A Cloud-Based Architecture for BIM as an Asset for Project Management

by

Elvina Constance MERY

A thesis

presented to the University of Waterloo

in fulfillment of the

thesis requirement for the degree of

Master of Applied Science

in

Civil Engineering

Waterloo, Ontario, 2019

© Elvina Constance MERY 2019

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

After more than 30 years of research, better solutions continue to be sought for nuclear power plant decommissioning and radioactive waste management. As some approaches are interesting, improvements are still required in order for them to become generalized solutions.

This thesis is a part of a larger research project that focuses on developing robotic and automated technologies that could support the decommissioning of the nuclear power plant in Pickering, Ontario. The overarching research project is divided into four main tasks: (i) automatic scanning of parts of a nuclear power plant; (ii) creation of BIMs (Building Information Models) from these scans for integrated asset management, and decommissioning planning and analysis; (iii) non destructive evaluation of elements in the nuclear power plant; and (iv) packing optimization of the radioactive waste for its storage and management.

This thesis concerns the second part of the project: creating BIM from the scans (point clouds) generated automatically by a robotic mobile platform. Using Revit® and Recap®, the point clouds are opened in the software and the BIM is created manually from them. A comparison with automatic recognition is made and the limits of both methods are analyzed in order to present the state of the art of automation in this process and the future improvements that can be done.

Dividing this larger research project into four tasks is necessary but creates data management problems, representative of the decommissioning planning challenge. In fact all the data is collected separately with no common storage. Because of the size of this project, it appears possibly advantageous to create an interface where all the data can be shared and accessible by all allowed members. However, the confidentiality of some information must be respected. The security aspect of the developed cloud-based interface is introduced in this thesis and its different functions are presented.

The working environment programmed here can be utilized as an approach for BIM-based asset and project management. To prepare for future modifications and generalization to other domains or fields of construction, it has the advantage of being customizable. Indeed, all the functions here are coded in Javascript and are designed for this nuclear power plant decommissioning project. But other functionalities can be developed and existing ones can be suppressed to suit perfectly another project.

Acknowledgments

I would like to thank my supervisor Dr. Carl Haas for his support and help during the two years of my thesis and for his trust in my work and my contribution to this project. I also acknowledge the support of my collaborators Dr. Sriram Narasimhan and Dr. Giovanni Cascante, who bring a lot of passion into this project. I loved sharing this international experience with my project partners Yinghui Zhao, Stephen Philips, Nicholas Charron and Piotr Wiciak.

Dedication

I want to dedicate this thesis to my friend Auréline Vallée, who will never have the chance to write her own, nor become the brilliant architect and engineer she was intended to be.

Table of contents

List of figures:

List of Acronyms

BIM: Building Information Modeling

OPG: Ontario Power Generation

SLAM: Simultaneous Localization and Mapping

COBie: Construction Operation Building Information

IFC: Industry Foundation Classes

CAD: Computer Aided Design

MEP: Mechanical Electrical Plumbing

ICP: Iterative Closest Point

RANSAC: RANdom Sample Consensu

FM: Facility Management

API: Application Programming Interface

HTML: HyperText Markup Language

NDE: Non-Destructive Evaluation

I. Introduction and Outline of the Thesis

I.1. Background

While other Canadian provinces have natural resources such as coal or natural gases, Ontario turned to nuclear energy in the 1960s and has on its ground all Canadian nuclear reactors, except for Point Lepreau, New Brunswick. However, some power plants in Ontario have now more than 30 years of age and are often shut down for maintenance. Decommissioning projects are taking place but face high costs and very long term prospects. Indeed, because of the lack of information on as-built data, and also because of the "Safe enclosure" which lasts more than 25 years, this process can last more than 80 years. Moreover, because of the hazardousness of the environment and the difficult access to every part of nuclear power plants, robotics and automation have been introduced to these projects. Ontario Power Generation associated with five Universities across Ontario in order to develop a more convincing solution for the decommissioning of the Pickering nuclear power plant. In the University of Waterloo, four different phases of our project can be highlighted: (i) SLAM: scanning from a robotic mobile platform, (ii) 3D BIMs creation from the scans generation, (iii) Non-destructive evaluation of the nuclear power plant, (iv) and packing of the nuclear waste optimization (**Figure 1**).

The first phase enables to have scans of hazardous environments and sometimes without the need for human presence, which is very suitable for decommissioning. The point clouds are automatically generated from the field in the software in minutes. It is an alternative to most laser scans where human needs to be there to change the position of the scanner for each scan and which can last hours, even days for big facilities. The point clouds then need to be manually registered and sent for BIMs creation. The main objective of this phase is to automatically collect data from the nuclear power plant, especially the reactors, without human presence. The third phase helps to have more information on the material composing the nuclear power plant. The technology can be placed on the mobile robot and evaluate with lasers the state of the concrete and other objects. It will procure helpful information for demolition. Finally, the last phase focuses on optimizing nuclear waste packing, one of the biggest problems in decommissioning nowadays. Indeed, only low-contaminated elements can be contained and buried, but several other criteria add some difficulties to the process. With mathematical functions and simulations and taking into consideration the weight and size of the waste, a process will be developed for optimizing the packing of different radioactive elements into containers.

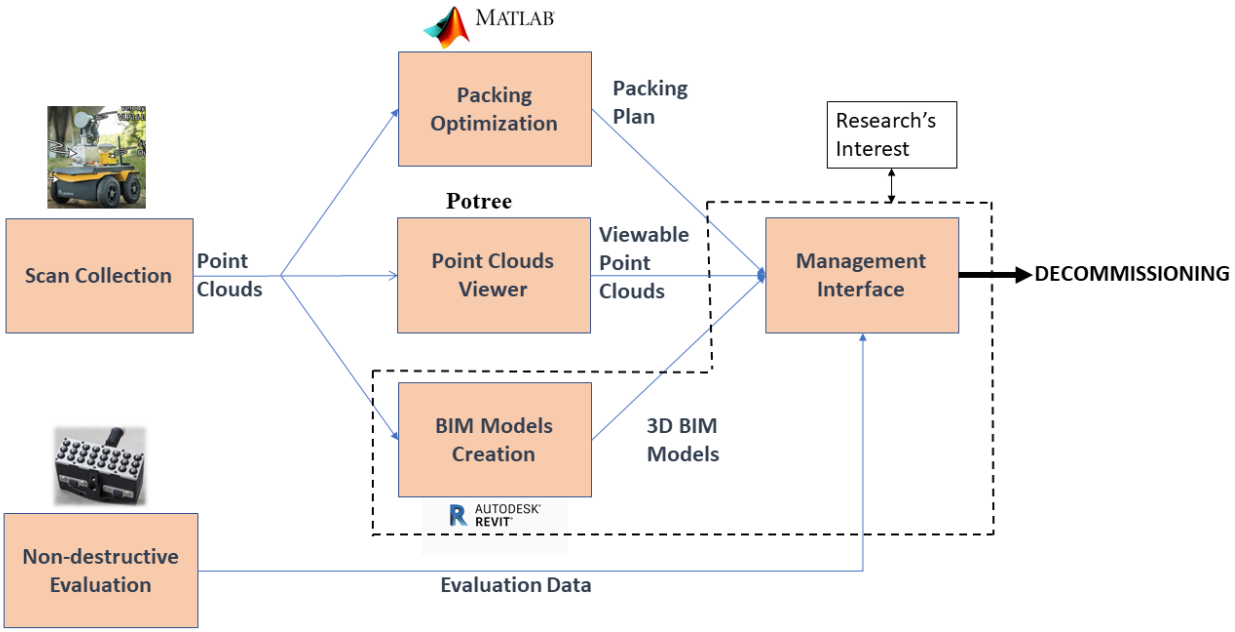


Figure 1. *Presentation of the project.*

In this thesis, the second phase is presented. BIM (Building Information Modeling) has had a huge part in construction projects and has become an essential tool for management and design. It is important to have a general view of the current BIM state of the art. We can divide BIM and its information into different levels and different dimensions (**Figure 2**). The levels represent the interoperability of a BIM project. The more interoperable it will be, the higher the level of the project. The current common level and current standardization of the British government is level 2. It represents a file-based collaboration, meaning that several files of a project can be shared between the different members. But the exchange stays very basic (e-mail, USB devices…). The next level will be reached if a way of integrating the data into a web service, or in a local server, is found.

A BIM project can also have numerous dimensions. The five most classic ones are the three geometrical dimensions, time and cost. But new dimensions such as sustainability (energy evaluation) and facility management (external data) are getting integrated into BIM projects. Knowing the ecological priority of the current society, sustainability and how to integrate analysis to a project have now become a real interest for researchers. And in big projects, linking BIM and management in an intuitive and understandable way have also become a necessity (see **II.3** for more details).

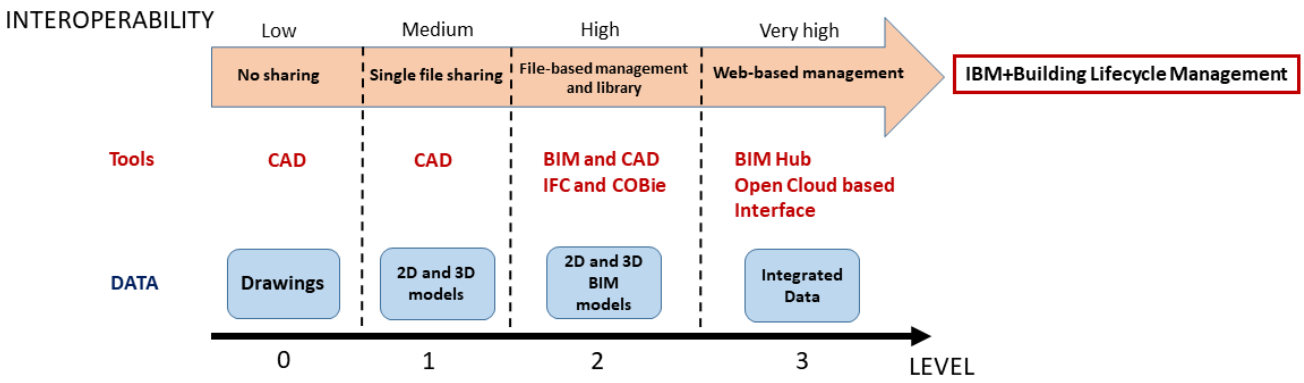


Figure 2. *Levels of BIM.*

A part of the thesis will also be focusing on Scan-to-BIM since the models will be created out of the scans collected by the mobile device. Indeed, for more efficiency and to be closer to reality, generating a scan-based model of the building in BIM seems like an interesting solution. The interest for it has mainly grown for renovation or deconstruction projects where the as-built assessment is necessary and often different from drawings. In the case of decommissioning, this solution appears like the best, since drawings of power plants are often 30 years old and a complete understanding of the infrastructure is impossible because of the complexity of the piping systems.

I.2. Scope of the project

Our project was proposed by Ontario Power Generation (OPG) in order to help the decommissioning of the Pickering power plant, which should start by 2025. This nuclear power plant is the biggest in Ontario, with eight reactors (**Figure 3**).

Figure 3. *Pickering Nuclear Power plant (from SNL-Lavalin Website)*

However, the facility is huge and our project is still in the research phase. Prototypes are not efficient enough to offer a good solution for such a big place. And they also need to be tested on smaller spaces to be improved and become ready for the final objective. Therefore, the prototype and the models are done on the structure lab of the E3 building in the University of Waterloo (**Figure 4**).

Figure 4. *Structure Lab in E3 building, University of Waterloo*

This space is chosen because of its numerous pipelines and its industrial aspect. These similarities with a power plant are interesting for us because the main errors met during research phase would be expected in a power plant facility. That allows us to anticipate future errors and solve them before deploying our project to the nuclear power plant.

To decommission this facility, OPG has in mind to create models from the laser scans and use them to operate. Indeed, this facility's construction started in 1966 and continued for twenty years. The remaining drawings are not enough to provide the necessary information for decommissioning. Simple CAD models were also developed but do not gather all the data needed (**Figure 5**). BIMs from laser scans offer an interesting multi-dimensional solution to collect all the data needed because they represent the as-built facility. By the development of an interface, the operation and deconstruction phases' management will also be facilitated.

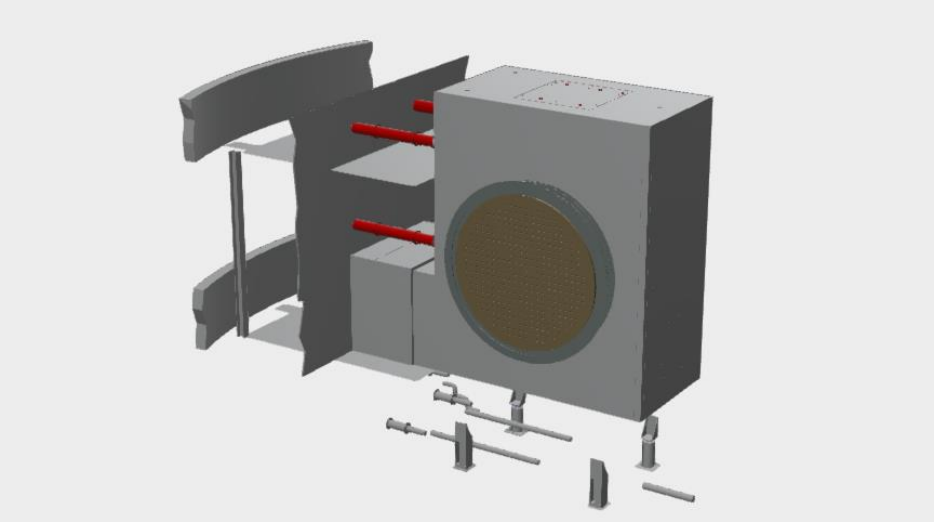


Figure 5. *A very simplified CAD model of one of the reactors.*

Moreover, the most important part of the decommissioning phase is waste packing. Optimizing it in the containers is extremely important in order to limit their numbers, in order to reduce costs and underground volume. This process can be done computationally, based on the shapes of the waste or the elements of the buildings. The BIMs could provide this information and their similarity with the as-built facility is essential. The cloud-based aspect of the interface offers also a good solution for storing and exchanging the data.

Also, once the models are created, very few solutions exist to use them for project management. In our case, the main objective is to use them directly for decommissioning planning and operations. In order to achieve that goal, we have to think of an approach not only to gather the data, but also to use it during the whole decommissioning project. This thesis has to propose an architecture which enables to view the models, update them and use their properties. And it has to take into consideration that the project is expected to last more than 70 years. Developing a cloud-based solution is interesting for easy updates and keeping track of all the uploaded data.

