.

Qt és el software emprat per tal de definir aquesta interfície gràfica, i la PyQt4 [23] és la llibreria que oferirà funcionalitat en aquest àmbit.

- **xml.dom**

Tenint en compte que els arxius en format IndoorGML es basen estructuralment en un esquema XML, el mecanisme emprat en el plugin IndoorGML Reader per tal de poder recórrer i extraure'n informació es basa en la utilització de l'API xml.dom (The Document Object Model API) [24].

4.1.3. Plugins de QGIS

L'entorn de treball per la creació del plugin requereix, alhora, fer servir d'altres complements que altres contributaris posen a disposició de la comunitat, i que incorporen eines útils per moltes de les operacions previstes en les tasques de desenvolupament d'aquest.

- **Plugin Builder**

Generar tots els arxius necessaris per tal de implementar el plugin, i fer-ho de manera controlada, garantint-ne la funcionalitat, pot acabar resultant una tasca més complexa del que es podria haver avaluar inicialment. Així, el Plugin Builder [25] entrega una bona alternativa per fer-ho garantint-ne unes bones condicions de definició i capacitat operativa.

Fent servir aquest plugin només serà precís definir i entregar un conjunt de paràmetres per tal que el complement generi tot allò necessari. La Figura 31 mostra l'exemple de la primera pantalla del complement.

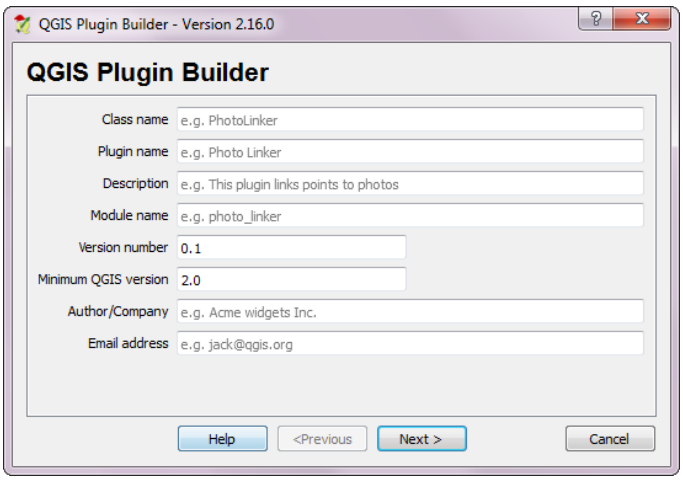


Figura 31. Exemple del QGIS Plugin Builder.

- **Plugin Reloader**

En el transcurs de les diferents tasques relacionades amb la definició del plugin, serà necessari amb molta freqüència recarregar aquest per tal

d'incorporar les darreres modificacions implementades, ens servirem del complement Plugin Reloader [26] per tal d'evitar haver de reiniciar el QGIS contínuament amb aquest propòsit.

En la Figura 32 observem com és duria a terme una recàrrega del plugin IndoorGML Reader.

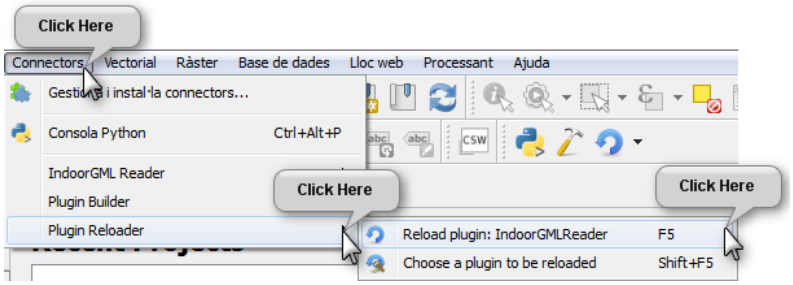


Figura 32. Recàrrega de plugin mitjançant Plugin Reloader.

4.2. Disseny i ús de la interfície gràfica del plugin

Es defineix la interfície gràfica del plugin com el conjunt d'elements que permeten la interacció entre l'usuari, l'aplicació QGIS i el propi plugin IndoorGML Reader.

4.2.1. Mecanismes per l'execució del plugin

En l'àmbit de la interacció del plugin dissenyat i com aquest interacciona amb l'usuari de l'aplicació QGIS, s'han establert en primer lloc dos mecanismes diferents per executar-lo. Ambdós han estat generats automàticament en el procés de creació inicial, amb el complement QGIS Plugin Builder [25] que hem presentat a la secció 4.1.2 Plugins de QGIS.

La primera opció per executar el plugin IndoorGML Reader és fer clic a la icona que s'estableix a la barra d'eines de la pròpia aplicació, veiem el detall a la Figura 33.

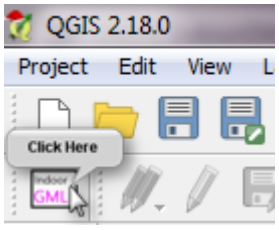


Figura 33. Detall de la icona per executar el plugin.

A més, també es possible cercar la opció adient al menú desplegable de l'aplicació per tal d'executar el plugin, és; Plugins → IndoorGML Reader → IndoorGML Reader. S'observa la representació gràfica a la Figura 34.

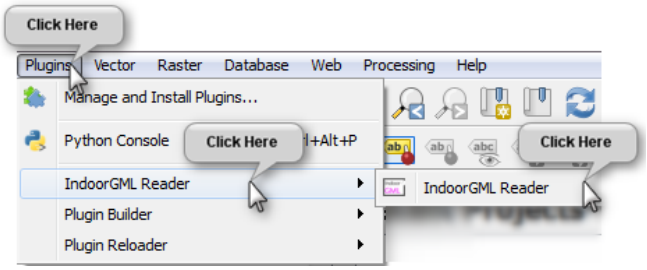


Figura 34. Representació de la opció al menú per obrir el plugin.

4.2.2. Interfície gràfica del plugin

Els plugins de QGIS tenen com a objectiu principal incorporar una nova funcionalitat a l'aplicació que aquesta no ofereix de manera nativa, com és el cas de l'exploració i representació gràfica de conjunt de dades en format IndoorGML. Si en l'apartat anterior s'ha fet revisió dels elements que permeten executar el complement, en aquesta secció s'exposarà la interfície pròpia del plugin, així com el procediment per la seva creació i l'ús que es fa dels elements propis de QGIS per fer la representació gràfica.

- **Creació de la finestra de càrrega i exploració**

L'objectiu principal del complement és oferir possibilitats d'exploració dels diferents elements que composen un arxiu IndoorGML, així com la seva representació gràfica.

El disseny de la pròpia finestra de càrrega i exploració del conjunt de dades IndoorGML s'ha fet amb el programari Qt [20], estretament relacionat amb la comunitat QGIS, i erigit com el principal mecanisme per fer dissenys gràfics de plugins per complementar les funcionalitats d'aquest.

Aquesta finestra de càrrega queda definida al fitxer del plugin anomenat indoorgml_reader_dialog_base.ui, a la Figura 32 es mostra el seu disseny en el propi editor Qt:

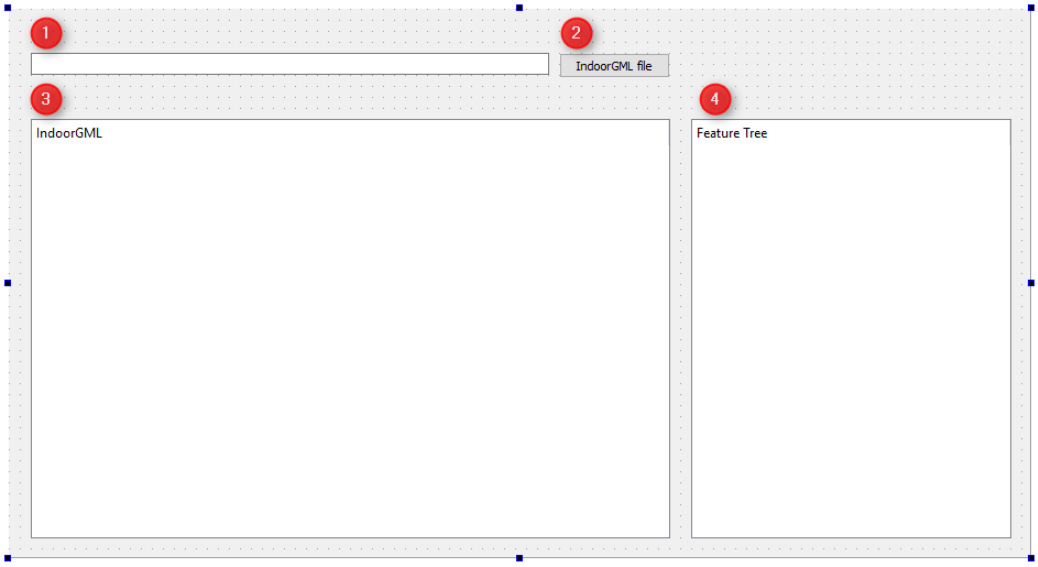


Figura 35. Disseny de la finestra de càrrega a Qt.

La interfície de la finestra de navegació queda definida principalment per quatre elements que cerquen satisfer diferents necessitats del plugin, i que son diferents tipus d'objecte a Qt. Així, i referenciats numèricament a la Figura 35:

1. **gmlLine**: Aquest és un element del tipus QLineEdit [27], mostrarà el contingut de la ruta de l'arxiu IndoorGML carregat, i servirà de referència en el treball d'aquest en l'entorn del plugin. A la interfície ofereix interacció amb l'usuari mitjançant l'edició dels seus continguts, i també és interoperable mitjançant el codi Python associat al complement.

2. **loadFile**: El loadFile és el botó que implementarà el mecanisme per què l'usuari pugui navegar en el seu sistema d'arxius i escollir el conjunt de dades IndoorGML. És del tipus QPushButton [28], permet enviar senyals en ser polsat des de la interfície gràfica al codi de Python.

   La càrrega del panell d'exploració i la representació gràfica del contingut del conjunt de dades quedarà enllaçat a la selecció del conjunt de dades mitjançant aquest element.

3. **tree**: Queda definit conceptualment dins el disseny com el panell principal d'exploració dels elements que conformen el conjunt de dades carregat des de l'arxiu IndoorGML. El grau d'interacció per l'usuari és força alt, ja que és permet anar explorant els seus continguts, veure nodes dependents de manera jeràrquica, i tornar a contraure si és necessari.

És defineix com un objecte de tipus QTreeWidget [29], que ofereix un ampli conjunt de funcionalitats i mecanismes de interacció amb el codi Python que defineix el complement, i que permetrà disparadors en el moment de diferents tipus de interacció amb la superfície de l'objecte.

4. **secTree**: És un objecte que novament oferirà capacitats de navegació a l'usuari, però que en aquest cas té com a objectiu mostrar detalls d'elements enllaçats provinents del panell principal d'exploració tree.

   Com en el cas anterior, és un objecte de tipus QTreeWidget, i ofereix el mateix tipus de funcionalitats.

Tot i que en l'àmbit de treball de l'elaboració d'aquest complement no s'ha identificat la necessitat de fer adaptacions massa complexes de cap dels elements exposats, sí és interessant identificar clarament el tipus d'objecte generat i el nom que l'identifica, per tal de poder fer-ne invocacions en el codi Python. Aquest detall s'observa en la Figura 36, que mostra el contingut de la vista Object Inspector al programari Qt.

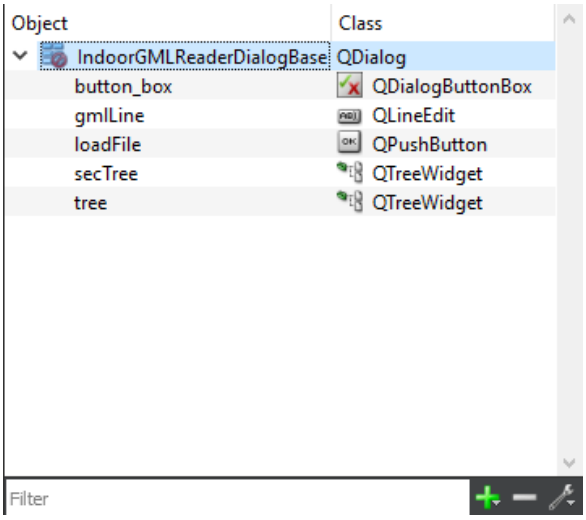


Figura 36. Qt Object Inspector.

