Escuela Técnica Superior de Ingenieros de Caminos, Canales y Puertos.

UNIVERSIDAD DE CANTABRIA

DYNAMO PARA CIVIL 3D: PROGRAMACIÓN VISUAL Y BIM EN OBRA CIVIL

TRABAJO FINAL DE MÁSTER

Trabajo realizado por:

Pablo Eizaguirre García

Dirigido:

César Otero González

Titulación:

Máster en Ingeniería de Caminos, Canales y Puertos

Santander, Julio 2020

RESUMEN

Las herramientas informáticas empleadas en proyectos de ingeniería civil están sufriendo una transformación motivada por la implantación de la metodología BIM. De entre las nuevas tendencias se encuentra la Programación Visual vinculada a entornos de modelado 3D.

En el contexto del modelado de obra civil -más específicamente, obra lineal- el programa *Autodesk Civil 3D* lanza en octubre de 2019 su extensión para Programación Visual en el entorno *Autodesk Dynamo*. Dynamo es una plataforma de diseño computacional paramétrico que se vincula a otros entornos (Revit y Civil 3D) y desarrolla para ellos rutinas y algoritmos. La capacidad de aplicación de Dynamo para Civil 3D en proyectos BIM viene determinada por los flujos de trabajo BIM en torno a Civil 3D y la situación de IFC actual.

Los flujos de trabajo en BIM están definidos por un conjunto de modelos virtuales generados por softwares de múltiples disciplinas -modelado 3D, cálculo estructural, planificación de obra, presupuestos…- que logran trabajar colaborativamente gracias a su unificación en un Modelo Federado. El Modelo Federado interpreta todos los modelos virtuales a través de IFC, pero no es capaz de editarlos o llevar a cabo modificaciones. Esto se realiza en cada modelo virtual.

El modelo virtual, entonces, debe ser capaz de adaptarse a cambios y mantenerse actualizado durante todo el ciclo de vida del proyecto -concepción, diseño, licitación, construcción y operación- en acorde con el Modelo Federado. Este escenario impulsa la aplicación de Dynamo en Civil 3D a través de dos casos de estudio.

El primer caso de estudio analiza, desde la perspectiva de la fase constructiva del proyecto, la discretización de obras lineales en tramos vinculados a la planificación de obra y sus certificaciones. Se crean dos algoritmos en Dynamo que permiten segmentar la obra lineal y gestionar la distribución de los tramos a lo largo de la obra lineal. Los algoritmos se basan en programación del *API.NET* de Civil 3D a través de lenguaje *Python* en Dynamo.

El segundo caso de estudio trata la exportación de movimientos de tierras de Civil 3D a través de IFC2.3. Se ejemplifica la exportación a Presto a través de Cost-It para asignar -dinámicamente- partidas presupuestarias a los movimientos de tierras. Se observa que para conseguir la exportación a IFC 2.3 es necesario generar en Civil 3D sólidos que conformen el volumen del movimiento de tierras. Para ello se crea un algoritmo parametrizado que genera automáticamente los sólidos y exporta además las mediciones a Excel. El desarrollo se basa en la Programación Visual de Dynamo.

Con todo, se representa la situación y la interacción de un entorno de Programación Visual y el desarrollo de proyectos BIM en obra civil.

ABSTRACT

The computer tools used in civil engineering projects are undergoing a transformation due to the implementation of the BIM methodology. Among the new trends is the Visual Programming linked to 3D modeling environments.

In the context of civil works modeling - more specifically, linear works - the Autodesk Civil 3D program launches in October 2019 its extension for Visual Programming in the Autodesk Dynamo environment. Dynamo is a parametric computer design platform that links to other environments (Revit and Civil 3D) and develops routines and algorithms for them. Dynamo's application capability for Civil 3D in BIM projects is determined by the BIM workflows around Civil 3D and the current IFC situation.

BIM workflows are defined by a set of virtual models generated by software from multiple disciplines - 3D modeling, structural calculation, work planning, budgeting... - that manage to work collaboratively thanks to their unification in a Federated Model. The Federated Model interprets all the virtual models through IFC, but it is not able to edit them or carry out modifications. They are done in each virtual model.

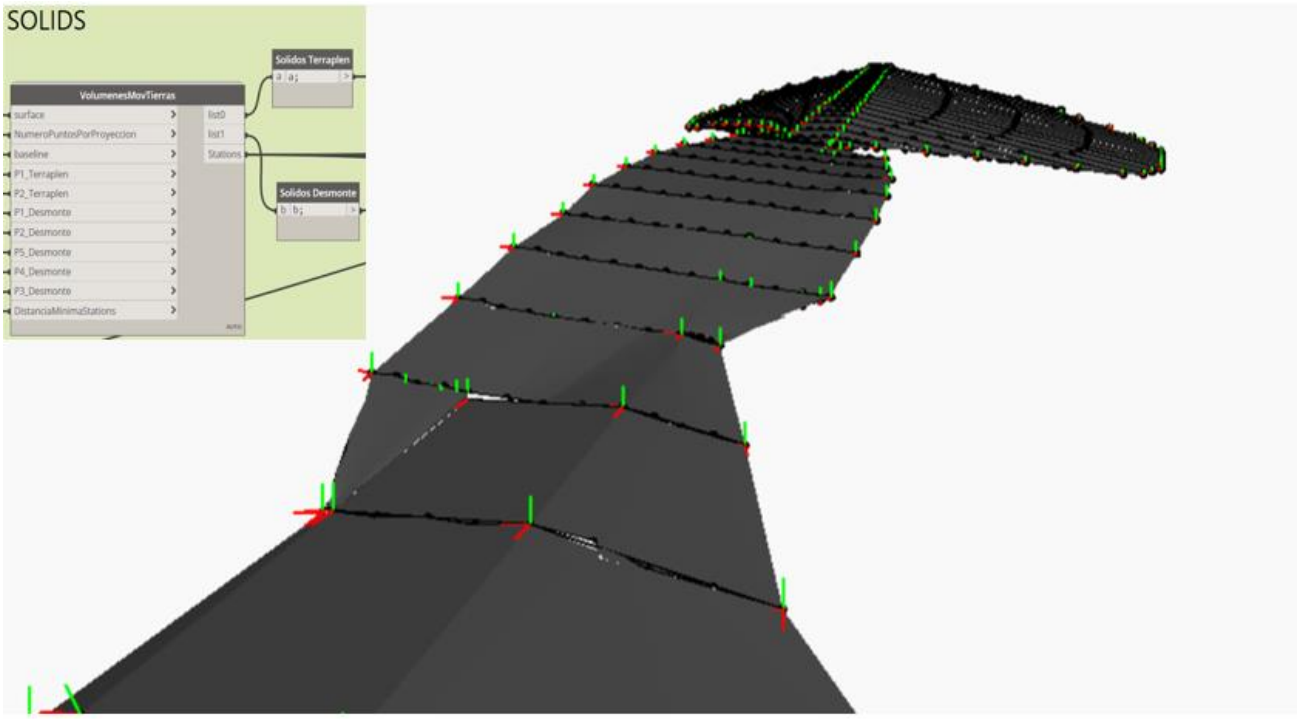
The virtual model, then, must be able to adapt to changes and keep updated during the whole project life cycle -conception, design, bidding, construction and operation- in accordance with the Federated Model. This scenario drives the application of Dynamo in Civil 3D through two case studies.

The first case study analyzes, from the perspective of the construction phase project, the discretization of linear works in sections linked to the work planning and its certifications. Two algorithms are created in Dynamo that allow to segment the corridor and to manage the distribution of the sections along the corridor. The algorithms are based on Civil 3D API.NET programming through Python language in Dynamo.

The second case study deals with the export of Civil 3D earthworks through IFC2.3. It is exemplified the exportation to Presto through Cost-It to assign - dynamically - budget items to the earthworks. It is observed that, to achieve export to IFC 2.3, it is necessary to generate in Civil 3D solids that make up the volume of the earth movement. For this purpose, a parameterized algorithm is created to automatically generate the solids and also exports the measurements to Excel. The development is based on Dynamo's Visual Programming.

However, it represents the situation and interaction of a Visual Programming environment and the development of BIM projects in civil works.

INVESTIGACIÓN DE DYNAMO PARA CIVIL 3D: PROGRAMACIÓN VISUAL Y BIM EN OBRA CIVIL



AUTOR: PABLO EIZAGUIRRE GARCÍA

TUTOR: CÉSAR OTERO GONZÁLEZ

Escuela Técnica Superior de Ingenieros de Caminos, Canales y Puertos || Julio 2020

UNIVERSIDAD DE CANTABRIA

Grupo de I+D EgiCAD

UC UNIVERSIDAD DE CANTABRIA

Índice

TABLA DE ILUSTRACIONES

AGRADECIMIENTOS

Querría agradecer en primer lugar al Grupo de I+D EgiCAD que me ha guiado y servido de inspiración durante el desarrollo del Trabajo Fin de Máster. En especial, a mi tutor César Otero que tan brillantemente logra transmitirme confianza y seguridad. Mi experiencia en el grupo es para mí un ejemplo de las cosas bien hechas.

A la Escuela de Caminos de Santander porque todas las oportunidades que me ha brindado por desarrollarme como persona han sido excepcionales.

A mi madre, mi padre y mi hermano, que me han enseñado a esforzarme, a ser mejor cada día y que tanto me han arropado durante toda la carrera. Sin ellos nada habría sido posible.

A mi pareja porque en ella encuentro una fuente sincera de inspiración creativa, curiosidad y motivación por mejorar el entorno, transmitiéndose a todos los aspectos de mi vida.

Gracias.

1. MOTIVACIÓN

Es conocido por la comunidad de Ingenieros de Caminos, Canales y Puertos, que las herramientas informáticas de diseño, control, cálculo y planificación de obra civil están experimentando avances en el marco de la metodología BIM. El uso del BIM ha comenzado a ser obligado en proyectos de obra civil según la Ley 9/2017, amparada por la Directiva Europea 2014/24/UE. Debido a estas exigencias y a que el sector de la construcción está instaurando progresivamente los medios técnicos y tecnológicos para conseguir su cumplimiento, es de interés conocer la situación actual del BIM en proyectos de obra lineal, su implementación, estandarización y sus flujos de trabajo. Asimismo, la exploración de nuevas herramientas informáticas que faciliten el uso del BIM es requisito para mantenerse actualizado y estar a la vanguardia de su técnica.

Dynamo es un software de programación visual de la casa Autodesk que sirve, desde 2015, como herramienta de soporte en BIM para Revit, el programa de modelado para edificación de Autodesk. En octubre 2019, Dynamo comienza a dar soporte a Civil 3D, el programa de modelado para obra lineal de Autodesk. Con este hito se abre una ventana de trabajo que lleva a investigar y aplicar las posibilidades de *Dynamo for Civil 3D* para el desarrollo BIM de una obra lineal.

Flujos de Trabajo BIM, *Programación Visual*, *Diseño Computacional*, *Computación en Ingeniería (civil), Modelos Virtual y Federado BIM, Parametrizado de los elementos BIM de Ingeniería Civil, BIM 4D* y *BIM 5D* son las Palabras Clave que enmarcan este Trabajo Fin de Máster. A lo largo del documento serán desarrolladas e incorporadas en un ejercicio de desarrollo de Software orientado a la optimización de productividad en la Oficina BIM.

2. OBJETIVOS DEL TFM

Son los siguientes:

- Estudiar y documentar técnicamente el entorno Dynamo para Ingeniería Civil (Civil 3D). Esto se lleva a cabo en el capítulo 3.
- Estudiar y documentar metodológicamente el Marco de Desarrollo BIM en el que se mueve actualmente cualquier Compañía de Ingeniería de un tamaño medio o grande (sin menoscabo de muchas pequeñas ingenierías). Esta labor se hace identificando palabras y términos clave a lo largo del capítulo 4.
- Aplicar la metodología de trabajo Dynamo para Civil 3D, propia del diseño computacional, a dos casos de estudio de plena vigencia e interés profesional: uno relativo a la fase de proyecto (BIM5D) y otro relativo a la planificación en la fase de construcción (BIM4D). Todo ello se lleva a cabo de manera pormenorizada en el capítulo 5.

3. DYNAMO

El uso de herramientas de programación en softwares dedicados al campo de la ingeniería civil requiere de conocimiento especializado en escritura de código. Ante esta dificultad aparecen softwares auxiliares de programación visual como Dynamo (enlazado a Revit y Civil 3D) o Grasshopper (enlazado a Rhinoceros 3D) que tratan de facilitar el desarrollo e implementación de rutinas y algoritmos en aquellos programas a los que están enlazados.

3.1. Programación Visual

La Programación Visual consiste en la generación de algoritmos sin necesidad de escribir código, sino a partir de una conexión lógica de pequeños cuadros denominados nodos. El nodo es la unidad fundamental, en la cual se almacena código. Los nodos son generados y ubicados por defecto en la librería, jerarquizados en categorías, de manera que el flujo de trabajo que resulta es identificar los nodos de interés y unirlos de forma lógica y ordenada.

Los nodos tienen entradas y salidas de datos, mostrando el tipo de dato que necesita en su entrada para funcionar correctamente y el tipo de dato que devuelve el nodo como salida. En la *Figura 1* se muestra el nodo correspondiente a la generación de un punto (dato de salida "output") recibiendo como entrada de datos "inputs" el valor numérico de las coordenadas x, y, z del punto.

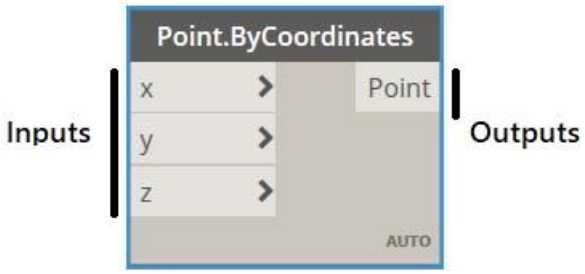


Figura 1. Nodo Point.ByCoordinates de Dynamo

El espacio de trabajo está compuesto por dos componentes. El navegador del gráfico de nodos y el navegador 3D. Mientras uno de los dos esté activo, en primer plano, el otro seguirá viéndose y ejecutándose a tiempo real en segundo plano. Esto permite ver reflejado, como en la Figura 2 el resultado geométrico del algoritmo y observar cómo varía frente a cambios en las entradas de datos o en la estructura del algoritmo (reemplazo de nodos, modificación del orden de ejecución de subrutinas…).

Otro elemento de una interfaz de programación visual es la librería. En ella se encuentran categorizados todos los nodos que el programa incorpora por defecto, reflejando el modelo de datos de Dynamo.

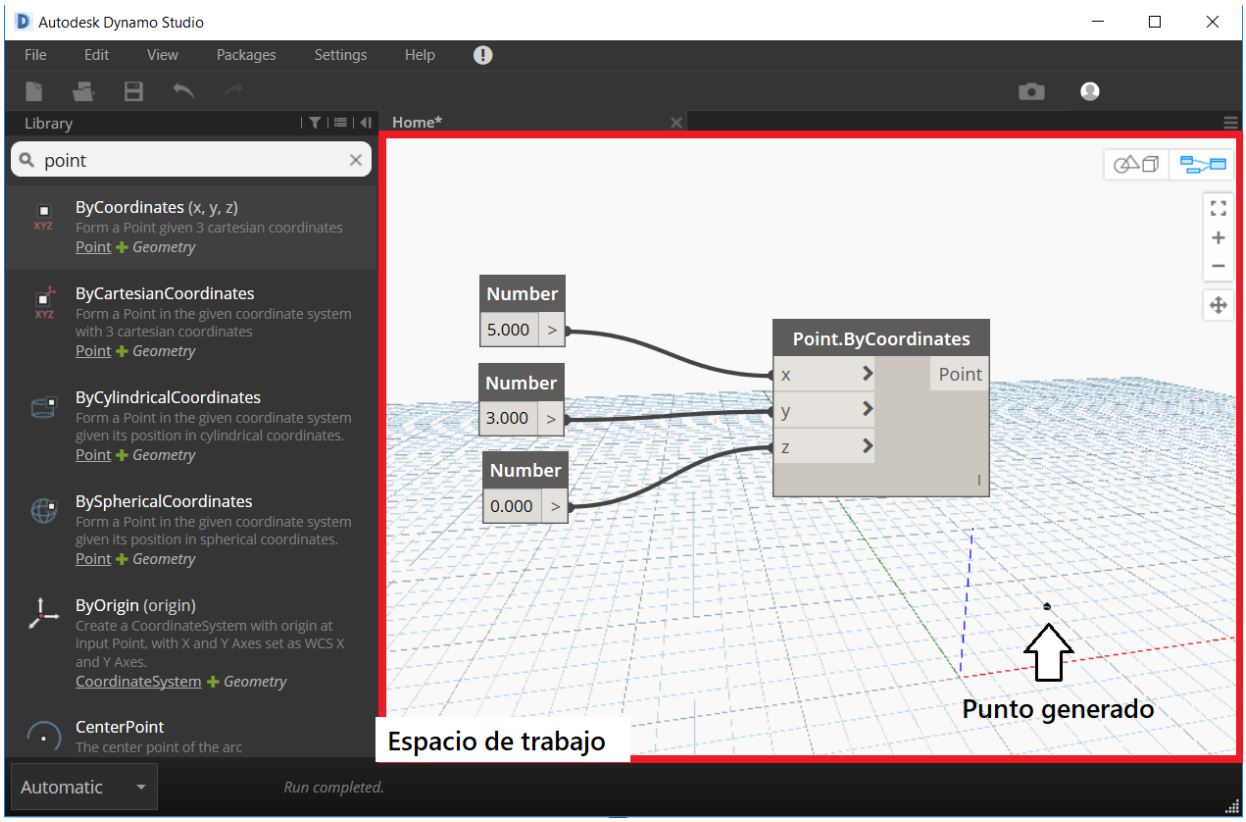


Figura 2. Espacio de trabajo en una interfaz de programación visual

3.2. Dynamo

Dynamo es una aplicación de modelado y diseño computacional desarrollada por Autodesk, capaz de implementarse y trabajar junto a otros softwares de Autodesk, hasta la fecha Revit y Civil 3D. Su característica fundamental es el uso de Programación Visual como método de programación. Cuenta, además, con la posibilidad de programar a través de código, dando lugar a una herramienta intuitiva donde el usuario es libre de trabajar con un gran abanico de posibilidades, creando sinergias entre sus técnicas de codificación.

Soporta los siguientes lenguajes de programación (*Figura 3*):

- *DesignScript*: Es el lenguaje que gobierna Dynamo y en el cual están escritos los nodos por defecto. Escritura en nodo *CodeBlock*. Permite la generación de ciclos, pero no el acceso a librerías externas.
- *Python*: A través del nodo *Python Script* permite una programación más avanzada que el DesignScript, pudiendo generar ciclos y acceder a librerías externas, por ejemplo, el API de Civil 3D o Revit.
- *C#*: Dynamo permite la importación de archivos .dll, en forma de nodos, originados en cualquier editor de código en lenguaje C#. Su mayor limitación es no poder generar ciclos.

	Looping	Recursion	Condense Nodes	Ext. Libraries	Shorthand
DesignScript	Yes	Yes	Yes	No	Yes
Python	Yes	Yes	Partially	Yes	No
ZeroTouch (C#)	No	No	No	Yes	No

Figura 3. Capacidades de los distintos lenguajes de programación soportados por Dynamo [1]

Dynamo no solo está concebido para el diseño computacional. Es capaz de gestionar datos exportando/importando a Excel y otros formatos. Su potencial reside en su adaptabilidad y versatilidad.

Existe la posibilidad de que el usuario cree sus propios nodos o use nodos creados por otros usuarios de la comunidad Dynamo en forma de paquetes descargables gratuitamente. Los paquetes amplían las posibilidades de uso de Dynamo. Estos serían ejemplos de paquetes:

- *Mesh Toolkit*: Paquete para trabajar con mallas 3D. [2]
- *Project Refinery*: Consigue a partir de diseño generativo, optimizar geometrías mediante el análisis de simulaciones con unos parámetros de diseño dados. Por ejemplo, maximizar la sección de un túnel minimizando el material a usar. [3] [4]
- *CivilConnection*: Permite a Dynamo, lanzado desde su extensión de Revit, enlazarse a la vez con Civil 3D. Dedicado a unir las fortalezas de Revit y Civil 3D para incrementar su interoperabilidad y permitir un flujo ágil de trabajos en metodología BIM. [5]
- *Raindrops*: Enlaza a Dynamo con la plataforma Google Drive

3.3. Dynamo en Revit, Dynamo en C3D

Dynamo trabaja colaborativamente con otros entornos Autodesk. En primer lugar, se enlazó con Revit 2015, siendo esta la primera versión en tener una extensión integrada de Dynamo. Revit es un software de modelado 3D usado en disciplinas como la edificación. Progresivamente, las funcionalidades de Dynamo en Revit han ido evolucionando. En octubre de 2019 Civil 3D 2020, el software de modelado 3D para obras lineales, presenta de forma oficial su primera extensión de Dynamo.

La particularidad de ejecutar Dynamo como extensión de otro entorno, por ejemplo, Civil 3D, es que a la librería de nodos por defecto -de Dynamo- se la incluye una nueva categoría de nodos dedicados al programa con el que se enlaza.

3.3.1. Dynamo en Revit

Revit lleva integrando su extensión para programar rutinas y macros con Dynamo desde 2015. La versión más reciente, Revit 2020, cuenta con una extensa librería en Dynamo dedicada a funciones y operaciones propias de Revit.