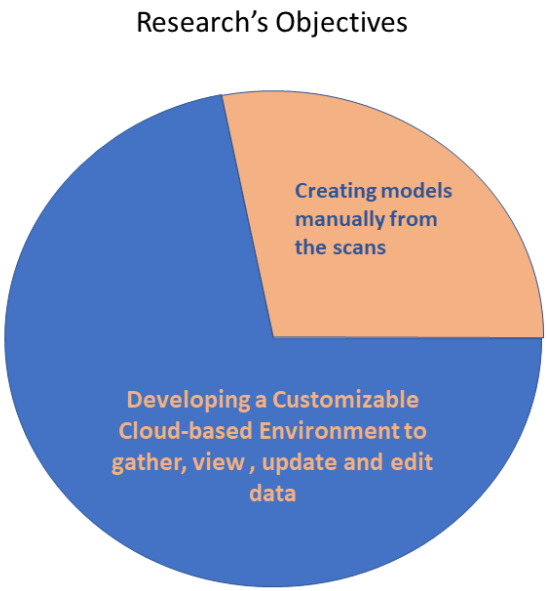
I.3. Objectives of the thesis

The objectives of the research are to give OPG as-built models of the facility in a useful collaborative online environment so they can plan the decommissioning phase with the best information in their hands.

The first part of this thesis is creating BIMs from laser scanning. The human work would thus be reduced. This leads to a cut of expenses and time spent in the facility. Also, a nuclear power plant is a hazardous environment. Therefore by choosing this approach, not only are human errors

avoided, but also the employees' safety is prioritized. But designing models cannot stay the only objective of the research because a solution has to be found to share them.

To be able to see and edit a model, a BIM licensed software is needed. For some companies, which have no interest or budget in buying a license to see the models, it is important to offer another solution. The second objective is then defined as to share models openly but without losing confidentiality. By deploying an interface for the management during the decommissioning phase, another finality of the research is to give OPG all the data they need, and the possibility of updating it during the whole process. Knowing that this decommissioning phase is planned to last until 2100, the updates of the BIMs and the data will be numerous. This had to be taken into consideration for this research and had to be a priority for the development of the interface. It will also be developed with useful functionalities for the project, which will be customizable in order to suit perfectly the demands. In the prototype, the classic functionalities will be developed with the possibility of making changes if needed. The main challenge is to assure confidentiality of the data. When it comes to cloud-based infrastructures, it is always hard to assure security. We use a three step authentication process and deploy the interface at a local level. Another challenge is to be able to customize most of the application and functions programmed. In our case, we work on a decommissioning project, but we want to keep this solution open to other construction or deconstruction projects. We have to keep this in mind when we code our cloud-based environment.



I.4. Methodology

The methodology for this research follows a logical path (**Figure 6**). After laser scanning the structures and doing the registration, the point clouds have to be integrated into the BIM software. The BIMs are designed from these scans. Two solutions can be chosen: manual recognition or an automatic one. Because the study case is a single room, the manual solution is chosen for this research. It allows entering the properties of the recognized elements simultaneously. It also is not in the research's interest to develop an automatic solution, since commercial ones already exist. Some of this software uses algorithms developed by research team related to that to which this thesis belongs. Another interest is to be able to choose the Level of Detail (LOD) appropriately. With uninteresting elements in the room and on the scans, a manual approach enables not to model them. An automatic approach will represent them or demand a previous task where the useless elements are filtered from the scans. The time saved for modeling with an automatic recognition could thus be lost. Better performance of automated recognition is necessary and is an entire research interest by itself.

The interface is then developed, with all the functions needed for decommissioning management: scans visualization, models viewer, live sensor data graph, schedule and link to the documents of the project. Before coding the interface, it is important to choose the coding language and the software used. Here quick research on Forge Viewer and coding websites led to the choice of Javascript, HTML, and Node.js as the languages for the development of the interface.

Once it is operable, the models have to be uploaded in order to be able to view them. The data linked to the project also needs to be integrated. For the research, demonstration data or random numbers are used. Indeed, in the research case, the data is mostly introduced to have a visual aspect of the final result. But the real data is not shared by OPG yet, and the prototype is based on the structural lab, not on the nuclear power plant. The models and data get linked in the interface, which gathers every information from the project and let anyone who has access to it to see, comment, modify and export accordingly to their status.

The last phase, which is considered as future work, would be to deploy the interface on a local server internal to OPG, or on a website if the confidentiality is not necessary. This work is also on the stage of a prototype, and developers have to work on a more commercial aspect of it. It can also bring more convenience and make it more intuitive, thus easier to work with.

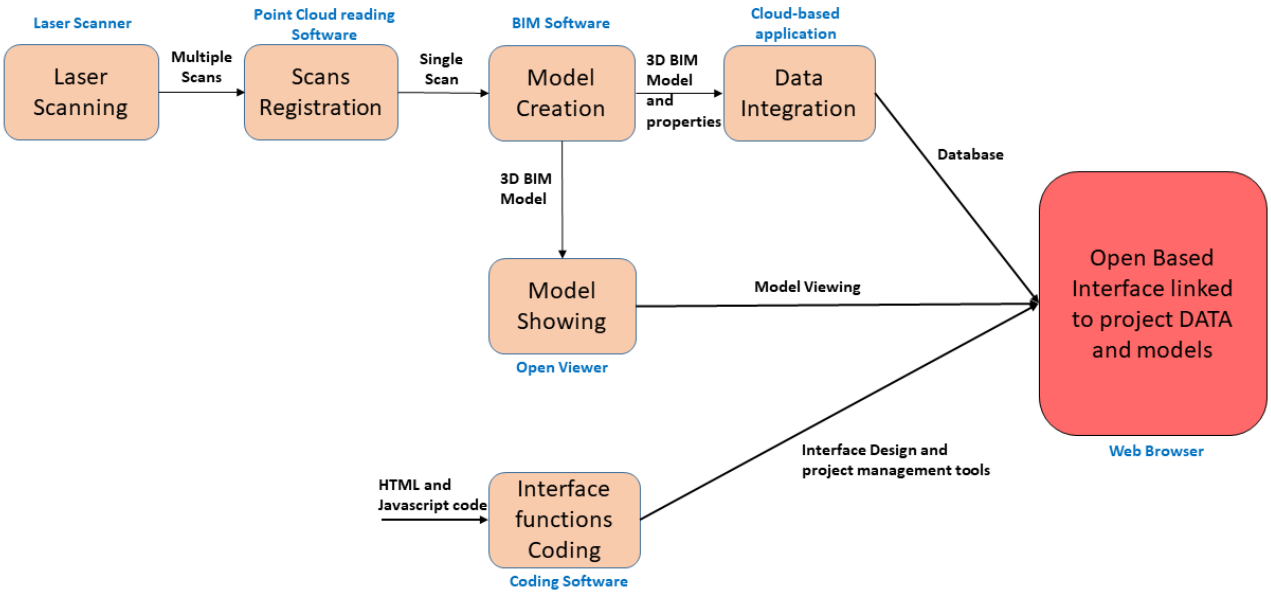


Figure 6. *Methodology of the research.*

I.5 Structure of this thesis

First, the different concepts of Scan-to-BIM, Scan-vs-BIM and automatic recognition will be introduced in a literature review. The focus will also be on BIM as a tool for construction management and the state of art in integrating BIMs in facility management. The limits and advantages of current cloud-based BIM solutions will also be presented. And a state of the art of the issues of nowadays decommissioning will be discussed.

The second part of this thesis will have an interest in creating BIMs from point clouds. The research is based on the case of study of the structure lab of the University of Waterloo, in the building E3. The manual chosen approach is presented. But to certify our choice, the automatic recognition modeling will also be tested (a Scan-to-BIM extension in Revit) and a comparison between both solutions will be made. Both methods have advantages and limits for decommissioning, and it is important to highlight them in order to have the best information in hand to take a future decision.

When the models are created, the next step is to share them. The problem met instantly is the need for a license for BIM software. Clients who are not in the design domain will not pay only to view models. The development of an interface for managing the project and sharing models have thus been decided and will be presented in the third part of this thesis. This interface will not only be able to show the BIMs but also help the clients with managing the project. It has several other functions that facilitate the progress of big projects that are often hard to process

because of the numerous actors, documents, and the density of information transmitted. As presented earlier, by programming such an interface, we could reach level 3 of BIM.

II. Background and Literature Review

II.1. Scan-to-BIM and automatic recognition improvements

As construction management improves, BIM (Building Information Modeling) has had new functions. First used as a tool for design and constructions phases of a project, it has now had been extended to quality controls and refurbishments. In fact, Scan-to-BIM has been developed in order to extract as-built information to create a 3D model [**1**]. While some work has been focusing on BIMs from photogrammetry [**5**], [**6**], laser scanning has quickly become more interesting because of its ability to collect directly 3D information in a single scan and give as-built information. However, this technology is now already used by companies for big components of a building, such as walls, openings, and roofs. The most recent research has been focusing on other solutions:

- Detection of smaller components, like MEP cylindrical elements. [**1**], [**3**]
- Automatic recognition of all elements of a construction project beyond their geometry.
- Quality control of the point cloud used for 3D model generation. [**4**]

Differentiating the different techniques is important. Scan-to-BIM and Scan-vs-BIM are two solutions based on laser scanning and 3D models and can be matched together for better results. The first one has four strategic steps:

1. Data collection, from laser scanning.
2. Preprocessing: the point cloud is registered and if needed, filtered.
3. Geometric 3D modeling: the simplified 3D shapes of the elements are modelized.
4. BIM: categories, properties, material, and relationships are assigned to the detected objects.

Scan vs BIM has a different approach to this problem. It compares a point cloud from an as-built facility to its BIM. Therefore, it is mostly used for quality control, life cycle monitoring or progress on a project. But because it is limited by an accurate 3D registration, it can also be combined with a Scan-to-BIM method for more robustness or precision [**3**].

These methods opened a bridge to new solutions in this domain, but are not the focus of current research. In fact, well developed and commercialized nowadays, they are not the point of interest anymore. Now the research tries to improve the 3D modeling from laser scanning by looking to the automatic aspect. This is a step forward from the technology in order to eliminate human interaction and therefore some errors or imprecision that it can bring to a project. It also allows reducing costs, when 90% of the budget goes to modelling objects from scans [**24**].

The first research on this topic was done on CAD models [**6**]. But once accurate solutions had been found, the transfer to BIMs was fast and did not have to change much in the process. A review by Tang et al. in 2010 [**7**] presents a limited approach to BIM objects recognition. At that time, geometric shapes could be recognized by: Gaussian method for curves; Hough transforms or Ransac for planes. The further steps introducing to BIM objects was then introduced by 2013 [**8**]. By basing their research on improvements of Bosché and Haas in 2008 [**12**], [**8**] and [**11**] have proven their contribution to automatic recognition from laser scanning. Two types of registration offer the perfect option for detecting accurate shapes: a coarse registration, to roughly align data; and a fine registration for optimizing the alignment of the data. Some research has focused on only using fine registration or coarse registration, but find themselves limited. Coarse registration is usually based on points picked manually, and therefore the final results depend on the choice made at the beginning. Most research use matching points method, with a manual setup of the three first pairs. It is also possible to use a target based or a feature-based approach. However, the feature based alone can be limited to complex structures. However, combining it to another coarse registration can give suitable results [**22**]. Concerning fine registration, most of the research follows the Iterative Closest Point approach [**10**], first introduced in 1992 but developed by Bosché et al. [**12**]. It can also be a base method for more complex registrations, like in [**13**] where a multiple origin approach is developed. The next steps then feature extraction and recognition [**14**]. But combining these two methods has proven its worth. In fact, Bosché [**23**] and Wang et al [**15**] eliminate the imperfections of the coarse registration by adding as a second step the ICP fine registration. Based on [**5**] and [**16**] they use all the data and not just some points and can recognize 3D BIM objects automatically. However, the limits of this approach are that only primary objects, such as walls, roofs, and openings are correctly detected. Ahmed et al [**17**], [**18**] get around the 3D problem by dividing the point cloud into "slices" and applying Hough transform to detect circles. Then, the centers of the circles are matched to extract pipes with few millimeters precision, with a faster registration and therefore a reduction of costs.

The issue of large point cloud makes it difficult to register correctly. This is why Gao et al [**19**] try to use another method to register the whole data without coping with this problem. Their approach is to progressively detect objects but focusing on small parts of the point cloud at a time. The same issue was the main focus of research made by Ochmann et al. [**20**]. By only getting the indoors data, and dividing it by rooms or space types, they do not have to manage a large point cloud, and can even filter all the outside information and noise. Czerniawski's et al [**21**] approach is based on reducing the point cloud size to not have to deal with big point clouds. Indeed, they know that recognizing planes is easier than small components, so they use the Gaussian approach to a density-based clustering algorithm to find planes in point clouds and delete them. This way they only keep other types of elements on which the recognition can be done. Another solution will be to only focus on the shapes of the building [**22**]. For more complex structures, sometimes the registration is difficult and limited even with the simplified approach. This is why [**23**] uses a semi-automatic method, by detecting planes with RANSAC, removing the noise and defining

only the boundaries and the facility. But all final recognition and classification are then made manually.

Some of the latest research (2018, [**9**]) shows the limits of coarse registration. As the fine registration is commonly made by ICP method, their second step is to add a new dimension to the coarse registration for more accuracy. By integrating the fourth plane in the congruent method, they add robustness and improve the final results.

As the automatic 3D BIM objects recognition and classification are still in progress, some results are already promising. The shapes of buildings and primary objects can now be detected. Some efforts still need to be done concerning MEP structures and other secondary objects. In our case, nuclear power plants sadly have a lot of pipelines and hidden elements, which makes automatic detection harder. (See **Chapter III-3**). Agapaki et al [**24**] highlight the remaining issues of automatic extraction by evaluating Edgewise®, the most developed automatic recognition software on the market. They point out the importance of focusing on the most common object types found in a facility to narrow the automatic recognition and upgrade the precision. However, even with this approach, the total labor hours are still high and precision too low compared to the expectations (62% in the best case scenario, but 22% in the worst case).

II.2 BIM and asset management: state of the art and remaining gaps

BIM is part of current construction projects but is mostly developed and used during the design phase. However, for built facilities and in our case of decommissioning an already built nuclear power plant, it is interesting to know where the research stands for BIM and management during the operation and maintenance phases. Liu and Issa [**29**] interviewed practitioners and showed that half of them highlighted the importance of BIM for operating and design phases. In the US alone, studies showed that a loss of 15.8billion US$ in 2002 because of the lack of interoperability between the actors of the project, and the weak consideration for maintenance and operation during the construction and design phases. The top priorities of BIM enabled-FM systems are: locating components, facilitating real-time data access, checking maintenance and automatically creating digital assets. Unfortunately, none of these are easy and can be done directly.

The most complete article in this topic is the review from Edirisinghe et al [**25**] published in 2017 and presenting the work on this topic based on more than two hundred reviewed articles. However, only eighty were focusing on BIM in FM. It is settled that IFC format was created to add interoperability to BIM software. But the files could not be opened or linked to facility management software. And they are very heavy, therefore not the best solution for exchanging information. COBie files were then introduced in 2005 in order to address this issue. Presented as an Excel table with different sheets, it lists all properties of the project and can be read by all. It can also be easily linked to facility management software by different methods [**27**]:

- The most common one: Ontology-based. It extracts heterogeneous information and creates a common model using ontology models. This means classifying and exchanging information by specific classification or vocabulary.
- Schema-based: integrate heterogeneous data and help information exchange between them.
- Services-based: request and extract the heterogeneous model for integrating and converting heterogeneous data
- Process-based: guideline to integrate heterogeneous information into a standard model like IFC (it is mostly manual).
- System-based: open libraries, components, commercial software are used to integrate heterogeneous data.