L'aplicació Qt, permet, mitjançant la interacció directa amb el disseny que es presentava a la Figura 35, modificar les dimensions de cadascun dels elements. També és possible fer edició dels textos que acompanyen cada element, o fins i tot modificar el comportament dels mateixos. A la Figura 37 s'exposa l'exemple de la vista al propi programari Qt del Property Editor per un objecte de la classe QPushButton [28], que ofereix un mecanisme general per fer modificacions de diferents paràmetres de cadascun dels objectes.

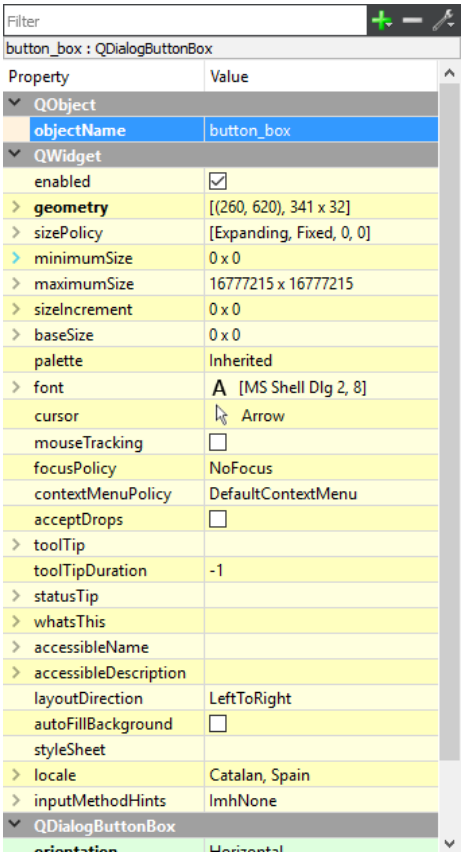


Figura 37. Vista Property Editor de Qt.

- **Elements de la finestra de càrrega i exploració**

La finestra de càrrega i exploració és l'únic element gràfic del plugin que no queda completament encastat en la pròpia eina QGIS, aquest és el motiu pel qual és especialment important que la seva usabilitat sigui bona i l'estètica concordant amb la resta de l'aplicació.

Amb anterioritat s'ha relatat la seva construcció amb el programari Qt, i el detall tècnic de les classes a les que pertanyen els elements que la composen, així com les seves possibilitats funcionals. Ara la proposta serà fer-ne una descripció plenament funcional i des de la perspectiva d'usuari.

S'estableix mitjançant un disseny clar i senzill, que en l'aspecte funcional permet fer la pujada de l'arxiu que conté el conjunt de dades IndoorGML que es vol representar, així com l'exploració dels elements que el composen.

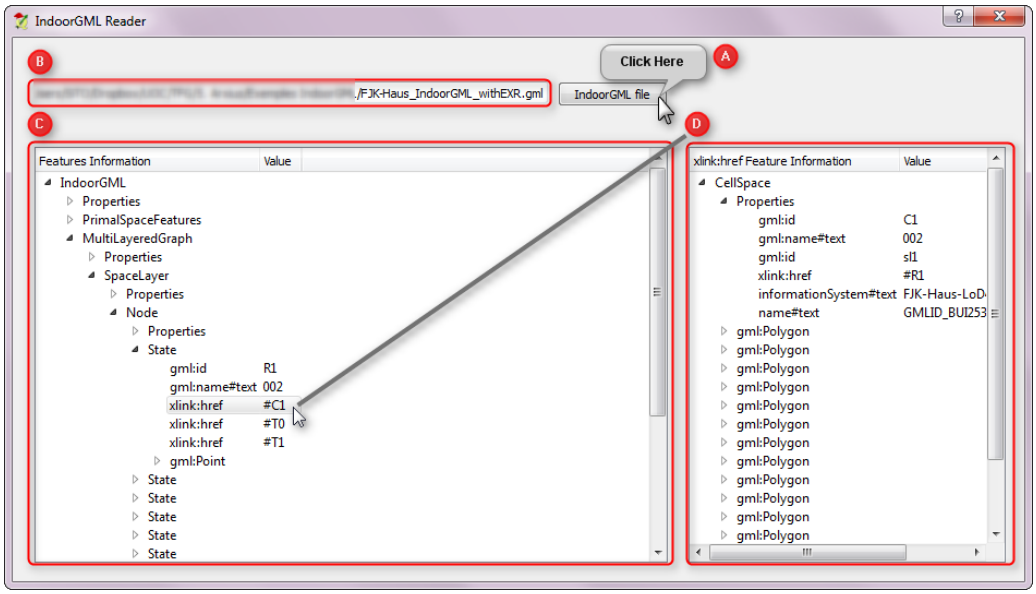


Figura 38. Exemple de la finestra de càrrega i exploració del plugin.

A continuació, donant seguiment als identificadors generats a l'exemple de la finestra de càrrega i navegació que mostra la Figura 38, es farà un inventari dels principals àrees i elements que la composen:

A. El botó IndoorGML file permet fer la selecció de l'arxiu que conté el conjunt de dades IndoorGML.

B. En aquest espai es mostra la ruta de l'arxiu que s'ha carregat a través del botó de càrrega.

C. Aquest element de la Figura 38 s'estableix com el panell principal d'exploració dels components de les diferents classes definides a l'arxiu IndoorGML carregat. Aquest panell quedarà automàticament completat al carregar el conjunt de dades, i permetrà fer-ne una exploració plena.

   Quan l'usuari polsi sobre un objecte del tipus xlink:href, la interfície, de manera automàtica, generarà una instància al panell d'exploració D de l'objecte al que enllaça.

D. En aquest cas, aquest panell d'exploració d'elements del conjunt d'arxius IndoorGML només mostrarà informació dels elements enllaçats a característiques d'objectes seleccionats al panell d'exploració C.

- **Visualització de l'arxiu IndoorGML a QGIS**

Tant la representació eminentment geomètrica dels espais que composen el conjunt de dades IndoorGML de l'espai primari carregat amb el plugin IndoorGML Reader, com la informació de les altres capes que poden conformar l'espai dual, seran representades a l'espai de treball del propi programari QGIS.

Immediatament després de fer la càrrega de l'arxiu IndoorGML amb el plugin IndoorGML Reader, aquest genera de manera automàtica una nova capa vectorial, de manera que quedin representats l'espai primitiu (si aquest està definit al conjunt de dades) i l'espai dual.

A la Figura 39 es mostra el detall del Layers Panel de QGIS després de la càrrega d'un conjunt de dades amb el plugin IndoorGML Reader, en el que s'identifica una capa per CellSpaces, que mostrarà informació de l'espai primitiu, i dues capes més, una per les Transitions i l'altre per els States que conformen el graf de l'espai multi capa dual.

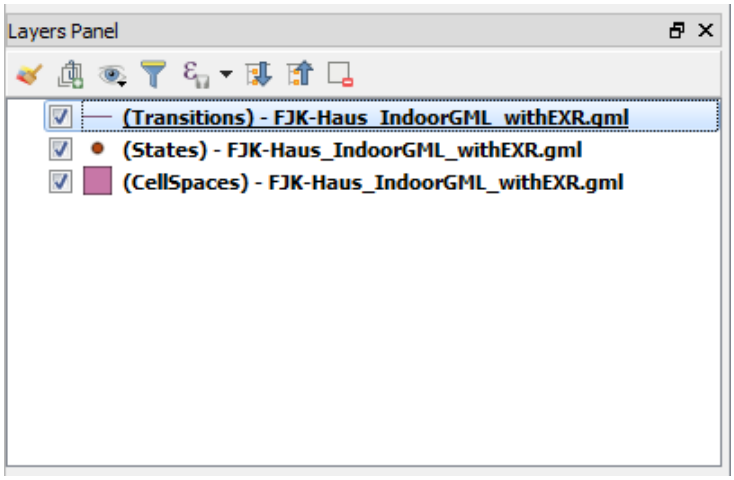


Figura 39. Layers Panel després de la càrrega d'un conjunt de dades.

La Figura 40 mostra el detall de la representació gràfica de l'exemple d'un conjunt de dades carregades mitjançant el plugin IndoorGML Reader.

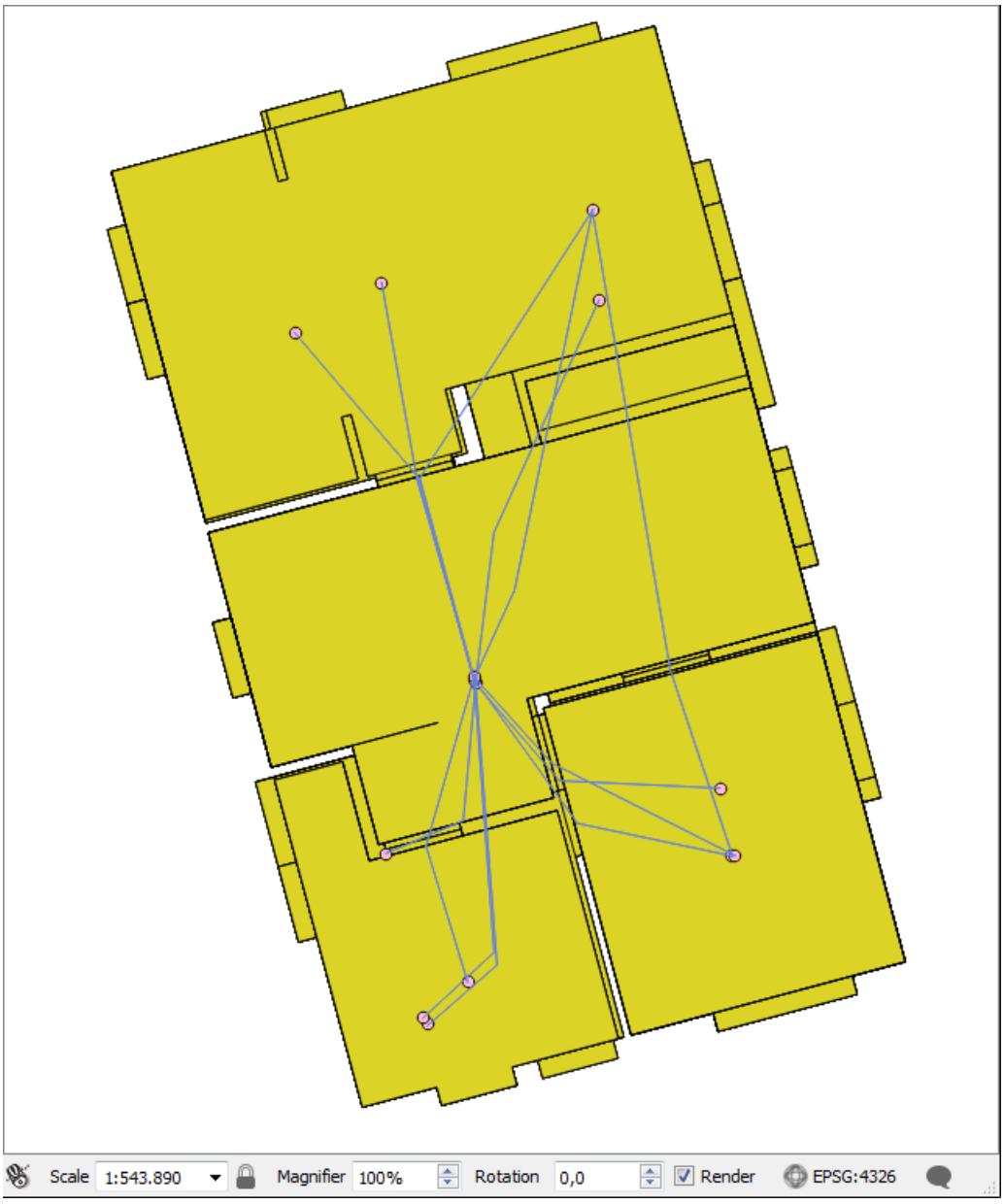


Figura 40. Representació gràfica de conjunt de dades IndoorGML.