De entre las funciones que Dynamo en Revit 2020 incorpora están:

- Selección de elementos Revit y su visualización en Dynamo permitiendo modificar parámetros de diseño de elementos adaptativos estudiándolos de manera aislada.
- Creación de estructuras a partir de elementos adaptativos Revit.
- Acceso a todo el rango jerárquico de elementos Revit (categorías, familias, tipos y ejemplares) o en inglés (*categories, families, types* and *instances*).
- Apoyo en la documentación y exposición del trabajo realizado, a través de la importación/exportación a Excel y GoogleDrive (*paquete Raindrop*), o generando mapas de calor en función de parámetros geométricos.
- Enlazar el modelo Revit con un modelo Civil 3D para el desarrollo de un modelo de obra lineal de forma conjunta (*paquete CivilConnection)*.
- Empleo de otros paquetes de Dynamo.

Para el interés de este Trabajo Fin de Máster, las funciones más importantes son aquellas derivadas de la gestión de datos enfocando a Dynamo como herramienta gestión de los mismos y su función conectora de Revit y Civil 3D.

La interconexión entre Revit y Civil 3D supone un avance en interoperabilidad y agilidad en el desarrollo de modelos en metodología BIM, aprovecha las ventajas de cada uno de los programas creando sinergias. Las ventajas de Revit se encuentran en el diseño estructural por medio de elementos discretos (muros, elementos prefabricados…) mientras que Civil 3D modela obras lineales (carreteras, canales…). En este sentido, Dynamo cobra el papel de intermediario y se convierte en el centro de operaciones.

3.3.2. Dynamo en Civil 3D

En el transcurso de este trabajo, tuvo lugar un esperado acontecimiento por la comunidad de desarrolladores de Dynamo y BIM para obra lineal. Se lanzó en octubre de 2019 la extensión de *Dynamo for Civil 3D* compatible con Civil 3D 2020.

Hasta la fecha, se habían explorado alternativas como el uso del paquete CivilConnection desde Revit (la primera versión trata 2017) o la modelización de una obra lineal desde Revit a partir de sólidos extraídos de Civil 3D (operación costosa y con muchas limitaciones).

La librería de nodos en *Dynamo for Civil 3D* no es tan extensa ni completa como la de Revit. La razón es que la versión de Revit lleva años actualizándose con nuevo contenido de forma periódica. Por ahora las funciones que permiten realizar los nodos son suficientes como para generar operaciones geométricas derivadas de elementos de la obra lineal engendrada por el modelo, por ejemplo, el caso expuesto en el capítulo 5.2 (Figura 4)

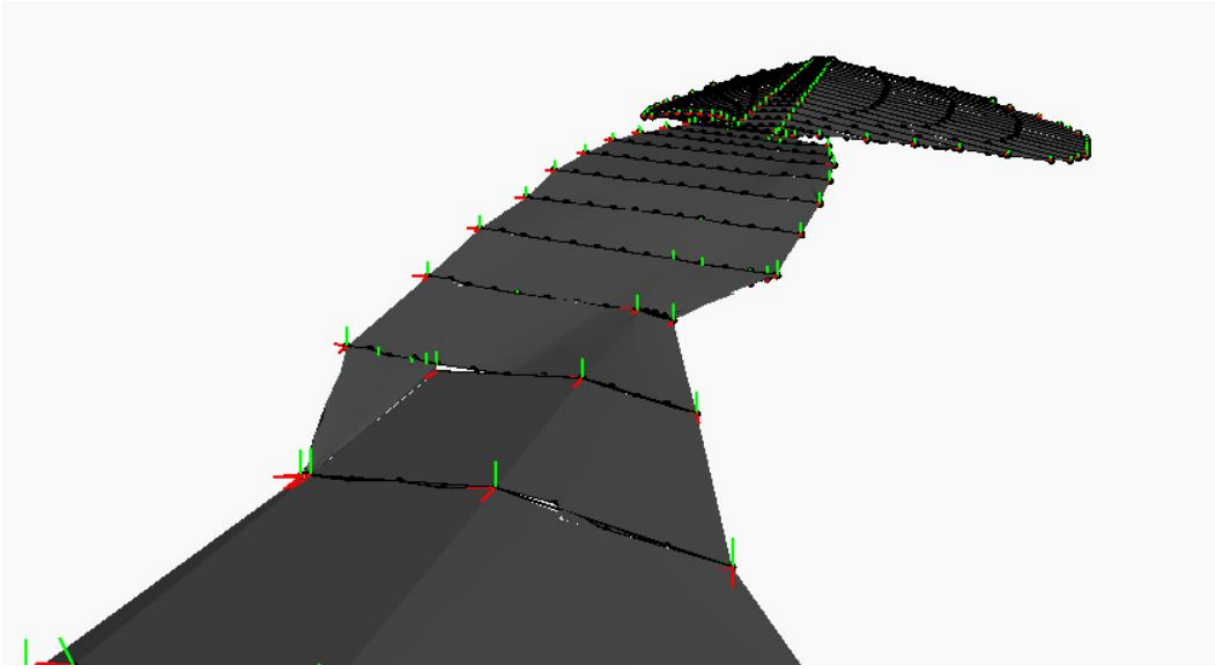


Figura 4. Modelado de una obra lineal y sus movimientos de tierras en Dynamo for Civil 3D

Un resumen de las funcionalidades que *posee Dynamo for Civil 3D* a la hora de programar rutinas es:

- Selección, visualización y obtención de parámetros de elementos Civil 3D (alineaciones, líneas características, superficies…).
- Creación de elementos geométricos sólidos en el modelo Civil 3D parametrizados y referenciados con datos del modelo (muros, movimientos de tierras, barreras de contención…)
- Obtención del volumen de las capas de firme en forma de sólidos.
- Empleo de los diferentes paquetes descargables de Dynamo.
- Programación en el Civil 3D API.NET a través de Python:
    - Creación y modificación de elementos Civil 3D: obras lineales, superficies, alineaciones, ensamblajes (sección tipo), regiones, etc.

En este caso, con Civil 3D, la comunidad de desarrolladores en Dynamo no ha contribuido tanto como en Revit para la creación de paquetes dedicados específicamente a *Dynamo for Civil 3D*.

La única manera de realizar procedimientos fuera del alcance de los nodos por defecto es a partir de la programación en Python. El nodo de programación en Python (nodo *Python Script*) tiene la capacidad de importar librerías y programar sin ninguna clase de restricción. Por eso es capaz de importar las librerías del API.NET de Civil 3D y emplear todas las funciones, propiedades y procedimientos que posee.

CivilConnection posee las mismas características que *Dynamo for Civil 3D*, añadiendo además la interoperabilidad directa con Revit. Sin embargo, su instalación y preparación son más complejas y salvo que se desee trabajar conjuntamente con Revit, por sencillez y porque es una herramienta oficial de Autodesk, es preferible trabajar desde *Dynamo for Civil 3D*.

3.4. Dynamo y el modelado Computacional Civil 3D

El desarrollo de aplicaciones o rutinas personalizadas en programas como AutoCAD o Civil 3D, requiere de técnicas de transmisión de información entre el desarrollador y el motor del programa. Para lograrlo existe la interfaz de programación de aplicaciones denominada API. Ofrece una serie de subrutinas, funciones y procedimientos relacionados con el programa al que da soporte, en este caso del Civil 3D, para su uso en otras aplicaciones como Dynamo.

Dynamo está limitado al uso de los nodos que almacena la librería. Para conseguir programar sin esa limitación es necesario desenvolverse a través del API y conocer el modelo de datos de Civil 3D, sus clases y jerarquías.

3.4.1. Modelo de datos en Civil 3D

Un API permite comprender cómo se estructuran y jerarquizan los espacios de trabajo, las clases y, en general, todos los datos de los que depende un programa. A la hora de programar, se requiere conocer previamente la estructura de datos con el fin de reconocer el flujo de información que siguen los distintos elementos del programa y tener capacidad de manipularlos.

Se analizan las entidades y elementos propios de la generación de obras lineales. Para definir tales entidades la *Figura 5* muestra su flujo de trabajo. A partir de una alineación, se define el trazado, ésta se empareja con un perfil longitudinal, que marca la elevación, y una sección tipo que determina la tipología y configuración de la obra lineal. La entidad superficie de Civil 3D representa la superficie altimétrica referida, habitualmente, al terreno a través de modelos cartográficos y sirve como referencia durante el proceso de realización del perfil longitudinal y como objetivo *target* de los taludes en los posibles movimientos de tierras de la obra lineal. La generación de la obra lineal crea automáticamente líneas características y líneas base, propias de la obra lineal.

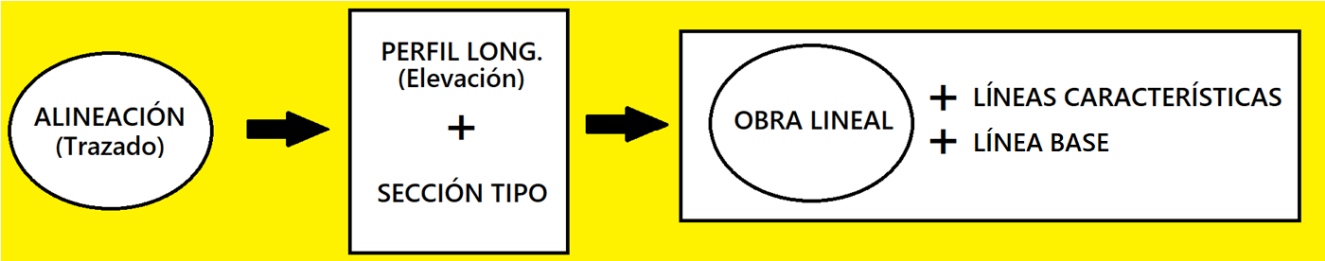


Figura 5. Flujo de trabajo habitual para la generación de una obra lineal

Este es el único flujo de trabajo que pueden llevar a cabo usuarios no programadores, pues la interfaz de usuario aporta las herramientas necesarias para llevarlo a cabo automáticamente. Sin embargo, mediante programación es posible generar nuevos flujos de trabajo.

En lenguaje de programación todos los términos emplean su traducción al inglés:

Alineación = ***Alignment*** || Perfil Longitudinal = ***Profile*** || Sección tipo (ensamblaje) = ***Assembly***

Obra lineal = ***Corridor*** || Líneas características = ***Featurelines*** || Línea base = ***Baseline***

Superficie = ***Surface*** || Subensamblaje = Subassembly

Definidos los elementos fundamentales de la generación de obras lineales, se analiza cómo el modelo de datos los estructura, plasmádose en el API.NET donde aparece su contenido. Se ha determinado la estructura de datos de la *Figura 6* atendiendo a la especificación funcional -métodos y propiedades- de las entidades.

El flujo comienza con la llamada al documento activo de Civil 3D, la clase *CivilDocument*, que posee la propiedad de seleccionar todas las entidades, según su clase, contenidas en el documento. Sin embargo, no recoge directamente las entidades, sino que recoge sus *ObjectID,* es decir, recoge el identificador de cada uno. El identificador o *ObjectID* es un número aleatorio con el que se referencia a una única entidad existente en el documento. Para invocar a la entidad, por ejemplo un *corridor,* es necesario establecer una transacción con la base de datos del documento en la cual se incluya el *ObjectID* de la entidad.

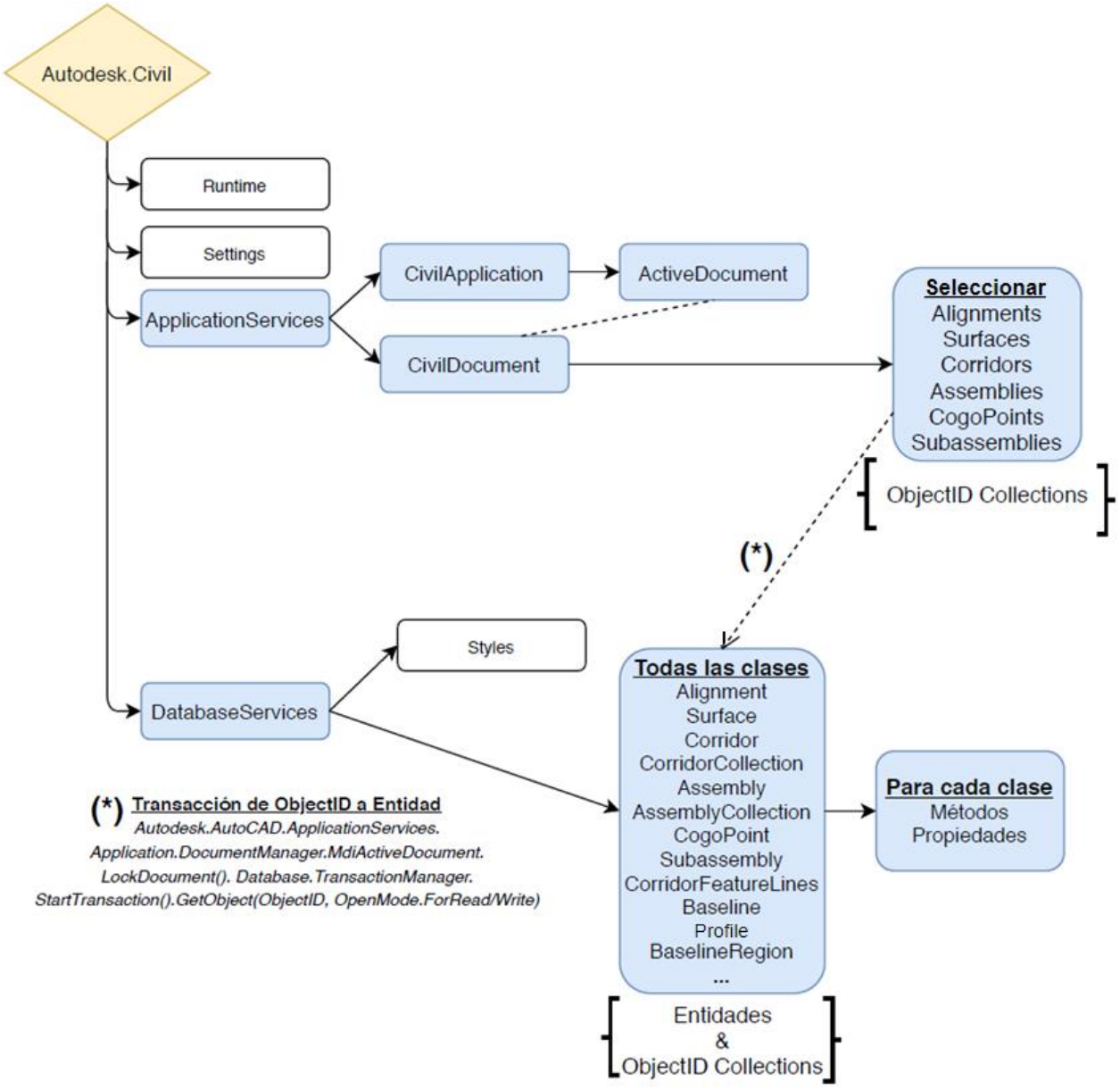


Figura 6. Resumen del modelo de datos de Civil 3D

3.4.2. API.NET: Librería C#, VB.NET y C++

Para permitir el acceso al modelo de datos de Civil 3D desde editores de código y que programar en Civil 3D sea posible, existe el API.

El API funciona como intermediario en la transmisión de información entre el editor de código y Civil 3D. Es una herramienta común en toda clase de softwares, no pertenece exclusivamente a Autodesk, sin embargo, cada software que lo emplea tiene su propia librería con sus funciones, sus clases y sus "normas". Por ello al referirse al API, se está refiriendo en realidad a la librería API

de Civil 3D. Como el API de Civil 3D está construido sobre .NET (plataforma o *framework* de Microsoft, como alternativa a Java, para desarrollo de aplicaciones, soportado en C#, C++ y VB.NET) se denomina finalmente API.NET. Así que la manera de programar en Civil 3D es importando al editor de código externo la librería del API.NET de Civil 3D junto con una serie de módulos que posee la instalación de Civil 3D (*AcDbMgd* por ejemplo) y permiten verificar y trabajar con la versión de Civil 3D instalada en la computadora.

El soporte de Civil 3D incluye en su ayuda para desarrolladores una guía de referencia para programar con el API .NET. Proporciona un listado de las subrutinas, funciones y procedimientos que posee el API en lenguaje Visual Basic .NET, C# y C++, incluyendo información sobre las clases del modelo de datos de Civil 3D, sus propiedades y métodos, además del módulo que debe estar importado en el editor de código para que cada procedimiento funcione.

Todo editor de código que soporte un lenguaje compatible con la plataforma .NET es susceptible de usarse para programar con el API.NET de Civil 3D. Además de los ya citados VB.NET C# y C++, existen adaptaciones de otros lenguajes para operar en .NET. En el caso de Python existe su implementación .NET llamada IronPython.

3.4.3. PYTHON: Nodos Dynamo y metodología de creación

En el momento en que programar con los nodos Dynamo por defecto queda demasiado limitado se plantea el uso del nodo de escritura en Python, llamado *Python Script* (ver Figura 7). El lenguaje con el que opera este nodo es concretamente el IronPython 2.7, la implementación .NET de Python, haciendo posible la comunicación entre un nodo Dynamo y el API.NET de Civil 3D.

Ha sido necesario el aprendizaje del lenguaje Python, adaptado a la programación en el API.NET de Civil 3D. El contenido del API es recogido en lenguaje VB.NET y C# por el soporte para desarrolladores de Autodesk, teniendo así que interpretar estos lenguajes y adaptarlos al Python.

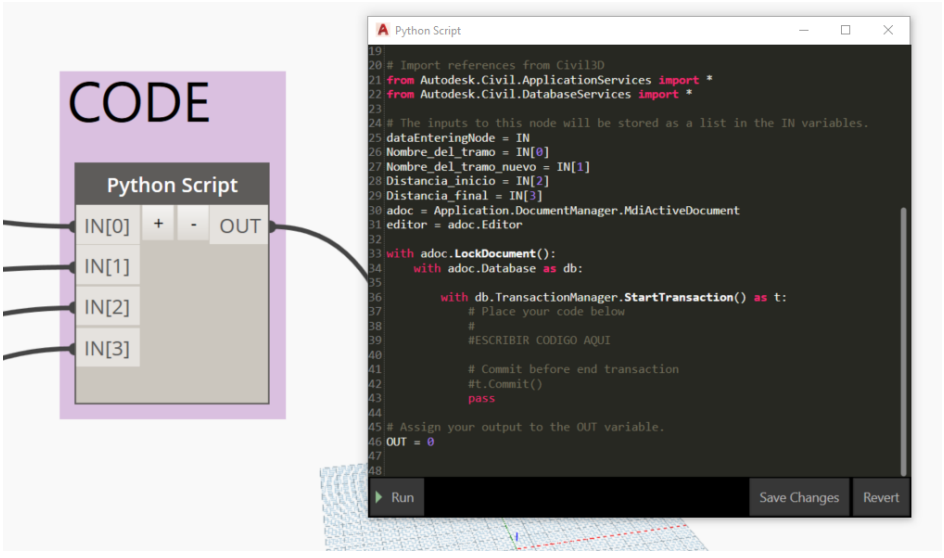


Figura 7. Nodo Python Script

4. BUILDING INFORMATION MODELING (BIM)

4.1. El ciclo de vida de la obra civil y el modelo Federado BIM

Citando la interpretación de la ISO 19650 [6] ***Building Information Modelling (BIM) es el uso de una representación digital compartida (modelo de información) de un activo construido para facilitar los procesos de diseño, construcción y operación, y proporcionar una base fiable para la toma de decisiones***.

Respecto a esta definición, se recalcan dos apreciaciones. La primera, referente a la representación digital, de un activo, que deberá haber sido definida bajo unos protocolos de desarrollo claros y comunes, en conocimiento de todos los agentes actuantes del proyecto (cliente, propietario, contratante principal y diferentes partes contratantes). La segunda, a que esa representación digital tendrá que evolucionar y adaptarse a las diferentes fases de proyecto durante su ciclo de vida. Ambas apreciaciones están conectadas, pues en cada fase de proyecto intervendrán distintos agentes con distintas competencias y necesidades del modelo.

En [6] se define el modelo de información como: ***un conjunto formado por información estructurada (modelos geométricos, propiedades y atributos, programaciones, etc.) e información no estructurada (documentos, imágenes, videoclips, etc.) que facilita la toma de decisiones. [...] EL modelo de información podrá componerse de un conjunto de modelos propios de cada disciplina o proyectos parciales (arquitectura, estructura, instalaciones, etc.) organizados para que puedan ser federados de forma apropiada y facilitar la colaboración durante el desarrollo del proyecto.***

El flujo de trabajo que gobierna el Modelo Colaborativo o Federado (ver ***Figura 8)*** se fundamenta en la interconexión mediante un software (por ejemplo, Autodesk Navisworks o Autodesk Infraworks) que sirva como plataforma colaborativa en la que todas las disciplinas pueden realizar exportaciones de sus modelos. El Modelo Federado interpreta de forma unificada los modelos, facilitando su comunicación y presentación. Estos softwares están limitados a la representación de los modelos, es decir, no pueden realizar modificaciones o tareas de edición del modelo. Este hecho obliga a tener un flujo de trabajo unidireccional, obligando a los softwares de cada disciplina a llevar a cabo toda la labor de definición y edición del modelo virtual.