For example, Kang and Hong link the data from Cobie files to the BG-ETL software (architectural software, also used for maintenance, management, and operation) by using a GIS (Geographic Information System). By this approach, they can have all the information from BIMs in their architectural software. However, the process is not direct, since they have to go through the COBie file, and define classifications.

As most agree on the necessity of creating BIM-enable systems which will integrate BIMs and FM data together and then transfer them to FM software, they also show that COBie files are not the best option for management [**28**]. Mostly because they are unreadable for big projects, because of the numerous lines, columns, and sheets, which limit the rapidity of the process, and induced errors. But also because safety problems appear when it comes to the exchange of information [**26**]. In fact, COBie files are Excel sheets and can be read by anyone who has access to it, also it is hard to create confidential information since all the data are contained in it. Therefore, other solutions have to be found. Unfortunately, most of the research done has been to evaluate the IFC and COBie solutions. Only a little research is currently focusing on the creation of interface or new systems, but they are still in progress.

Also, since Scan-to-BIM is improving and aiming full automatic recognition, we can, therefore, imagine that models will be updated often and that the data exchange has to be quick and precise. Singh et al [**30**] highlighted the importance of as-built information exchange an integration but think that the current technology is not good enough for providing facility management information, integration, and interoperability of the different systems. Indeed, Lin and Su [**31**] present a web camera and mobile devices system, with BIM, integrated to it, but also point out the long downloads, errors and constant updates of the BIMs.

II.3. Cloud-based collaborative models delivery with BIM

BIM as an asset for management during the operation and maintenance phase has proven to be necessary and demanded by practitioners. In a study made by Redmong et al [**38**], members from construction companies and thus possible future users are interrogated concerning their expectations on BIM and its interoperability. A cloud-based interface could be a solution if security and training are assured. It could solve some issues of COBie and IFC highlighted in **II.2**. Indeed, IFC files are written in XML, which is very hard to be encoded and understood for a novice. And these files are usually very heavy so hard to share. COBie on the other hand is hard to read and is indecipherable for big projects. An alternative to these American standards had to be found. This lead in 2010/11 [**32**] to newly published articles presenting cloud-based approaches. Not only could it lead to time-saving but also costs reduction. This approach could be an important tool for big projects.

Three types of cloud deployment can be used [**35**],:

- SaaS (Software as a Service): using a cloud provider, with available mainstream packages offered on the cloud. It is an already prepared application.
- PaaS (Platform as a Service): using SaaS but with additional functions than the ones provided. Users can create and then deploy their own applications thanks to the provider's services. It gives the tool to build and customize applications.
- IaaS (Infrastructure as a Service): renting a computing and storage infrastructure to the provider but everything else needs to be done to have an application as a result.

The importance of sharing is important in any stage of a project. The design phase is when all the key decisions are made and need to have all the data possible. And not having interoperability during maintenance and operation seems irresponsible. However, only recent research focuses on operation and maintenance phase, most of the early research is interested in construction and design tasks, as explained in the review from Wong et al [**32**]. Indeed, Chuang et al [**33**] are detailing a Graphical User Interface (GUI) capable of visualization and manipulation of BIM objects. As they create a platform to view models and share them, they do not add more functions to facilitate project management. It is also the case of Goulding et al [**34**] who create a construction site simulator based on cloud interface developed in C#. It is used to gather data collected by mobile devices But it is limited to helping design and schedule the project by creating a space capable of grouping several information formats that could not be gathered otherwise. This is one of the main advantages of the cloud. In big projects where different field and actors are interacting, a cloud-based solution enables to store and share files without converting them. That is one of the interest for Juan et al [**35**] who developed OpenBIM, a platform where data is independent of software or format. Trying to improve the only neutral existing solution which is IFC, they developed a layered cloud-based platform. They divide the information into five layers: infrastructure layer, data layer, model layer, application layer, and interactive layer. Whereas model, data and infrastructure layers regroup the different data and resources collected, the interactive and application help the user with functions and possibilities to use this data.

Jiao et al [**36**] combine this approach with augmented reality and use the cloud as a storage for all the data collected on site. Not only does it help with data sharing but also with the organization and management of the project schedule. All the data is stored in one place, so it is easier to know where to get it. Meza et al [**37**] go a bit further by improving the AR technology in order to compare BIM design and real on-site data. There are still some limits to the process, like applying this solution to 4D or more BIMs and big projects. Moreover, AR, like RFID (Radio Frequency Identification), barcode and Laser tagging technologies, is extensively used in the field but does not have a good efficiency since it needs to be adapted to every facility. Therefore, these solutions are rarely generalized and prepared for any sites, and need to first be calibrated with on-site data [**43**], which is a loss of time and effectiveness.

Because a cloud-based interface regroups different field and all types of works and anteriority of the members, there are also doubts concerning education teaching. The question of necessary formations so anyone could understand the data in the cloud and use it is asked [**32**].

In a nutshell, the biggest advantages to Cloud-BIM are: accessibility, scalability, reliability (of the storage), advance interoperability for BIM application, automatic back up, cost reduction and green analysis [**40**], [**42**]. However most of the research is done for design and construction phases, and there is a lack of solutions for maintenance and operation phases. Matthews et al [**43**] are one of the rare research presenting a solution for on-site maintenance. But it depends on manual checking and updates, meaning that an engineer needs to upload all the data manually in the cloud to update the BIMs. Also here, the data needs to be converted before being integrated into the cloud, which brings an intermediate stage where information can be lost.

Some other key concerns are still remaining. The security is the main issue when talking about sharing data. And the cloud is basically an open space where all data can be seen by anyone. It already is a problem for confidential projects. Developers and engineers have to come up with new technical solutions and legal documentation [**40**]. Big concerns are also related to the access to the data, the loss of the privacy and the loss of ownership once the data is put in the cloud. The example of Google® owning the data uploaded to Google® Drive is set in people's mind and show the high uncertainties and vulnerability of this solution [**39**]. It is obvious that at least an authentication process needs to be integrated into the interface to assure security and help set rules in terms of access and ownership. Yang et al [**41**] present another approach on a case study of a big company. They use the different servers of the company, that they divide into three layers: platform communication, distributed file system, and distributed database. They secure some privacy by only opening the platform communication layer to a web server and data sharing. The distributed file system and database stay private and safe. However, like this one, very interesting approaches need to be improved. Yet they are limited to the IFC formats and thus not using the heterogeneous advantage of the cloud, which is its main interest [**42**].

Finally, Alreshidi et al [**40**] highlight the fact that all the proposed collaborative solutions are more focusing on the technical aspect and the advantages of the cloud rather than the business and

legal aspects. They imply that governance needs to be thought of in these scenarios. In their case, they create GovernBIM, a governance BIM platform with UML (Unified Modified Language). Using cloud-based platforms raise new legal questions. Adapting business contracts ([**38**], [**39**]) by adding a section concerning the cloud, the data's ownership and security rules seems necessary in this situation.

II.4. Objectives and Issues of current decommissioning projects

With its hazardous aspect and the large size of the projects, decommissioning a nuclear power plant has many issues. First, it seems to be an individual matter since the license holder is responsible for it and has to come up with the best solution possible [**49**]. Also, not only are the nuclear power plant all different, and thus their waste management and decommissioning will be proper to each one of them and vary depending on the case and the situations [**46**]. But European countries for example also have different regulations and methods concerning nuclear energy and the way to act on it [**45**]. And differs from the International Atomic Energy Agency's standards as well. And that does not include the accidents that can take place and after which quick solutions need to be found, changing the way of thinking decommissioning [**46**]. Several points need to be answered, and could be rushed in case of incidents:

- How to reduce waste volume effectively?
- How much storage capacity is needed?
- How to choose waste that needs special treatment?
- How reliable will it be long term?
- How to choose the radiation threshold?
- Where is the waste going to be stored?
- How to divide elements by their level of contamination when we know it is not homogenous? [**47**]

It also needs to be taken into consideration that the volume reduction facilities are built specially for decommissioning phase and will have to be added to the contaminated waste in the end.

The need for research to bring solutions to reduce the costs and improve the efficiency of the technologies is high. Indeed, some decommissioning projects talk about billions of dollars [**57**]. Choi et al [**48**] present a site architecture to facilitate and reduce waste management and fasten the decommissioning process. They build a new nuclear power plant on the same site as the one being decommissioned at the same time. They thus process cycles of 72 years for building and decommissioning and reduce the waste of 90%. But this is a very risky process since the waste has to stay safe and not be contaminated by the new nuclear power plant in action. Therefore, the management of both construction and decommissioning is very complicated. It is also applied to one particular nuclear power plant, and it is difficult to see how general this solution can be.

Since the first decommissioning plans, teleoperation robots have been seen as the perfect solution for manipulating nuclear waste ([**58**],[**59**]) and limiting human interaction with the radiations. Most of the robotic technologies are used for inspections or simple tasks such as holding and stacking. As the first robot capable of handling waste management has been developed in 1958, the first solutions for inspection appeared in the 1980s'. But at that time sensors were not good enough so it was impossible to develop autonomous robots without taking big risks [**41**]. Lee et al [**44**] present a two-parts robot: a mobile device for positioning and acting inside the nuclear power plant and an exoskeleton to give instructions and input. Marturi et al [**50**] highlight the importance of automation in the process by comparing tasks (three tests of grasping and stacking) effectuated by humans and robots. Humans are more distracted by the environment, and that is without taking into consideration the exposure to radiation. Chen et al [**53**] focus on the inspection function by developing a monitoring robot linked to a cloud-based platform to store the data collected. Following [**54**]'s approach, they monitor real-time waste and have faster access to the data in order to facilitate management and analysis.

Multi-tasks robots are still rare and improving. Qian et al [**52**] develop a teleoperated robot with nuclear and chemical detection and capable of collecting samples. They enable different functions thanks to three levels of detection, giving which sensors are working at the same time. However, they do not advocate complete automation because of the complexity of the mission planning and the precision of the sensors that this would induce. In such an environment, it is almost essential to have robots capable of responding to random situations and be fully automated [**51**] but a compromise has to be done between automation and time spend on the planning, which also means human intervention.

However most of the research still needs to improve, especially technologies need to be tested in a radioactive environment to see the impact radiation have on the robotics [**44**]. Robots are often solutions imported from other field and are not quite adapted to nuclear, missing, for example, some useful sensors [**50**]. Vision-guided state estimation could contribute to collect and send the missing input but still needs to be improved. Another issue is communication with the robot. Nuclear power plants are big facilities and some problems can appear if wireless communication is chosen [**55**].

Little research focuses on how to classify the waste, which is another objective of decommissioning. Suffat et al [**56**] use a single monocular camera and Otsu and Suzuki's method to extract the shape of waste elements placed on a sorting table and then classify the waste. But research still needs progress for these tasks. Most of them are not focusing on what happened after demolition, meaning the process of classifying and packing nuclear waste.

II.5. Conclusions on the state-of-art and lead to this research

Thanks to this literature review, the remaining gaps of the state-of-art in automatic feature extraction and BIM as an asset for management can be highlighted.

When it comes to automatic recognition and Scan-to-BIM, the research is currently mostly focusing on secondary and cylindrical elements. As it is now pretty convincing for the primary elements and shape of the facility, improvements still need to be done concerning smaller objects. This is why a manual approach is chosen in this thesis. Also, the quality of the point cloud plays a big part in the recognition of the elements. Laser scanners are very precise but are also heavy files with a lot of information. This means that the automatic process will have to compare much more points. These processes can be very long for uncertain results. And a gap is still remaining concerning objects that do not have classic shapes. However, an automatic recognition test is also done in order to compare both methods and discuss the advantages of state-of-art processes.

Concerning BIM as an asset for management, the research is still very recent and need time to improve and find solutions. Some approaches start to emerge for linking BIM and management software or using BIM as a management tool but they are too isolated to become a generalized solution yet. This comes mostly from the fact that this topic has been questioned very recently and that research has not found convincing solutions yet. The gaps in that field and the possibilities to fill them are therefore numerous but not well developed. The thesis is thus presenting an idea of a solution for linking BIM to management in order to bridge this gap in this decommissioning case.

III. Scan-to-BIM: From laser scans to 3D models

As presented in **Chapter I and Chapter II,** many approaches can be chosen for Scan-to-BIM. Nowadays, two solutions exist and have their advantage and inconvenient: manual one or semi-automatic one. Even if some research is close to full automation, a part of human intervention is always necessary. To favorize decision making and precision, a manual method is preferred here. It is a long process with important costs but enables to create complex 3D models with a high level of details. It demands patience and concentration from designers to be the more precise possible. This approach is first presented. But even if automatic recognition is not fully developed yet, this approach can be chosen for big structures or envelops design. Indeed, details would be hard to model during that process. It is chosen in cases where the level of details expected is low and time and costs can be saved. In this project, the most important elements to detect are pipelines and structures. Therefore, a test of semi-automatic extraction is also done. And a discussion will be presented at the end of the chapter to compare both methods.

III.1.Converting the Scans

III.1.a. Scanning and registration

As presented in **Chapter I**, the Pickering Power plant could not be chosen as the case of study. It will be open for tests after the complete shutdown and once the decommissioning has started, probably in five years. Therefore, the structure lab is chosen as the case study of this thesis. Because of its industrial aspect and the numerous pipes, it has some similarities with a nuclear power plant. Working on a space with the same aspect as the power plant is very important. If errors occur or if the robot is not well calibrated, a solution will be found and will enable to have a system prepared and adapted to hazardous environments. The impact of radiation on the system will be one of the only remaining interrogations.

To scan the lab, a laser scanner and targets are used. Before scanning, a plan has to be done concerning the right place to put the targets. Depending on the size of the room and the precision of the scanner, five to twenty targets are necessary. Once the place of the targets is decided, the different positions of the scanner have to be decided. The more scan collected, the more precise the final point cloud will be. But each scan lasts six minutes. A compromise has to be done whereas which scan is really essential and which position will add precision and details. If time is not a problem, it is always better to do more scans and then suppress the useless ones during the registration phase. But if time is limited, good preparation and brainstorming are necessary. In our case, half a day was dedicated to scanning the room, so the second solution (having too many scans and choosing the best ones during registration) is chosen.

The laser scanning of the structure lab is made with twenty-one different targets and thirteen different positions. The numerous scans and targets are not all useful, but it is safer to have too many scans and targets to be able to keep only the good ones. In this situation, for instance, only eight to ten laser scans could have been done, and ten to fifteen targets could have been enough to cover the whole lab.

With this quantity of scans and targets, the registration step is important. Only the best scans are kept. In Scene®, the registration software, three main choices can be made (and can be combined to one another) for the registration: (i) top view registration, (ii) target based registration, (ii) cloud-to-cloud registration. The process done in this thesis is detailed in **Appendix 2**.