Les capes vectorials generades automàticament per el complement IndoorGML Reader a l'espai de treball de QGIS son operables en els mateixos termes que si haguessin estat carregades mitjançant els mètodes tradicionals – i.e. Add Vector Layer dels propis menús de QGIS –.

4.3. Elaboració del codi Python

La comunitat de desenvolupament de QGIS estableix de manera majoritària Python com el codi de programació per excel·lència, tot i no ser la única alternativa en aquest sentit [30], donat que ofereix també

compatibilitat amb C++. Els motius argumentats acostumen a ser l'agilitat del llenguatge i la facilitat en el seu aprenentatge.

Aquest capítol oferirà informació del desenvolupament del copi Python necessari per tal que el plugin IndoorGML Reader satisfaci els requeriments presentats, de manera segmentada en funció de les diferents funcionalitats implementades.

Tot el codi Python que capacita les funcionalitats del plugin es troba a l'arxiu indoorgml_reader.py.

4.3.1. Funcionalitats genèriques

A més de les funcionalitats específiques a satisfer, per tal que el plugin sigui totalment funcional, serà necessari generar mecanismes genèrics per les funcionalitats estàndard, com per exemple la càrrega del mateix complement en l'entorn de QGIS, o la seva desactivació.

- **Importacions.**

Aquest codi de Python queda subscrit principalment a l'àmbit de l'ús de llibreries que permeten la interacció amb diferents objectes i elements de l'aplicació QGIS, així com de la interfície de càrrega i exploració del propi plugin IndoorGML Reader. A la Figura 41 es mostren totes les llibreries importades. Les principals havien quedat ja esmentades a la secció 4.1.2 d'aquests materials.

```
# Import qgis libraries in order to interoperate with QGIS elements.
import qgis
from qgis.core import *
from qgis.gui import *
from qgis.utils import iface
# Import PyQt4 libraries in order to satisfy functional needs. (QMessageBox, QFileDialog)
from PyQt4 import QtCore, QtGui
from PyQt4.QtCore import QSettings, QTranslator, qVersion, QCoreApplication
from PyQt4.QtGui import QAction, QIcon, QFileDialog, QMessageBox
# Import libraries in order to explore xml file.
from xml.dom import minidom
import sys
# Initialize Qt resources from file resources.py
import resources
# Import the code for the dialog
from indoorgml_reader_dialog import IndoorGMLReaderDialog
import os.path
```

Figura 41. Importació de llibreries.

- **Càrrega i descarrega del plugin a QGIS.**

Amb la importació de llibreries necessàries establertes, el següent objectiu del codi serà inserir els botó del plugin a la interfície de treball de QGIS, així com el registre al menú contextual. El codi necessari per capacitar aquesta funcionalitat va ser automàticament definit en el procés de creació del plugin amb el Plugin Builder, i que ha estat modificat i adaptat per cobrir la consecució dels objectius del projecte.

El codi defineix una primera funció anomenada initGui, la definició de la qual s'observa a la Figura 42, que definirà un conjunt de paràmetres que més endavant seran utilitzats per entregar a la funció add_action.

```
def initGui(self):
    """Create the menu entries and toolbar icons inside the QGIS GUI."""

    icon_path = ':/plugins/IndoorGMLReader/icon.png'
    self.add_action(
        icon_path,
        text=self.tr(u'IndoorGML Reader'),
        callback=self.run,
        parent=self.iface.mainWindow())
```

Figura 42. Funció initGui.

Aquests paràmetres aporten informació de la ubicació del fitxer d'imatge que utilitzarà la icona del plugin a la barra de QGIS, el text que apareixerà tant en el botó com en el menú, i el comportament del plugin en ser executat de manera consecutiva.

Un cop els paràmetres s'entreguen a la funció add_action, que es mostra a la Figura 43, aquesta agafa la informació dels paràmetres i els fa servir per, finalment, interactuar amb el programari QGIS afegint el botó i el registrant l'entrada al menú contextual.

```
def add_action(
    self,
    icon_path,
    text,
    callback,
    enabled_flag=True,
    add_to_menu=True,
    add_to_toolbar=True,
    status_tip=None,
    whats_this=None,
    parent=None):

    icon = QIcon(icon_path)
    action = QAction(icon, text, parent)
    action.triggered.connect(callback)
    action.setEnabled(enabled_flag)

    if status_tip is not None:
        action.setStatusTip(status_tip)

    if whats_this is not None:
        action.setWhatsThis(whats_this)

    if add_to_toolbar:
        self.toolbar.addAction(action)

    if add_to_menu:
        self.iface.addPluginToMenu(
            self.menu,
            action)

    self.actions.append(action)

    return action
```

Figura 43. Funció add_action.

En aquesta mateixa línia també es troba la funció unload, que es defineix amb l'objectiu d'entregar al programari QGIS les funcionalitats necessàries per desinstal·lar el plugin de la seva instància d'execució un cop l'usuari decideix treure'l de l'entorn de treball. A la Figura 44 és visible com aquesta funció interopera amb QGIS per treure el botó de la barra d'eines així com el registre al propi menú contextual.

```
def unload(self):
    """Removes the plugin menu item and icon from QGIS GUI."""
    for action in self.actions:
        self.iface.removePluginMenu(
            self.tr(u'&IndoorGML Reader'),
            action)
        self.iface.removeToolBarIcon(action)
    # remove the toolbar
    del self.toolbar
```

Figura 44. Funció unload.

- **Mètode constructor i execució del plugin**

El codi Python generat per fer la implementació del plugin es crea mitjançant la implementació d'una classe anomenada IndoorGMLReader, i que, com en qualsevol implementació, disposarà d'un mètode constructor. En aquest llenguatge, els mètodes constructors tenen sempre la definició __init__, s'observa la seva definició concreta a la Figura 45.

```
class IndoorGMLReader:
    """QGIS Plugin Implementation."""

    def __init__(self, iface):
        """Constructor.

        :param iface: An interface instance that will be passed to this class
            which provides the hook by which you can manipulate the QGIS
            application at run time.
        :type iface: QgsInterface
        """

        # Create the dialog (after translation) and keep reference
        self.dlg = IndoorGMLReaderDialog()
        # Save reference to the QGIS interface
        self.iface = iface
        # initialize plugin directory
        self.plugin_dir = os.path.dirname(__file__)
        # initialize locale
        locale = QSettings().value('locale/userLocale')[0:2]
        locale_path = os.path.join(
            self.plugin_dir,
            'i18n',
            'IndoorGMLReader_{}.qm'.format(locale))

        if os.path.exists(locale_path):
            self.translator = QTranslator()
            self.translator.load(locale_path)

            if qVersion() > '4.3.3':
                QCoreApplication.installTranslator(self.translator)


        # Declare instance attributes
        self.actions = []
        self.menu = self.tr(u'&IndoorGML Reader')
        # TODO: We are going to let the user set this up in a future iteration
        self.toolbar = self.iface.addToolBar(u'IndoorGMLReader')
        self.toolbar.setObjectName(u'IndoorGMLReader')
        # Initialize load button and field.
        self.dlg.gmlLine.clear()
        self.dlg.loadFile.clicked.connect(self.load_indoorGML_file)
```

Figura 45. Funció __init__.

A la mateixa Figura 45 s'observa que s'estableix la plataforma de comunicació amb la instància d'execució de QGIS, que es anomenada dlg, i amb la interfície de l'aplicació, que es denotarà per iface. També quedarà inicialitzat el nom de la finestra de càrrega del plugin, i es definirà el comportament del botó de càrrega de l'arxiu IndoorGML, anomenat loadFile, definit en l'apartat 4.2.2 d'aquests materials.

4.3.2. Càrrega de l'arxiu amb el conjunt de dades

En l'apartat anterior s'ha fet revisió del conjunt d'elements necessaris per tal que el plugin quedés instal·lat en el programari QGIS, així com la importació de llibreries necessàries, construcció de la classe i establiment de les vies i mecanismes bàsic per la interoperació amb el programari.

La càrrega del conjunt de dades s'estableix mitjançant l'accionament del botó IndoorGML file de la Figura 38. En fer accionament d'aquest botó, l'usuari seleccionarà un conjunt de dades IndoorGML, i aquest serà carregat a la instància d'execució del codi Python del plugin IndoorGML Reader. A la Figura 46 es mostra en detall què fa la funció load_indoorGML_file un cop l'usuari ha polsat el botó:

```
# Load the IndoorGML File to the plugin.
def load_indoorGML_file(self):
    indoorGMLFile = QFileDialog.getOpenFileName(self.dlg, "Load IndoorGML file ","", '*.gml')

    # Once the file is defined, we upload the information to a variable.
    if indoorGMLFile:
        self.dlg.gmlLine.setText(indoorGMLFile)
        file = self.dlg.gmlLine.displayText()
        gml = minidom.parse(file)
        # Once the file is uploaded, the exploration tree and the layers are also created.
        self.printCellSpaces(gml)
        self.printStates(gml)
        self.printTransitions(gml)
        self.create_tree(gml)
    else:
        self.showMessage()
```

Figura 46. Funció load_indoorGML_file

La funció load_indoorGML_file() genera una interfície de selecció a l'usuari, en la que aquest pot escollir un document del sistema de fitxers del seu propi entorn de treball – el codi queda definit per evitar mostrar arxius que no siguin de format .GML – i l'introdueix a la variable indoorGMLFile. La ruta absoluta d'aquest document es defineix després a la interfície de càrrega per donar-li visibilitat, com podem veure a la Figura 35.

A continuació, la funció carrega el document en una variable d'objecte minidom [24], que farà servir com a paràmetre que entregarà a les funcions printCellSpaces(), printStates(), printTransitions() i create_tree() que seran presentades més endavant.

Amb intenció d'oferir més robustesa a l'aplicació, es controla que s'hagi seleccionat un arxiu, o que aquest sigui adient, amb una clàusula if – else, en cas d'error, l'usuari rebrà un missatge d'error com el de la Figura 47.

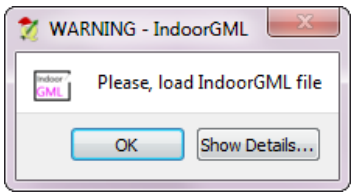


Figura 47. Missatge d'error de càrrega d'arxiu IndoorGML.

El mecanisme per generar aquest missatge d'error es genera mitjançant la funció showMessage(), el codi Python per la seva definició és mostra a la Figura 48.

```
# Offer a message in order to inform user about the need of upload an IndoorGML file.
def showMessage(self):
    msg = QMessageBox()
    msg.setIconPixmap(QtGui.QPixmap(":/plugins/IndoorGMLReader/icon.png"));
    msg.setText("Please, load IndoorGML file")
    msg.setWindowTitle("WARNING - IndoorGML")
    msg.setDetailedText("Click on 'IndoorGML file' button.\nSelect the IndoorGML file.")
    msg.exec_()
```

Figura 48. Funció showMessage()

4.3.3. Creació dels arbres d'exploració

El mètode seleccionat per fer representació lògica de la composició del conjunt de dades IndoorGML carregat al plugin és mitjançant els arbres d'exploració.

Aquest mecanisme s'implementa amb objectes de les classes QTreeWidget [29] i QTreeWidgetItem de la llibreria del programari Qt. Aquestes classes permeten fer que la representació de la relació entre les diferents classes de característiques que conformen un conjunt de dades a un arxiu IndoorGML sigui molt visual i interactiva, permetent expandir i contraure nodes de manera àgil i intuïtiva.

Al codi Python del plugin IndoorGML Reader s'hi estableixen tot un conjunt de funcions que engranen aquesta composició, per les que a continuació farem una descripció més detallada.