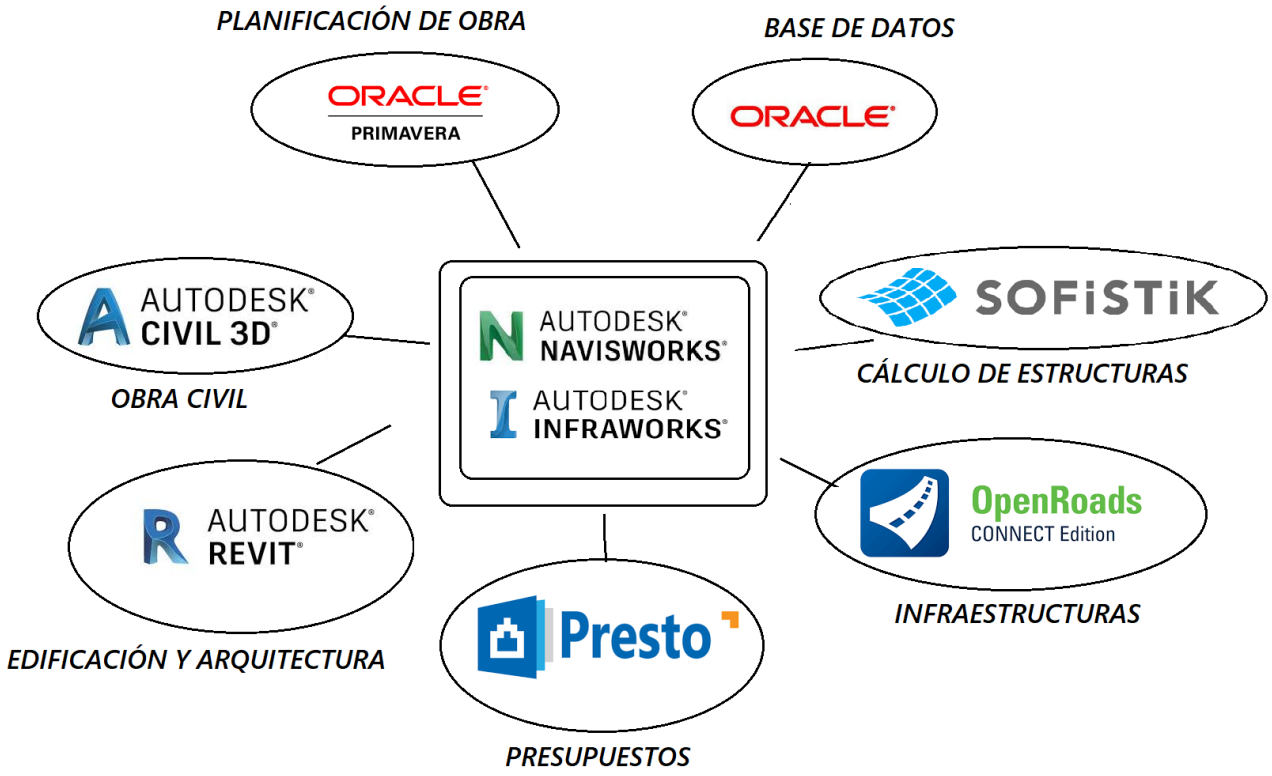


Figura 8. Ejemplo de un Modelo Federado

Con el fin de medir la calidad y el coste del modelo de información se definen tres niveles de detalle del modelo:

- Level of Detail (LOD): Nivel de detalle en la representación geométrica.
- Level of Information (LOI): Nivel de la información con la que cuentan los elementos del modelo.
- Level of Accuracy (LOA): Nivel de precisión o sensibilidad del modelo.

En las fases preliminares del proyecto estas categorías tienen niveles reducidos y alcanzan valor suficiente al concluir la fase de diseño detallado. El modelo se actualiza a medida que transcurren las posteriores fases de proyecto, admitiendo correcciones que pudieran aparecer en fases de construcción y operación debido a errores en mediciones. Las correcciones permiten tener una información aún más detallada en el modelo. No obstante, no siempre será fructífero trabajar con el modelo en alto detalle, pues en función del agente que lo utilice habrá información que no aporte valor e impida un manejo fluido del modelo.

En la *Figura 9* se muestra la distribución de esfuerzos a lo largo del ciclo de vida de un proyecto constructivo, comparándose un proyecto BIM con un proyecto convencional. El planteamiento de la curva de MacLeamy [7] es que, gracias a generar una representación digital del activo a construir, se acometen con mayor brevedad aspectos de diseño que de otra forma no es posible abordar

hasta fases posteriores, materializándose en una reducción del impacto económico de los susceptibles cambios en el diseño.

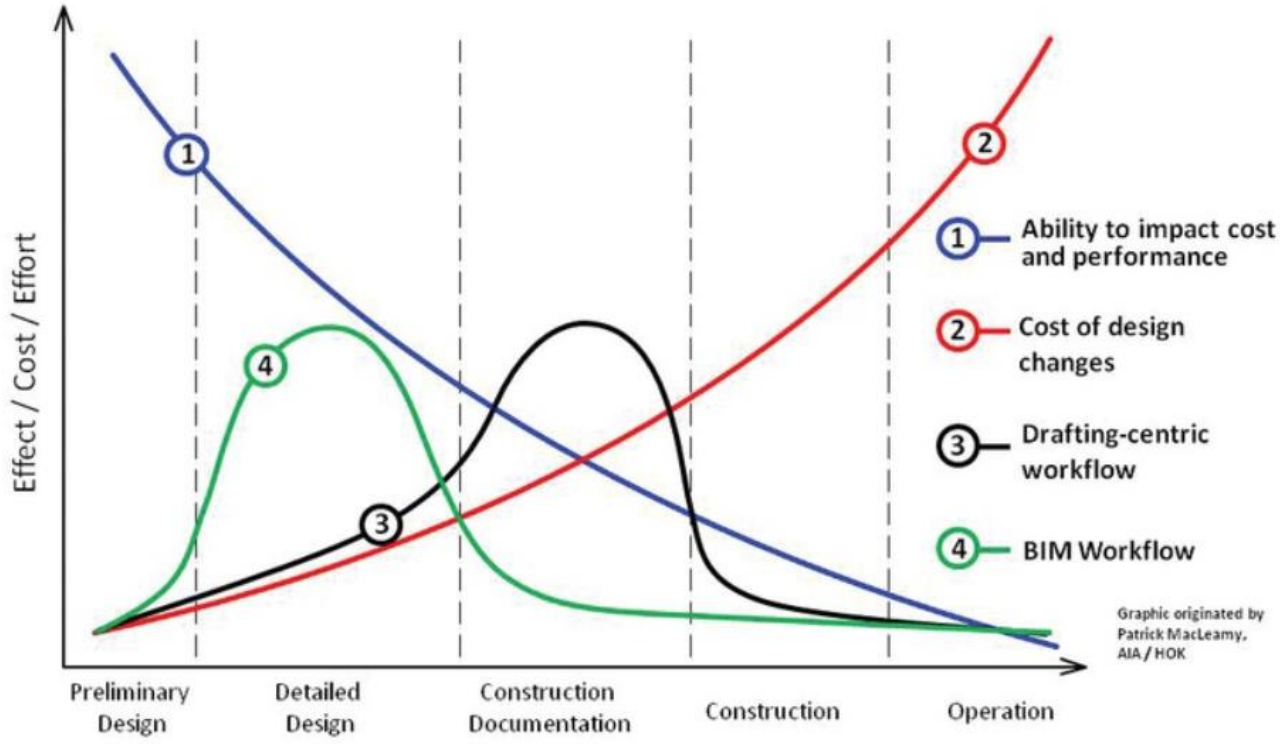


Figura 9. Curva de MacLeamy [7]

4.2. Los estándares ISO para el intercambio de modelos BIM

El proceso de generación de una representación digital colaborativa supone un reto en cuanto a definir los protocolos de transmisión y creación de información comunes y estandarizados para todos los agentes implicados en el uso de un modelo. En proyectos multidisciplinares como a los que se enfrenta la ingeniería civil existe dificultad, y cobra aún más importancia, conseguir la interoperabilidad de todas aquellas plataformas específicas de cada disciplina que interviene en el modelo. La manera en que se está solucionando esta cuestión es a través de normas ISO, en colaboración con buildingSMART, que normalicen protocolos en el procesamiento de datos con el fin de que los fabricantes de software para ingeniería desarrollen programas y aplicaciones interoperativas con el resto generando un entorno colaborativo de todas las disciplinas (Modelo Federado).

La aplicación de estándares no termina en la definición de los procesos informáticos, sino que toda la gestión y planificación de recursos (humanos, intelectuales y tecnológicos) sigue una normativa basada en cinco principios básicos ya estandarizados:

- Descripción de los procesos de intercambio de información.
- Hacer posible los intercambios.
- Gestionar los cambios de coordinación del uso del modelo.

- Definir toda la terminología que recoge el modelo.
- Interpretar los requisitos técnicos de los procesos (geometría, costes, comportamientos estructurales…).

Sus normativas asociadas aparecen en la *Figura 10* junto con los formatos de archivo que se emplean para llevarse a cabo. A esa tabla creada por [8] se la ha sumado la reciente aprobación de la EN-ISO 19650.

Formato	Norma	Función
IDM *Manual de Entrega de Información*	ISO 29481-1 ISO 29481-2	Descripción de los procesos
IFC *Industry Foundation Class*	ISO 16739	Intercambio de información y datos
BCF *Formato de Colaboración BIM*	buildingSMART BCF	Gestión de cambios de la coordinación
IFD *Marco Internacional de Diccionarios*	ISO 12006-3 buildingSMART Data Dictionary	Recoger la terminología
MVD *Definiciones de la Vista del Modelo*	buildingSMART MVD	Interpretar requisitos técnicos
	ISO 19650	Definición del uso de la información

Figura 10. Estándares básicos de la metodología BIM [9] [8]

La EN-ISO 19650 es [6] *un conjunto de normas internacionales que definen el marco, los principios, y los requisitos, para la adquisición, uso y gestión de la información en proyectos y activos, tanto en edificación como de ingeniería civil, a lo largo de todo el ciclo de vida de los mismo.* Por el momento se han aprobado las dos primeras partes de la norma, la ISO 19650-1 que establece los conceptos y principios recomendados para el desarrollo y gestión de la información a lo largo del ciclo de vida del activo, y la EN-ISO 19650-2 que define los procesos de desarrollo y gestión de la información durante la fase de desarrollo (diseño, construcción y puesta en servicio) [6]. La parte tercera lo estudiará para la fase de operación (operación y mantenimiento), mientras que la quinta parte, la última de la que se tiene información, establece los requisitos de seguridad. Se establecen con esta norma ISO las pautas, fases y gestiones a realizar para abordar un proyecto BIM, es decir, define la hoja de ruta para su desarrollo.

4.3. Industry Foundation Classes (IFC) y sus versiones: IFC 2.3, IFC 4.0

La organización buildingSMART trabaja como colaboradora de ISO en la generación de estándares en BIM. Fue fundada en 1994 como un consorcio de compañías sin ánimo de lucro llamado por entonces *International Alliance for Interoperatibility.* Las compañías eran invitadas por Autodesk con motivo de mejorar el intercambio de datos entre los softwares empleados en el sector de la construcción. Para ello desarrollaron el término IFC *Industrial Foundation Class* que es una especificación funcional utilizable por los softwares de ingeniería civil y que permite describir semánticamente, detallar geométricamente y compartir modelos procedentes de distintos softwares, de forma colaborativa. Cada clase de

elemento en la ingeniería civil posee, o poseerá, su definición en IFC, formando una base de datos común, abierta y estandarizada para todo el sector de la construcción.

Para incluir nuevas funcionalidades en las clases que el IFC recoge es necesario crear nuevas versiones del formato, ya que una versión actualizada del IFC implicará cambios en el resto de los aspectos del modelo virtual, concretamente en el MVD (Model View Definition). Por esa razón existen las siguientes versiones de IFC:

- IFC 2.3: Publicada en 2006, recoge una amplia base de datos referentes al sector de la edificación, desde elementos estructurales, arquitectónicos hasta fontanería, electricidad y control del edificio. Ha sido el formato más popular y sigue en uso debido a que permite la coordinación geométrica y espacial, incluyendo atributos y comentarios en los elementos, en proyectos de edificación en BIM.
- IFC 4: Recoge mejoras de la versión 2.3 y amplía los flujos de trabajo posibles permitiendo la interoperabilidad con software de planificación de obra (BIM 4D) y con softwares de presupuestos (BIM 5D). Incorpora los requisitos definidos por los estándares ISO.
    - IFC 4.1: Incorpora funciones básicas para las distintas extensiones del dominio de las infraestructuras que están actualmente en fase de desarrollo: carreteras, túneles, líneas de ferrocarril, puertos y canales.
    - IFC 4.2: Incorpora a IFC 4.1 la extensión de puentes
    - IFC 4.3: El propósito de la versión es describir el conjunto de elementos que cubren los distintos dominios de las infraestructuras (continuación de IFC 4.1) y de aquellos elementos comunes a varios dominios.

En definitiva, IFC se ha usado de forma exitosa en el dominio de la edificación, sin embargo, el conjunto de clases IFC que forman el dominio de las infraestructuras está aún en desarrollo. Para el desarrollo del dominio de las infraestructuras buildingSMART genera proyectos de desarrollo dirigidos por expertos de cara a englobar todos aquellos elementos relacionados con cada categoría (IFC Road - carreteras, IFC Rail – ferrocarriles, IFC Bridge – puentes y IFC Tunnel - túneles).

El proyecto de desarrollo de IFC Road [10] está compuesto por equipos de la administración pública de transportes de Alemania, Suecia y Finlandia además de expertos de la industria, con el objetivo de definir los estándares digitales en generación de modelos de carreteras. El objetivo es que a final de 2020 pueda estar listo para implementación. La Figura 11 muestra cómo se generaría una carretera con IFC Road.

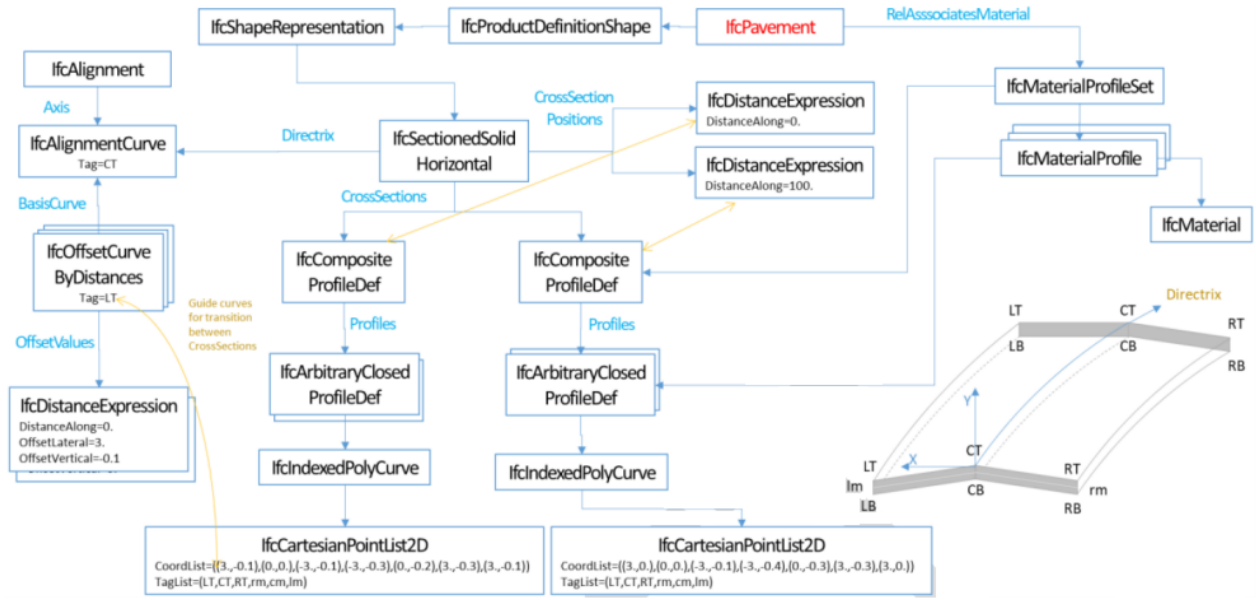


Figura 11. Ejemplo de un flujo de datos para generar una carretera mediante IFC (IFC Road -en desarrollo-) [11]

Por otro lado, el proyecto de desarrollo de IFC Rail [12] cuenta con el apoyo de instituciones públicas de transporte europeas e instituciones chinas. El objetivo común es habilitar mejores flujos de trabajo digital y desarrollar un estándar abierto e internacional para toda la industria del ferrocarril. Continúa en fase de desarrollo.

Finalmente, IFC Bridge [13] se centró en estandarizar el proceso de desarrollo y la estructura organizativa de un modelo de datos de las principales tipologías de puente (puente de losa, puente de vigas, puente de viga-cajón y puente de marco). El desarrollo se completó en abril 2019, participaron equipos procedentes de Alemania, Francia, Finlandia, EEUU y China, bajo la guía de buildingSMART.

4.4. Flujos de trabajo BIM y relevancia de Dynamo en la metodología BIM

A los efectos de este Trabajo Fin de Máster, el flujo de trabajo BIM se basa en la transmisión unidireccional de datos de un modelo virtual al Modelo Federado que conecta y unifica modelos virtuales de softwares de todas las disciplinas (ver Figura 8). El Modelo Federado no tiene capacidad de editar el modelo virtual, tan solo lo interpreta.

De cara a proyectos de obra lineal modelados en Civil 3D, la conclusión es que para trabajar en BIM se ha de asegurar la correcta exportación del modelo Civil 3D mediante IFC y se ha de conseguir un modelo virtual capaz de adaptarse a cambios de diseño de forma flexible y ágil. Estas necesidades originan un creciente interés en el uso de herramientas de programación como Dynamo.

Con el objetivo de estudiar la capacidad que tiene Dynamo para mejorar las prestaciones del entorno Civil 3D en el contexto de un flujo de trabajo BIM, se plantean dos casos de estudio que requieren tiempo de trabajo manual de ingeniería y que a partir de programación de algoritmos se pueden realizar automáticamente.

5. APLICACIÓN DE *DYNAMO FOR CIVIL 3D* EN UN CONTEXTO BIM.

La programación de rutinas es el mejor soporte de cara a afrontar tareas repetitivas y/o complejas. Como *Dynamo for Civil 3D* es un módulo de programación nuevo, no existe apenas documentación dedicada a estudiar su aplicación en un contexto de proyecto BIM real.

Para analizar y reconocer la viabilidad de uso de Dynamo en un flujo de trabajo BIM se plantean dos casos de estudio adaptables a procesos reales de modelado BIM en Civil 3D. El procedimiento seguido para evaluar la capacidad de Dynamo, en cada caso de estudio, se compone de:

- Análisis de un problema resoluble manualmente
- Resolución mediante programación en Dynamo
- Estudio la estructura de los datos que intervienen en el proceso de resolución.
- Metodología de trabajo seguido para el desarrollo de los algoritmos
- Especificación mejoras que ha tomado el algoritmo durante su desarrollo
- Definición del algoritmo completo y sus procedimientos
- Valoración técnica y económica del algoritmo.

Los casos de estudio son:

- Edición y división de obras lineales para su control y planificación en obra.
- Exportación del movimiento de tierras mediante IFC.

5.1. Caso de estudio I: Edición de obras lineales para su control y planificación en obra

5.1.1. Fase I. Análisis

El modelado de una carretera está fundamentado en ser ágil y óptimo en su diseño, permitiendo modelar carreteras de decenas de kilómetros como una sola entidad de obra lineal. Sin embargo, en fase de construcción las necesidades del proyecto cambian, y tener entidades de obra lineal demasiado grandes puede resultar ineficiente en tareas como el control de la certificación de mediciones o la planificación de obra.

Como consecuencia del flujo de trabajo BIM, el modelo virtual tiene que estar preparado para soportar modificaciones de diseño a lo largo de todo el ciclo de vida de la obra. Una de las modificaciones que deben tenerse en cuenta son los cambios de planificación durante la fase de construcción.

De cara a adaptar el modelo a la planificación de obra resulta cómodo y accesible discretizar el modelo en tramos independientes de los que se puede conocer de forma automática las propiedades geométricas. De este modo se estructura

Consecuentemente, se plantean los siguientes condicionantes:

- Condicionante 1: Existe interés en discretizar una obra lineal en tramos
- Condicionante 2: Cada tramo, o conjunto de tramos simultáneos, corresponde a la planificación mensual que debe seguir la ejecución de obra
- Condicionante 3: La planificación de obra es cambiante

Para resolver los condicionantes se pueden emplear procedimientos manuales de creación y edición de obras lineales. Supone una tarea repetitiva en la que es sencillo que se produzcan errores humanos.

5.1.2. Fase II. Resolución con Dynamo

Teniendo en cuenta los condicionantes del apartado 5.1.1 se generan dos algoritmos en Dynamo capaces de resolver la problemática de forma automática, ante cualquier escenario.

El primer algoritmo es el llamado "Creación de tramo de obra lineal", que genera una entidad de obra lineal nueva con la misma geométricas, pero distinta distribución de regiones, que otra obra lineal ya existente. A la nueva obra lineal generada se la denomina también por el término tramo. El algoritmo recibe como entrada de datos (ver Figura 12) el nombre de la obra lineal ya existente, el nombre con el que se desea generar la nueva obra lineal, y el rango, es decir, el PK (punto kilométrico) inicial y PK final, respecto de la obra lineal ya existente, en el que se generará el tramo. De esta forma se resuelven los condicionantes 1 y 2.