The first one enables to automatically gather all the point clouds together, centered on the same point. For the target-based registration, the technician has to manually choose the targets on the different point clouds and link them together. It can only be made with two scans at once. This method is thus very restrictive when a lot of scans have to be registered. In our case, this registration is not chosen. However, the top view does not give fully acceptable results. The Cloud-to-Cloud registration is therefore combined to add precision and make sure the best registration is done. It combines point clouds together by finding similar points and their belonging to the same plane or surface. Because the top view registration was made before, the point clouds are all aligned on the same center, and the Cloud-to-Cloud registration is done with more accuracy. It finishes to align all the point clouds together and reduce the errors. These two registrations done alone would give a good result but less precise.

The point clouds can then be extracted as one entity in a .xyz file.

III.1.b. Converting the scans

The first problem faced is the fact that laser scans' registration software gives point clouds in .xyz or .pcs format. This is understandable since the software used are not Autodesk® products. These files are too heavy for Revit to be able to open them directly or are just not recognized formats. They need to be converted to .rcs or .rcp formats. These two are products of Autodesk® Recap. The first step is thus to convert the data from laser scanning to data usable in Revit.

Developed by Autodesk®, Recap is software for point clouds and photo (with Recap Photo) editing. It enables reality capture and 3D scanning processes. It is very complete, and commercial software is developed from it, linking it to UAVs (Unmanned Aerial Vehicle) or other mobile laser scanning technologies [**59**]. As an Autodesk® product, it is able to convert point clouds into .rcs and .rcp files. In this research, it is mainly used as a converter and the interest in it comes from the fact that it is an Autodesk® product. Therefore, it is the only software able to convert the point clouds as wanted. If the point clouds have to be edited after registration, it is also possible to do so.

The first step is to open the point cloud in Recap®. Once it is loaded, it is possible to extract it as a .rcs file. **Figure 7** shows the point cloud in Autodesk Recap®.

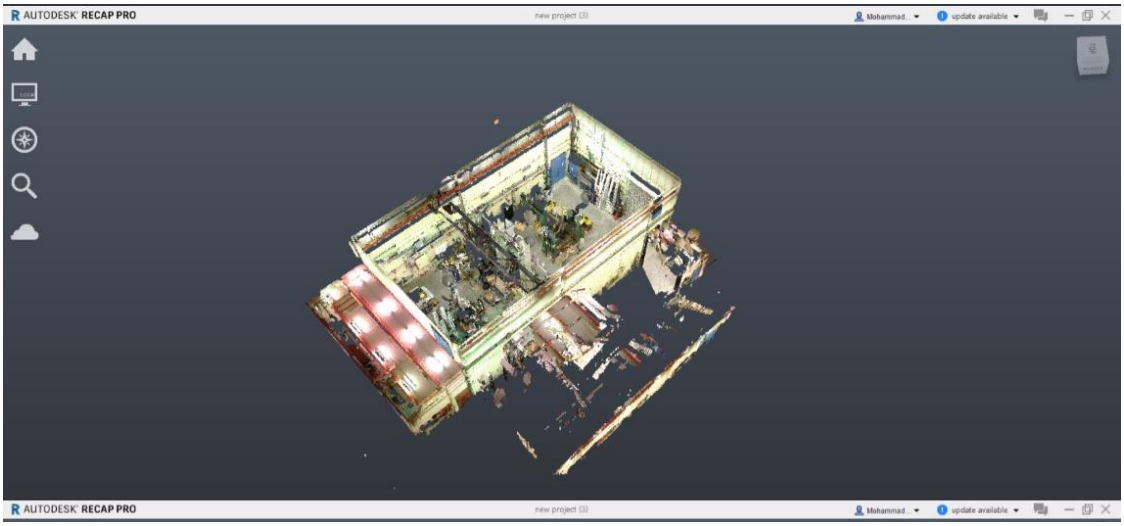


Figure 7. *Structure Lab point cloud in Autodesk Recap®*

Once the files are converted, a new project can be created in Autodesk Revit®. The point cloud is then opened inside this project (**Figure 8**) by using the function "Insert a point cloud".

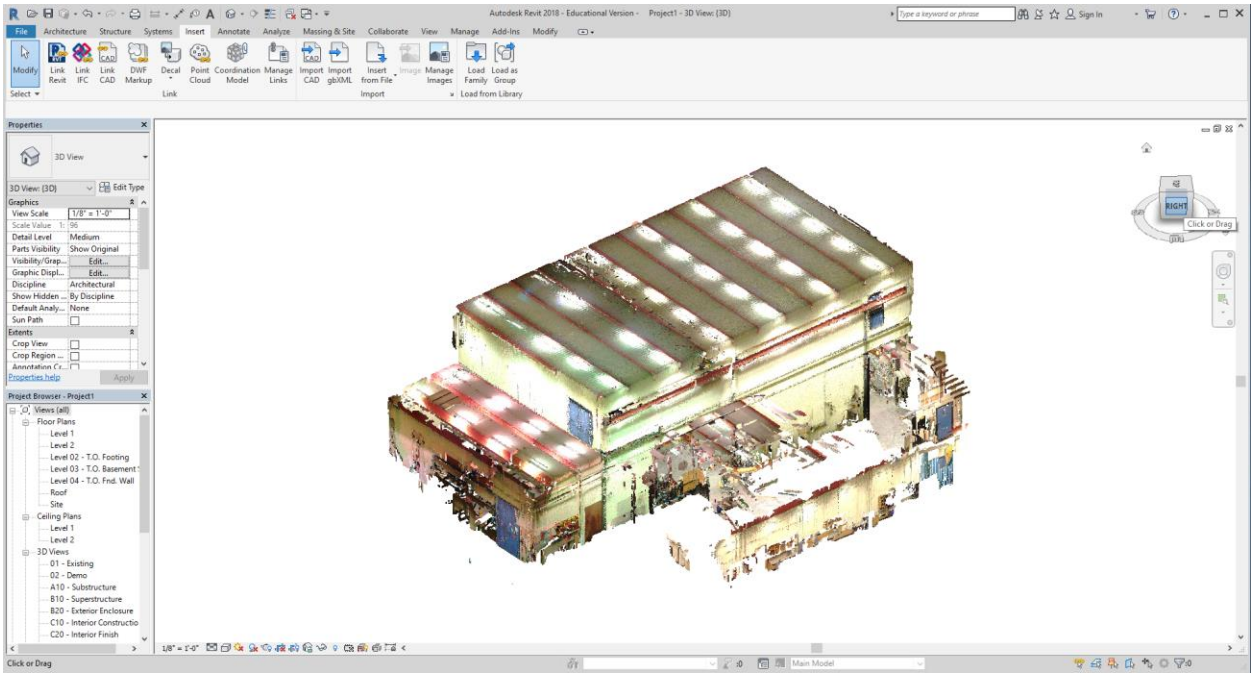


Figure 8. *Point cloud of the structure lab in Revit®*

III.2. Creating a model in BIM: important steps

With the rise of BIM in construction management, numerous software propose models designs and configuration. The first decision when it comes to modeling is to choose between CAD and BIM. The first one is very good for structural and MEP elements. In our case, it would be interesting to work with such models. However, for a three-or-more dimensions aspect, BIM has proven itself as a better solution. As the final model will be the entire facility and not just a room, not using a CAD model does not seem doubtful. But CAD models can be developed to add precision and accuracy to the final BIM.

Once this choice is decided, Autodesk® Revit appears as good software to develop such a model. Not only is it intuitive and very commonly used, but it is also part of the Autodesk® products. And it is better to use most products from the same company, to prevent file conversions and data loss. With its possibility of designing MEP, construction, or architectural models, it comes as an asset for developing a BIM of a nuclear power plant. And it is simple to find pre-designed objects in free libraries because it is used by many companies and designers.

III.2.a. First step: Aligning the levels

When the point cloud is opened in Revit, the levels set up are still the default ones. The first step is to align the levels (first floor, second floor, basement, roof…) in accordance with the ones on the point cloud (**Figure 9**). In this case of study, three levels are defined: level 1, which will represent the floor; level 2, which will represent the ceiling of the extricated part; and the roof/ceiling, which will define a level for the beams on the ceiling. Indeed, to improve the clarity of the model, the real ceiling is often not represented. This enables to see the inside of the model without adjusting the limit box or hiding the ceiling. Defining levels is extremely important since it will enable to work and view plan views. Indeed, in Revit, the plane views proposed are linked to the levels the designer choose to define. If a level is not defined, it is not possible to link a plane view to it and work on it, the only possibility of being viewing a blank page.

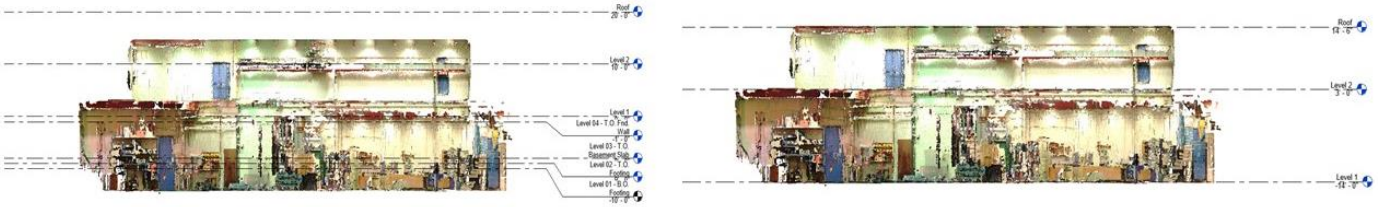


Figure 9. *Aligning the levels in Revit® with the scan*

Once the levels are defined, creating a model is possible and facilitated. First, the primary elements will be created in 2D views, then details will be added in the 3D view.

III.2.b Second step: Designing the primary elements

The primary elements of a building are defined as the biggest elements, which can be modelized more easily than others. Often because they are planes. As primary elements, we can list walls, openings, doors, roof, floors, and ceilings. Once the levels of the scan are defined, the next step is to model these elements. The work starts in 2D view, where the walls should appear clearly if the levels' definition is well done (**Figure 10**).

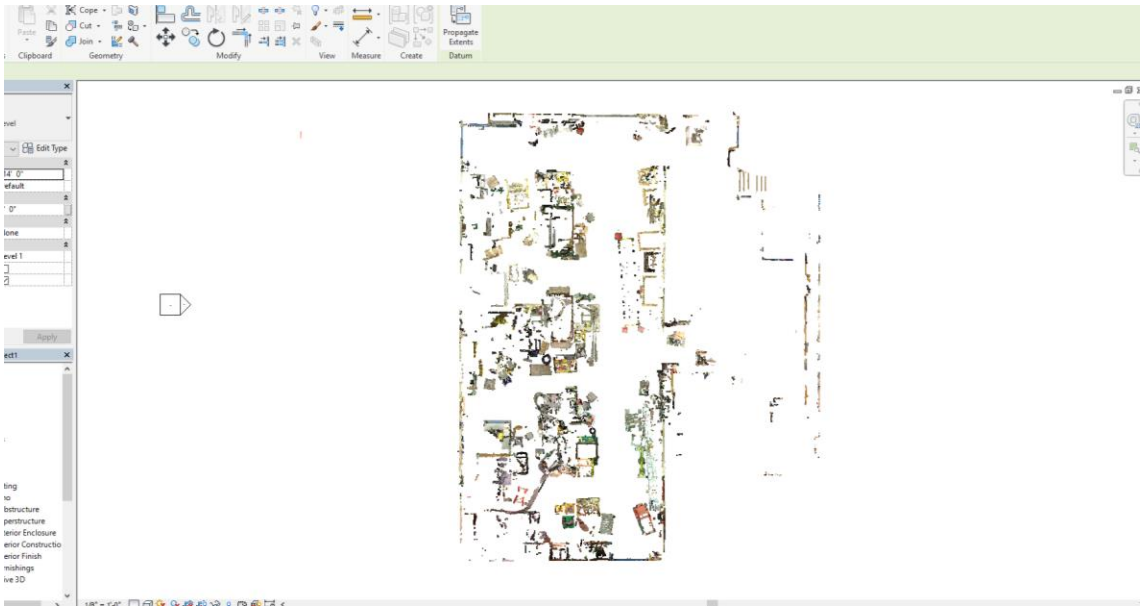


Figure 10. *2D view of the first floor after the levels' definition*

Once the walls are designed, the floor is easily made by having the walls as its limits. The tool of automatic creation is used. It takes the edges of the room closed by the walls and designs automatically the floor. This is not only useful because of the precision of the shape of the floor, but also because it links the floor directly to the walls, preventing any "holes" creation. The same tool is applicable to generate ceilings or a flat roof. The 3D model is thus created on top of the scan as if it was a tracing paper. To model the doors and opening, the scan has to be highlighted by clicking on it. Otherwise, only the model's walls are visible. Indeed, the issue that occurs when creating a model on top of the scan is that the modelized elements will hide the scan. To highlight it and make all the details appear, it is necessary to select it. However, this process enables to gain time, because no measures have to be done. The place of the elements are already defined and it is the designer's work to be the most precise possible. After this phase, the edges of the model are created and the other parts of the scans are still visible (**Figure 11**).

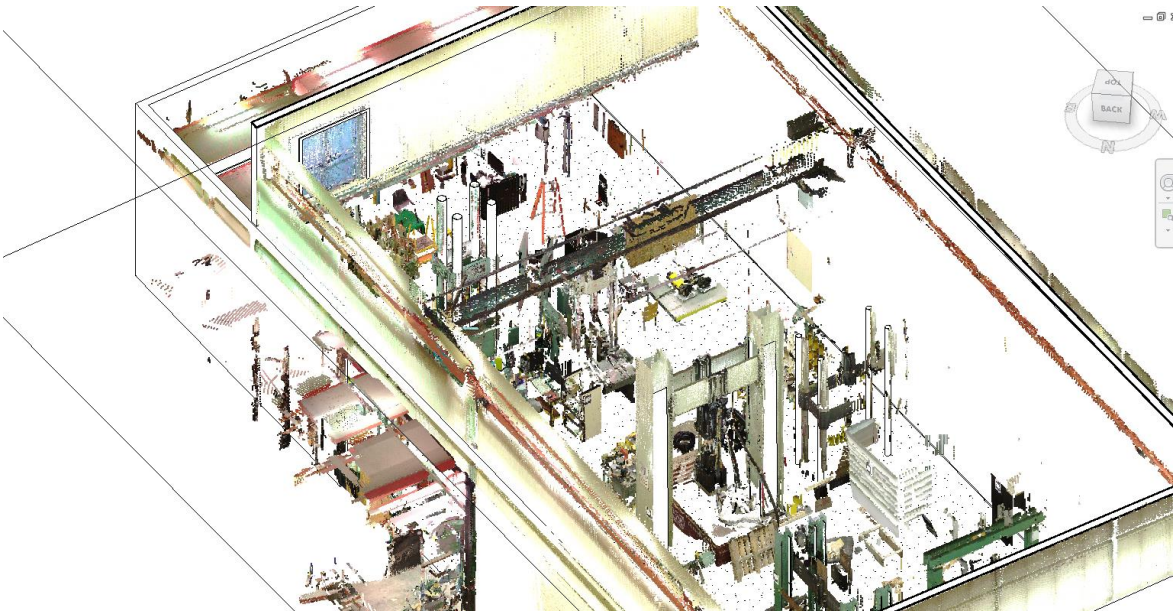


Figure 11. *3D model of the primary elements from the laser scanning point cloud*

III.2.c. Third step: Designing the secondary elements

The secondary elements regroup all elements that are not primary. In our case, it represents the pipelines, beams and structures and some parts of the equipment that could be represented by BIM objects. Basically, every element that could be found in a nuclear power plant are worth a representation. Every other object in our case is not worth representing. To model these types of elements, working on the plan view alone is not enough. The use of different views in 3D is necessary: (i) Top view, for the position of the elements; (ii) side view for their height and length, (iii) general 3D view for the precise matching with the scan, adding precision to their position. Additionally, as highlighted in **III.2.b**, the walls are already created and hide some parts of the scans. It is possible that hiding walls or reducing the size of the limit box have to be done in order to be able to view some details. Especially for the elements very close to the walls.