- **Funció create_tree()**

La funció load_IndoorGML_file() passa com a paràmetre el conjunt de dades carregat al plugin a la funció create_tree(), per tal que en faci processament generant un arbre d'exploració que es portarà a la finestra de càrrega i exploració del plugin amb la que l'usuari podrà interactuar.

```
# Create the exploration tree of the IndoorGML file.
def create_tree(self, gml):
A   tree = self.dlg.tree
    header = QtGui.QTreeWidgetItem(["Features Information","Value"])
    tree.setHeaderItem(header)
    tree.header().setResizeMode(QtGui.QHeaderView.ResizeToContents)
    tree.header().setStretchLastSection(False)
    self.dlg.tree.clear()

B   # In order to launch a trigger when we click on a item.
    self.dlg.connect (self.dlg.tree, QtCore.SIGNAL ("itemClicked(QTreeWidgetItem*, int)"), self.displayExtendedInformation)

C   # We set "IndoorGML" tag name as the root of the exploration tree.
    root = QtGui.QTreeWidgetItem(tree, ["IndoorGML"])
    IFs = gml.getElementsByTagName('IndoorFeatures')
    if IFs.length >= 1:
        var = QtGui.QTreeWidgetItem(root, ['Properties'])
        self.getAttributes(IFs[0],var)

D   # The IndoorGML document will be set under a PrimalSpaceFeatures item.
    PSF = gml.getElementsByTagName('PrimalSpaceFeatures')

E   # The seconday main tree of the exploration tree will be the MultiLayeredGraph
    MLG = gml.getElementsByTagName('MultiLayeredGraph')
```

Figura 49. Primera part de la funció create_tree().

A continuació es detallarà el contingut de cada segment de la primera part de la funció, representada a la Figura 49.

A. Es genera l'arbre de navegació, establint-se la capçalera i configurant-se de manera que adapti la amplitud de les columnes al contingut que mostra en cada moment.
B. L'arbre es parametritza per enviar un senyal, que podrà tractar-se més endavant, cada cop que es seleccioni un dels elements que el composen.
C. Es defineix l'arrel de l'arbre, i s'hi incorporen altres atributs de la instància de classe IndoorFeatures com les propietats del conjunt de dades, donat que serà l'arrel d'aquest.
D. Es genera una nova variable que contindrà tot el conjunt de dades que correspon a la instància de classe PrimalSpaceFeatures, i que representarà l'espai primari.
E. Definició de l'objecte que contindrà el conjunt de dades corresponent al MultiLayeredGraph, que oferirà l'espai dual.

La segona part de la funció es centra en el desenvolupament de la part de l'arbre corresponent a l'espai primari, els detalls queden exposats a la Figura 50.

```
if PSF.length >= 1:
F   for psf in PSF:
        # We go through the whole of "PrimalSpaceFeatures" items, and display their attributes.
        A = QtGui.QTreeWidgetItem(root, ['PrimalSpaceFeatures'])
        varA = QtGui.QTreeWidgetItem(A, ['Properties'])
        self.getAttributes(psf,varA)
        # Now we search for "CellSpace" elements inside of this PrimalSpace.
        CS = psf.getElementsByTagName('CellSpace')
G       for cs in CS:
            varA = QtGui.QTreeWidgetItem(A, ['CellSpace'])
            varAA = QtGui.QTreeWidgetItem(varA, ['Properties'])
            self.getAttributes(cs,varAA)
            varAB = QtGui.QTreeWidgetItem(varA, ['Geometry'])
            # Exploring child nodes and their attributes.
            childs = cs.childNodes
            for child in childs:
                self.getAttributes(child,varAB)
                self.getNodes(child.childNodes,400,varAB)
            # We define a new identation to polygon elements.
            polygons = cs.getElementsByTagName('gml:Polygon')
            for polygon in polygons:
                varABA = QtGui.QTreeWidgetItem(varAB, [polygon.nodeName])
                self.getAttributes(polygon,varABA)
                self.getNodes(polygon.childNodes,400,varABA)
```

Figura 50. Segona part de la funció create_tree()

F. El PrimalSpaceFeatures queda definit com un dels nodes generats directament a partir de l'arrel de l'arbre. S'explora i registren els seus atributs, i es genera un nou objecte per contenir els CellSpace que el composen.
G. En aquesta part del codi es recorreran tots els espais de cel·les, registrant les seves propietats a l'arbre, així com els seus nodes dependents. En aquest àmbit obtenen un tractament especial els objectes de tipus gml:Polygon [31], ja que composen la definició geomètrica dels CellSpace.

La darrera part de la funció te com a propòsit inserir a l'arbre de navegació tots els elements que composaran el MultiLayeredGraph que conté el conjunt de dades carregat al plugin. La Figura 51 mostra la composició d'aquesta secció de la funció:

```
if MLG.length >= 1:
    for mlg in MLG:
H       B = QtGui.QTreeWidgetItem(root, ['MultiLayeredGraph'])
        varB = QtGui.QTreeWidgetItem(B, ['Properties'])
        self.getAttributes(mlg,varB)
        # Identifying spaceLayers within the MultiLayeredGraph
        SL = mlg.getElementsByTagName('SpaceLayer')
I       for sl in SL:
            varB = QtGui.QTreeWidgetItem(B, ['SpaceLayer'])
            varBA = QtGui.QTreeWidgetItem(varB, ['Properties'])
            self.getAttributes(sl,varBA)
            # Getting nodes and edges of each spaceLayer
            N = sl.getElementsByTagName('nodes')
            E = sl.getElementsByTagName('edges')
J           for n in N:
                varBA = QtGui.QTreeWidgetItem(varB, ['Node'])
                varBAA = QtGui.QTreeWidgetItem(varBA,['Properties'])
                self.getAttributes(n,varBAA)
                # Getting the States of each Node
                S = n.getElementsByTagName('State')
K               for s in S:
                    varBAA = QtGui.QTreeWidgetItem(varBA, ['State'])
                    self.getAttributes(s,varBAA)
                    self.getNodes(s.childNodes,400,varBAA)
                    points = s.getElementsByTagName('gml:Point')
                    for point in points:
                        varBAAA = QtGui.QTreeWidgetItem(varBAA, [point.nodeName])
                        self.getAttributes(point,varBAAA)
                        self.getNodes(point.childNodes,400,varBAAA)
L           for e in E:
                varBB = QtGui.QTreeWidgetItem(varB, ['Edge'])
                varBBA = QtGui.QTreeWidgetItem(varBB,['Properties'])
                self.getAttributes(n,varBBA)
                # Getting the Transitions of each edge
                T = e.getElementsByTagName('Transition')
M               for t in T:
                    varBBB = QtGui.QTreeWidgetItem(varBB, ['Transition'])
                    self.getAttributes(t,varBBB)
                    self.getNodes(t.childNodes,400,varBBB)
                    lineStrings = t.getElementsByTagName('gml:LineString')
                    for lineString in lineStrings:
                        varBBBA = QtGui.QTreeWidgetItem(varBBB, [lineString.nodeName])
                        self.getAttributes(lineString,varBBBA)
                        self.getNodes(lineString.childNodes,400,varBBBA)
```

Figura 51. Segona part de la funció create_tree()

H. S'extrauen en aquest segment les propietats relatives al objecte MultiLayeredGraph del conjunt de dades, i es carreguen a l'arbre d'exploració. A més, es genera un array que conté tots els SpaceLayer que el conformen.
I. Per a cada SpaceLayer es registren les seves propietats, i es defineix també dos nous jocs d'array; un per els objectes de tipus node i un altre per als objectes de tipus edge.
J. Aquesta secció de codi agafarà les propietats de cada objecte de tipus node i les registra al arbre d'exploració. També extreu tots els State que pertanyen al node sobre el que es treballa.
K. Per a cadascun dels estats identificats, s'extreu la informació de propietats més rellevant. És fa un tractament especial a la posició geomètrica de l'estat, gml:Point [31], ja que entrega informació rellevant a l'usuari del posicionament d'aquest al conjunt de dades.

L. En aquest fragment de codi es treballa sobre els objectes de tipus edge, obtenint el conjunt de les seves propietats i enregistrant els objectes de classe Transition que en depenen.
M. Dintre de cada objecte edge quedaran definits tots els elements de classe Transition que en depenen, amb les seves propietats. En aquest cas, la informació geomètrica que defineix l'objecte és el gml:Linestring [31], motiu pel qual té una entitat pròpia i s'hi defineixen les seves propietats dins cada element Transition.

Com a exemple, la Figura 52 mostra el contingut d'un arbre d'exploració generat amb el plugin IndoorGMLReader en que es mostra l'objectiu d'exploració de la funció create_tree().

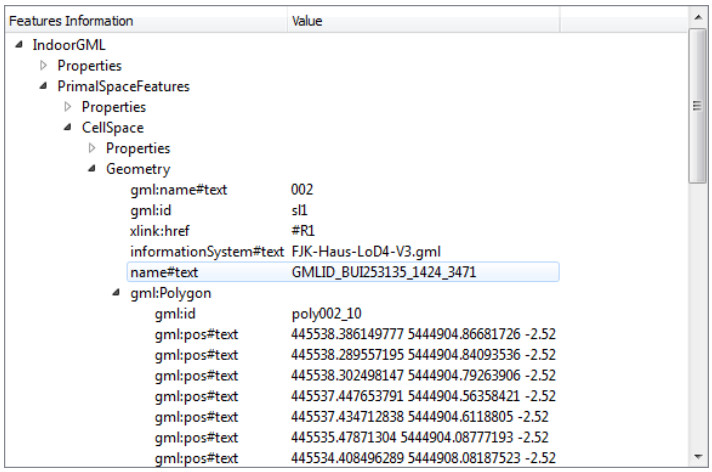


Figura 52. Arbre d'exploració de conjunt de dades IndoorGML.

- **Funció getAttributes()**

La funció getAttributes() és generada dintre el codi Python, a l'arxiu indoorgml_reader.py, com a accessori funcional de la funció create_tree().

El focus de treball d'aquesta funció és completar amb tot el conjunt d'atributs de cada node XML l'arbre d'exploració de manera que els inclou automàticament en el node explorat.

S'observa a la Figura 53 com la funció getAttributes() explora els atributs del node XML que se li entrega com a paràmetre, així com els atributs dels seus nodes dependents de tipus text, i els registra al node apropiat a l'arbre d'exploració.

```
# Get the attribute information of a given node.
def getAttributes(self, item, parent):
    if item.localName != None:
        keys = item.attributes.keys()
        if item.attributes.length >= 1:
            for key in keys:
                QtGui.QTreeWidgetItem(parent, [str(key),item.attributes[key].value])
        datas = item.childNodes
        for data in datas:
            if (data.nodeName == '#text'):
                if "\n" not in data.data:
                    QtGui.QTreeWidgetItem(parent, [data.parentNode.nodeName + data.nodeName,data.data])
```

Figura 53. Funció getAttributes().

- **Funció getNodes()**

La funció getNodes() extreu els nodes fills d'un determinat objecte XML, i acull les seves propietats per registrar-les a l'objecte del que depenen. Aquest comportament té excepcions per els casos en que s'analitzi un objecte de tipus geomètric, ja que aquests tenen entitat pròpia i no seran tractats de la mateixa manera.

La funció rep com a paràmetre l'objecte XML a processar, el nombre de vegades que serà processada, per tal de ser executada de manera recursiva, i l'objecte al que es relacionaran les propietats extretes, els detalls són a la Figura 54.

```
# Exploring the whole of child nodes of a given node, getting attributes and data.
def getNodes (self, item, level, parent):
    if level != 0:
        for node in item:
            if (node.nodeName == 'gml:Polygon') or (node.nodeName == 'gml:Point') or (node.nodeName == 'gml:LineString'):
                break
            self.getAttributes(node,parent)
            self.getNodes(node.childNodes, level - 1, parent)
```

Figura 54. Funció getNodes().

- **Funció displayXlinnkInformation()**

L'arbre d'exploració es defineix per tal d'enviar un senyal cada cop que l'usuari selecciona un dels elements que el composen. Aquesta senyal genera una crida a la funció displayXlinnkInformation(), passant com a paràmetre de manera estàndard el propi QtreeWidgetItem seleccionat.