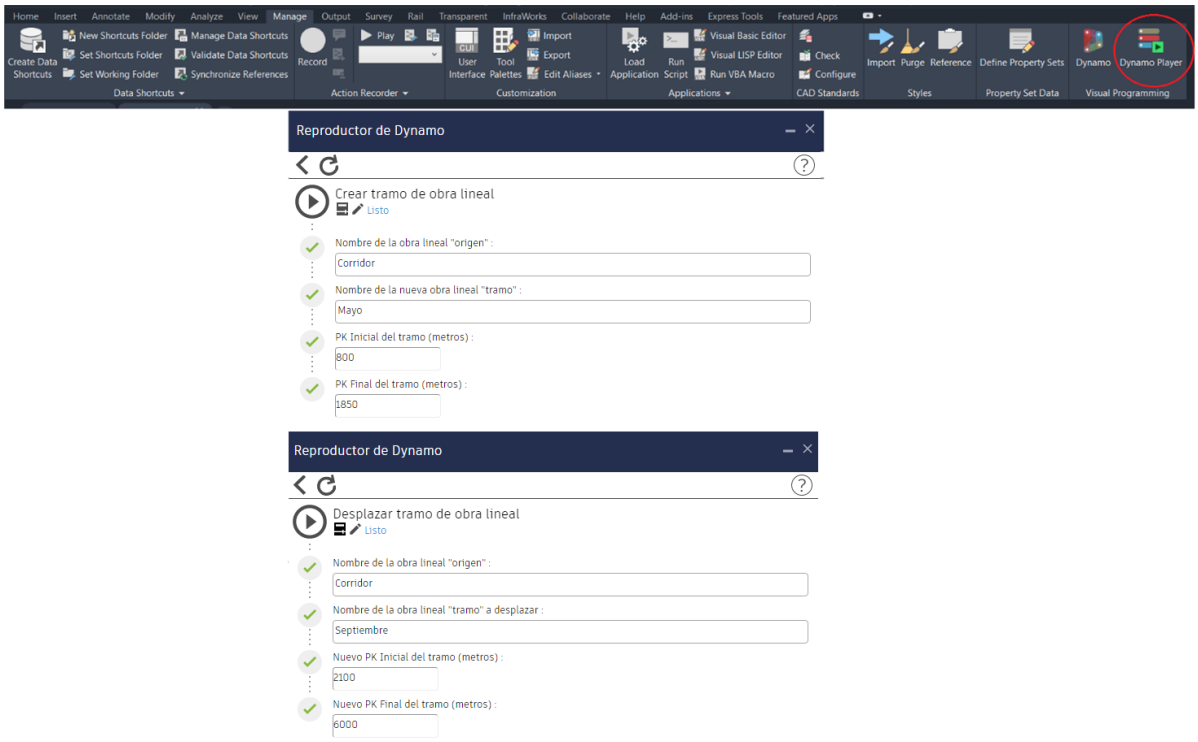


Figura 12. Ejemplo de entrada de datos de los algoritmos "Crear tramo de obra lineal" y "Desplazar tramo de obra lineal" visto desde el Reproductor Dynamo.

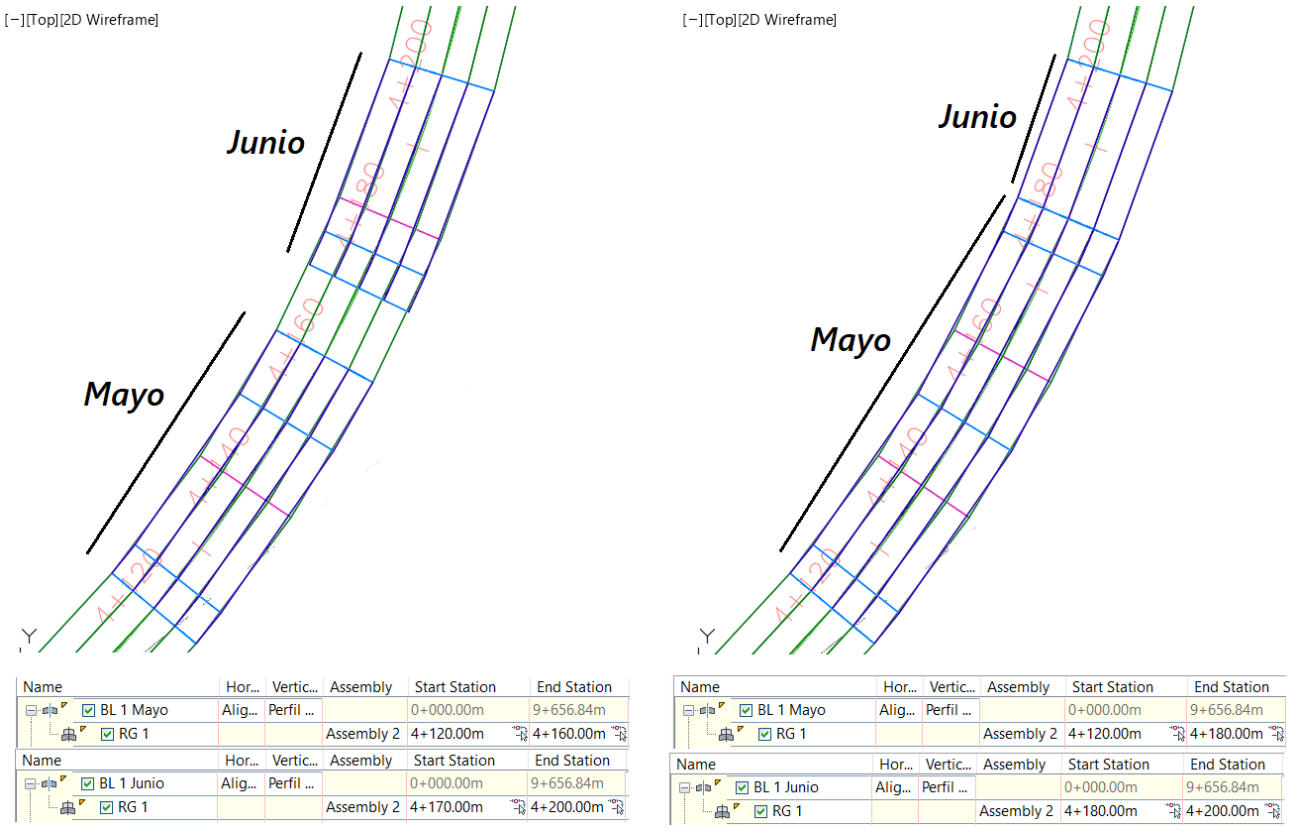


Figura 13. Ejemplo de uso del algoritmo "Desplazar tramo de obra lineal"

Combinando los dos algoritmos se puede gestionar completamente la discretización de una obra lineal en tramos, y en definitiva, añadir al modelo Civil 3D el factor tiempo.

5.1.3. Fase III: Estructura de datos

Los procesos llevados a cabo por los dos algoritmos, tanto "Crear tramo de obra lineal" como "Desplazar tramo de obra lineal", se encuentran dentro del marco de trabajo de la clase *corridor* (obra lineal) y de todas sus clases raíz. En la Figura 14 se expone el flujo de datos en el proceso de generación de una obra lineal, y que envuelve a todas las clases raíz de *corridor*.

En el proceso manual de generación de una obra lineal, esta se crea a partir del perfil longitudinal (*profile)*, la alineación *(alignment)* y la sección tipo (*assembly),* pudiendo incluir una superficie *(surface)*. La alineación y el perfil longitudinal se combinan y crean las líneas base (*baselines)* entidades con la geometría en planta de la alineación, con su trazado, pero con la propiedad de la elevación que aporta el perfil longitudinal. El siguiente paso del proceso es incluir a la línea base la sección tipo y crear, así, una región (*BaselineRegion)*. Una región solo puede tener una sección tipo vinculada, por lo que obras lineales con distintas secciones estarán definidas por varias regiones.

Las regiones o *BaselineRegions* son los elementos que poseen toda la información de una obra lineal (trazado, elevación y sección tipo) por eso es posible definir una obra lineal a partir de sus regiones.

Por otro lado, la superficie (normalmente la superficie altimétrica del terreno), además de servir como referencia para generar el perfil longitudinal, servirá como objetivo (*target)* de los taludes de movimientos de tierras. Un objetivo se refiere a un elemento geométrico superficie, o línea, que sirva como referencia a elementos de la sección tipo *(subassemblies)* que requieran de una referencia geométrica para su definición, por ejemplo, carriles de anchura variable.

Por último, las secciones tipo se generan, normalmente de forma manual, a partir de subensamblajes (*subassemblies)*, es decir, elementos concretos de una sección tipo como un carril, una mediana o un talud.

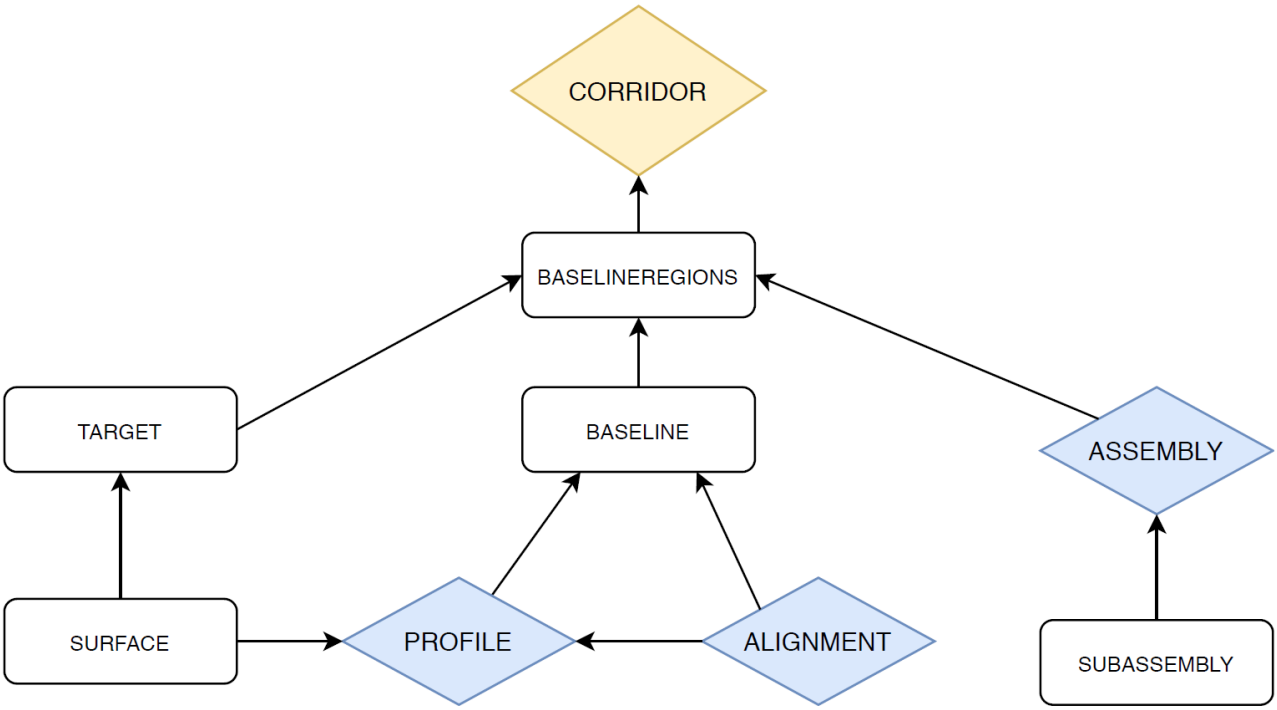


Figura 14. Jerarquía de los elementos raíz de la clase "corridor"

5.1.4. Fase IV: Metodología y ensayos para el desarrollo

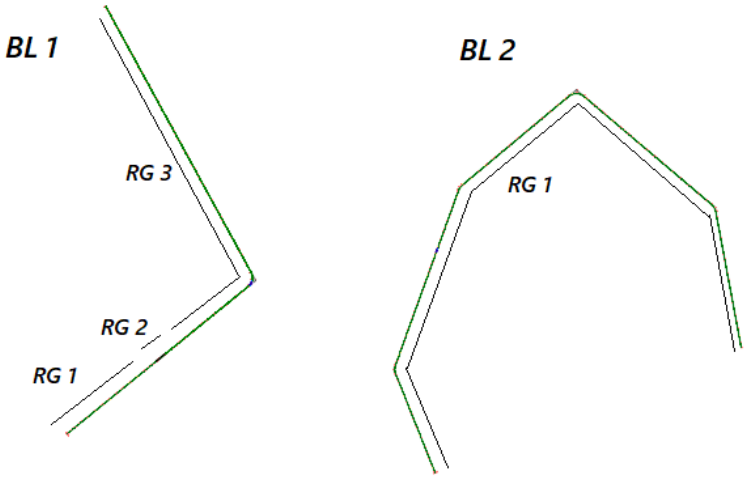
El modelo de ensayos (Figura 15) se compone de una obra lineal compuesta por dos alineaciones, obligando a componerse la obra lineal de dos líneas base (*BL* procedente de *Baseline)*, esta composición es frecuente en autopistas con calzadas separadas.

La línea base 1 tiene tres regiones con diferentes secciones tipo:

- Assembly: Sección de dos carriles

- Assembly 2: Sección de cuatro carriles
- Assembly Transition: Sección de anchura variable de transición (está enlazada a un objetivo *target*)

La composición de la línea base 1 permite evaluar el comportamiento del algoritmo frente a cambios de regiones a la hora de crear o modificar tramos de obra lineal. En cambio, el interés que tiene añadir al modelo de pruebas una segunda línea base es observar el comportamiento del algoritmo frente a obras lineales con varias alineaciones.

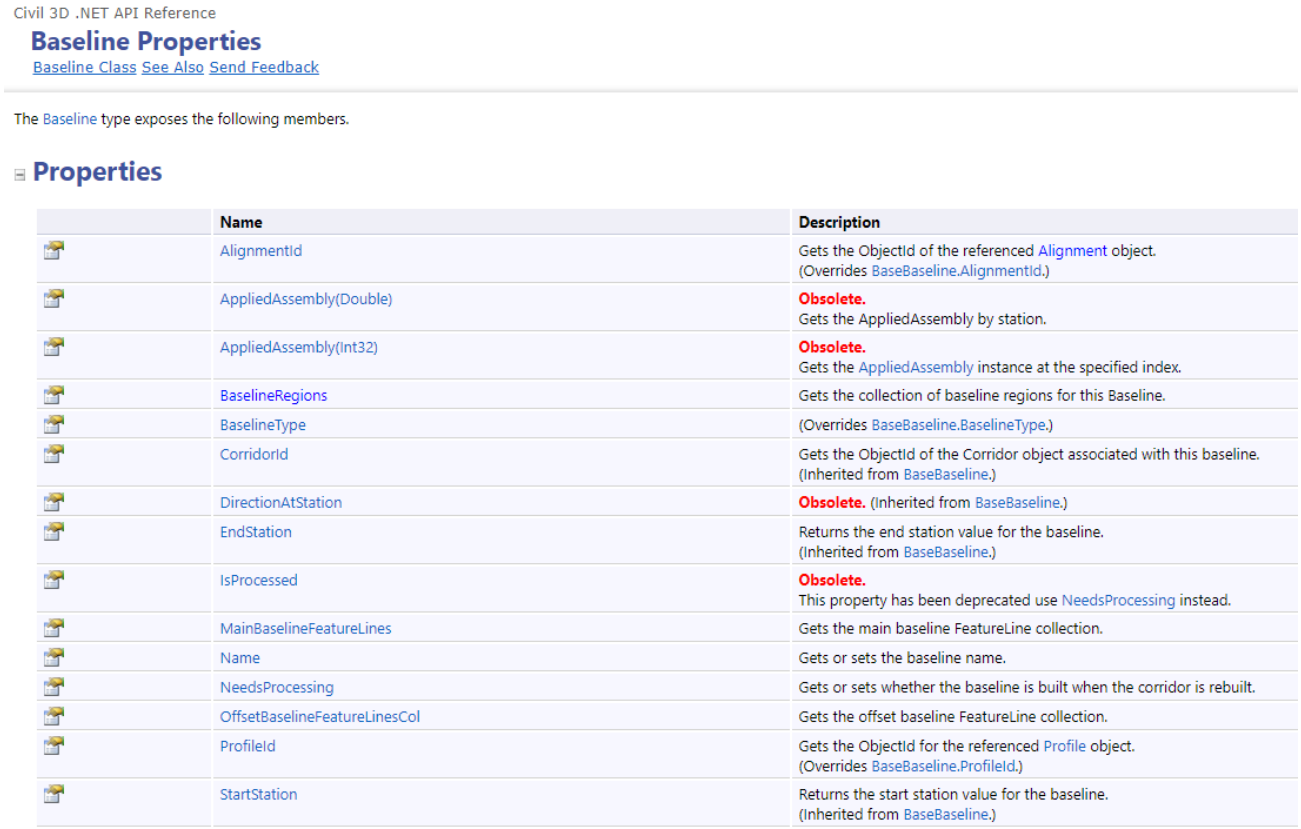


Name	Horizontal Baseline	Vertical Baseline	Assembly	Start Station	End Station
BL 1 - Alignment 1	Alignment 1	Perfil Longitudinal 1		0+000.00m	9+656.84m
RG 1 - Assembly			Assembly	0+000.00m	2+000.00m
RG 2 - Transition			Assembly Transition	2+000.00m	2+180.00m
RG 3- Assembly 2			Assembly 2	2+200.00m	9+656.84m
BL 2 - Alignment 2	Alignment 2	Perfil Longitudinal 2		0+000.00m	13+584.16m
RG 1 - Assembly			Assembly	0+000.00m	13+584.1...

Figura 15. Modelo de ensayos para el Caso I: Edición de obras lineales para su control y planificación en obra

La metodología para el desarrollo de los algoritmos fue generar un nodo Python al que se le importa la librería API.NET de Civil 3D y comenzar con el desarrollo de "Crear un tramo de obra lineal". Para ello, el primer paso fue recoger la información de una obra lineal ya existente a través de sus propiedades del API. Con esa información, crear una obra lineal nueva adaptada al recorrido que se quiera dar al tramo, es decir, su PK inicial y final. El proceso fue experimental.

Para afrontar este proceso se requería una profunda exploración de la librería API.NET de Civil 3D [14]. La exploración del API se realizó identificando las clases del API que intervienen en la jerarquía de objetos de una obra lineal (Figura 14) y observando los métodos y propiedades que cada clase posee (ver Figura 16).



Civil 3D .NET API Reference

Baseline Properties

Baseline Class See Also Send Feedback

The Baseline type exposes the following members.

Properties

	Name	Description
	AlignmentId	Gets the ObjectId of the referenced Alignment object. (Overrides BaseBaseline.AlignmentId.)
	AppliedAssembly(Double)	**Obsolete.** Gets the AppliedAssembly by station.
	AppliedAssembly(Int32)	**Obsolete.** Gets the AppliedAssembly instance at the specified index.
	BaselineRegions	Gets the collection of baseline regions for this Baseline.
	BaselineType	(Overrides BaseBaseline.BaselineType.)
	CorridorId	Gets the ObjectId of the Corridor object associated with this baseline. (Inherited from BaseBaseline.)
	DirectionAtStation	**Obsolete.** (Inherited from BaseBaseline.)
	EndStation	Returns the end station value for the baseline. (Inherited from BaseBaseline.)
	IsProcessed	**Obsolete.** This property has been deprecated use NeedsProcessing instead.
	MainBaselineFeatureLines	Gets the main baseline FeatureLine collection.
	Name	Gets or sets the baseline name.
	NeedsProcessing	Gets or sets whether the baseline is built when the corridor is rebuilt.
	OffsetBaselineFeatureLinesCol	Gets the offset baseline FeatureLine collection.
	ProfileId	Gets the ObjectId for the referenced Profile object. (Overrides BaseBaseline.ProfileId.)
	StartStation	Returns the start station value for the baseline. (Inherited from BaseBaseline.)

Figura 16. Propiedades de la clase Baseline en el API.NET de Civil 3D

El segundo algoritmo, "Desplazar tramo de obra lineal" se comenzó a desarrollar una vez terminado "Crear tramo de obra lineal". En este caso, el modelo de ensayos era idéntico al expuesto para el algoritmo "Crear tramo obra lineal", con la salvedad de que se incorporan dos tramos de obra lineal (generados por dicho algoritmo) al modelo para poder simular el proceso de planificación de obra mediante discretización mensual en tramos como ocurre en la Figura 13.

Para crear el algoritmo, el primer paso fue el de conseguir desplazar un tramo dado a través de todo el recorrido de la obra lineal de origen, adaptándose a sus diferentes regiones, y por lo tanto diferentes secciones tipo. El siguiente paso fue solidarizar ese desplazamiento con el resto de los tramos que pudieran verse afectados (ejemplo de la Figura 13), con el fin de cumplir el condicionante 3.

5.1.5. Fase V: Mejoras implementadas durante el desarrollo

Durante el proceso de creación del algoritmo se han producido mejoras en el desempeño de la rutina. La mayor parte se han desarrollado durante el desarrollo de "Creación de un tramo de obra lineal".

Las mejoras que experimentó el algoritmo "Creación de un tramo de obra lineal" fueron:

- Mantener las líneas base de la obra lineal ya existente:

En primera instancia, se planteó que para definir la nueva obra lineal se usaría una línea base recortada, que tuviera el recorrido del tramo que se desea generar, lo que implicaba crear la línea base desde cero, a partir de una alineación recortada y su perfil longitudinal.

Para realizarlo era necesario definir una nueva alineación vacía y copiar las entidades que componen la alineación de origen y que se encuentran dentro del recorrido del tramo.

El proceso era posible, sin embargo, con vistas al algoritmo "Desplazar tramo de obra lineal" este modo de proceder, además de ser computacionalmente más costoso, al mantener la misma línea base de partida, los tramos la usan como referencia y modo de comparación de puntos kilométricos.

- Implementar la función de copiar los *targets* de la obra lineal de origen a un tramo nuevo:

  Los objetivos o *targets* resultan necesarios para definir correctamente regiones con secciones tipo variables. A pesar de ser un requisito de diseño, se considera una mejora del algoritmo porque debido a su complejidad computacional los diseños previos no contaban con esta función.

  La dificultad de la implementación proviene [15] de que los targets se almacenan en colecciones desconectadas, generando inconsistencias de código. Finalmente, se implementó la solución dada en [16] que se consiguió ajustar perfectamente al algoritmo.