The beams are designed with the structural tab and pipes with the MEP one. The project is limited concerning these two tabs because the working environment is a construction template. However, it is also possible to open new structural and MEP projects, to create the model of the corresponding elements, and to integrate them to the final primary project. This enables to have more choice in terms of structures and piping systems. However, it limits their editing once they are integrated into the primary project. Indeed, to edit these parts of the project once they are integrated, it is necessary to go back to the MEP or structure project, edit it and reinsert it with the modifications. It is important to know if such a process is wanted, or if freedom in the primary project is preferred.

Because the beams and structures are common enough, it was not necessary to do it in our case. However, such a process is necessary for the piping system because it is easier to create vertical pipes and there are a lot of them in the scan. The pipes here are also very close to the walls. Knowing that the scans are not visible anymore once the primary elements are modeled on top of it, it is very hard to distinguish the MEP objects. Creating a new MEP project, inserting the scan and only modeling the piping system facilitates the process. There is no need to hide walls or highlight the point cloud for each modelized pipe. Because the same scan is used, once the MEP project is inserted into the general project, their position does not need to be modified and it is loaded at the right location. This approach saves some time and add precision to the piping system creation. But it is only necessary because of the closeness of the pipes from the walls.

After modeling the beams, structures, and pipeline, some details can be added depending on the LOD (level of detail) demanded by the client. In our case, the most important parts are the industrial elements and the pipeline, as they are elements that can be found in a nuclear power plant (**Figure 12**). To be closer to reality, some industrial elements are added to the model, like ladders or storage. But they are not essential. The scan can then either be deleted to only keep the 3D model, or just hidden if some future modifications want to be done.

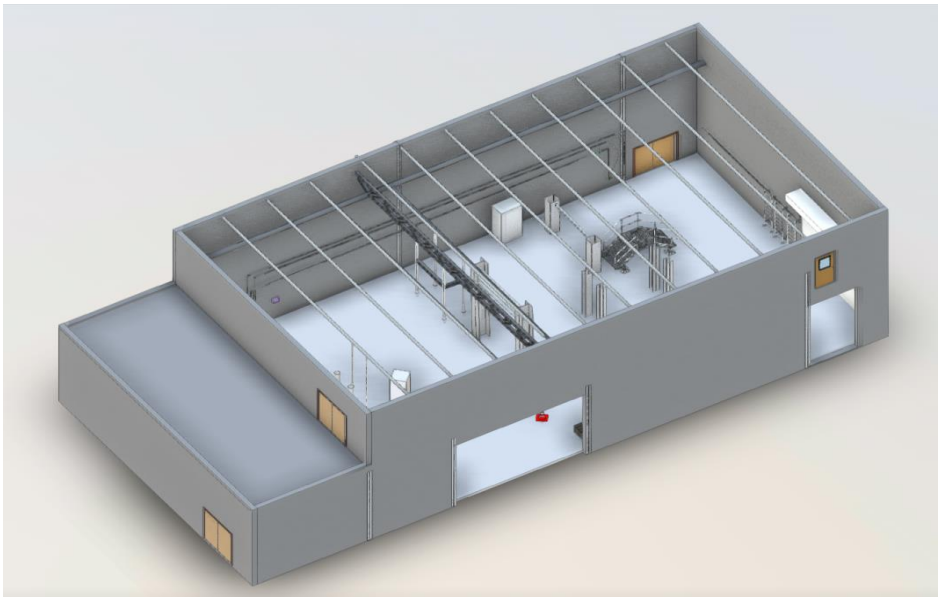


Figure 12. *Final project after modeling of the secondary elements.*

III.3. Automatic feature extraction: a convincing solution?

New convincing solutions are being developed and enable to save time on the designing phase. All the process presented in **III.2** is done manually. However, in a big project like a nuclear power plant, it could be interesting to use a semi-automated process. In this thesis, a Revit extension

which extracts BIM object from scans is presented. Following, a discussion is made on the interest and accuracy of this process, and if it could be used instead of manual design.

III.3.a. Automatic feature extraction extension results

To extract features from the point cloud in Revit, the Scan-to-BIM extension is downloaded and installed. This commercial product can extract pipes, ducts, and walls in a semi-automatic way. Indeed, the designer has to choose points from the wanted object and the software will then extract the corresponding feature. In this part, two tests were made: a pipe extraction and wall extraction.

It is important to notice that the point cloud is very dense and precise. The precision is the highest reached by this type of laser scanner. Therefore, the selection of one single point demands precision and can be a repetitive task when it comes to big projects.

A random point is selected in a pipe next to the wall. To assure the best accuracy, the pipe chosen is far from any other element like beams or small objects. This is to prevent some errors. However, even by taken all the precautions, the extracted feature is not convincing (**Figure 13**). In fact, the extracted element is not in the right direction, not going along the wall. Also, it is a lot shorter than the scanned pipe. This could be a choice from the company developing the product. They try to avoid the errors by only extracting small part at a time. But the direction problem is probably because it does not differentiate the points from the pipe and the one belonging to the wall. This is an error that could occur if extended to a nuclear power plant decommissioning project. Indeed, most of the pipes in these facilities are close to either other pipes or walls. Both errors are important in our case because mean that a manual correction has to be done afterward. Only the diameter of the pipe is almost correctly done.

Figure 13. *Results of an automatically extracted pipe.*

Another test is run with walls. Since it is bigger elements, the possibility of having more accurate results is higher. The process is this time a bit different. The designer does not choose one point but three points on the same plan. Also, it has to be done in a 2D view. That confirms the fact that

the levels assignation has to be very precise, not only for the model creation but also for object extraction. Like for the pipe extraction, the wall chosen for this test is the one with the fewer objects relying on it. Three points are chosen on the entire length of the wall. However, the same problem occurs, the element created is only one part of the selection (**Figure 14**). This means that the process has to be done many times to have the entire wall. Or that the designer has to manually extend the wall created, which will make this process a manual one.

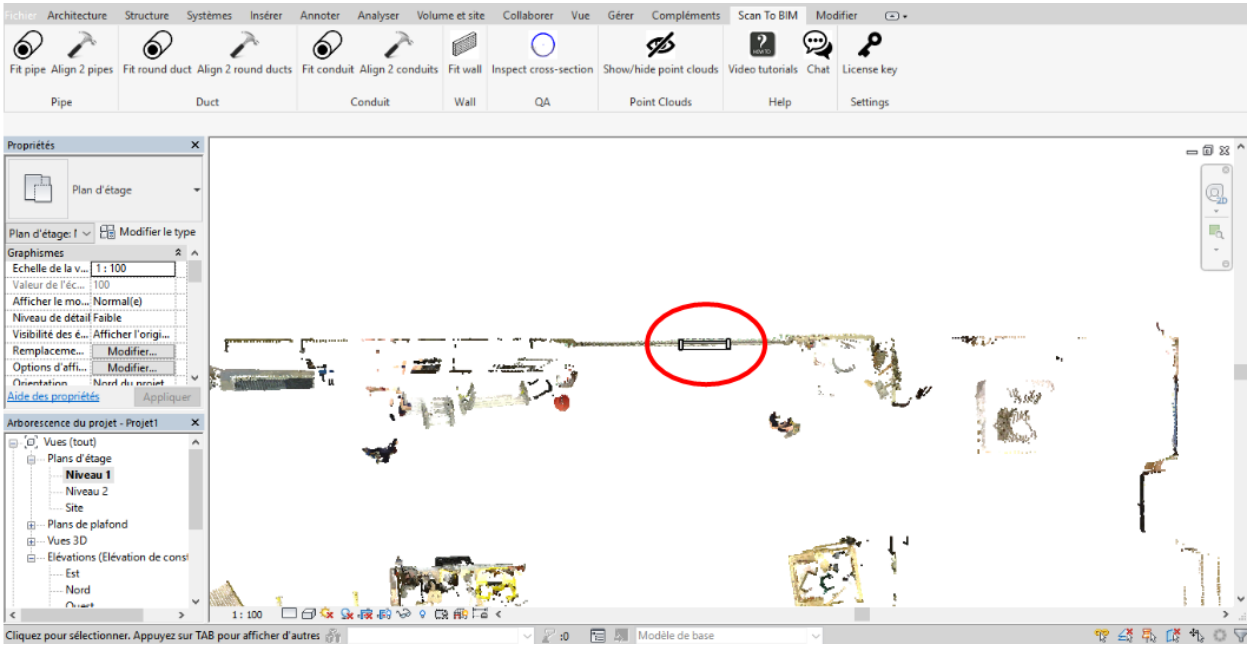


Figure 14. *Results of an automatically extracted wall.*

III.3.b. Discussion on the accuracy of the results and comparison with a manual approach

In big projects like nuclear power plant facilities, automatic detection could be the key for BIM. Most of the objects are similar in shape because they belong to the same type of families, which are mostly: pipes, ducts, walls and electrical elements. Knowing that the commercial products extract these kinds of objects, it was in our interests to test them to see how accurate the final model could be. However, the limits of the software appear quickly. First of all, this approach is not fully automatic. The designer needs to choose the points belonging to the feature wanted. But most importantly, the objects are not completely designed. This means that manual work is needed to continue the automatic extraction.

For walls, an extension of the element needs to be done. Their direction is properly created thanks to the selection of three points instead of one. But their length never matches the real one. Two solutions can be chosen: extending the one element manually, or creating several elements for one wall. The second option is very bad in our case because it would distort the data and charts linked to the model. Indeed, it will show more walls than existing ones.

For the pipes, the software is even less convincing. Not only does it meet the same issue of creating short elements, but also the direction is not right. This comes from the fact that only one point is chosen and makes it complicated to fully know which other point belongs to the same pipe. It could be even worse in the nuclear power plant, where pipes are very close to each other. Here the only option is to extend the pipe and reset the right direction. The only real input that could be kept being the diameter, if only the feature is correctly generated.

Automatic recognition has shown its limits during these two tests. With the errors faced for a simple lab, it is unthinkable to use it in a nuclear power plant right now. Some commercial products are certainly more improve than the Revit extension used here. But it seems logical to think that there is not a huge difference of recognition between two commercial products. As explained in **Chapter II**, limits and issues are also met when it comes to more developed products like Edgwise®. Agapaki et al showed [**24**] the low precision of the software, especially in the case of curly roof shape. Even if the errors are not as common, there is still some progress to make in this field. It is important to point out that automatic recognition is recent research, and is still in progress. The products created out of it are not yet good enough for complex cases. It is also important to highlight the fact that this study is dealing with big point clouds. When it comes to this much information, automatic recognition reaches quickly its limits. Bosché [**22**] and Wang [**15**] tried to solve it by combining two methods. While other researchers ([**19**], [**20**], [**23**]) try to divide the point cloud into smaller parts and limit the region of detection, but without convincing results yet. And it also adds complexity to this process. It shows that when it comes o big industrial infrastructure, automatic recognition is not yet fully ready. But it would be very interesting to see how good they can get in a close future, because of the time saving they could offer this kind of projects.

IV. Developing an asset management interface based on BIM

This chapter will highlight the development of a prototype to facilitate management and help use BIMs during operation and maintenance phases. As presented in **Chapter 2**, a lack of methods and research concerning BIM and management during the whole life cycle of the building is limiting the use of this tool. As BIM could change the way of managing big projects after the design phase, it is not yet used for this prospect. In this chapter, a presentation of other management solutions is firstly done to understand why creating an interface was selected as the final option. Then, the prototype is introduced and its functions are detailed to better see why they can be useful for this case of study.

IV.1 Why develop an interface?

IV.1.a. Other solutions for management

As introduced in the literature review **Chapter II**, the state-of-art of BIM for management is currently still limited. No existing solution has been yet can be extended to a general aspect and be developed as software or an extension of existing software. Research tends to reach a six or seven dimensions BIMs, by adding life cycle assessment and sustainability analysis. But these are not yet convincing enough to be developed as a general approach and limited by the second level of interoperability. However, products can help construct a viable solution to help management from a BIM.

- **Revit**

Since the BIMs were created in Revit, the first logical option that can be thought of is the scheduling option it offers. Indeed, we can generate tables concerning specific elements of the model directly in the software. It is also possible to then export them as Excel files. The criteria are chosen when it is created and enables to have a customized table. The common criteria are: type, family, area, surface or volume, lifetime, location in the building… However, for big projects or complex ones that have a lot of different elements, numerous tables have to be generated (one for each category of element). It is also the case if different criteria want to be applied to the same category of elements. A new table has then to be created. It becomes then almost unreadable and the information cannot be used in an efficient way.

Another tool found in Revit is construction phases. A project can thus be divided into different phases which would show the corresponding elements belonging to that phase and constructed at that time. However, this process is a little time consuming, especially for big projects where numerous phases have to be created and do not concern the entire facility, so have to be specific to one part of it. This adds more complexity to this approach.

Therefore, Revit is not a good solution for big projects with multiples categories of objects. It should be limited to models with only primary elements.

- **Navisworks**

To schedule and simulate projects' advancement, Navisworks was developed by Autodesk®. It can be seen as a model managing software, where tools can be created from the models' data and properties. The BIMs cannot be created or edited inside this software. However, other management helping tools can be generated, such as schedules of construction or deconstruction. With the Timeliner, Scripter and Animator options, it is possible to create simulation with specific criteria. The Clashdetective option is also very useful as it enables to find clashes in the model. This means that if two elements collapse on a point and therefore will create issues during the construction and operation phases, it is highlighted by the clash detector. It then offers solutions to modify the model in order to solve the clash issue. Finally, the quantification option can count elements in the models and their materials. It has more clarity than a Revit® table but does not have as many possibilities of choice for the properties listed and counted.