Aquesta funció, alhora, comprova si l'objecte seleccionat és de tipus xlink:href, i si ho és comprova al conjunt de dades si és de tipus CellSpace, State o Transition, donat que son aquells que seran representats en un arbre d'exploració secundari, mitjançant la crida a la funció createSecTree(). Els detalls apareixen a la Figura 55.

```
# Get feature information, through tree item click, on the QText element.
def displayXlinkInformation(self, item, column):

    if (item.text(0) == 'xlink:href'):
        file = self.dlg.gmlLine.displayText()
        gml = minidom.parse(file)
        extendedNodes = gml.getElementsByTagName('Transition')
        for node in extendedNodes:
            if ((node.attributes['gml:id'].value == item.text(1)) or ('#' + node.attributes['gml:id'].value == item.text(1))):
                self.createSecTree(node)
                break
        extendedNodes = gml.getElementsByTagName('CellSpace')
        for node in extendedNodes:
            if ((node.attributes['gml:id'].value == item.text(1)) or ('#' + node.attributes['gml:id'].value == item.text(1))):
                self.createSecTree(node)
                break
        extendedNodes = gml.getElementsByTagName('State')
        for node in extendedNodes:
            if ((node.attributes['gml:id'].value == item.text(1)) or ('#' + node.attributes['gml:id'].value == item.text(1))):
                self.createSecTree(node)
                break
    else:
        self.dlg.secTree.clear()
```

Figura 55. Funció displayXlinkInformation().

- **Funció createSecTree()**

La funció createSecTree() generarà un segon arbre d'exploració, a partir de l'objecte entregat com a paràmetre, per tal de representar l'objecte de tipus xlink:href seleccionat a l'arbre d'exploració principal. A la Figura 56 s'observa el codi que la composa.

```
# The secondary tree is created. It will show infomration of xlink:href linked features.
def createSecTree (self, node):
    secTree = self.dlg.secTree
    header = QtGui.QTreeWidgetItem(["xlink:href Feature Information","Value"])
    secTree.setHeaderItem(header)
    secTree.header().setResizeMode(QtGui.QHeaderView.ResizeToContents)
    secTree.header().setStretchLastSection(False)
    self.dlg.secTree.clear()
    root = QtGui.QTreeWidgetItem(secTree, [node.nodeName])
    varA = QtGui.QTreeWidgetItem(root, ['Properties'])
    self.getAttributes(node,varA)
    self.getNodes(node.childNodes,400,varA)
    if node.nodeName == 'State':
        points = node.getElementsByTagName('gml:Point')
        for point in points:
            varB = QtGui.QTreeWidgetItem(root, [point.nodeName])
            self.getAttributes(point,varB)
            self.getNodes(point.childNodes,400,varB)
    if node.nodeName == 'CellSpace':
        polygons = node.getElementsByTagName('gml:Polygon')
        for polygon in polygons:
            varB = QtGui.QTreeWidgetItem(root, [polygon.nodeName])
            self.getAttributes(polygon,varB)
            self.getNodes(polygon.childNodes,400,varB)
    if node.nodeName == 'Transition':
        lineStrings = node.getElementsByTagName('gml:LineString')
        for lineString in lineStrings:
            varB = QtGui.QTreeWidgetItem(root, [lineString.nodeName])
            self.getAttributes(lineString,varB)
            self.getNodes(lineString.childNodes,400,varB)
```

Figura 56. Funció createSecTree().

Aquest arbre secundari es generarà a l'espai del disseny expressament creat a la finestra de càrrega i exploració pels objectes de tipus xlink:href seleccionats a l'arbre principal, com es veu a la Figura 57.

Figura 57. Arbre d'exploració secundari del plugin IndoorGML Reader.

4.3.4. Representació gràfica del conjunt de dades

Un dels principals objectius funcionals que persegueix el plugin per QGIS vers els conjunts de dades en format IndoorGML és la representació gràfica de la informació geomètrica que conté el conjunt de dades carregat.

A diferència de la representació de l'arbre d'exploració, QGIS si permet fer la representació gràfica mitjançant la seva pròpia interfície gràfica, aquest és el motiu per el qual el plugin generarà una nova capa vectorial a QGIS per a representar els objectes de tipus Transition i State del MultiLayeredGraph – que es recorda representaran l'espai dual – i els CellSpace que faran el mateix amb l'espai primitiu.

Per tal d'oferir aquesta representació, l'arxiu Indoorgml_reader.py conté un conjunt de funcions que capaciten aquesta representació, i dels que a continuació se'n fa revisió.

- **Funció printCellSpaces()**

Amb l'objectiu de fer la representació d'espais de cel·les continguts a l'espai primari dins el conjunt de dades IndoorGML, la funció printCellSpaces() explora el contingut de l'arxiu IndoorGML carregat al plugin cercant la informació geomètrica poligonal que permetrà fer-ne la representació gràfica.

En primer lloc, s'estableix el nom que se li assignarà a la capa vectorial al Layers Panel de QGIS, en funció del nom de l'arxiu. També es controla que no existeixi ja una capa amb un nom coincident, per evitar duplicar continguts.

La generació dels objectes de tipus polygon es farà registrant el conjunt de punts que el composen, i introduint-lo a l'objecte QgsGeometry.fromPolygon [32] que quedarà inserit a la capa vectorial creada, com es pot veure a la Figura 58.

```
def printCellSpaces (self,gml):
    # We get the filename in order to give name the layer.
    fileName = os.path.basename(os.path.normpath(self.dlg.gmlLine.displayText()))
    layerName = '(CellSpaces) - ' + fileName

    # In order to avoid repeated layers.
    layers = self.iface.legendInterface().layers()
    layerExists = 0
    for layer in layers:
        if layer.name() == layerName:
            layerExists = 1

    polygons = gml.getElementsByTagName('gml:Polygon')

    if ((polygons.length >= 1) and (layerExists == 0)):

        layer =  QgsVectorLayer('Polygon', layerName, "memory")
        pr = layer.dataProvider()

        for polygon in polygons:
            positions = polygon.getElementsByTagName('gml:pos')
            points = []
            for position in positions:
                stringPosition = position.childNodes[0].data
                splitedPosition = stringPosition.split(' ')
                point = QgsPoint(float(splitedPosition[0]),float(splitedPosition[1]))
                points.append(point)

            pol = QgsFeature()
            pol.setGeometry(QgsGeometry.fromPolygon([points]))
            pr.addFeatures([pol])
            layer.updateExtents()

        QgsMapLayerRegistry.instance().addMapLayers([layer])

        # Setting the zoom scale to this layer.
        canvas = qgis.utils.iface.mapCanvas()
        canvas.setExtent(layer.extent())
```

Figura 58. Funció printCellSpaces().

Un cop totes les features que conformen la capa de CellSpaces han estat generades, s'incorpora la capa al Layers Panel de l'entorn de treball de QGIS, adaptant-ne el zoom per facilitar la visualització a l'usuari.

- **Funció printTransitions()**

Els elements de la classe Transition poden representar-se com línies rectes que connecten dos o més components de classe State.

La funció printTransitions() cerca al conjunt de dades carregat al plugin IndoorGML Reader tots els elements XML de classe Transition, i registra tots els punts geomètrics que el composen en un array. Cadascun d'aquests arrays generaran una feature de tipus QgsGeometry.fromPolyLine [32]. El conjunt de tots aquests elements conformarà una nova capa vectorial que inclourà novament el nom de l'arxiu carregat. Els detalls apareixen a la Figura 59.

```
def printTransitions (self,gml):
    # We get the filename in order to give name the layer.
    fileName = os.path.basename(os.path.normpath(self.dlg.gmlLine.displayText()))
    layerName = '(Transitions) - ' + fileName

    # In order to avoid repeated layers.
    layers = self.iface.legendInterface().layers()
    layerExists = 0
    for layer in layers:
        if layer.name() == layerName:
            layerExists = 1

    transitions = gml.getElementsByTagName('Transition')

    if ((transitions.length >= 1) and (layerExists == 0)):

        layer =  QgsVectorLayer('LineString', '(Transitions) - ' + fileName, "memory")
        pr = layer.dataProvider()

        for transition in transitions:
            positions = transition.getElementsByTagName('gml:pos')
            points = []
            for position in positions:
                stringPosition = position.childNodes[0].data
                splitedPosition = stringPosition.split(' ')
                point = QgsPoint(float(splitedPosition[0]),float(splitedPosition[1]))
                points.append(point)

            l = len(points)
            i = 0

            for point in points:
                if l >= 2:
                    line = QgsFeature()
                    line.setGeometry(QgsGeometry.fromPolyline([points[i],points[i + 1]]))
                    pr.addFeatures([line])
                    layer.updateExtents()
                    i = i + 1
                    l = l - 1

        QgsMapLayerRegistry.instance().addMapLayers([layer])

        # Setting the zoom scale to this layer.
        canvas = qgis.utils.iface.mapCanvas()
        canvas.setExtent(layer.extent())
```

Figura 59. Funció printTransitions().

Per tal d'evitar duplicitats, es comprova la no existència d'una capa vectorial amb un nom coincident amb el de la capa tractada, i s'ajusta la visualització per adaptar el visor al contingut d'aquesta.

- **Funció printStates()**

La funció printStates() capacita el plugin IndoorGML Reader per tal de fer representació gràfica del conjunt d'elements de la classe State identificats al conjunt de dades carregat.

Com s'observa a la Figura 60, la capa vectorial creada contindrà el nom de l'arxiu en base al que es genera, i se'n controla la no existència prèvia al Layers Panel.

Es genera després un array amb el conjunt d'elements XML de tipus State, i per a cadascun d'ells s'explora el punt geomètric que el representarà en forma de QgsGeometry.fromPoint [32], conformant una capa que contindrà totes les features d'aquest tipus. La funció s'encarrega finalment d'adaptar el focus de visualització a la nova capa vectorial generada.

```
# Creating States layer.
def printStates (self,gml):
    # We get the filename in order to give name the layer.
    fileName = os.path.basename(os.path.normpath(self.dlg.gmlLine.displayText()))
    layerName = '(States) - ' + fileName

    # In order to avoid repeated layers.
    layers = self.iface.legendInterface().layers()
    layerExists = 0
    for layer in layers:
        if layer.name() == layerName:
            layerExists = 1

    states = gml.getElementsByTagName('State')

    if ((states.length >= 1) and (layerExists == 0)):

        layer = QgsVectorLayer('Point', '(States) - ' + fileName , "memory")
        pr = layer.dataProvider()

        for state in states:
            pos = state.getElementsByTagName('gml:pos')
            stringPosition = pos[0].childNodes[0].data
            splitedPosition = stringPosition.split(' ')
            pt = QgsFeature()
            point1 = QgsPoint(float(splitedPosition[0]),float(splitedPosition[1]))
            pt.setGeometry(QgsGeometry.fromPoint(point1))
            pr.addFeatures([pt])
            layer.updateExtents()

        QgsMapLayerRegistry.instance().addMapLayers([layer])

        # Setting the zoom scale to this layer.
        canvas = qgis.utils.iface.mapCanvas()
        canvas.setExtent(layer.extent())
```

Figura 60. Funció printStates().

5. Exemple d'ús del plugin IndoorGML Reader

El plugin per a QGIS IndoorGML Reader ha estat dissenyat amb l'objectiu que l'experiència d'usuari sigui senzilla i àgil, aquest és el motiu pel qual la capacitat d'exploració i representació del conjunt de dades en format IndoorGML Reader per part de l'usuari només requereix de la càrrega de l'arxiu que conté conjunt de dades, desencadenant la conformació de l'arbre d'exploració del conjunt de dades així com la seva representació gràfica.

La primera intervenció per part de l'usuari per carregar el fitxer en format IndoorGML al programari QGIS mitjançant el plugin IndoorGML Reader serà fent clic a la icona del plugin a la barra d'eines de l'aplicació, o navegant fins la opció al menú contextual. S'observa el detall a les Figures 61 i 62 respectivament.