- Incorporación de entradas de datos dinámicas:

  Debido a la necesidad de pedir al usuario la introducción de datos de entrada para que el algoritmo funcione correctamente, se decidió incorporar la entrada de datos "inputs" al algoritmo a través del uso de Dynamo Player (Figura 12).

- Nomenclatura dinámica de los tramos generados:

  Se incorporó una rutina capaz de adaptar el nombre con el que se genera el tramo en "Crear tramo de obra lineal". El nombre se adapta a los siguientes escenarios (ver Figura 17):

  - El algoritmo no recibe ningún nombre:
    - ➔ Se impone el mismo nombre de la obra lineal de origen seguido de "Tramo 1… 2 … 3 … " hasta que se encuentre un nombre no utilizado previamente
  - Se recibe un nombre:

➔ Se evalúa si el nombre ya existe. En ese caso se pondrá el nombre seguido de (2)..(3)...(4).. hasta que se encuentre un nombre no utilizado.

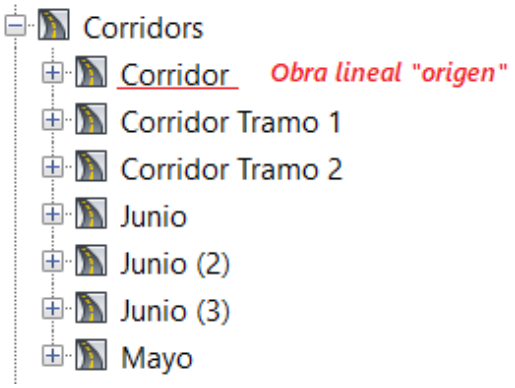


Figura 17. Nomenclatura dinámica aplicada en un ejemplo

- Albergar varias líneas base en una misma obra lineal:

  En diseños previos del algoritmo, por simplicidad, él código se ejecutaba para una sola línea base. Para casos en los que la obra lineal de origen posea varias alineaciones, es decir, varias líneas base, se implementa un nuevo ciclo que recorre y actúa sobre todas las líneas base que contenga la obra lineal.

5.1.6. Fase VI: Definición del algoritmo final

La definición del algoritmo refleja el proceso programado que sigue el algoritmo al ejecutarse. Cada algoritmo de los estudiados "Crear tramo de obra lineal" y "Desplazar tramo de obra lineal" se basa en la ejecución de una función definida en un nodo Python de Dynamo. El código contenido en el nodo Python, en cada caso, es expuesto y explicado en forma de pseudocódigo.

El aspecto del proyecto Dynamo en ambos algoritmos es el mismo (ver Figura 18), se compone de cuatro entradas de datos, que tienen la propiedad de ser dinámicas (editables desde Dynamo Player), conectadas a un nodo Python que se encarga de generar la rutina en formato de código escrito (ver Figura 7).

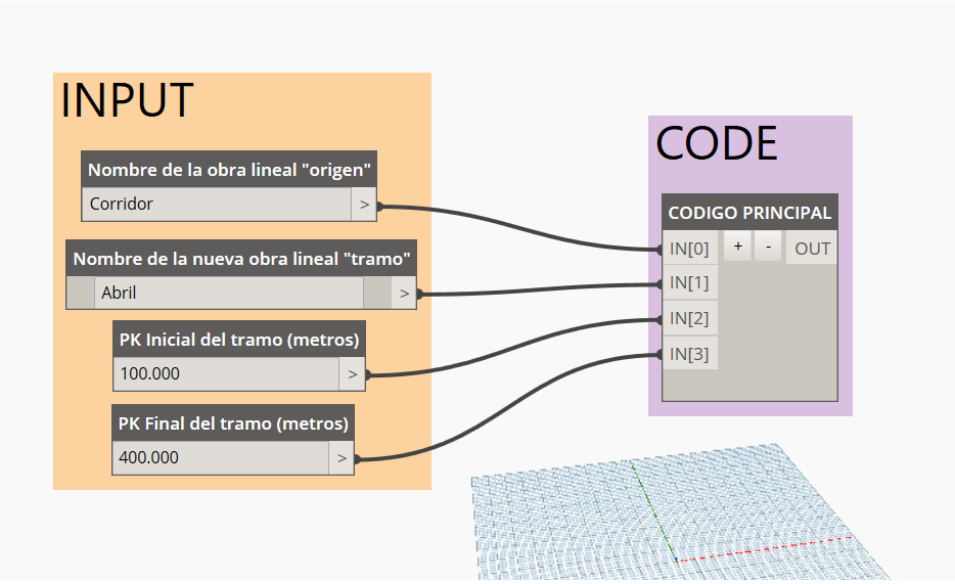


Figura 18. Estructura de nodos en la generación de algoritmos: "Crear tramo de obra lineal" y "Desplazar tramo de obra lineal"

Crear tramo de obra lineal

Las variables de entrada son introducidas en la función denominada CrearTramoObraLineal(), que se encarga de generar, en el recorrido limitado entre el PK inicial y el PK final, a partir de la obra lineal origen una nueva obra lineal con su misma geometría. No tiene parámetros de salida. Al terminar de ejecutar la función, el algoritmo finaliza.

El método AssignTargets() [16] Recibe como parámetros de entrada una región origen y una región destino. Tiene como objetivo copiar los objetivos *targets* de la región origen a la región destino. Dentro de esta función se ejecuta la función AssignTarget() que recibe como entrada dos colecciones de t*argets* y tiene como objetivo copiar la colección de *targets* de una a otra.

Pseudocódigo **FUNCIÓN: CREAR TRAMO DE OBRA LINEAL**

INPUT: **Nombre de la obra lineal "origen"**

Nombre de la nueva obra lineal "tramo"

PK Inicial del tramo a generar

PK Final del tramo a generar

OUTPUT: None

1. **Invocar a la obra lineal de origen**
2. **Invocar la alineación de la obra lineal de origen**
3. **Establecer el nombre de la nueva obra lineal (ver 5.1.5 - Nomenclatura dinámica)**
4. **Generación de una nueva entidad obra lineal vacía (añadiendo una nueva entidad a la colección de obras lineales del documento), con el nombre del punto 3 -se tendrá que ir añadiendo información a la nueva obra lineal para definirla geométricamente-.**
5. **Se inicia un ciclo por el cual, para cada línea base de la obra lineal de origen:**
   - 5.1. **Se copian sus regiones y se almacenan en una colección llamada "bregions"**
   - 5.2. **Se almacenan los ObjectId[1] de las secciones tipo de cada región y de sus perfiles longitudinales.**
   - 5.3. **Se añade una línea base a la colección de líneas base de la nueva obra lineal aportando el ObjectId de la alineación y del perfil longitdinal que lo engendran (ver Figura 19).**
   - 5.4. **Se recorta cada región almacenada en el punto 5.1 para que cumplan la limitación de recorrido entre el PKInicial y PKFinal. Si la región está fuera del rango de valores es eliminado.**
   - 5.5. **Se añade cada región a la línea base generada en el punto 5.3, aportando el PK donde se inicia, el PK donde termina y el ObjectId de la sección tipo.**
   - 5.6. **Se emplea el método "AssignTargets()" [16] para asignar los objetivos de las regiones de origen a cada región generada para que finalmente queden definidas por completo con los parámetros de la obra lineal de origen.**
6. **Actualizar la nueva obra lineal para que se efectúen los cambios en el modelo Civil 3D**

Pseudocódigo **FUNCIÓN: ASSIGN TARGETS**

INPUT: **Región origen**

Región destino

OUTPUT: None

1. **Llamada a la colección de *targets* de cada región**
2. **Se inicia un ciclo que recorre la colección de *targets* de la región origen**
   - **2.1. Se ejecuta AssignTarget()**
   - **2.2. Solventar inconsistencia computacional: Completar la asignación de targets aportando a Region.SetTargets() una colección de targets proveniente de si misma, es decir de Region.GetTargets(), si no, no funcionará.**

Pseudocódigo **FUNCIÓN: ASSIGN TARGET**

INPUT: **Colección de *target* origen**

Colección de *target* destino

OUTPUT: None

1. **Se genera una colección de ObjectIds para copiar los ObjectIds de los *targets* de origen**
2. **Definir la colección de ObjectIds de *targets* de origen como la colección de *targets* destino**
3. **Se copia la condición de funcionamiento de los *targets* de origen a los de destino.**

Civil 3D .NET API Reference

BaselineCollection.Add Method (String, ObjectId, ObjectId)

BaselineCollection Class See Also Send Feedback

Adds a baseline with the given baseline name, alignment and profile.

Namespace: Autodesk.Civil.DatabaseServices
Assembly: AeccDbMgd (in AeccDbMgd.dll) Version: 10.5.768.0

Syntax

C#

```
public Baseline Add(
        string baselineName,
        ObjectId alignmentId,
        ObjectId profileId
)
```

Visual Basic

```
Public Function Add ( _
        baselineName As String, _
        alignmentId As ObjectId, _
        profileId As ObjectId _
) As Baseline
```

Figura 19. Definición del método BaselineCollection.Add(), para generar una nueva línea base. Extraído del API.NET de Civil 3D como ejemplo de su aplicación en el desarrollo de algoritmos

Desplazar tramo de obra lineal

El algoritmo trata en definir y ejecutar la función DesplazarTramo(). A su vez esa función llama a otras dos funciones que también deben ser definidas: LimpiarZona() y AssignTargets() [16].

El objetivo de la función LimpiarZona() es el de despejar la zona del recorrido comprendido entre el PK inicial y el PK final de otros tramos de obra lineal existentes. De esta manera se consigue que el desplazamiento del tramo elegido sea solidario con el resto de los tramos (ver Figura 13).

Pseudocódigo **FUNCIÓN: DESPLAZAR TRAMO**

INPUT: Nombre de la obra lineal "origen"

Nombre de la obra lineal "tramo" a modificar

Nuevo PK Inicial del tramo

Nuevo PK Final del tramo

OUTPUT: None

1. Invocar a la obra lineal de origen
2. Almacenar las líneas base de origen en una colección y los ObjectId de sus alineaciones en otra colección de ObjectId
3. Invocar al tramo de obra lineal
4. Se inicia un ciclo en el que para cada línea base almacenada en el punto 2:
   - 4.1. Se almacenan en la colección "bregions" las regiones origen que se encuentran en el recorrido al que se desplazará el tramo, junto con sus secciones tipo almacenadas en otra colección de ObjectId.
   - 4.2. Se eliminan las regiones del tramo que ya no estén en el dominio.
   - 4.3. Para desplazar el tramo de obra lineal a regiones que no posee será necesario copiarlas de la obra lineal origen y generarlas en la obra lineal tramo. Se inicia un ciclo que recorre la colección "bregions".
     - 4.3.1. Se crea una variable tipo bool (True/False) llamada "Copiar" para evaluar en cada región almacenada en "bregions" si es necesario generar esa región en el tramo a desplazar. "Copiar" = True
     - 4.3.2. Se inicia un ciclo que recorre las regiones del tramo de obra lineal y las compara con la región de "bregions" sacada del ciclo del punto 4.3. Se establecen las siguientes condiciones:
       - 4.3.2.1. Si la región de "bregions" forma parte del tramo de obra lineal no hace falta copiarla → "Copiar" = False
       - 4.3.2.2. Si la región de "bregions" forma parte del tramo de obra lineal, pero de forma incompleta, se amplía el recorrido de la región del tramo para que alcance a completarse o llegue hasta los límites establecidos por el PK inicial o final. No hará falta copiarla → "Copiar" = False
     - 4.3.3. Si la variable "Copiar" se mantiene en True se genera la región de "bregions" en el tramo de obra lineal a partir de su sección tipo (almacenada en el punto 4.1) y su recorrido, limitado si fuese necesario, por el PK inicial y final establecido. Se ejecuta además el método "AssignTargets()" [16] .
5. Se actualiza el tramo de obra lineal para materializar el desplazamiento
6. Se ejecuta el método "LimpiarZona()"

Pseudocódigo **FUNCIÓN: LIMPIAR ZONA**

INPUT: **Nombre de la obra lineal "origen"**

Colección de alineaciones de la obra lineal "origen"

Nombre de la obra lineal "tramo" a modificar

Nuevo PK Inicial del tramo

Nuevo PK Final del tramo

OUTPUT: None

1. **Se inicia un ciclo por el cual a cada obra lineal del documento:**
   - 1.1. **Se evalúa si coincide con la obra lineal de origen o la obra lineal tramo que se desea desplazar, en esos casos el ciclo se termina aquí y continua con la siguiente obra lineal del ciclo.**
   - 1.2. **Se inicia un ciclo por el cual a cada línea base de la obra lineal que contenga una alineación en común con la obra lineal de origen (por lo tanto, se considere un tramo de la obra lineal de origen):**
     - 1.2.1. **Se evalúa si la línea base posee regiones dentro del recorrido entre el PK inicial y final. En este caso, se eliminan o recortan las regiones conflictivas para que el espacio entre el PK inicial y final quede vacío.**
     - 1.2.2. **Se actualiza la obra lineal para materializar los cambios**

5.1.7. Fase VII: Valoración técnica y económica

El tiempo de ejecución de los algoritmos es muy reducido, ambos tardan menos de un segundo en ejecutarse, En cuanto a su funcionalidad, admiten cualquier tipo de escenario y se adaptan a cualquier composición de modelado de obra lineal, siendo eficaz en secciones tipo variables referenciadas a objetivos *targets* y obras lineales compuestas por un conjunto de alineaciones.

La única limitación es que en el caso de haber alguna modificación de la obra lineal origen, los tramos generados tendrían que actualizarse volviendo a ejecutar la rutina. Sin embargo, los algoritmos podrían dar lugar a ser la base para otras rutinas como la discretización automática de una obra lineal en tramos equidistantes, o la actualización de los tramos frente a cambios de diseño en la obra lineal origen, solventando tal limitación.

Técnicamente es una solución más robusta y rápida que realizar la discretización manualmente, eliminando posibles errores humanos. No obstante, para su desarrollo es necesario conocer el API.NET de Civil 3D y desenvolverse con el nodo Python de Dynamo, es decir, una serie de conocimientos muy específicos y poco practicados (muchos programadores de Civil 3D usan otro tipo de editores en C# y VB.NET).

En cuanto a la valoración económica, el tiempo necesario para desarrollar el algoritmo se estima a 40 horas. El tiempo de discretización manual no requiere de un desarrollo previo. No obstante, con el algoritmo ya desarrollado la discretización pasa a ser prácticamente automática, por lo que los tiempos se reducen considerablemente en comparación a hacerse manualmente. Si al tiempo de ejecución manual se le tiene en cuenta posibles fallos humanos, el uso de algoritmos es la solución más favorable.

Con la Figura 20 como referencia, se entiende el algoritmo como una inversión de tiempo, que en la medida en cuanto más se use, mayor beneficio y recorte de tiempos tendrá. Por esa razón, puede ser especialmente interesante su desarrollo destinado a oficinas técnicas en la que se puede aprovechar su ejecución en multitud de proyectos. La duración de las modificaciones son las estimadas en un caso general. La proporción de horas empleadas se mantiene constante.

(Coste: 20€/h)	HORAS DE DESARROLLO	COSTE INICIAL	HORAS POR MODIFICACIÓN DE PLANIFICACIÓN	COSTE POR MODIFICACIÓN DE PLANIFICACIÓN
Algoritmo	40	800	0.1	2
Manualmente	0	0	0.5	10

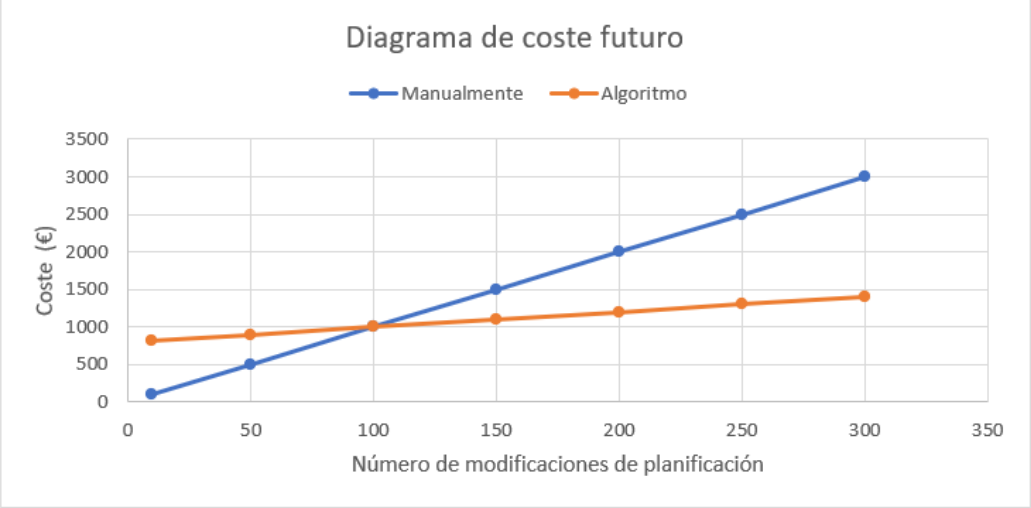


Figura 20. Datos de referencia para la evaluación económica del Caso de estudio I

5.2. Caso de estudio II: Exportación del movimiento de tierras

5.2.1. Fase I: Análisis

Uno de los aspectos más enriquecedores de trabajar con BIM resulta ser la conexión de modelos virtuales geométricos, como Civil 3D -que son capaces de calcular las mediciones y cubicaciones del modelo- con un software dedicado a bases de datos presupuestarios, como Presto -capaz de diseñar el modelo presupuestario del proyecto-. El modelo presupuestario se actualizará dinámicamente con cambios que se puedan producir en la geometría del modelo virtual.

En el interés por conectar Presto con programas de modelado 3D Presto desarrolla para Revit una extensión denominada Cost-It. Gracias a esta extensión se pueden asociar elementos IFC de Revit a partidas presupuestarias de Presto. Las propiedades presupuestarias (precio, descripción de la partida y código de montaje) que se atribuyan a través de la partida de Presto serán visibles en el elemento IFC desde Revit.

Civil 3D permite conocer las mediciones de una obra lineal distinguiendo los movimientos de tierra y la medición de firmes, sin embargo existe un problema que impide exportar las mediciones de los movimientos de tierras. La exportación a IFC de los taludes de terraplén y desmonte se realiza como cuerpos (superficies) y no como sólidos (volúmenes), de este modo no es posible calcular el volumen y utilizarlo en las mediciones de Presto, a través del módulo Cost-It integrado en Revit.

La solución manual es generar los sólidos del movimiento de tierra mediante el comando Esculpe, generando un sólido comprendido entre los taludes de desmonte y terraplén, y la superficie original del terreno, para poder exportarlo correctamente a formato IFC 2.3. El formato IFC 4.1 no está definido para exportar sólidos sino para exportar, por ejemplo, alineaciones. Resolviendo la exportación IFC, no solo se vuelve exportable a Revit sino que a todo el conjunto de programas con capacidad de interpretar archivos IFC, como por ejemplo Naviswork.

Este flujo de trabajo e intercambio de información entre Civil 3D y Presto se genera gracias al empleo de Revit como Modelo Federado, es decir, como un software plataforma que une programadas de distintas disciplinas. Aunque no es su uso habitual, en este caso es una solución que se adapta de forma natural y que resulta práctica.

5.2.2. Fase II. Resolución con Dynamo

Se plantea una solución que permita automatizar la generación de sólidos del movimiento de tierras en una obra lineal y así mejorar y agilizar el proceso de exportación del movimiento de tierras a IFC como sólidos.

Para ello se desarrolla un algoritmo en Dynamo llamado "Volúmenes de Movimiento de Tierras" el cual genera sólidos en el modelo Civil 3D, distinguiendo entre volúmenes de terraplén y de

desmonte, y exporta las mediciones a una hoja Excel. Los sólidos generados en Civil 3D son exportables a IFC 2.3 como sólidos, resolviendo la exportación de sus mediciones (ver Figura 21).

Los sólidos se generan en capas diferentes (ver Figura 22) según sean de desmonte o de terraplén.