As it offers multiple interesting options, Navisworks® cannot be enough for the management of decommissioning a power plant. First of all, it is an Autodesk® product and only people with a license can use it. This limits its utilization to designers and is hard to convince other fields to use it. For big projects that gather people from different fields and backgrounds, it does not seem like a good solution. Also, even if it can integrate laser scans and is able to compare point clouds to models, it does not operate data from sensors. This last option seems interesting because offers a comparison that we cannot find in Revit®, but is however limited by the size of the model or point cloud. The as-built information is thus limited because of the lack of precision of the comparison point cloud/3D models, but also because all the data collected cannot be processed. Finally, the scheduling in Navisworks is entered manually. In the case of decommissioning a power plant, the project is expected to last at least until 2100. A better solution would be to be able to link directly the project's schedule, and not have to integrate it manually.

- **IFC and COBie file standards (Figure 15)**

Another common option for management is to use IFC and COBie (Construction Operations Building Information Exchange) files. It became a standard in terms of interoperability and data exchange concerning BIM. As these two file formats were created to add interoperability in construction projects, it is impossible to talk management and BIM without introducing them. As presented in the Introduction **Chapter 1**, BIM can be defined by its levels and its dimension. Levels represent the level of interoperability, meaning how easy it is to exchange documents and the variety of information it is possible to share. For example, level 1 is defined as a simple file exchanged between two protagonists. Level 2 adds some interoperability by increasing the numbers of people and file. Level 3 is not yet fully developed but is presented as a very interoperable level, with web-based or local server based exchanged. It is considered as a contemporary goal for full interoperability and facility management. All the files can be accessible

on an open space, where only specific actors who have access to it can share and edit them. The dimensions concern the information type that can be integrated into a BIM project. The first three dimensions are of course the three geometric ones. Then cost and time have been added, by the schedules option. Now, some new dimensions like energy savings and facility management can be integrated. However, they are still in the research phase and are not fully developed.

The first format introduced in order to add interoperability to BIM was IFC. It is a specific and standardized type of file, often in form of a text or an XML file, with all the information of a construction project. It defines elements of the project by their object definition, relationship with other objects and its properties. It is divided into four layers :

- Domain layer: defines the specific concepts and information entities.
- Resource layer: gathers basic and general information.
- Core layer: regroups basic framework and organizes the data of the resource layer.
- Sharing layer: defines the shared concepts and objects, and the interactions between them.

However, it demands a lot of knowledge on XML and coding to be able to read or write an IFC file. It is mostly open by a software able to view the encrypted data. Having different layer enables to organize the information but is rarely used as raw data if they are not used by a specialist.

The IFC files are also very voluminous and cannot be opened or created by non-specific software. It leads to creating a new standard based on the IFC one: COBie. These files take the form of a spreadsheet and thus can be open by Excel. It is more accessible and shareable than IFC files. All the properties, elements, contacts and other information concerning the BIM project are separated in different sheets, lines, and columns. The simplicity of the presentation was first the advantage of this standard but quickly became a limit. In fact, for big projects (which are common in the construction field), the COBie file is unreadable because of its unclarity. Too many columns, lines (thousands!) and sheets are created and it is almost impossible to find specific information. Many engineers who were first thrilled by the new format, had to go back to IFC standard. A lot of them also question the interest of converting a 3D model, therefore complex information, in a simple spreadsheet. It seems that most of the information is lost during conversion and that the remaining one is unreadable.

StructureFaro.ifc

```
#1= IFCORGANIZATION($,'Autodesk Revit 2018 (ENU)',$,$,$);
#5= IFCAPPLICATION(#1,'2018','Autodesk Revit 2018 (ENU)','Revit');
#6= IFCCARTESIANPOINT((0.,0.,0.));
#9= IFCCARTESIANPOINT((0.,0.));
#11= IFCDIRECTION((1.,0.,0.));
#13= IFCDIRECTION((-1.,0.,0.));
#15= IFCDIRECTION((0.,1.,0.));
#17= IFCDIRECTION((0.,-1.,0.));
#19= IFCDIRECTION((0.,0.,1.));
#21= IFCDIRECTION((0.,0.,-1.));
#23= IFCDIRECTION((1.,0.));
#25= IFCDIRECTION((-1.,0.));
#27= IFCDIRECTION((0.,1.));
#29= IFCDIRECTION((0.,-1.));
#31= IFCAXIS2PLACEMENT3D(#6,$,$);
#32= IFCLOCALPLACEMENT(#213450,#31);
#35= IFCPERSON($,'','ecmery',$,$,$,$,$);
#37= IFCORGANIZATION($,'','',$,$);
#38= IFCPERSONANDORGANIZATION(#35,#37,$);
#41= IFCOWNERHISTORY(#38,#5,$,.NOCHANGE.,$,$,$,1537976528);
#42= IFCSIUNIT(*,.LENGTHUNIT.,$,.METRE.);
#43= IFCDIMENSIONALEXPONENTS(1,0,0,0,0,0,0);
#44= IFCMEASUREWITHUNIT(IFCRATIOMEASURE(0.3048),#42);
#45= IFCCONVERSIONBASEDUNIT(#43,.LENGTHUNIT.,'FOOT',#44);
#46= IFCSIUNIT(*,.AREAUNIT.,$,.SQUARE_METRE.);
#47= IFCDIMENSIONALEXPONENTS(2,0,0,0,0,0,0);
#48= IFCMEASUREWITHUNIT(IFCRATIOMEASURE(0.09290304),#46);
#49= IFCCONVERSIONBASEDUNIT(#47,.AREAUNIT.,'SQUARE FOOT',#48);
#50= IFCSIUNIT(*,.VOLUMEUNIT.,$,.CUBIC_METRE.);
#51= IFCDIMENSIONALEXPONENTS(3,0,0,0,0,0,0);
#52= IFCMEASUREWITHUNIT(IFCRATIOMEASURE(0.028316846592),#50);
#53= IFCCONVERSIONBASEDUNIT(#51,.VOLUMEUNIT.,'CUBIC FOOT',#52);
#54= IFCSIUNIT(*,.PLANEANGLEUNIT.,$,.RADIAN.);
#55= IFCDIMENSIONALEXPONENTS(0,0,0,0,0,0,0);
#56= IFCMEASUREWITHUNIT(IFCRATIOMEASURE(0.0174532925199433),#54);
#57= IFCCONVERSIONBASEDUNIT(#55,.PLANEANGLEUNIT.,'DEGREE',#56);
#58= IFCSIUNIT(*,.MASSUNIT.,.KILO.,.GRAM.);
#59= IFCDERIVEDUNITELEMENT(#58,1);
#60= IFCDERIVEDUNITELEMENT(#42,-3);
```

Presse-papiers | Police | Alignement | Nombre | Styles | Cellules | Édition | Idées

B2 | n/a

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	Name	CreatedBy	CreatedOn	Category	Description	AssetType	Manufacturer	ModelNumber	WarrantyGuarantorParts	WarrantyDurationParts	WarrantyGuarantorLabor	WarrantyDurationLabor	WarrantyDurationUnit	ExtSystem	ExtObject	ExtIdentifier	ReplacementCost	ExpectedLife	DurationUnit	WarrantyDescription	NominalLength	NominalWidth	NominalHeight	ModelReference	Shape
2	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Ceilings	944e3c42	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
3	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Doors	31261f36-	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
4	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Doors	704e4689	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
5	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Doors	8b6bab35	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
6	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Doors	8b6bab35	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
7	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Doors	8b6bab35	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
8	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Doors	8b6bab35	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
9	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Floors	febf851d-	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
10	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Furniture	8b6bab35	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
11	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Furniture	8b6bab35	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
12	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Furniture	51d1547c	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
13	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Furniture	51d1547c	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
14	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Furniture	51d1547c	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
15	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Furniture	51d1547c	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
16	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Furniture	8b6bab35	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
17	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Furniture	8b6bab35	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
18	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Furniture	51d1547c	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
19	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Furniture	51d1547c	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
20	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Railings	b6d169bd	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
21	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Roofs	9693c920	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
22	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Specialty	51d1547c	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
23	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Specialty	51d1547c	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
24	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Stairs	d3f3a8df-	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
25	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Autodesk	Landings	d3f3a8df-	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a

Instruction | Contact | Facility | Floor | Space | Zone | Type | Component | System | Assembly | Connection | Spare | Resource | Job | Impact | Doc...

Figure 15. *(a) Example of an IFC file, (b) Example of a COBie file.*

- **Microsoft® Project and Primavera®**

It is impossible to talk about management without mentioning these two software. They allow users to plan a project with optimal management tools such as GANTTs, resource classification, and organization, data analysis, export to Excel (for MS® Project) or integrate financial or human capital aspect (Primavera®). Primavera® is a cloud-based SaaS platform, so it is also deployable to mobile devices via an application. As research is still in progress to optimize 4D BIMs with this software, an already existing solution enables to generate 4D BIMs by linking this management tool to 3D BIMs: Visual Simulation of Innovaya®. It is a 4D analysis tool allowing BIM created in Revit to be integrated into MS Project® or Primevera® files. It uses Revit construction phases to create sequence based projects in this software, that are automatically linked to the BIM components. It is also possible to view tasks and associated model objects. As this solution appears interesting, it is also composed of non-open software and lead to trying to integrate construction designs to management projects. Primavera® is an Oracle® product and is easily linkable to other products from the brand but it is harder for other connections. An approach using this software would mean using an intermediate and having a license for these products, which means more costs and possible data loss.

Other solutions exist but are not as interesting or common as these three ones. It is clear that the management aspect was neglected and is slowly improving. And a lot of work still has to be done concerning other phases than design and construction one. Still, no permanent solution seems to be intuitive yet.

IV.1.b. Server based interface: the right solution?

To improve BIM as an asset for management, a good solution would be a file, software or interface where it is possible to see the scans collected and the 3D models based on them, as well as live data collected by sensors, schedules of the decommissioning phase and links to additional information related to decommissioning in general or specific parts of the project. In a nutshell, a place where every information linked to a project can be accessible or viewed. After doing the literature review on the state of art concerning BIM and management, and knowing more IFC and COBie standards, it appeared that such a solution was not yet on the market. Some research is interested in web servers and cloud-based BIM but these products are not yet opened and commercialized. Moreover, none focus on all aspect of the project and stay close to BIM as the main interest. The approach of developing an interface is thus selected and can also be interesting thanks to its possibility of being customized from the beginning. Indeed, by creating a local web page with different coding languages, it is possible to add numerous options and extension to our interface. One of the priority is to keep this prototype customizable in the future. It is convenient in case a function has to be added later on since the decommissioning project should last until 2100.

The decision of having an interface is also based on the restraint choices to show BIMs openly. In fact, without any license, not many options exist and are usually limited in terms of actions. For

example, Revit® Viewer is open to all but only offers the possibility of seeing the 3D model. After research on this topic, Forge Viewer is selected as a good solution for this project (see **IV.2.a** for more details). And with this choice comes the possibility of developing an interface around it.

However, the confidentiality of this project has a huge role in the development of a cloud-based solution. And it is well known that web pages are not the safest places to upload confidential data. The coding has to follow strict rules to keep the data secured and open only to people to whom the access was given. Therefore, local server deployment and authentication before having access to the interface are necessary (see **IV.2.c** for more details).

IV.2. Developing the interface

IV.2.a. Choosing a viewer and data manager

As explained in **IV.1**, as no application offers exactly all the functions at once, the interface development approach is chosen. The two main functions of this interface are viewing the 3D models and storing the data from the project. It is logic to find an application or software that has both to start with and then develop more functions from that base. Therefore, the decision of creating an interface is directly linked to what Autodesk Forge® offers, a set of web services [**53**]. Introduced in 2016, Forge® was developed by Autodesk® in order to help the expanse of their other products. Indeed, its different tools and API (Application Programming Interface) help build applications for design and engineering that Autodesk® products alone could not have been able to do. Some companies are using Forge® to develop visualization, 3D printing or other functions to their applications. It enables more designing options and easy customization. It is also interesting to note that Forge is linkable with a lot of applications that do not have to be Autodesk products. For example, it is possible to link a Google Drive account to Forge in order to view documents from it. Eight APIs (by February 2019) and tools are suggested when the designers create an app with Forge®:

- BIM 360 (see IV.3.b for more details): It is a cloud-based application, where information can be stored and exchanged between members of a project. It is also possible to chat and schedule meetings and deadlines. It is a very powerful tool for management and respects the confidentiality with its option of letting the administrator decide everyone's role.
- Data Management: It links Forge® to the management applications like BIM 360, Fusion or A360. By getting access to any stored data, it can make Forge® a cloud storage interface.
- Design Automation: It is not used in this research, because it deals with DWG files. It can create applications for objects and properties of a DWG file, convert these files to another format or create new ones.
- Model Derivative: Its role is more to convert design files for other API to use. For example, it prepares the file for the Viewer to be able to show them.

- Viewer (see IV.3.a for more details): It shows 2D and 3D models, as well as drawings. It is also able to open files format like PDF. It has multiples functions that can help the visibility of a model and viewing some of its specificity. A big advantage of the viewer is its capacity to be customized by coding.
- Reality Capture: It is the tool to use when working with aerial mobile devices. It helps the photogrammetry and camera capture. It also facilitates the computing of these scans and images.
- Token Flex Usage Data: It gives every information you need on your tokens, from their consumption to their usage. Tokens are a big part of the security in our interface, so having access to their properties is essential.
- Webhooks: It is basically the customer service of Autodesk. It gives you notifications when an event that may interest you is happening.

Forge is used in many applications and has a lot to offer. From virtual reality development to kinematics animation, a lot of examples of products can found [**61**]. Some web designers use it to animate 3D models, others to manage data or even fluid mechanics controls. Forge brings freedom to design and customization, which is exactly what is interesting in our case. Indeed, the interface has to be based on main functions but cannot be limited in order to be able to generalize this approach to other AEC projects.

With several of Forge's APIs combined, the functions we are looking for the interface can be developed. BIM 360 Team is kept for the data storage and exchange, and of course, Data Management links all information to the interface. The Viewer enables to see all the models from the scans. Sadly, the format of the scan is not borne yet. The interface is based on Forge Viewer® with direct access to the BIM 360 account and all its data. All other functions can either be integrated into the Viewer or on the web page.

IV. 2.b. Choosing the coding language and software

A lot of possibilities exist in terms of choosing a coding language. Knowing that the prototype will be a web page, HTML is inevitable. However, to code the viewer's function, a choice has to be made. The best approach is to find which coding language is mostly used for the viewer customization and see if it correspond to this research. The viewer can be customized by coding in Javascript. Several existing extensions and code can be found in different languages. To facilitate the coding process and development of the interface, it is better to choose one of these languages. This is for future prospects, to enable non-specialists to develop their applications and functions. There are two options that seem to be highlighted: ASP.NET and Node.JS. However, most of the extensions found on Github are in Node.js. As no preferences were sensed, the choice of Node.js is almost logical. Finally, the design of the page is done in CSS. It is a very common design language tool for HTML pages, which correspond to our case.

Concerning the software, the choice is also easy. Because the work done on the interface is not for a professional web design, basic software can be used and is enough for the coding of a web page. Visual Code Studio is an open software developed by Microsoft. It can be downloaded directly from the website and has every option needed for the coding of the interface. It also contains Node.js and Firefox extensions, which will be needed for the deployment of the interface locally. Other browsers' extension can directly be download from the software as well.