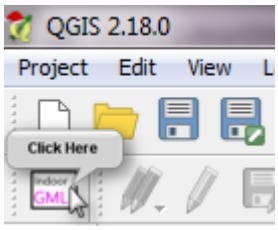


Figura 61. Detall de la icona per executar el plugin.

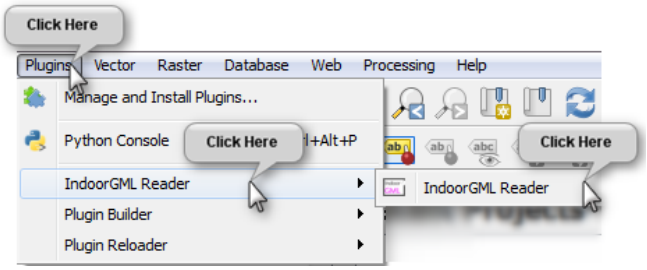


Figura 62. Representació de la opció al menú per obrir el plugin.

A continuació, apareixerà la finestra de càrrega i exploració, l'usuari haurà de polsar sobre el botó IndoorGML file, el detall apareix a la Figura 63.

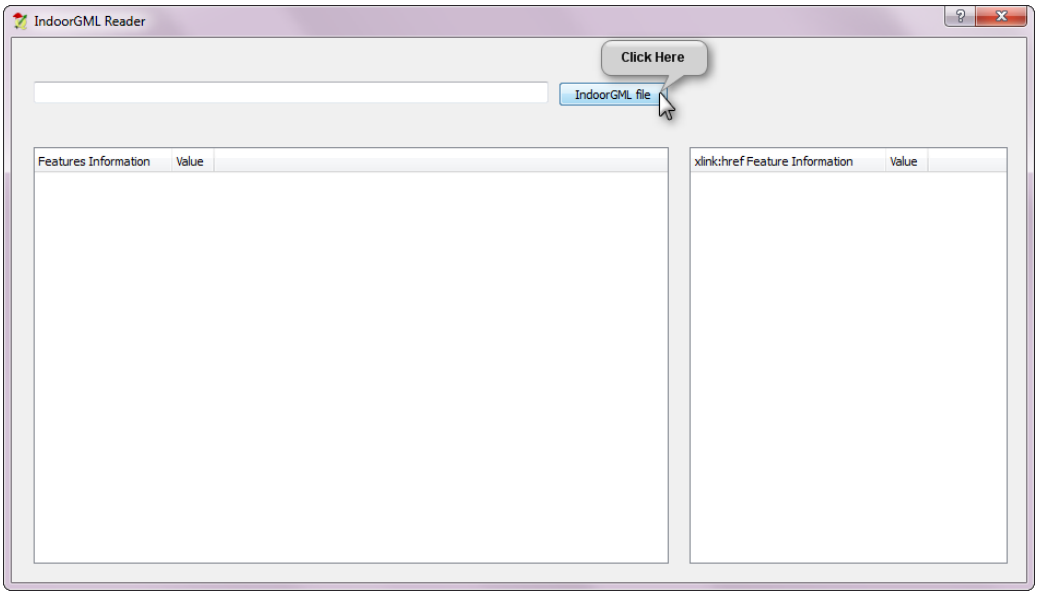


Figura 63. Detall de càrrega d'arxiu en format IndoorGML.

Aquesta acció provocarà l'aparició d'una finestra emergent, per tal de seleccionar un arxiu a l'entorn local d'usuari, d'un explorador d'arxius semblant al de la Figura 64. Aquest explorador només mostrarà arxius amb extensió GML, serà precís seleccionar un arxiu vàlid i fer clic a Obrir.

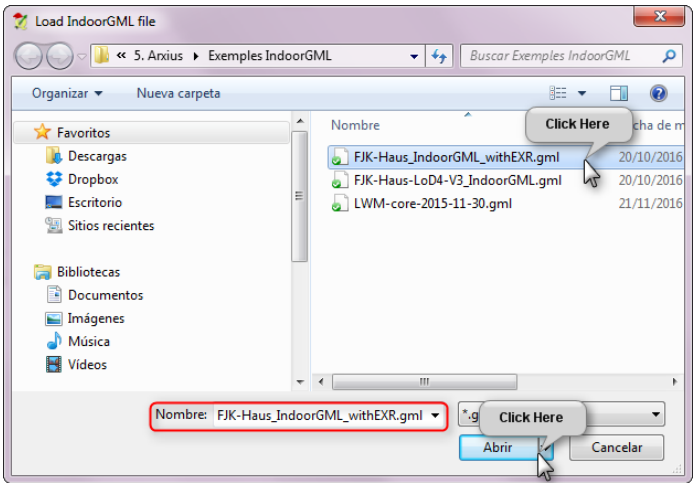


Figura 64. Finestra d'exploració d'arxius IndoorGML.

En generar les noves capes vectorials, el programari de QGIS demana automàticament a l'usuari, mitjançant una finestra de selecció, el CRS [6] en el que les noves capes, generades a partir del conjunt de dades, seran creades. La Figura 65 mostra el diàleg que QGIS ofereix per establir-ho, un per cada capa generada.

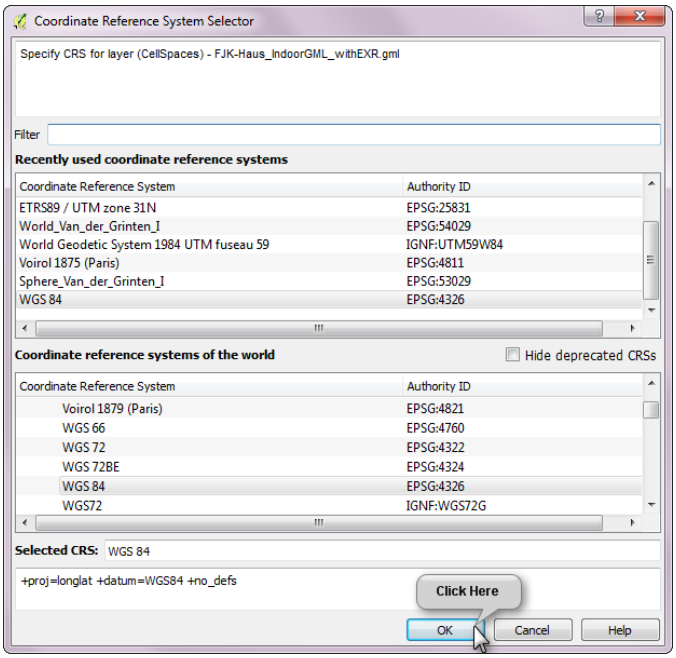


Figura 65. Selecció de CRS per la capa vectorial.

De manera desatesa, el codi Python del plugin genera a la finestra d'exploració l'arbre del conjunt de dades extret de l'arxiu seleccionat; al panell d'exploració principal l'usuari pot explorar el conjunt de dades, mentre que a la part dreta apareixerà informació relativa als elements de tipus xlink:href - que recordem són elements relacionats amb el node explorat -. El detall és a la Figura 66.

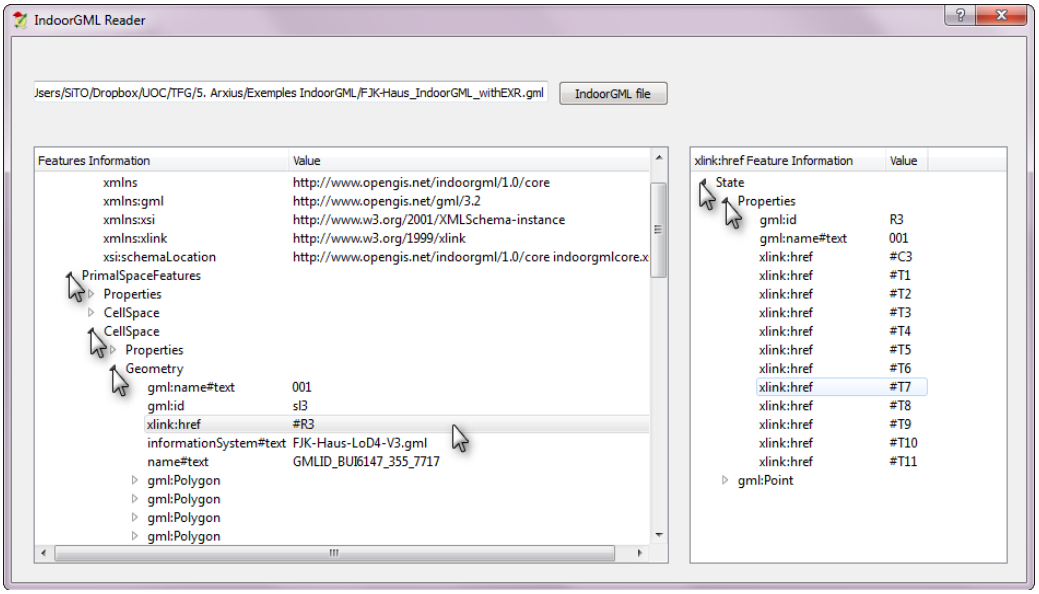


Figura 66. Finestra de càrrega i exploració del plugin IndoorGML Reader.

Al mateix temps, també es genera la representació gràfica dels elements de l'espai primitiu i dual del conjunt de dades a l'entorn de treball de QGIS, l'usuari tindrà oportunitat d'explorar les diferents capes, i cadascuna de les features en elles creades, com s'observa a la Figura 67.

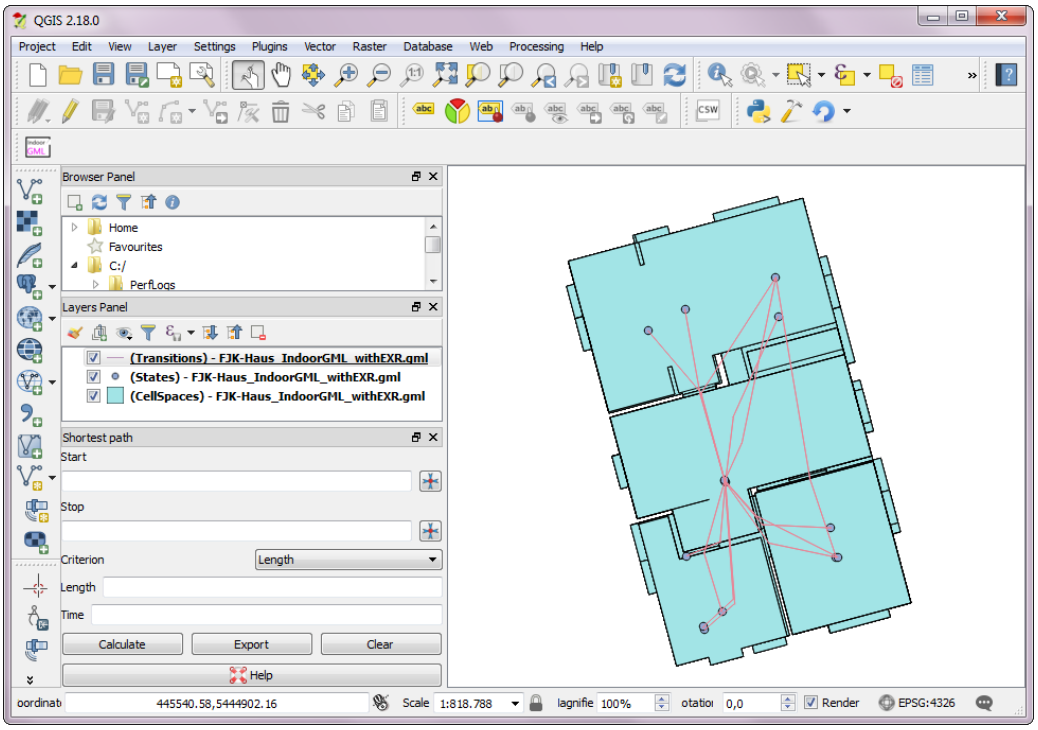


Figura 67. Entorn de treball QGIS amb les capes del plugin.