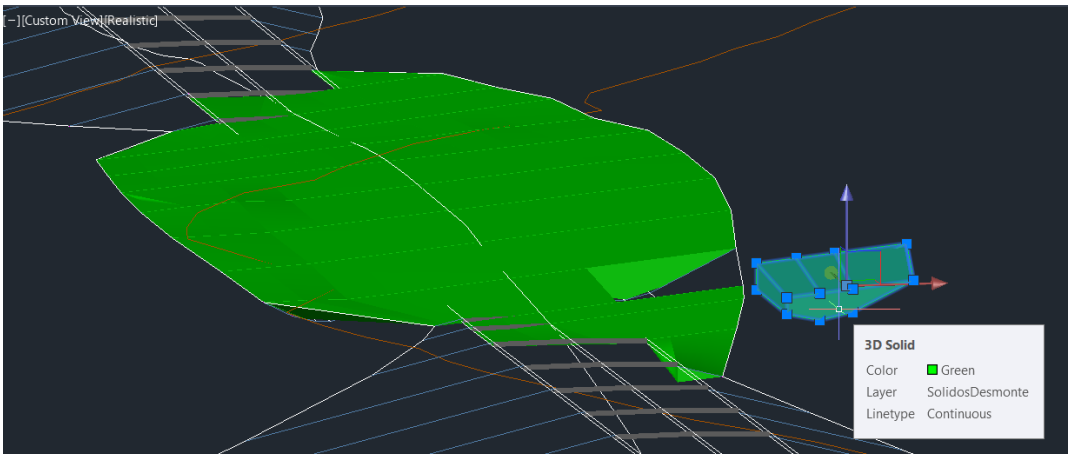


Figura 21. Muestra gráfica del modelado del movimiento de tierras como sólidos 3D

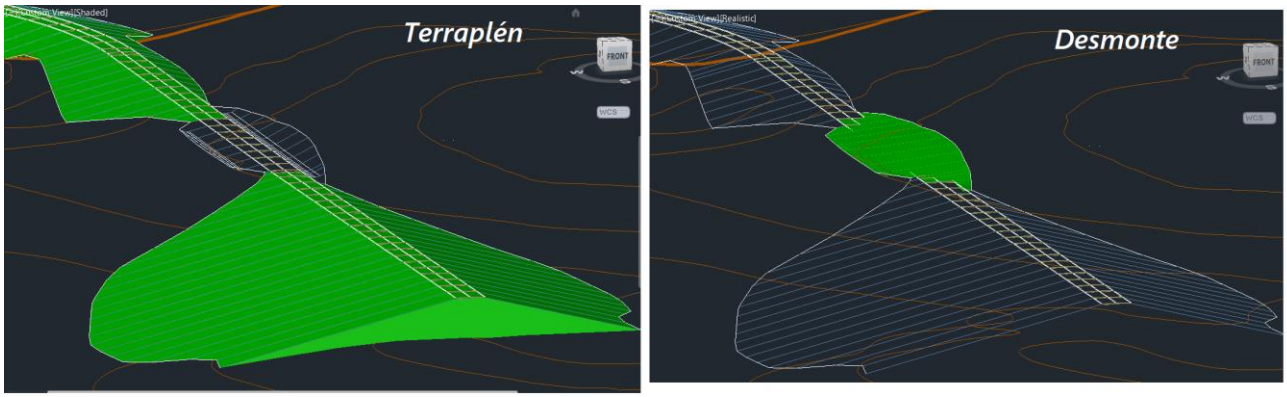


Figura 22. Vista de los sólidos de movimiento de tierras generados por el algoritmo "Volúmenes de Movimiento de Tierras", distinguiendo entre aquellos generados en la capa VolumenesTerraplén y los generados en la capa VolumenesDesmonte

El algoritmo requiere como entradas de datos el nombre de la obra lineal de la que se desea realizar la exportación, el nombre de la superficie y los códigos de las líneas características que definen la sección tipo -estos códigos requieren de un conocimiento previo de la estructura de datos que interviene en la generación de líneas características-. Con esas entradas de datos, el algoritmo puede ejecutarse de forma dinámica a través del *Dynamo Player*.

5.2.3. Fase III: Estructura de datos

Para ser capaz de parametrizar el recorrido de una obra lineal, se definen en Civil 3D unas secciones transversales virtuales llamadas *stations*. Computacionalmente, se definen como el *double* (número con decimales) correspondiente al P.K. (punto kilométrico) de dicha sección.

Dynamo for Civil 3D, permite generar sistemas de coordenadas relativos contenidos simultáneamente en ***stations*** y en elementos que definen la obra lineal (***featurelines*** o líneas características, líneas base, alineaciones y perfiles).

Las secciones transversales se definen, en cada ***station***, con la unión de puntos llamados ***Point Codes*** o puntos de código. Cada subensamblaje (subelemento que compone la sección tipo -carril, arcén, talud…) posee sus propios puntos de código, pudiendo tener desde dos hasta más de cinco puntos de código. Para identificar y diferenciar unos puntos de código con otros se les da una codificación diferente a cada uno. La codificación es el nombre que tiene cada punto de código. Como estos puntos se generan en cada ***station***, la unión longitudinal de los puntos de código con misma codificación genera la entidad llamada ***Feature Line***, o línea característica (ver Figura 23).

El sistema de referencia empleado habitualmente es a través de las líneas características, creando en cada ***station*** un sistema de coordenadas relativo situado en el punto de código que define la línea característica.

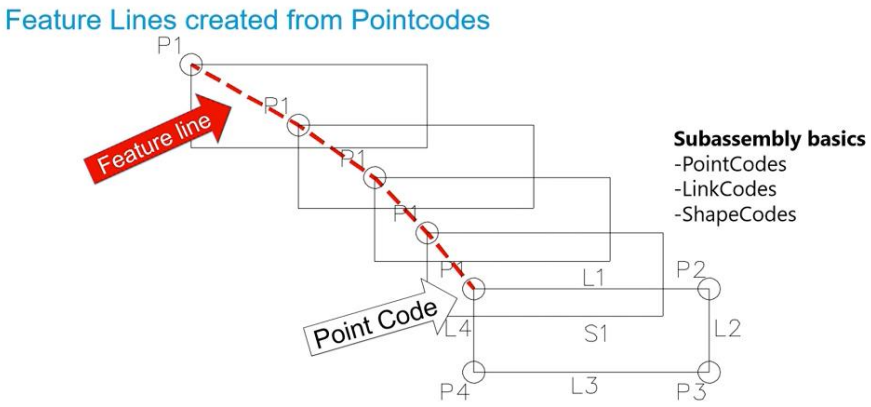
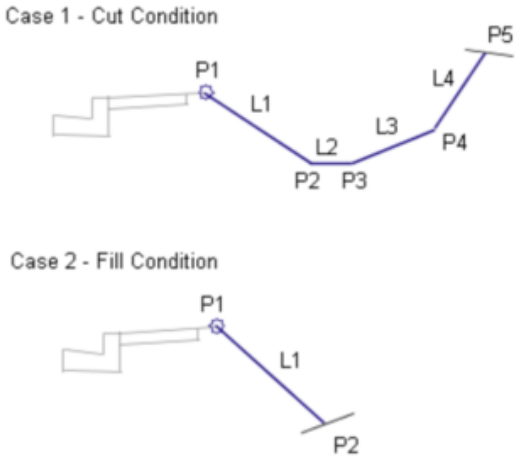


Figura 23. Esquema de la generación automática de una línea característica "Feature Line" definida a través de su punto de código "PointCode". Además el resto de elementos que componen la sección tipo. [17]

Los códigos de los puntos de código que se asocian a los taludes dependen de la clase de talud que se defina en la sección tipo. En el caso de la obra lineal de ensayos, se ha utilizado un talud de uso frecuente "BasicSideSlopeCutDitch" que tiene los códigos que aparecen en la Figura 24



Point, Link, or Shape	Codes
P1	Hinge Hinge_Fill
P2	Ditch_In Daylight Daylight_Fill
P3	Ditch_Out
P4	Hinge_Cut
P5	Daylight Daylight_Cut

Figura 24. Puntos de código pertenecientes al talud "BasicSideSlopeCutDitch" [18]

En la introducción del caso de estudio se mencionaba la capacidad que por defecto tiene Civil 3D de extraer sólidos de una obra lineal. El diagrama de la Figura 25 refleja cómo origina Civil 3D esos sólidos, que nacen de las secciones tipo y los elementos que la definen (ver Figura 23) como los ***Point Codes, Link Codes*** (la unión de ***Point Codes*** de una misma sección) y ***Shape Codes*** (superficie originada por la unión cerrada de ***Link Codes***). Con esta definición de sólidos, Civil 3D consigue modelar los volúmenes de firme correctamente, sin embargo, como los movimientos de tierras están relacionados con la superficie del terreno y salvo para definir la longitud de los taludes, esta no se ve involucrada en la generación de la sección tipo y por lo tanto tampoco en los sólidos. Esa es la razón que explica que Civil 3D extraiga los movimientos de tierras como superficies (ver Figura 26).

En el contexto de esta estructura de datos, Dynamo posee herramientas suficientes como para plantear procesos de diseño paramétrico referenciado a elementos de la obra lineal, como un movimiento de tierras, a partir de sus nodos por defecto sin necesidad de programar en el API.

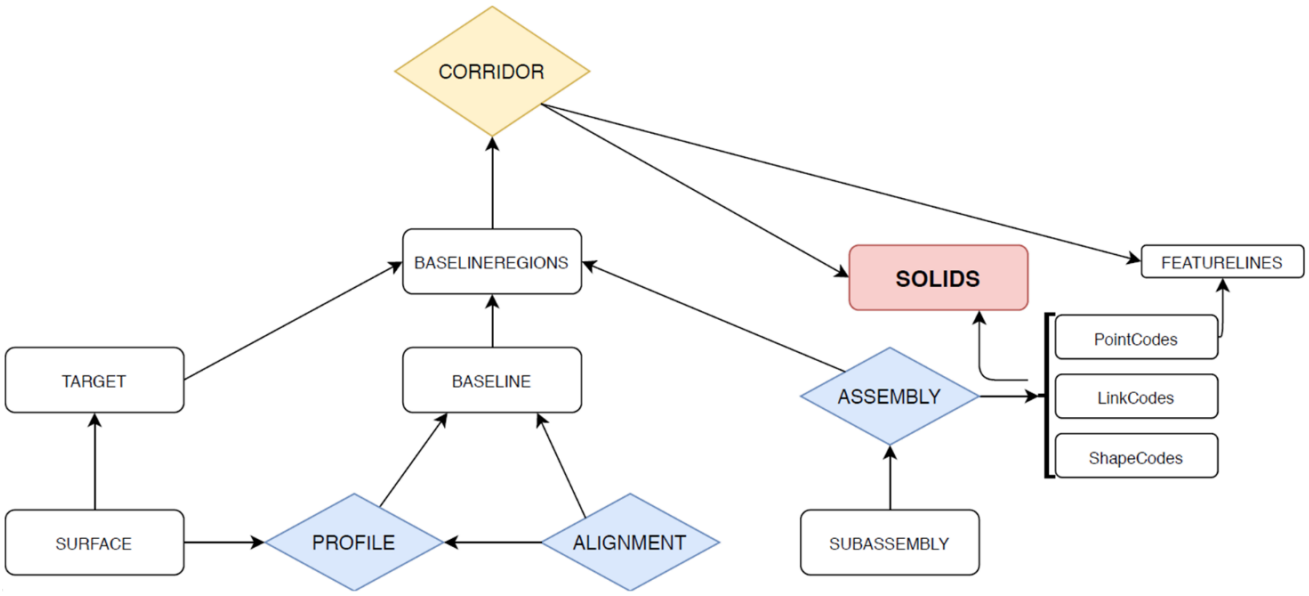


Figura 25. Diagrama del flujo de datos derivados de la generación de una obra lineal en Civil 3D. Origen de su extracción de sólidos.

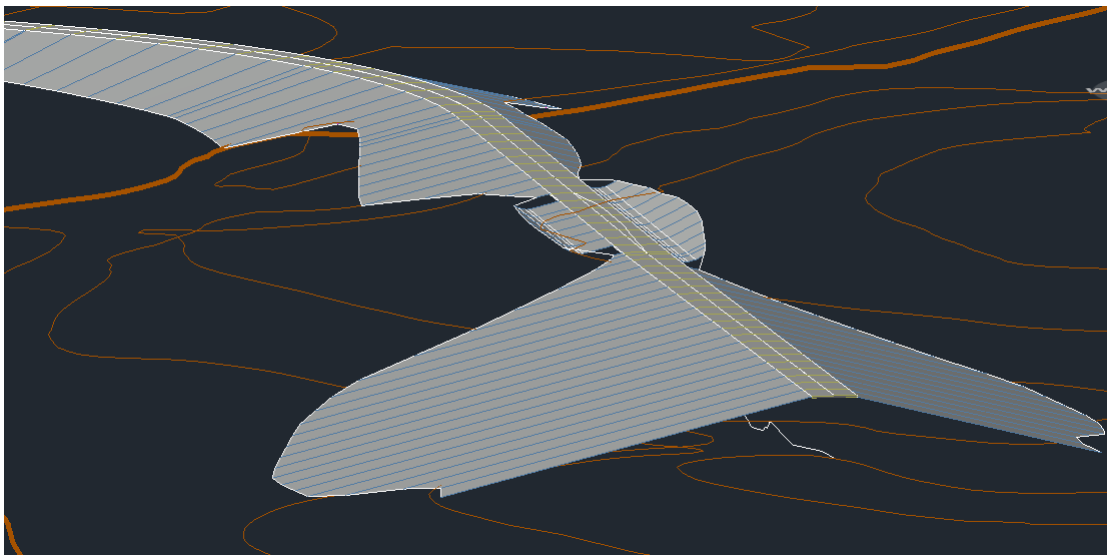


Figura 26. Extracción de sólidos por defecto de Civil 3D -Movimiento de tierras extraído como superficie en vez de como sólido-

5.2.4. Fase IV: Ensayos y metodología para el desarrollo

En este caso, Dynamo posee nodos por defecto suficientes como para soportar todas las operaciones geométricas necesarias para generar sólidos en un movimiento de tierras sin necesidad de programar en el API. Por esa razón, todo el desarrollo del algoritmo se realiza mediante Programación Visual a partir de los nodos Dynamo.

El método seguido para crear sólidos de movimiento de tierras es que, para cada *station* de muestreo de la obra lineal, se dibuje una policurva que encierre el área comprendida entre el talud

de desmonte o de terraplén y la superficie terreno. Mediante extrusión de una policurva a la policurva inmediatamente siguiente, la de la siguiente ***station***, se crea un sólido comprendido entre las dos secciones de muestreo. De esta forma se termina creando una serie de sólidos que componen el movimiento de tierras al completo.

Se generó un modelo de Civil 3D para ensayar la ejecución del algoritmo durante su desarrollo. El modelo de ensayos cuenta con una obra lineal (ver Figura 27), compuesta por dos alineaciones (dos líneas base), y construida sobre una superficie de terreno real perteneciente a una Base Topográfica Armonizada en el municipio de Torrelavega [19].

Los perfiles longitudinales están definidos para que en ambas líneas base se generen tramos de terraplén y desmonte y así poder evaluar experimentalmente ambos escenarios y la suavidad de la transición entre terraplén y desmonte.

En cuanto a su sección tipo, tan solo se usa una, que es común a ambas líneas base y recorre toda la obra lineal. Se trata de la sección tipo más sencilla posible que admitiera taludes de terraplén y desmonte. La razón de no incluir otras secciones dentro de la obra lineal es que la parte de la sección tipo que influye directamente en los taludes y su volumen, es tan solo el elemento exterior que incluya el talud de desmonte o terraplén. En el caso de existir secciones más complejas la rutina seguirá ejecutándose de la misma forma.

Por esa razón, el último paso del proceso es validar el algoritmo en un caso real (ver Figura 28), más complejo que el modelo de ensayos, del que se disponen las mediciones para poder así comparar los volúmenes obtenidos por el algoritmo a los obtenidos tanto manualmente como en la propia obra una vez ejecutada.

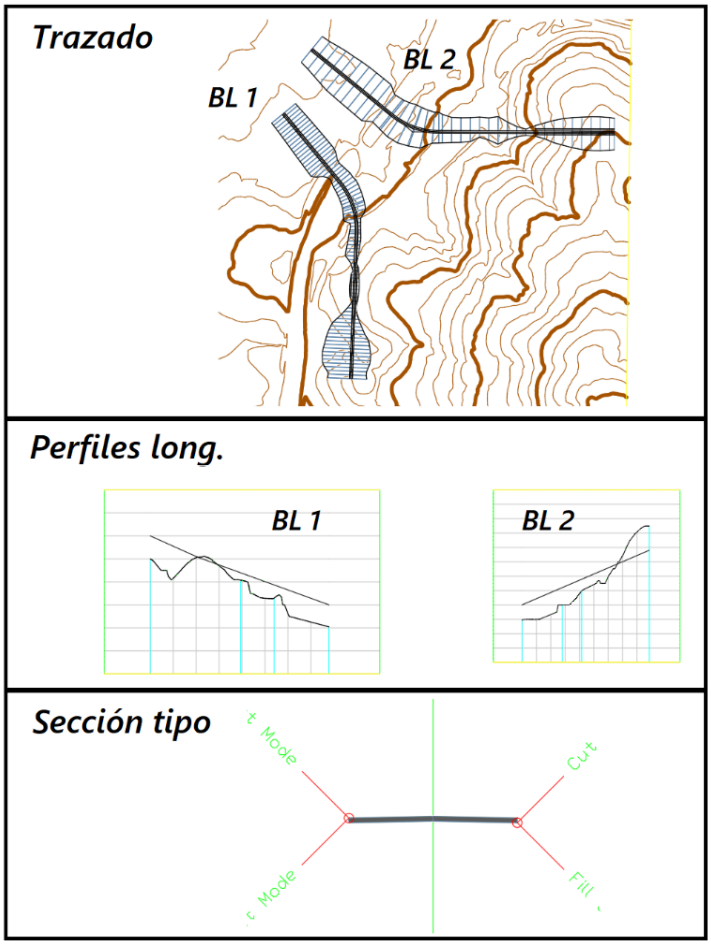


Figura 27. Descripción del modelo de ensayos empleado para desarrollar el algoritmo "Volúmenes del movimiento de tierras"

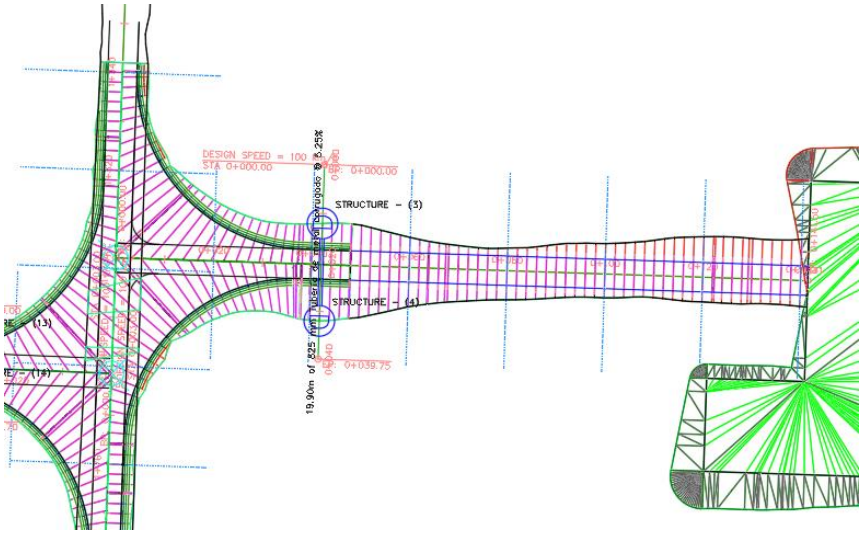


Figura 28. Modelo CIvil 3D de un parque eólico real

5.2.5. Fase V: Mejoras implementadas durante el desarrollo

Las mejoras que, conforme al progreso en el desarrollo, se fueron llevando a cabo son:

- Unión de la generación del terraplén y el desmonte, y su transición:

  Como simplificación del ensayo, en fases tempranas del desarrollo, se definieron los procesos de generación de sólidos de forma separada según, fuesen talud de terraplén o talud de desmonte. El de terraplén se desarrolló en primer lugar y una vez terminado, se generó el de desmonte, que por sus características de sección es más complejo.

  Terminados los procesos del algoritmo necesarios para generar sólidos de los taludes de terraplén y desmonte, se realizó el proceso de generación de sólidos por debajo (terraplén) y/o por encima (desmonte) de la calzada, incluyendo así todo el volumen correspondiente a un movimiento de tierras. Supuso también la definición de una transición entre los sólidos de terraplén y desmonte.

- Albergar varias líneas base en una misma obra lineal:

  Una vez se tenía confeccionado el algoritmo para generar sólidos a lo largo de una sola línea base, el proceso de ejecutarlo en varias líneas base de golpe requería una mejora técnica del algoritmo (la creación de un nodo personalizado) que implicaba un cambio en cómo estaba concebido el algoritmo.

  El requerimiento técnico consistía en almacenar todo el proceso de generación de sólido dentro de un nodo personalizado de Dynamo para que así, el proceso se repitiera para cada línea base.

- Parámetro dinámico de distancia mínima entre secciones de muestreo, ***stations***:

  Se observó experimentalmente que en situaciones donde se produjeran cambios de curvatura en la alineación (trazado en planta) de la obra lineal, si las ***stations*** están muy próximas, los taludes tienen el riesgo de solaparse y crear un conflicto en la geometría. Para solucionar esta situación, se tomó la medida de implementar un parámetro dinámico -que el usuario puede editar cuando lo desee- que establece una distancia mínima entre ***stations***, eliminando de la colección de ***stations*** aquellas separadas una distancia inferior a la mínima.

- Parámetro dinámico de número de puntos en los que se discretiza la sección del movimiento de tierras en una ***station***:

  Para generar los sólidos se necesita discretizar en un número de puntos determinado cada sólido que se genera. Para parametrizar el proceso se estableció un parámetro dinámico aportado por el usuario que define el número de puntos en los que se discretiza la proyección vertical de cada policurva del modelo. Así una sección completa estaría definida por un número de puntos correspondiente a seis veces el valor de este parámetro, menos uno.

Este parámetro permite adaptar la precisión de las mediciones obtenidas, pudiendo disminuirse para minimizar el tiempo de ejecución computacional del algoritmo en función de las exigencias y tolerancias necesarias.

- Exportación de mediciones a Excel:

  Una vez se generan los sólidos, mediante los nodos Dynamo se obtienen los volúmenes de cada sólido, es decir, el volumen de movimiento de tierras, distinguiendo terraplén de desmonte, de cada tramo entre *stations*. Se observa así la evolución de los volúmenes a lo largo de la obra lineal.

  La exportación se realiza de forma que en Excel se crea para cada línea base de la obra lineal, dos hojas de cálculo -una para los volúmenes en terraplén y otra para los de desmonte-.

5.2.6. Fase VI: Definición del algoritmo final

El algoritmo está programado mediante Programación Visual, es decir, a través de la unión de nodos de Dynamo. Dentro de la red de nodos (ver Figura 29) se encuentra el nodo llamado "VolumenesMovTierra"s el cual almacena la parte del algoritmo correspondiente a la modelización de los sólidos en Dynamo, que posteriormente serán exportados a Civil 3D.