IV. 2.c. Assuring the safety and the confidentiality

The main concern in a cloud-based solution is the privacy and safety of the data it contents. As resented in the literature review in Chapter II, with the ownership, it is the reason practitioners still have some doubts about this approach. It also means that if a convenable solution concerning security is brought, a cloud-based interface could be a very interesting management asset.

In this research, some dispositions were taken to offer the safest environment and protect data confidentiality. The first precaution is to never have the data on the interface. Indeed, it is a place where all the information is gathered and can be found by linking it to numerous other safe accounts. This way the data is not on the interface's page but in other accounts where registration is already needed. Then they are viewable on the web page at the same time.

To assure the security of the process, a 3-legged authentication is also essential. This means that three steps are necessary:

1. Signing in with Forge® ID (no license needed, just a registration).
2. A token is linked to the ID. It is perishable to assure that no one stays connected indefinitely.
3. An access notification is sent. The client has to accept that the interface has access to its BIM 360 account data.

This process assures the security of the interface and the different log in to it. With big projects and numerous actors, the safety of the information is a priority. Indeed, depending on their status, they will have limited access to some data. The ownership of the data can be determined when the actors become members of the project, by signing a contract. The logical solution would be that the data on the interface is owned by the person uploading it.

Finally, as a last precaution, this environment is not deployed to the Internet. Only a person attached to the local server can have access to it. Even if extending the interface to the Internet without giving the URL could also be a solution, it is not very safe. Some data can still leak and the web page could, unfortunately, be found by people outside the project. To prevent this risky situation, the interface is deployed locally on the computer. For bigger projects, it can be extended to the local server of a company for example.

IV.3. Adding functions to facilitate the management of BIM projects

In this part, the interface and its functions will be presented (**Figure 16**). As explained before, the functions can be changed, added or deleted depending on the needs in the project. Here, the ones developed are according to the demands of a decommissioning project. It can still be suitable for other deconstruction projects. The final result (**Figure 17**) is a web page, run locally, and divided into two main parts:

- Forge Visual Reports [**62**]. It regroups the BIM 360 account where the information is accessible, a viewer to show the 3D models, a document viewer to show the 2D views of the model or a PDF file, and Pie chart where the properties of the opened model are organized. In big projects, a model of the entire facility will be too much information. This is why it is often divided into smaller projects. However, only one model can be open at the same time. To see general information of the project, a second part was added.
- Functions of the project. As the first part concerns only one part of the project at a time, these functions enable to see some information about the entire project. It is useful to be able to access general information while viewing a model. It prevents from going back and forth between the data. Especially in big projects where a lot of data is gathered. For the same reasons, all the functions linked to other web pages are also coded in order not to leave the interface's page. New pages are opened in the browser instead of charging the function on the interface's page.

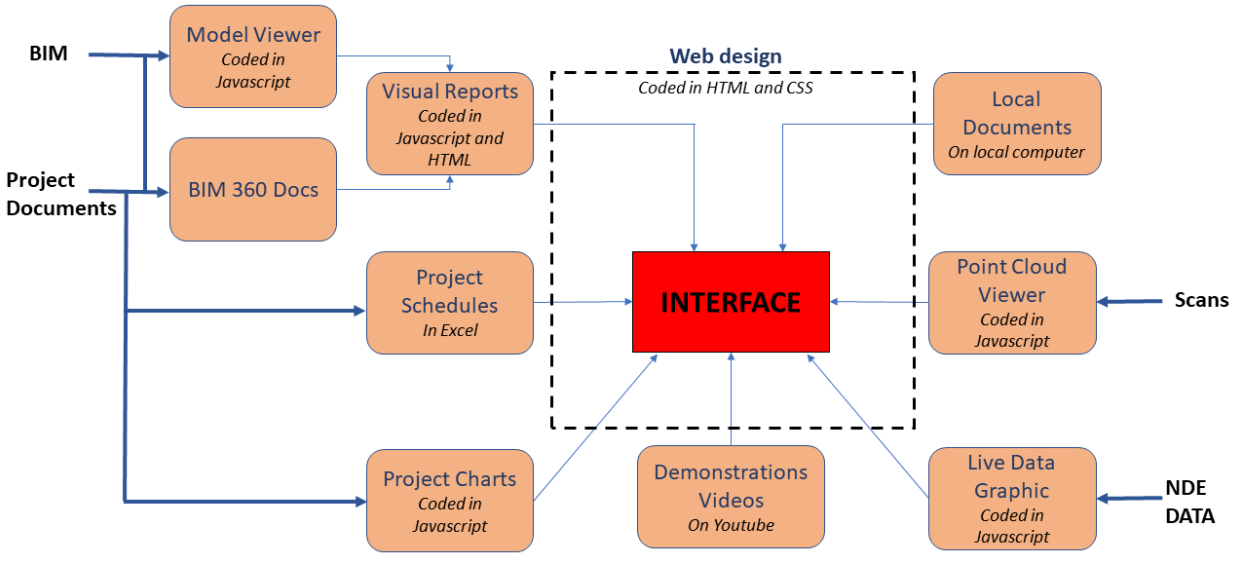


Figure 16. *General coding plan the interface.*

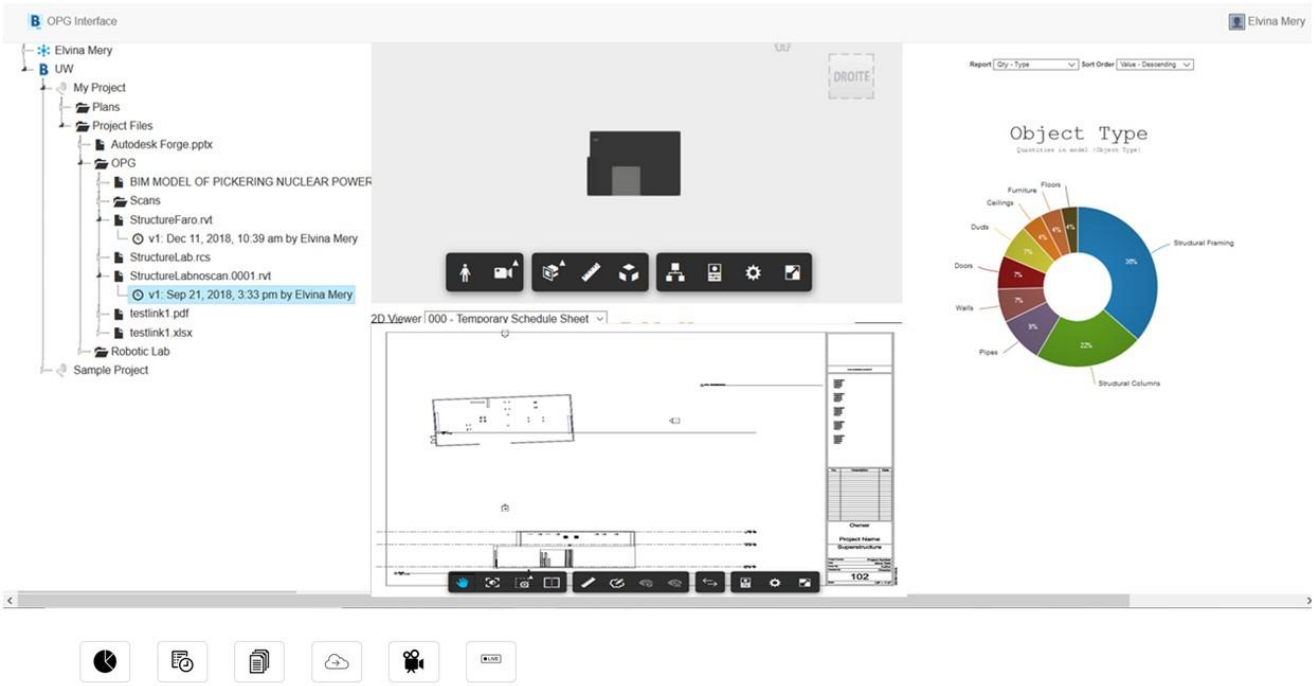


Figure 17. *General aspect of the interface.*

Each element is more detailed in this part and the code lines are attached in **Appendix 5**.

IV.3.a. Functions linked to the BIMs

IV.3.a.i. Forge Viewer

The first interest in Forge for this thesis is the viewer. In fact, one of the problems faced is to be able to show the models created without having to pay for a license of a product not used otherwise. For some clients, paying a license is not a problem but this project is huge. Therefore, small actors who will have minor actions would have to get Autodesk® license just to see 3D models and the information linked to them. To mitigate this problem, open source viewers are looked at. Forge Viewer is chosen because the models are developed with Revit®, and keeping Autodesk® products facilitates the transfer of information. This way loss of data is limited. And also it has very interesting options for a full viewing experience.

The Forge Viewer has also the advantage of being customizable. As this is more a coding engineer work for complex functions, some useful functions are still easy to code. The basic form of Forge® Viewer has also already some useful viewing functions (**Appendix 3**):

- Movement-based: It is possible to rotate the model, zoom in and out and create walkthroughs.
- Placement based: The inside of the model can be explored with a "first-person" option. It is basically a walkthrough without more freedom at that moment.

- Information based: Properties of the model or the selected elements can be seen, measures can be taken and parameters can be changed.
- Viewer based: The model can be exploded in its different elements in order to find or select small elements easily. It is also possible to select them directly from a classified list of all the objects composing the model. When selected, the viewer will directly zoom on the object inside the model to show its position.

In this thesis, the Forge Viewer is used to show the 3D models (**Figure 18**).

Figure 18. *Structure lab model in Forge Viewer.*

As presented previously, it is also possible to add some functions to the Forge Viewer and thus customize it for the project. **Figure 19** shows two extensions of the viewer: handle extension and dock panel. These two extensions are chosen because they are part of the viewer tutorial. Other extensions developed by Autodesk's engineers are free on Github and can be used. They cover a large panel of useful options a project could need (**Appendix 4**) and is constantly growing. Therefore, without many developing competences, the viewer can offer a lot of possibilities to get information out of the model. Of course, more complex functions can also be developed by computer science engineers or developers to match the project.

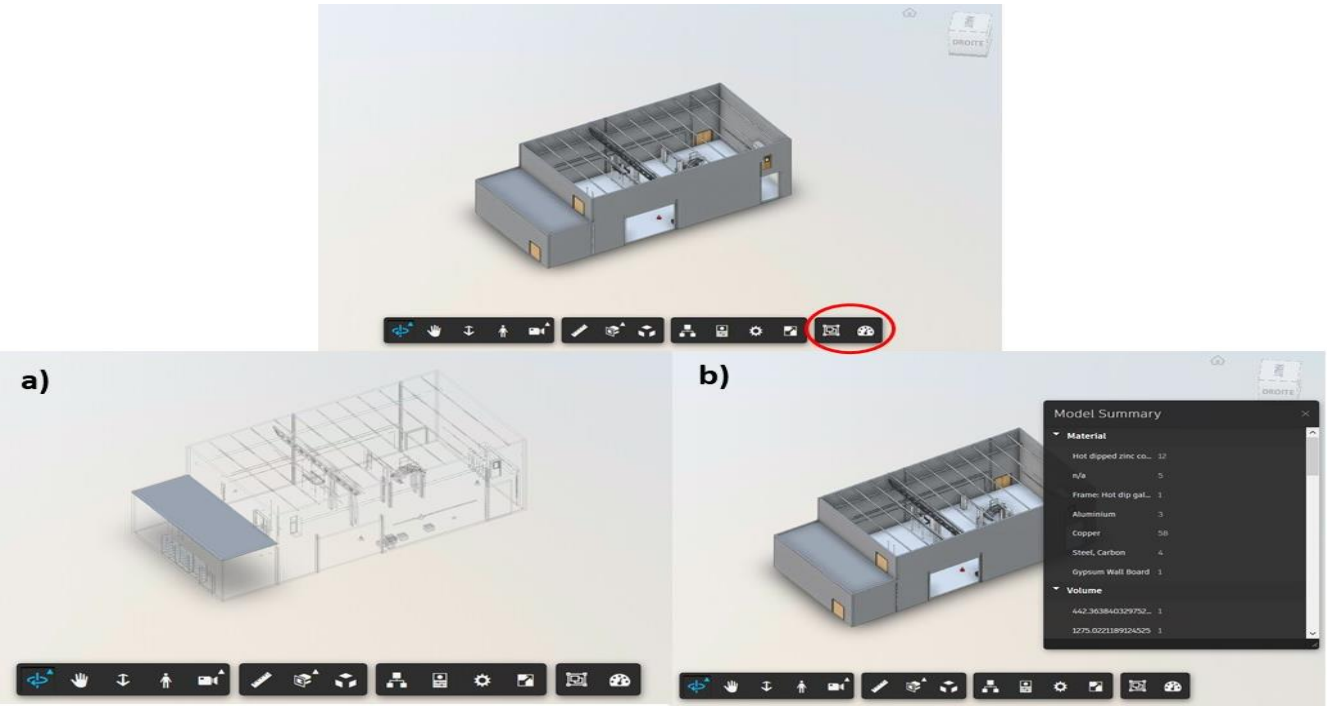


Figure 19. *Handle Selection a), and Dock Panel b) Extensions.*

IV.3.a.ii. BIM 360 Docs: Data Management application

Now that the models can be seen by any actor of the project, information needs to be linked to them. Indeed, having access to the models alone limits the actions. If the data of the project is also viewable at the same time, the management can be facilitated and showing the models becomes interesting. It is important to note that whatever data is viewable in the interface, none is actually uploaded on that web page, but on other safe accounts viewable from the interface.

Again, several solutions can be chosen but Autodesk® developed a good product for data management: BIM 360 Docs. It is cloud-based storage platform with interesting qualities. The most important one in our case is the level of security it offers. Indeed, the project is open by an administrator who invites the other members of the project. However, he also decides what kind of options are accessible to them and what kind of action they can do. They can either: have access to the document and modify or export it; or only have the access without the editing; or not have access to it at all. This enables some members to be part of the BIM 360 Docs project but only have access to documents they are concerned about. It keeps confidential information from small actors of the project, and administrative documents from technical engineers and vice versa.

BIM 360 Docs have several important management options (**Figure 20**):

- Viewer: It also is possible to view the models or documents in BIM 360 Docs. As it is also an Autodesk® product, it is very close to the Forge Viewer. However, it cannot be

customized. But some basic useful options can be used like markups and issues. The 2D sheets are also viewable.

- Chat: It is possible to chat with project members and see when they are online.
- Calendar: Meetings and deadlines can be added to the calendar and can be viewable by all members.
- Reviews: Issues and Reviews can be directly stored and discussed inside the platform.

Of course, files can also be downloaded directly from the platform to the local computer, if other changes need to be done with the need for specific software. **Figure 14** shows the home page of the project (a.), the different files uploaded in it (b) and an example of a file in the viewer (c), in BIM 360 Docs.