6. Conclusions i línies futures de treball

A continuació és descriuen, en termes analítics, un seguit d'aspectes relacionats amb les lliçons apreses fruit de l'esforç realitzat en l'execució de les diferents tasques que han composat aquest Treball de Final de Grau.

El focus es portarà en primer lloc al nivell de consecució dels objectius establerts en la proposta inicial del projecte respecte al producte entregat.

A més es farà revisió de les desviacions, en termes tant d'acompliment de previsió temporal com de mètode seguit, i es detallarà una projecció de línies de seguiment futures relacionades amb el treball.

6.1. Nivell de consecució d'objectius

Aquest projecte preveia la definició d'un seguit d'objectius genèrics que ajudarien a mesurar el nivell d'excel·lència i completesa en la consecució del projecte, són els següents:

6.1.1. Objectiu 1: Profunditzar en el coneixement de l'estàndard OGC IndoorGML

Les tasques relacionades amb objectiu d'assolir un grau de competència alt en el coneixement de l'estàndard IndoorGML han estat satisfactòries.

El coneixement conceptual dels espais de cel·les amb les seves diferents representacions és complert, de la mateixa manera que s'ha assolit un nivell alt en la comprensió dels models espacials, les relacions amb conjunts de dades externs i el model de dades proposat.

6.1.2. Objectiu 2: Conèixer Estat de l'art actual del software per la lectura del format IndoorGML

Un àmbit important de treball per la consecució del projecte entenia el coneixement de l'estat de l'art de les eines software comercials per la lectura del format IndoorGML, tant en el context d'eines d'ús lliure com de pagament.

L'estudi d'aquest mercat ha acabat derivant en l'aprofundiment en una selecció d'eines que es preveia interessant en el context de la lectura d'aquest format i, més enllà de les conclusions sobre les seves capacitats, en termes generals s'entén que s'ha assolit un bon grau de coneixement d'aquestes, en concret serien:

- FME Desktop 2016
- WebGL Indoor GML Viewer
- ArcGIS Pro amb extensió d'Interoperabilitat.
- QGIS amb plugin Complex GML Info

6.1.3. Objectiu 3: Implementar un plugin Python per QGIS per la lectura del format IndoorGML

Fruït de la necessitat de disposar d'una eina àgil i gratuïta per l'exploració i representació del format IndoorGML, s'estableix com a un dels objectius principals en el Treball de Final de Grau el disseny i implementació d'un plugin per a QGIS, codificat en Python, per la representació i lectura d'aquest model de dades.

En aquest cas, l'objectiu del disseny del plugin ha estat assolit en termes operatius, és a dir, aquest és capaç d'oferir mecanismes per l'exploració del model de dades i, a més, en fa la representació gràfica mitjançant la implementació de capes vectorials.

Malgrat això, problemes derivats de la immaduresa del format – essent difícil fins i tot trobar conjunts de dades d'exemple – i de la manca d'experiència en el desenvolupament de codi en llenguatge Python i relacionat amb el programari QGIS, han provocat que el rendiment del plugin i la seva robustesa no siguin tant bons com podria esser desitjable.

6.1.4. Objectiu 4: Publicar el plugin IndoorGML Reader al repositori oficial de QGIS

L'objectiu de publicació del plugin general per la lectura del format IndoorGML depenia absolutament de la consecució adequada, en termes de desviacions temporals i de mètode, amb l'objectiu 3.

En aquest sentit, tenint en compte aquests condicionants, no ha estat possible afrontar el procediment d'aprovació establert per la comunitat QGIS en els termes adients en els següents aspectes:

- No es disposava de les dues setmanes de marge per l'aprovació.
- És difícil garantir la seva robustesa en termes de rendiment.

6.1.5. Resum de consecució d'objectius.

Per tal d'oferir una representació visual i sintètica del grau de consecució dels objectius, a continuació es presenta una taula que entrega informació d'acceptació de la consecució de l'objectiu i, a més, ofereix una mesura del grau de compliment d'aquest.

	Assolit	Grau de consecució
Objectiu 1	Sí	Alt
Objectiu 2	Sí	Alt
Objectiu 3	Sí	Alt - Mitjà
Objectiu 4	No	N/A

6.2. Avaluació de desviacions

La previsió inicial del projecte establia uns termes, en l'àmbit temporal i de mètode, que en principi s'havien definit amb intenció de garantir la consecució total dels objectius establerts en els terminis requerits.

És interessant apuntar que, malgrat la previsió de tasques definida inicialment sí contemplava tots els elements necessaris per tal de fer un desenvolupament adequat del projecte i una consecució òptima d'objectius, si han hagut errades remarcables en l'estimació temporal d'aquestes. En concret, aquelles que han tingut una incisió més remarcable en la previsió són:

- L'anàlisi d'eines amb capacitats de lectura IndoorGML ha estat més complex del previst.
- La comprensió exigida del llenguatge Python i les llibreries necessàries pel desenvolupament del plugin eren molt més altes de la estimació.
- La creació de la memòria i la presentació és més exigent temporalment del pronòstic inicial.

Generalment, per tal d'ajustar temporalment les desviacions necessàries, ha estat precís fer renúncies en la consecució d'objectius, com la publicació del plugin al repositori oficial de QGIS, així com fer una inversió temporal molt més enllà del previst inicialment.

6.3. Línies futures de treball

Els objectius que persegueix aquest projecte s'establiren en el marc de la necessitat d'eines per la gestió de l'estàndard de dades d'informació geogràfica IndoorGML, en el que el context temporal de la recent oficialització per part de l'OGC com a referència en la navegació d'espais interiors entrega importància a aquesta necessitat.

Aquest projecte ha assolit un grau d'execució satisfactori en la comprensió i representació de models de dades en format IndoorGML, però es preveu que aquests mecanismes no seran suficients per satisfer les necessitats dels usuaris potencials d'aquesta. En aquest sentit, una formidable nova línia de treball contemplaria la possibilitat de implementar un plugin que incorporés noves funcionalitats, en serien exemple:

- Conversió del conjunt de dades a altres formats, com CityGML o KML.
- Modificació dels elements geomètrics de les classes que conformen el conjunt de dades.
- Edició d'atributs de les diferents features incloses al model de dades.
- Afegir nous elements al conjunt de dades, per exemple un nou CellSpace.

Pel que fa a la representació gràfica, i sense sortir de l'àmbit de les capacitats del plugin per QGIS, el mecanisme de visualització d'aquest no preveu fer representacions en format de tres dimensions. No obstant, existeixen alguns mecanismes de desenvolupament alternatiu que permeten fer una representació en tres dimensions que farien que aquesta es posés a l'alçada d'altres eines de pagament, com FME Desktop 2016.

D'altra banda, i donant més amplada a l'abast de la línia futura de treball, i amb l'objectiu de conèixer millor el mecanisme de implementació i oferir una eina didàctica per la seva comprensió, seria interessant generar un disseny per un espai interactiu on, a través de la representació gràfica del model de dades, l'aplicació entregués contínuament informació navegacional a l'usuari relativa a la coordenada exacta on hi ha el cursor. D'aquesta manera es disposaria d'una eina que mostraria en quin espai de cel·les de l'espai primitiu es troba, quin és el State que el representa en cada capa de l'espai dual, i quines Transitions pot fer servir per arribar als espais navegables contigus en cada capa.

7. Glossari

La redacció d'aquesta memòria de treball ha exigit l'ús constant de termes específics i acrònims relacionats amb l'àmbit del treball. A continuació se'n fa un inventari per facilitar-ne la comprensió:

- GPS – Global Positioning System
  - Sistema de posicionament basat en satèl·lits que entrega informació posicional als clients GPS en qualsevol espai obert de la terra.
- OGC – Open Geospatial Consortium
  - Organització reguladora sense ànim de lucre en l'àmbit de desenvolupament de contingut i estàndards amb propòsits de geolocalització.
- GML – Geography Markup Language
  - Gramàtica basada en XML, aprovada per la OGC, usada en la representació geogràfica i amb molta potència per capacitar els intercanvis de informació geogràfica a Internet.
- IndoorGML – Indoor based Geography Markup Language
  - Model de dades estàndard per la representació i intercanvi de models d'espais interiors especialitzat en la capacitat de navegació.
- CityGML – City based Geography Markup Language
  - Model de dades estàndard per la representació i intercanvi de models de ciutats i edificis.
- KML – Keyhole Markup Language
  - Model de dades basat en XML que s'especialitza en la expressió de notació i visualització geogràfica en navegadors d'Internet.
- SIG – Sistema d'Informació Geogràfica
- Open source – Eina de codi lliure.
- Plugin – Complement.
  - Element que complementa les funcionalitats d'un determinat programari software.
- PAC – Prova d'Avaluació Continuada.
- TFG – Treball de Final de Grau.
- XML – Extensible Markup Language
  - Llenguatge de referència basat en unes determinades regles que permeten que pugui ser llegit per màquines però alhora essent comprensible pels humans.
- Python
  - Llenguatge de programació d'alt nivell, de propòsit general i utilitzat de manera massiva.
- LBS – Location Based Service
  - Servei de capa software que utilitza el posicionament d'un dispositiu per entregar diverses funcionalitats a aquest.

- IFC – Industry Foundation Classes
    - Model de dades que pretén descriure informació relativa a la construcció d'edificis industrials.
- WiFi
    - Tecnologia dissenyada amb l'objectiu d'entregar capacitats de xarxa sota l'estàndard IEEE 802.11 sense la necessitat de connexió física mitjançant cable.
- CRS – Coordinate Reference System
    - Sistema de referència de localització de geogràfica que estableix una projecció per la representació d'informació geogràfica.
- NRG – Node Relationship Graph
    - Model conceptual per la representació de les relacions entre nodes.
- RFID – Radio Frequency Identification
    - Mecanisme que, basat en la difusió i recepció de camps electromagnètics, capacita un sistema per identificar objectes.
- UML – Unified Modeling Language
    - Llenguatge de modelatge amb el propòsit de representar el disseny d'un sistema.
- HTML – HyperText Markup Language
    - Llenguatge estàndard de referència emprat de forma massiva en la creació de planes web.
- WEB
    - Terme fet servir en termes genèrics per descriure un conjunt de recursos globals que es localitzen a Internet.
- IDE – Integrated Development Environment
    - Programari que es dissenya amb la intenció d'oferir un espai de treball idoni per la construcció de codi en un determinat llenguatge.
- API – Application Programming Interface
    - Conjunt de mecanismes, funcions i rutines que s'estableixen amb la intenció de capacitar la interoperabilitat d'una aplicació.

8. Bibliografia

- [1] http://docs.opengeospatial.org/is/14-005r4/14-005r4.html
  - OGC IndoorGML Definition
- [2] http://www.technavio.com/blog/top-33-indoor-location-based-services-lbs-companies-in-the-us
  - Informe que analitza l'adopció de sistemes LBS en el mercat els propers anys.
- [3] http://www.buildingsmart-tech.org/specifications/ifc-overview
  - Industry Foundation Classes (IFC) Overview
- [4] http://www.opengeospatial.org/standards/kml
  - OGC Keyhole Markup Language (KML) Definition.
- [5] http://www.opengeospatial.org/standards/citygml
  - OGC CitiGML Definition
- [6] http://inspire.ec.europa.eu/theme/rs
  - Coordinate reference Systems (CRS)
- [7] http://www.opengeospatial.org/node/2157
  - Nota del OGC oficialitzant l'adopció de IndoorGML com l'estàndard de referència en la representació de dades per la navegació en espais interiors.
- [8] http://www.safe.com/
  - Plana web de SAFE.
- [9] http://indoorgml.net/resource.html
  - Secció de recursos de la plana web oficial de la OFC per l'estàndard per la Informació Espacial Indoor.
- [10] http://english.pusan.ac.kr/uPNU_homepage/en/default.asp
  - Plana web de la Pusan National University.
- [11] http://uwcms.pusan.ac.kr/user/cseeng/index.action
  - Departament de Ciència de la computació i Enginyeria de la Pusan National University.
- [12] http://threejs.org/
  - Plana web de la llibreria JavaScript Three.js.
- [13] http://browserify.org/
  - Plana web de la llibreria JavaScript browserify.
- [14] https://github.com/highsource/jsonix
  - Plana web de la llibreria JavaScript jsonix.
- [15] https://opensource.org/licenses/MIT
  - Definició de llicencia de tipus MIT.
- [16] http://www.esri.com/
  - Plana web oficial del producte ArcGIS.
- [17] http://www.ogs.ny.gov/purchase/prices/7600020751prices.pdf
  - Document .pdf amb informació de llicenciament per ArcGIS.
- [18] http://www.qgis.org/ca/site/
  - Plana web del software QGis.
- [19] https://plugins.qgis.org/plugins/ComplexGmlInfo/
  - Plana web del plugin QGis Complex GML Info
- [20] https://www.qt.io/about-us/
  - Plana web de Qt Company.
- [21] https://notepad-plus-plus.org/
  - Plana web de Notepad++.