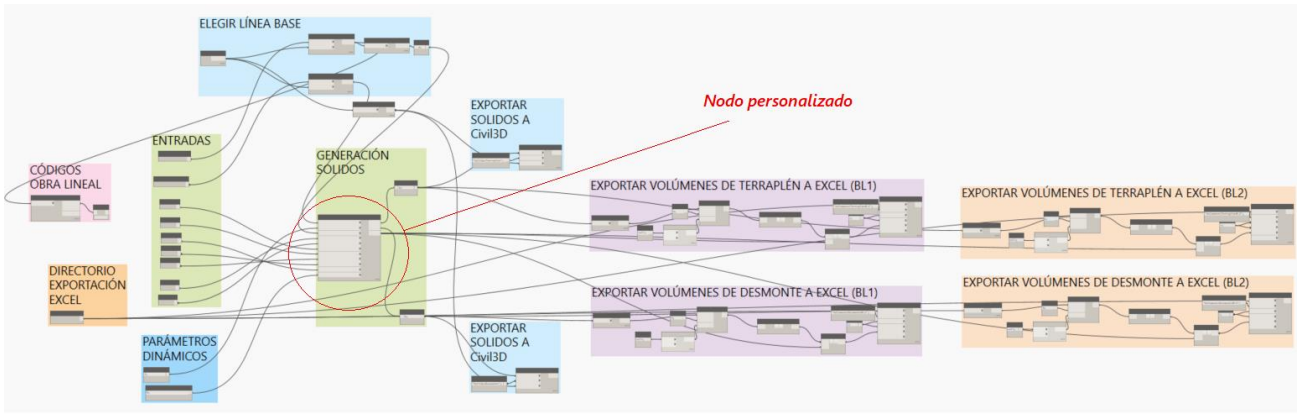


Figura 29. Red de nodos Dynamo del algoritmo "Volúmenes de movimiento de tierras"

Para explicar el desarrollo del algoritmo se definirá paso a paso los procesos, que aparecen agrupados en la Figura 29, por los que pasa la ejecución del código.

Pseudocódigo **RED DE NODOS DYNAMO: GENERACIÓN SÓLIDOS DE MOVIMIENTO DE TIERRAS**

Parte 1

INPUT: **Nombre de la obra lineal**

Nombre de la superficie terreno de referencia

Códigos de los puntos de código asociados al talud de desmonte

Códigos de los puntos de código asociados al talud de terraplén

Ruta del directorio de la exportación del Excel

OUTPUT: None

1. **Códigos de la obra lineal:**

   Muestra los códigos de la obra lineal, sirve de referencia para conocer los códigos a introducir en las entradas de datos.

2. **Elegir línea base:**
   - **2.1. Invoca la obra lineal y la superficie coincidente con el nombre dado en el punto 1.**
   - **2.2. A partir de la obra lineal invoca la colección de líneas base pudiendo continuar el algoritmo con toda la colección o seleccionando unas líneas base determinadas.**

3. **Generación de sólidos (Nodo personalizado):**

   La compleja red de nodos que generan los sólidos en Dynamo se simplifica a un nodo personalizado denominado "VolumenesMovTierras", cuya explicación se hará por separado. Las salidas de este nodo son:

   - **3.1. Lista de sólidos de terraplén**
   - **3.2. Lista de sólidos de desmonte**

4. **Exportar sólidos a Civil 3D:**

   Se exportan los sólidos Dynamo de las listas resultantes del punto 4 a Civil 3D a través del nodo Object.ByGeometry, como sólidos 3D. Los sólidos de terraplén y desmonte se exportan a capas diferentes -de no existir previamente estas capas se crean automáticamente- llamadas "SolidosTerraplen" y "SolidosDesmonte".

Pseudocódigo **RED DE NODOS DYNAMO: GENERACIÓN SÓLIDOS DE MOVIMIENTO DE TIERRAS**

Parte 2

5. **Exportar volúmenes de terraplén a Excel:**

   Para cada línea base hay que crear la siguiente rutina:

   5.1. **Se calcula el volumen de los sólidos Dynamo contenidos en cada colección sacada del punto 4, según se trate de desmonte o terraplén.**
   5.2. **Se fijan a cero aquellos elementos de la lista vacíos.**
   5.3. **Se recogen los datos válidos de los volúmenes concretando, de toda la colección de volúmenes calculados, aquellas que corresponden a la línea base correspondiente y se suman los volúmenes recogidos en la misma *station*.**
   5.4. **Se exporta a un documento Excel, con la ruta definida en la entrada de datos de "Directorio de Exportación a Excel", la lista de volúmenes de movimiento de tierras entre cada *station,* diferenciando, en hojas Excel separadas, para cada línea base, los volúmenes pertenecientes a terraplén y desmonte.**

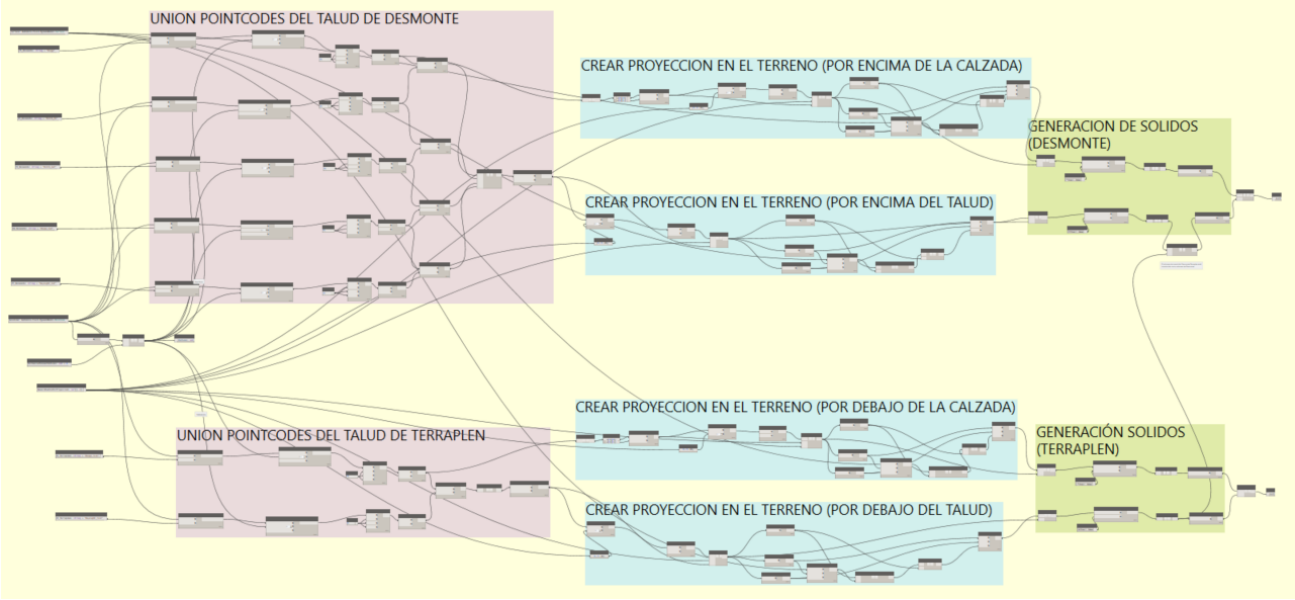


Figura 30. Red de nodos almacenado en el nodo personalizado "VolumenesMovTierras"

El nodo personalizado "VolumenesMovTierras" tiene la red de nodos Dynamo que muestra la Figura 30. Los sólidos que se generan resultan ser el volumen encerrado entre los taludes (y la calzada) y la superficie terreno. A continuación, se definen los procesos por los que transcurre la ejecución del algoritmo del nodo:

Pseudocódigo **NODOS PERSONALIZADO DYNAMO: VOLUMEN MOVIMIENTO DE TIERRAS**

Parte 1

INPUT: **La superficie Civil 3D del terreno**

La línea base o colección de líneas base a las que se va a aplicar el algoritmo

Los códigos de los puntos de código del talud de desmonte

Los códigos de los puntos de código del talud de terraplén

El parámetro dinámico de distancia mínima entre *stations*.

El parámetro dinámico de número de puntos que definen la discretización de los sólidos.

OUTPUT: Lista de sólidos de desmonte. Es la unión de las listas de sólidos 3.1.6 y 3.2.6 en desmonte

Lista de sólidos de desmonte. Es la unión de las listas de sólidos 3.1.6 y 3.2.6 en terraplén

Lista de *stations* del resultante del paso 2.1

1. **Unión de los puntos de código del talud de desmonte y terraplén:**

   Los procesos varían según el número de puntos de código involucrados, cuantos más puntos, más complejo. Por esa razón este proceso en desmonte es algo más costoso.

   1.1. **Paso previo: Eliminar las *stations* separadas una distancia inferior al parámetro dinámico de distancia mínima.**
   1.2. **Para cada punto de código se invoca su línea característica y en ella, para cada *station,* se crea un sistema de coordenadas relativo.**
   1.3. **Se crean puntos de coordenadas relativas (0, 0, 0) en cada sistema de coordenadas.**
   1.4. **En cada *station,* para los dos márgenes de la obra lineal, se crea una polilínea que une todos los puntos generados en dicha *station*.**

2. **Crear proyección en el terreno y generación de sólidos (ver Figura 31):**

   Para terraplén y desmonte, en cada caso, se realiza los siguientes dos procesos:

   2.1. **Generación de la proyección en el terreno por encima (desmonte) o por debajo (terraplén) de la calzada, sigue los siguientes pasos:**
      2.1.1. **Se unen los puntos generados en el paso 2.3 pertenecientes a los extremos de la calzada con una línea formando el perfil de la calzada.**
      2.1.2. **Se discretiza la línea en n puntos (n es el parámetro dinámico del paso 1.6).**

Pseudocódigo NODOS PERSONALIZADO DYNAMO: VOLUMEN MOVIMIENTO DE TIERRAS

Parte 2

2.1.3. Se proyectan los puntos de la discretización a la superficie terreno

2.1.4. Se almacenan los puntos que estén o por encima o por debajo de la calzada, según se analice el volumen de terraplén o desmonte.

2.1.5. Se unen los puntos de la proyección en el terreno con los puntos del perfil de la calzada (paso 3.1.1), formando una policurva cerrada.

2.1.6. Se identifican las posibles zonas de transición (ver Figura 32) y se incluyen como puntos de desponte.

2.1.7. Se generan sólidos, a cada margen de la calzada, a través de la extrusión de una policurva hacia su policurva contigua. El resultado es una lista de sólidos encerrados entre dos *stations*.

2.2. Generación de la proyección en el terreno por encima (desmonte) o por debajo (terraplén) del talud, sigue los siguientes pasos:

2.2.1. Se unen los puntos de código desde el punto extremo de la calzada hasta el extremo del talud con una línea formando el perfil del talud.

2.2.2. Se discretiza la línea en n puntos (n es el parámetro dinámico del paso 1.6).

2.2.3. Se proyectan los puntos de la discretización a la superficie

2.2.4. Se almacenan los puntos que estén o por encima o por debajo de la calzada, según se analice el volumen de terraplén o desmonte.

2.2.5. Se unen los puntos de la proyección en el terreno con los puntos del perfil del talud (paso 3.2.1), formando una policurva cerrada.

2.2.6. Se generan sólidos, a cada margen de la calzada, a través de la extrusión de una policurva hacia su policurva contigua. El resultado es una lista de sólidos encerrados entre dos *stations*.

Para llevar a cabo las zonas de transición (ver Figura 32), se opta por generar sus sólidos como sólidos de desmonte. Es una cuestión de identificación pues el modelo no cambia su geometría. Sin embargo, las mediciones de desmonte tenderán a ser mayores a la realidad.

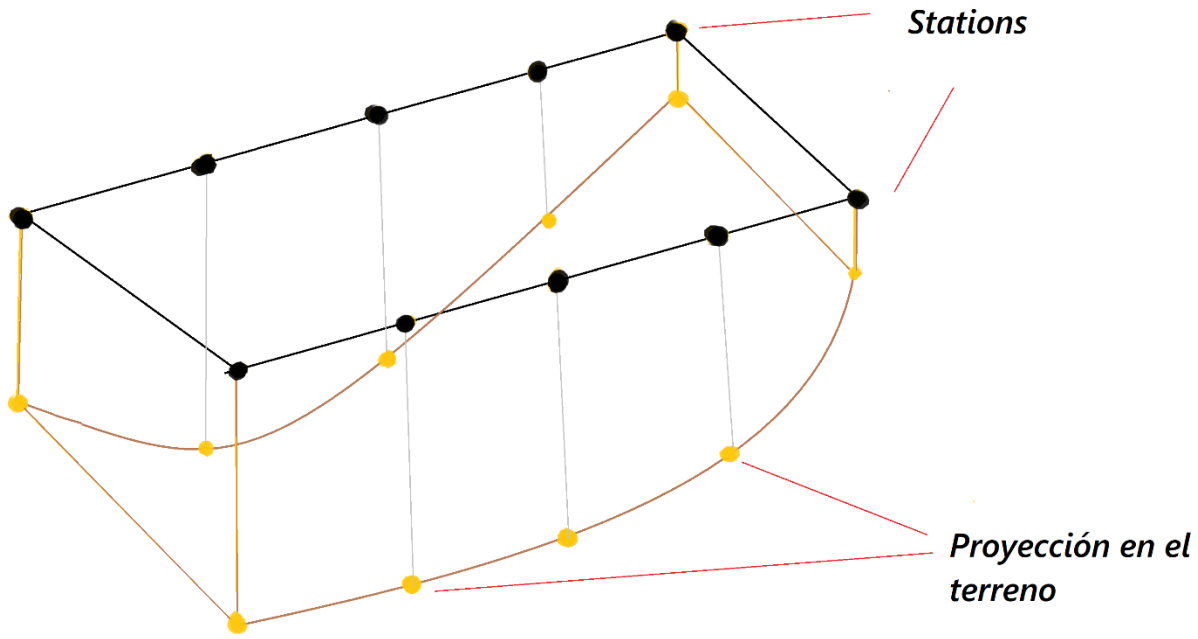


Figura 31. Esquema gráfico del método para generación de sólidos del movimiento de tierras

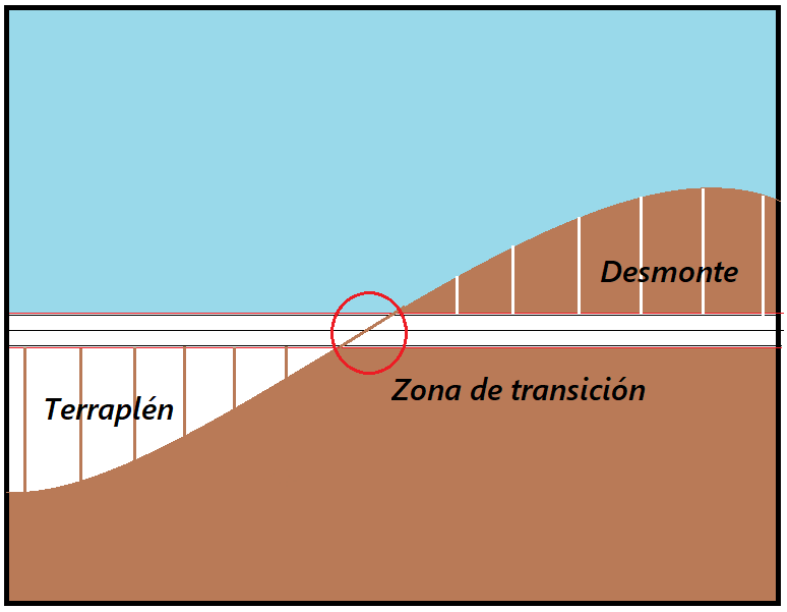


Figura 32. Zona de transición entre terraplén y desmonte (vista longitudinal)

Llamada a todos los inputs:

- Nombre de la obra lineal y de la superficie de referencia del terreno
- PointCodes del ensamblaje necesarios

Con la obra lineal, se extraen sus líneas base baselines y la distancia de sus secciones de control *stations* respecto el origen.

En cada *station* se generan los PointCodes uniéndose para generar el contorno que tendrá la sección de la carretera y sus taludes. Este contorno se proyecta a la superficie de terreno y se une con el anterior creando un polígono. El volumen se genera conectando los polígonos mediante el nodo Solid.ByLoft.

5.2.7. Fase VII: Contraste del algoritmo con casos reales

Para contrastar que los sólidos generados por el algoritmo poseen unos volúmenes coherentes y semejantes a los obtenidos mediante otros procedimientos en Civil 3D, y a mediciones de obra, se ensaya el desempeño del algoritmo frente a un modelo real de un parque eólico, del que se conocen las mediciones en obra y un cálculo estimado, a través de procedimientos manuales del modelo Civil 3D.

El parque eólico está compuesto por una carretera principal recta, con varios kilómetros de longitud, de la que salen ramales que dan acceso a una explanada donde se posicionará un aerogenerador. El caso de estudio se refiere al movimiento de tierras debido a la construcción de uno de los ramales, -no se tiene en cuenta la explanada del molino-. Debido a las características de las intersecciones entre la carretera principal y los ramales, el modelo Civil 3D está compuesto por dos tipos de obras lineales: las que definen las intersecciones, y las que definen los ramales. En la Figura 33 se muestra el ramal estudiado para ensayar el algoritmo.

Debido a que en las mediciones no se consideran los volúmenes de intersecciones -sino que estos se incluyen dentro del volumen de su ramal correspondiente-, se debe realizar un ajuste previo del modelo para que la obra lineal del ramal incluya la parte de la intersección. Para ello se genera una región de sección variable referenciada a la geometría de la intersección (ver Figura 34).

En cuanto a la sección tipo de la región propia del ramal (ver Figura 35), consiste en dos carriles adyacentes -carril izquierdo y carril derecho- tipo *LaneSuperelevationAOR* con un elemento talud a sus extremos tipo *BasicSideSlopeCutDitch*. Es el mismo talud del que se componía el modelo de los ensayos durante el desarrollo. Se advirtió en su momento que era un elemento talud comúnmente usado. La única salvedad es que tal y como está modelado el talud, en este caso, no se hay cuneta de drenaje, simplificando la sección del talud a desmonte. Para conservar la coherencia de secciones, se modela la sección variable correspondiente al tramo de intersección (ver Figura 35), como dos carriles de sección variable con el mismo elemento talud que la sección original del ramal, es decir, el elemento *BasicSideSlopeCutDitch*.

Surge un problema a la hora de invocar las líneas características de código *ETW_Sub* en la sección del ramal -carril tipo *LaneSuperelevationAOR*- necesarias para el modelado del desmonte. Como medida correctora, se opta por cambiar el tipo de carril y establecerlo como carril de transición fijando la anchura y espesores al mismo valor que el original.

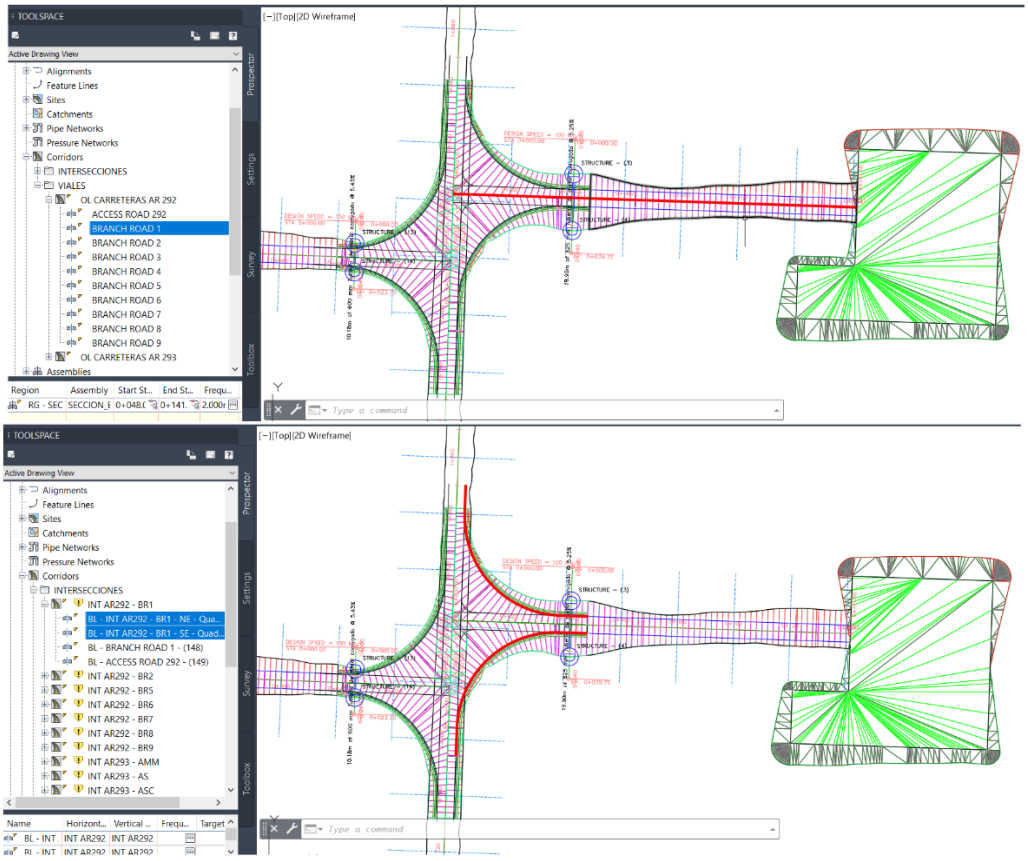


Figura 33. Trazado del ramal de estudio. En la imagen superior se visualiza la obra lineal propia del ramal. En la inferior la obra lineal de su intersección con la carretera principal.