- [22] https://qgis.org/api/modules.html
  - QGIS API Documentation
- [23] https://pypi.python.org/pypi/PyQt4
  - PyQt4 4.11.4
- [24] https://docs.python.org/2/library/xml.dom.html
  - The Document Object Model API
- [25] https://plugins.qgis.org/plugins/pluginbuilder/
  - Plugin Builder
- [26] https://plugins.qgis.org/plugins/plugin_reloader/
  - Plugin Reloader
- [27] http://doc.qt.io/qt-5/qlineedit.html
  - QLineEdit class
- [28] http://doc.qt.io/qt-4.8/qpushbutton.html
  - QPushButton class
- [29] http://doc.qt.io/qt-4.8/qtreewidget.html
  - QTreeWidget class
- [30] www.qgis.org/en/site/getinvolved/development/plugindevelopment.html
  - QGIS plugin development languages
- [31] http://www.georss.org/gml.html
  - Definició de gml:Polygon, gml:Point i gml:LineString
- [32] https://qgis.org/api/classQgsGeometry.html
  - QgsGeometry Class Reference
- [33] https://plugins.qgis.org/plugins/plugins.xml
  - Repositori oficial de plugins QGIS

9. Annexos

En aquesta secció s'inclouen apartats que amplien o detallen informació relativa al treball, però el contingut dels quals no tenen el pes específic exigible per aparèixer a la memòria.

9.1. Procediment d'instal·lació de QGIS

A continuació es detallaran els passos necessaris per la instal·lació del software de informació geospacial de llicència lliure QGIS.

9.1.1. Descàrrega de l'executable

El primer pas per tal de dur a terme la instal·lació del software de sistema de informació geospacial QGIS serà descarregar la darrera versió de l'aplicació. Aquesta aplicació està disponible per diferents sistemes operatius – i.e. Windows, MAC OS, Linux, BSD i Android – en la plana web de l'aplicació.

- http://www.qgis.org/es/site/forusers/download.html

En aquest procediment es farà servir la versió per Windows, en la Figura 68 es mostra el detall de la descàrrega de la versió per a arquitectures de 64 bits.

Figura 68. Detall de descàrrega de l'executable de QGIS.

9.1.2. Instal·lació del software

Un cop descarregat l'executable de l'aplicació QGIS, que en el cas de la creació d'aquest procediment serà la versió 2.18.0-1, s'ha de fer doble clic en el mateix, tal i com s'observa a la Figura 69:

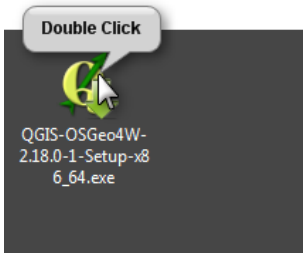


Figura 69. Executant el instal·lador.

Un cop el instal·lador ha efectuat les comprovacions inicials, apareix la primera pantalla de l'assistent de instal·lació. Com es veu a la Figura 70, s'ha de polsar sobre Siguiente:

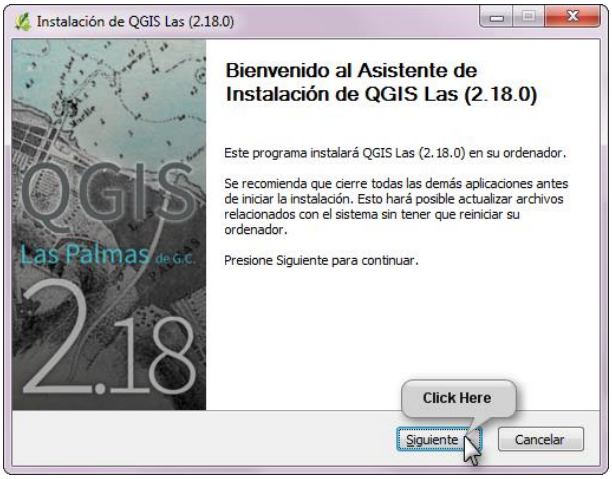


Figura 70. Primera pantalla instal·lació QGIS.

En el següent pas haurà d'esser acceptat l'acord de llicència de l'aplicació, a la Figura 71 es mostren les possibilitats de lectura de l'acord, i com l'acceptació és fa prement el botó Acepto:

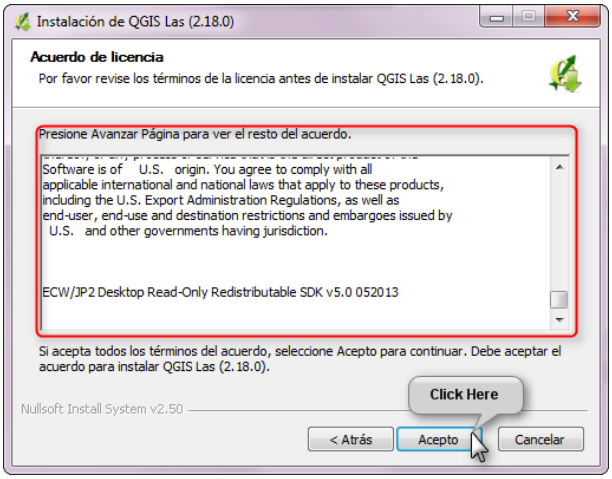


Figura 71. Acceptació de l'acord de llicència QGIS.

A continuació s'ha de definir la localització de la instal·lació del software, en el cas de l'exemple es mantindrà la ubicació suggerida, com es pot comprovar a la Figura 72. Prenem novament Siguiente.

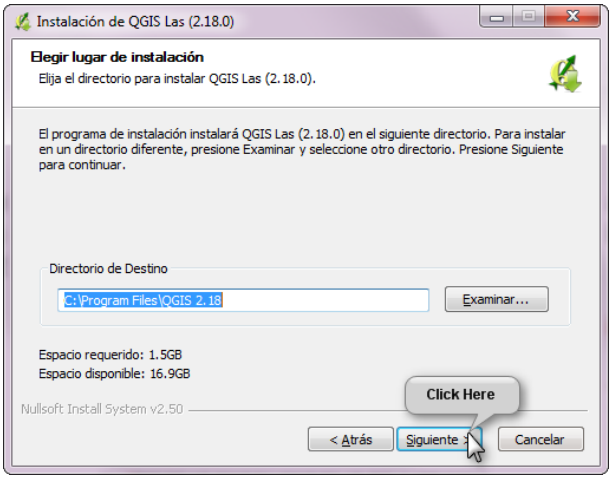


Figura 72. Ubicació de la instal·lació.

A continuació l'assistent suggereix fer instal·lació addicional d'alguns complements. En el cas de l'exemple no és necessari, es prem Instalar sense incloure'n cap, el detall és visible a la Figura 73.

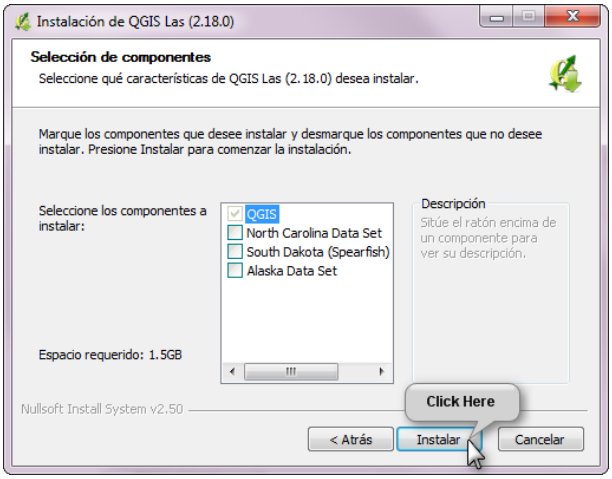


Figura 73. Inici de la instal·lació de QGIS.

Per últim, com es pot veure a la Figura 74, es rep confirmació de l'assistent de la correcta instal·lació de l'aplicació. S'ha de fer clic a Terminar.

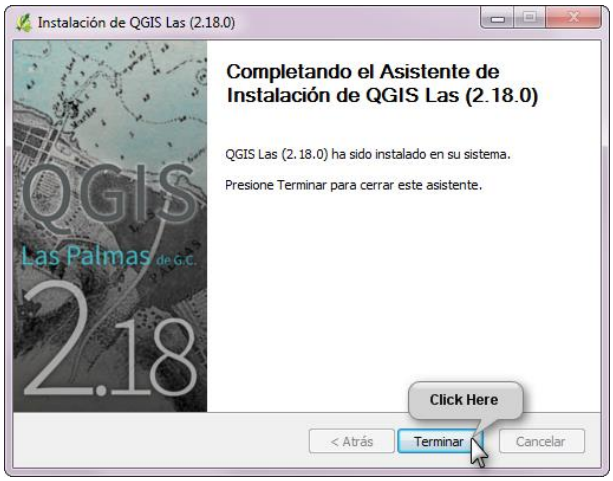


Figura 74. Final de la instal·lació de QGIS

9.2. Procediment d'instal·lació del plugin IndoorGMLReader per QGIS

La gestió dels complements o plugins amb els que l'aplicació QGIS treballa acostuma a dur-se a terme mitjançant el seu gestor de complements, que es troba a la barra Connectors, prement Gestiona i instal.la connectors. Es veu el detall a la Figura 75:

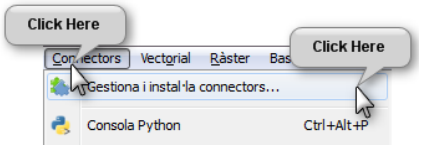


Figura 75. Gestor de complements QGIS.

No obstant això, si existeix interès en afegir en afegir un plugin que encara no ha estat publicat al repositori oficial de QGIS [33], s'haurà de fer una instal·lació manual.

Malgrat que pugui semblar encara més complicat, només serà necessari moure la carpeta del plugin sencera a la ruta on la instal·lació de QGIS deixa els complements – sempre sota el supòsit que s'estigui treballant amb una màquina amb sistema operatiu Windows –, serà la següent:

- C:\Users**<<UserName>>**\.qgis2\python\plugins\

És interessant remarcar que <<UserName>> ha de quedar substituït pel nom d'usuari en el que s'ha iniciat sessió a la màquina de la instal·lació.

En l'exemple de la Figura 76 es pot comprovar la ruta en la que els complements queden ubicats, i com s'hi localitzen, tant els instal·lats amb els gestor de plugins com els generats manualment.

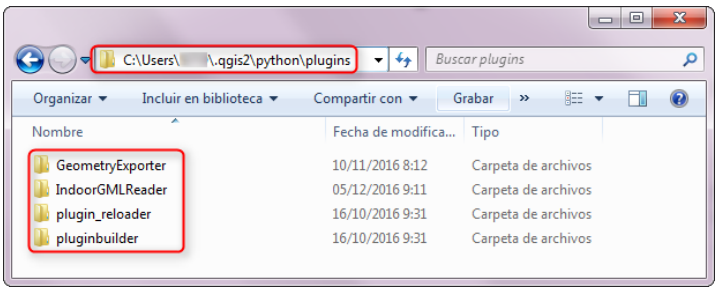


Figura 76. Detall de ruta de plugins.