Modificada	BRANCH ROAD 1	BRANCH ROAD 1	BRANCH ROAD 1 ...		0+000.00m	0+141.50m
	Transicion			SECCION_Variable	0+002.68m	0+048.00m
	Ramal			SECCION_Variable	0+048.00m	0+141.50m
Original	BRANCH ROAD 1	BRANCH ROAD 1	BRANCH ROAD 1 ...		0+000.00m	0+141.50m
	RG - SECCION_BRANC...			SECCION_BRANCH_RO...	0+048.00m	0+141.40m

Figura 34. Descripción de la obra lineal Modificada del ramal de ensayo y la obra lineal original del ramal de ensayo

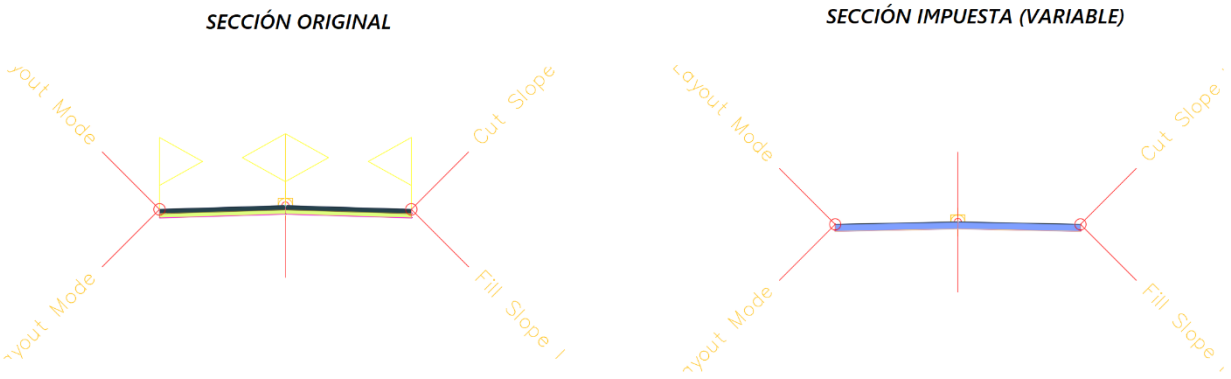


Figura 35. Secciones tipo empleadas en el modelado de obra lineal del ramal de ensayo

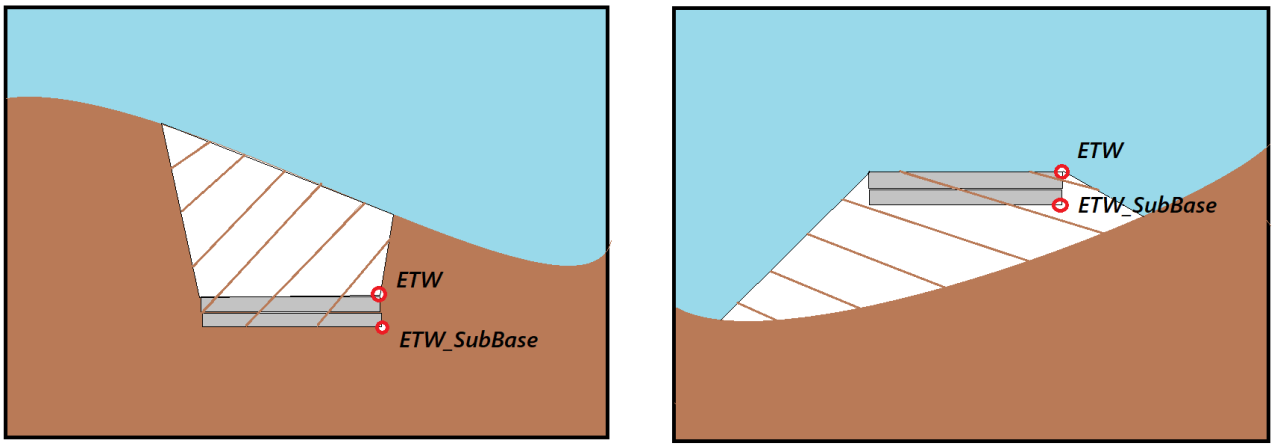


Figura 36. Esquema del movimiento de tierras producido y la influencia de los puntos de código en la medición de los volúmenes

De cara a referenciar los sólidos del movimiento de tierras se desea emplear como referencia el punto más bajo del extremo del carril, denominado el punto con código *ETW_Sub*Base para el desmonte, y el punto de código *ETW* para el terraplén (ver Figura 36).

Los resultados del algoritmo son por un lado el modelado de los sólidos de la Figura 37 y las mediciones de volúmenes obtenidas a partir de los sólidos. Las mediciones son comparadas con las registradas en obra y las registradas a partir de un procedimiento manual realizado en Civil 3D (ver Figura 38).

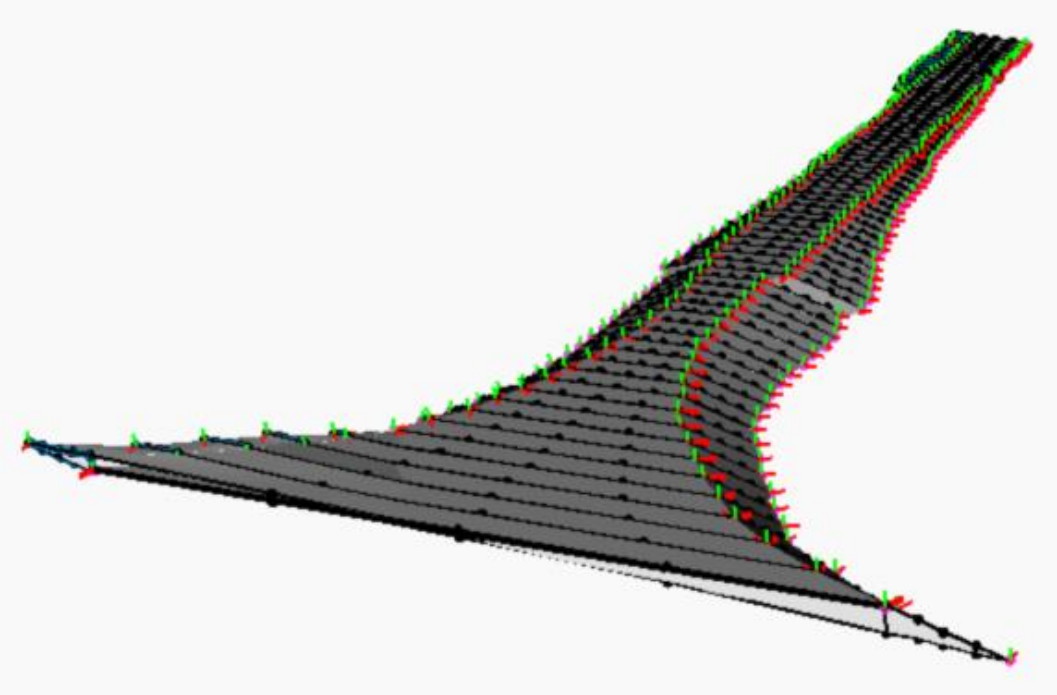


Figura 37. Modelado de sólidos del ramal de estudio en Dynamo

	CALCULADO CON CIVIL 3D		MEDICIÓN EN OBRA 1	
Volumenes (m3)	TERRAPLEN	DESMONTE	TERRAPLEN	DESMONTE
RAMAL	1610.50	98.91	1793.90	94.10
	MEDICIÓN EN OBRA 2		CALCULADO CON DYNAMO	
Volumenes (m3)	TERRAPLEN	DESMONTE	TERRAPLEN	DESMONTE
RAMAL	1661.20	115.70	1618.08	124.55

Figura 38. Tabla de resultados de las mediciones tomadas con Dynamo y su comparación con mediciones reales y mediciones mediante proceso manual en Civil 3D

En vista a los resultados obtenidos:

- Volúmenes de terraplén:

  El volumen obtenido se sitúa dentro del rango de valores mínimo y máximo obtenido en el resto de las mediciones. El volumen obtenido es inferior en un 4% al volumen de la media de las demás mediciones.

- Volúmenes de desmonte

  El volumen obtenido se sitúa por encima del máximo registrado por otros métodos. Es un 22% mayor que la media obtenida por otros procedimientos.

Es conocido que, por el procedimiento seguido en la modelización de las zonas de transición, el algoritmo tiende a incrementar las mediciones en desmonte. El volumen absoluto de desmonte es pequeño comparado con el de terraplén, por esa razón en el caso del terraplén el error relativo es inferior.

5.2.8. Fase VIII: Valoración técnica y económica

Este algoritmo presenta la dificultad de tener que ser adaptado para cada sección tipo al que se ejecute. La adaptación no solo se trata de adaptar los parámetros de entrada si no que puede ser necesario modificar nodos, o añadir series de nodos. Este proceso de adaptación puede complicarse hasta el punto de tener que cambiar el modelo Civil 3D para poder ejecutar el algoritmo, como ha sido el caso en el capítulo anterior. La composición de la red de nodos se determina en función del número de puntos de código que definen el elemento talud. El número de puntos que definen los elementos talud son variables como muestra la Figura 39.

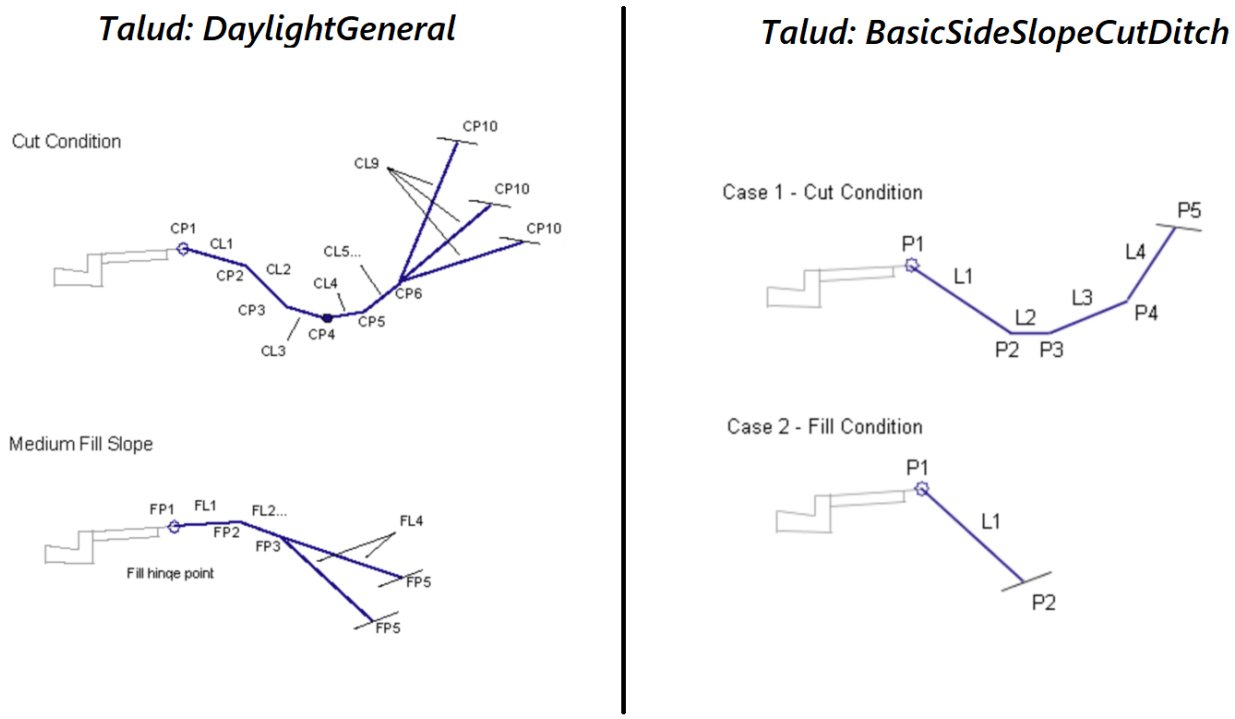


Figura 39. Puntos de código en taludes distintos. Motivo de adaptación de la red de nodos [18]

Durante el proceso de validación del algoritmo se ha observado que existían casos de líneas base -correspondientes a ramales adyacentes al estudiado en el anterior apartado- en los que Dynamo no conseguía ejecutar ningún tipo de acción.

El algoritmo está desarrollado a través de programación visual en Dynamo. Además, la red de nodos del algoritmo posee nodos Python generados para ordenar y gestionar las largas listas de entidades que maneja. Como resultado, el algoritmo está concebido para ser adaptable a los

posibles escenarios, además posee dos parámetros dinámicos que permiten variar la precisión y el tiempo de ejecución según las preferencias del usuario.

Su tiempo de ejecución es inferior a los 30 segundos y cuenta con errores en las mediciones que tienden a aumentar hasta un 22% los volúmenes de desmonte y minorar los de terraplén.

Sin conocer las características ni la valoración técnica y económica de otras propuestas, el algoritmo resuelve el problema de la generación de sólidos para el movimiento de tierras y además de extraer las mediciones a un Excel, es adaptable a otros casos de estudio. Dependiendo del modelo de obra lineal, el proceso de adaptación supondrá un coste de tiempo comprendido entre 0-3 horas.

Con apoyo de la Figura 40 se observa el impacto económico que tendría el empleo del algoritmo frente al uso de métodos manuales. Las horas empleadas en la adaptación a un nuevo modelo dependen en ambos casos de las características y complejidad del modelo, por lo que la diferencia de tiempos se considera una proporción relativa aplicable en un caso general.

(Coste: 20€/h)	HORAS DE DESARROLLO	COSTE INICIAL	HORAS POR ADAPTACIÓN A NUEVO MODELO	COSTE POR ADAPTACIÓN A NUEVO MODELO
Algoritmo	50	1000	1	20
Manualmente	0	0	1.5	30

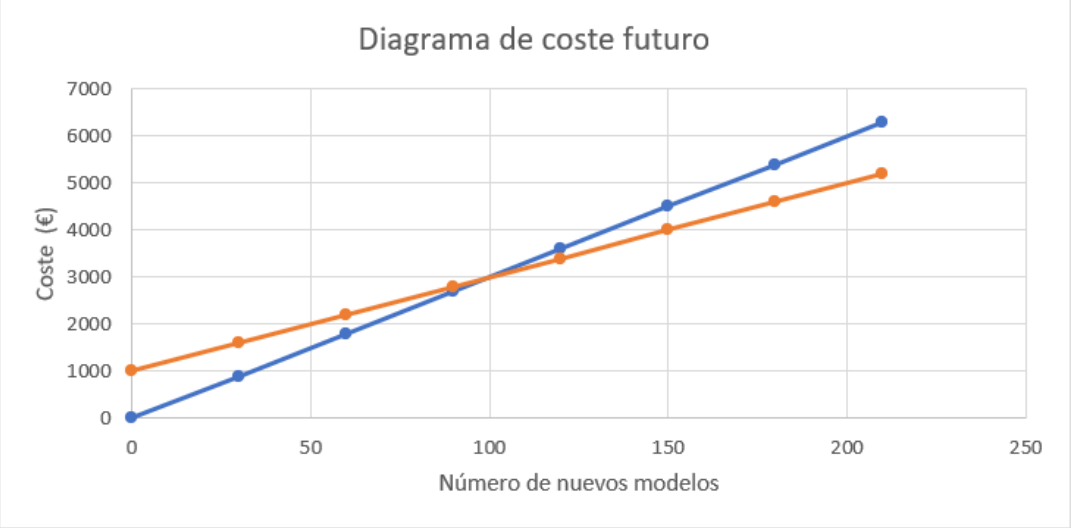


Figura 40. Datos de referencia para la valoración económica del Caso de estudio II

6. VÍAS DE CONTINUACIÓN

Se exponen las posibles líneas de estudio para dar continuación al Trabajo Fin de Máster:

- Unificación de los dos casos de estudio realizados:

  Discretizar una obra lineal (Caso de Estudio I) de tal modo que aplicando el Caso de Estudio II se puedan asignar diferentes tramos a diferentes partidas en Presto a través de la asociación de sólidos mediante Cost-It. De ese modo se puede adicionalmente simular el crecimiento de la obra lineal de forma secuencial.

- Mejorar la modelización de sólidos en la zona de transición del movimiento de tierras:

  Reducir los errores generados en situaciones de transición desmonte-terraplén.

- Agilizar la adaptación del algoritmo de generación de sólidos de movimiento de tierras frente a diferentes tipologías de sección tipo:

  Explorar la generación de una rutina que permita modificar la red de nodos de forma automática en función de la sección tipo empleada

- IFC Road y estandarización ISO:

  De cara a los próximos años, los flujos de trabajo en BIM para obra lineal se adaptarán en función de los avances en estandarización que experimente la disciplina. Es interesante adaptar los algoritmos a dichos estándares y nuevos escenarios de aplicación de Dynamo.

- Discretización completa de una obra lineal de forma automática:

  Implementación de mejoras funcionales al algoritmo de generación de tramos de obra lineal a fin de discretizar automáticamente obras lineales en función de parámetros como el rendimiento de obra.

- Actualización automática de la discretización frente a cambios de diseño.

7. CONCLUSIONES

Más allá del aprendizaje que ha supuesto el desarrollo de este trabajo fin de máster, son destacables las siguientes conclusiones:

- **Los métodos de Programación Visual constituyen una nueva ayuda para el ejercicio de la Ingeniería Civil. En ese sentido cabe destacar que muchas ingenierías de alto nivel (destacamos por ejemplo a Arenas y Asociados por su raigambre en esta Escuela de Caminos) ya lo tienen incorporado en sus procesos y flujos de trabajo.**
- **El aprendizaje de la Programación Visual es bastante cómodo para un profesional con la formación de Ingeniero de Caminos.**
- **Desarrollos DYNAMO pueden reducir considerablemente los tiempos de proyecto de ciertos problemas, particularmente los que implican FLUJOS DE EXPORTACIÓN de datos de carreteras que luego han de ser integrados en un modelo federado o procesados por otras aplicaciones específicas.**
- **En particular, los desarrollos para modificar la constitución interna de una obra lineal (carretera) por variación en sus regiones y los desarrollos orientados a optimizar el proceso de exportación de la carretera mediante sólidos han sido un éxito.**
- **Este TFM colabora a resolver un problema (el de la exportación de modelos C3D a IFC-ISO) que aún no está estandarizado a nivel internacional. Ello da idea de la trascendencia del problema elegido y de su aplicabilidad.**
- **Por lo anterior, destaca que este TFM propone unas vías de continuación de alto interés industrial.**

8. BIBLIOGRAFÍA

[1] «Dynamo Primer 13.2,» 2019. [En línea]. Available: https://primer.dynamobim.org/13_Best-Practice/13-1_Scripting-Strategies.html.

[2] «Dynamo Primer 11.2,» 2019. [En línea]. Available: https://primer.dynamobim.org/11_Packages/11-2_Mesh-Toolkit.html.

[3] «DynamoBIM Refinery,» 2018. [En línea]. Available: https://dynamobim.org/introducing-project-refinery/.

[4] P. Serra, «Knowledge Autodesk Refinery,» 2019. [En línea]. Available: https://knowledge.autodesk.com/support/civil-3d/getting-started/caas/screencast/Main/Details/7b5e54c6-15db-43de-a464-f7128349a5a8.html.

[5] P. Serra, «GitHub: CivilConnection y BIM,» 2019. [En línea]. Available: https://github.com/Autodesk/civilconnection/blob/master/Doc/Linear%20Structures%20Workflow%20Guide.pdf .

[6] buildingSMART Spain, Introducción a la serie EN-ISO 19650 Partes 1 y 2, 2019.

[7] P. MacLeamy, «Curva de MacLeamy,» de *American Institute of Architects*, 2005.

[8] buildingSMART, 2018. [En línea]. Available: https://www.buildingsmart.org/about/what-is-openbim/ifc-introduction/.

[9] G. Conejera, «BuildBIM,» 2018. [En línea]. Available: https://www.buildbim.cl/2018/07/20/ifc-principios-usos-y-mal-entendimiento-de-su-aplicabilidad/. [Último acceso: 2020].

[10] buldingSMART, «IFC Road,» [En línea]. Available: https://www.buildingsmart.org/standards/calls-for-participation/ifc-road/. [Último acceso: Junio 2020].

[11] S. G. d. Oliveira, «Open BIM for infrastructura: Recent standardisation efforts,» de *ISO/TC 211*, University of Maribor (Slovenia), 2019.

[12] buldingSMART, «IFC Rail,» [En línea]. Available: https://www.buildingsmart.org/ifc-rail-the-international-standard-progress-report/. [Último acceso: Junio 2020].

[13] A. Borrmann, S. Muhic, T. Chipman, S. Jaud, C. Castaing, C. Dumoulin, T. Liebich y L. Mol, «The IFC Bridge Project - Extending the IFS Standard to enable high-quality exchange of bridge information models,» de *European Conference on Computin Construction*, Chania, Creta, Grecia, 2019.

[14] Autodesk, «Civil 3D: API Reference Guide 2016,» 2016. [En línea]. Available: http://docs.autodesk.com/CIV3D/2016/ENU/API_Reference_Guide/index.html. [Último acceso: 2020].

[15] I. Rodriguez, «Civilized Development,» 2013. [En línea]. Available: https://civilizeddevelopment.typepad.com/civilized-development/2013/05/subassembly-targets-having-fun-yet.html. [Último acceso: 2020].

[16] tyronebk, «Autodesk Forums - Civil 3D Customization,» 2016. [En línea]. Available: https://forums.autodesk.com/t5/civil-3d-customization/api-to-set-target-mapping-gt-object-name-is-not-available-in/m-p/6473217#M12415. [Último acceso: 2020].

[17] M. Belïen y R. Zutt, «AutodeskUniversity,» 2019. [En línea]. Available: https://www.autodesk.com/autodesk-university/class/Computational-modeling-linear-structure-Civil-3D-Revit-Dynamo-CivilConnection-2019#video.

[18] Autodesk, *Subassembly Reference,* 2020.

[19] Gobierno de Cantabria, «Mapas Cantabria,» [En línea]. Available: https://mapas.cantabria.es/. [Último acceso: 2020].