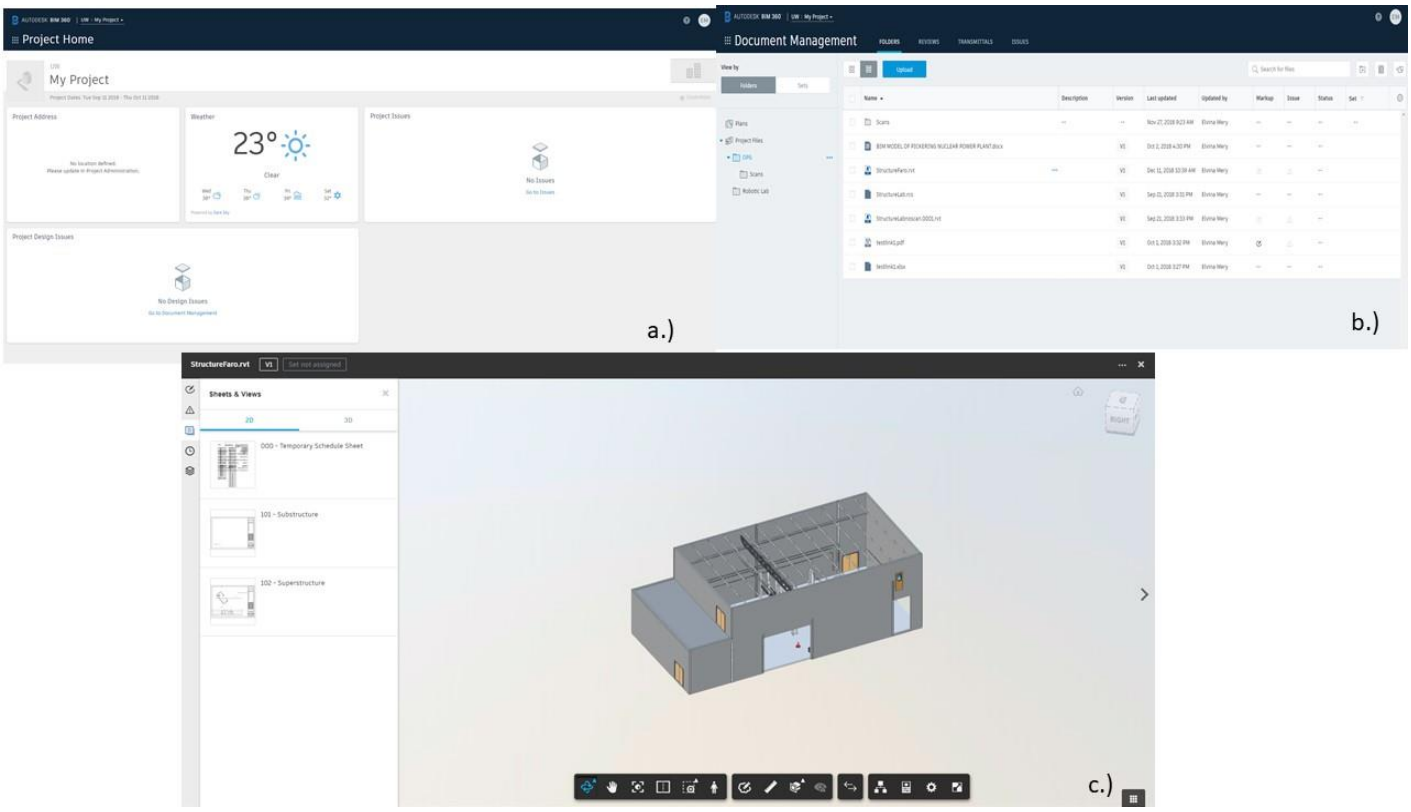


Figure 20. *Presentation of the BIM 360 Docs project: Home page (a), documents (b) and viewer (c).*

Because BIM 360 and Forge are both Autodesk® products, it is possible to link them. Thus by choosing BIM 360 to store the data of the project, it is possible to link it directly. On the interface, there is no need to upload the documents or the models one by one. The BIM 360 account of the project is opened and the files can be uploaded from there (**Figure 21**).

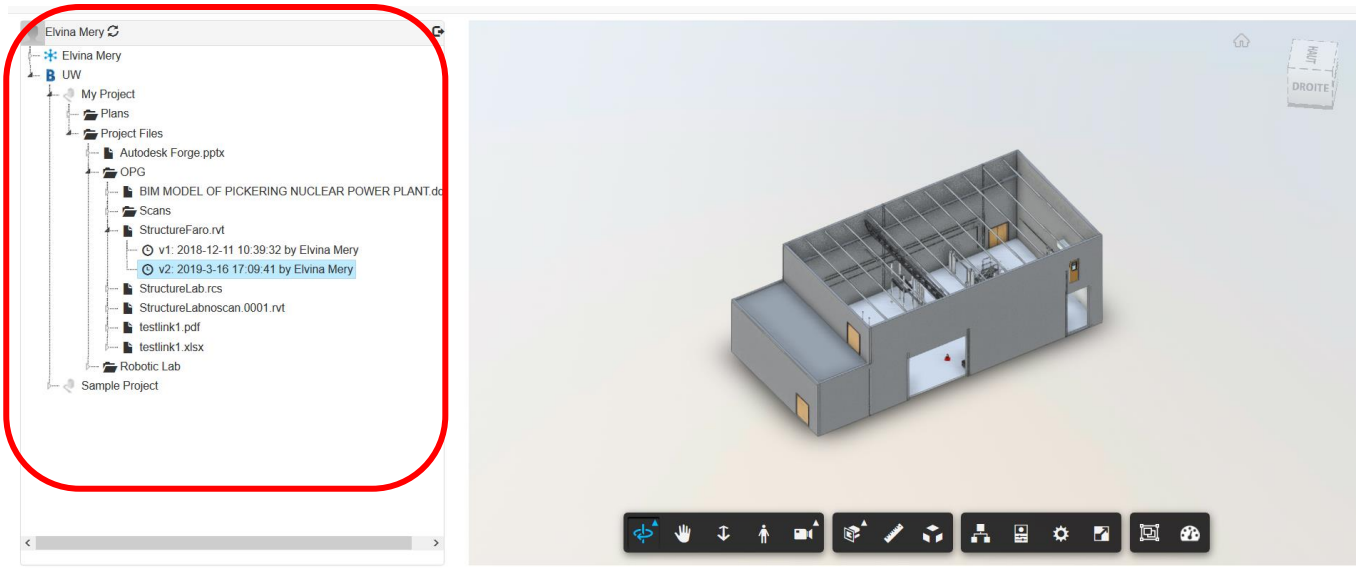


Figure 21. *BIM 360 account linked to Forge Viewer on the interface.*

IV.3.a.iii. Visual reports

As presented previously, Forge is an application facilitating the development of others thanks to its different APIs. One of the options of application is Visual reports. It is a viewer organization where reports can be generated from the model while viewing it and 2D plans. By using d3charts library for the code part, it is possible to create charts and reports directly linked to the opened model on the interface. The viewer can give properties of one selected element but cannot give information on the entire model. This application is run in Javascript, Node.js. Five main sheets need to be created:

- Forge Viewer, to organize the viewer and code a divided screen so the user can also view the 2D plan at the same time. The base of the code is necessary for most of the applications to determine how the user wants the model to be shown. But the divided screen and simultaneous 2D views are not common.
- Forge Tree, to link the BIM 360 Docs account. This is not only seen on Visual reports, but it is also done on every application which wants to be linked to that account.
- ReportData, to get the data and properties of the model and fill a database. It will then be used to generate reports and charts
- Oauth, to have an authentication before accessing the interface. It is also very common in an application where safety is a priority. Or when another application's account is linked, to authenticate and link it to the application.
- PieCharts. To generate the Pie charts by accessing the database in ReportData. In this application, d3pie Chart is used. It is a tool to help create pieCharts in Javascript.

The general idea of visual reports is detailed as followed. The data has first to be collected from the models into a database. To limit the errors and reduce the size of the data collected, only some categories can be transferred. For example, in our case only levels, volume, areas, names, and quantity are gathered. From this database, charts can then be created with the d3 library or Chart.js. They are then integrated into the interface (**Figure 22**). Reports and other types of charts can also be generated. Future work can also focus on the data shown by the charts to match the project. Indeed, if the level, names types are not important and the users are only focusing on the surface and volume of the elements, it is possible to extract only that information. If the viewer needs to be customized with new toolbar buttons and functions as presented before, it also has to be done in the ForgeViewer code in Javascript.

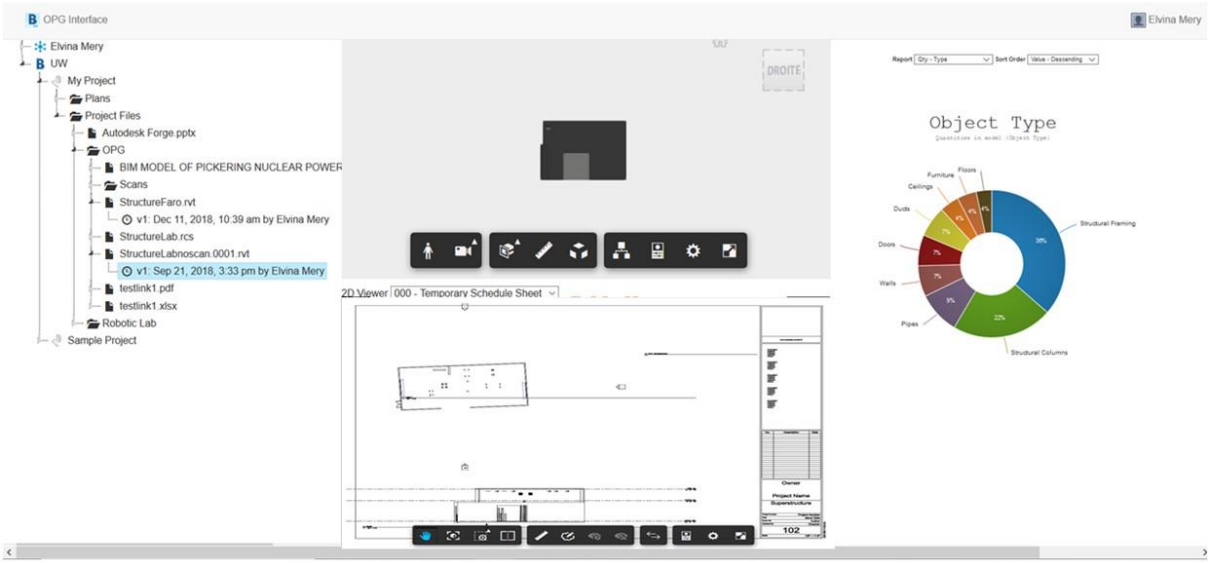


Figure 22. *Pie chart linked to the 3D model in the interface*

IV.3.b. Other functions linked to the general project

Before presenting the other functions, a clarification has to be made. All the previous functions are directly linked to Forge or the 3D models viewing. Therefore, Javascript and its extension are used for coding. But it can be important for management to have information on the entire project regardless of the model opened in the viewer. All the following functions are not part of the viewer but are still part of the interface, thus Javascript alone cannot be used anymore. All the coding is done in HTML with Javascript extensions because it is directly linked to the web page and not Forge. The interface is separated into two parts: One directly linked to the opened models in the viewer, and the other linked to the information of the project regardless of the model opened.

IV.3.b.i. Charts: Presenting the information about the entire project

It is interesting to be able to have a graphic view of the information about the model. It can bring visual to it and facilitate understanding. To do so, two charts are created with Chart JS. Like d3 charts, it is a tool to code charts on a web page. The properties of the chart have to be defined, as well as the data. Once again, it can also be linked to a database, like MySQL or an Excel file for example. Multiple charts can be chosen, from line to dynamic ones. Here, the data is separated into a bar chart and a pie chart. Other charts could be integrated as well if needed. To have an idea of what it is possible to do, these two types are chosen because they are the most common and representative of the information. For this case, the data is entered manually, representing the kind of material and volume present in the building (**Figure 23**). In fact, as this interface is a prototype, the goal is to show the different function of the interface and how they can be used. The real as-is data has then to be integrated later.

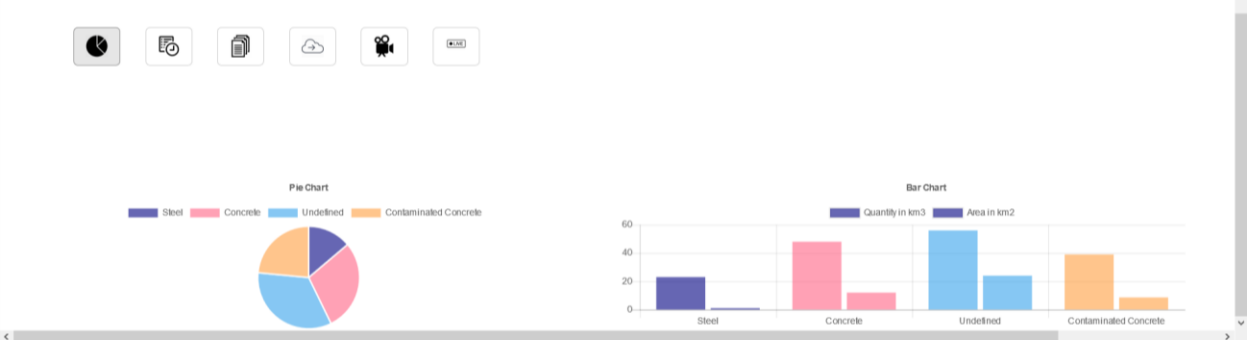


Figure 23. *Examples of bar chart and pie chart of the entire project in the interface.*

Another interesting option offered by Chart JS is the possibility to have several datasets on the same chart. That could be used for comparison. When multiple information is showed, it is also possible to only select one. The user can thus compare data or choose only one point of interest, depending on the need.

Once the code for charts is done, a button is added to the interface. It is coded in the HTML page and linked to the chart code which is the "source". The charts only appear when it is selected. As the user could maybe what to also compare the information with the one on the pie charts in Visual reports, the choice of not opening a new tab or page is made. Therefore the charts appear under the viewer part.

IV.3.b.ii. Schedules

To be able to manage properly a project, schedules are essential. Especially in big projects and long terms project, where multiple actors will take part. If it is possible to use the BIM 360 Calendar tool, it is not possible to link it to the interface via the code linking the BIM 360 Docs account. Unfortunately, only uploaded documents can be shown on the interface. All the tools found in this account are not transferable. Another solution has to be found. Because a part of the interface is coded in HTML, it is possible to add schedules to a web page. Indeed, an Excel file

can be integrated into an HTML code. To do so, the Excel file has first to be shared as a web page. This enables to have an HTML link. The format of the schedules becomes .html and can be integrated into the code easily. Like for the charts function, a button is created and the file is linked as the source. Again, it only appears when the button is selected. To not always have the schedule open, it is also possible to make it disappear by unclicking the button. To do this, lines of code have to be added in the main.css sheet. Main.css is responsible for the design of the page. It is where the position of the different buttons are decided. It is also there that it is decided if a "Click/Unclick" option has to be done. However, this code is a bit limited. For example, it cannot be combined with the Chart JS code. **Figure 24** shows an example of the Excel schedule integrated on the interface.

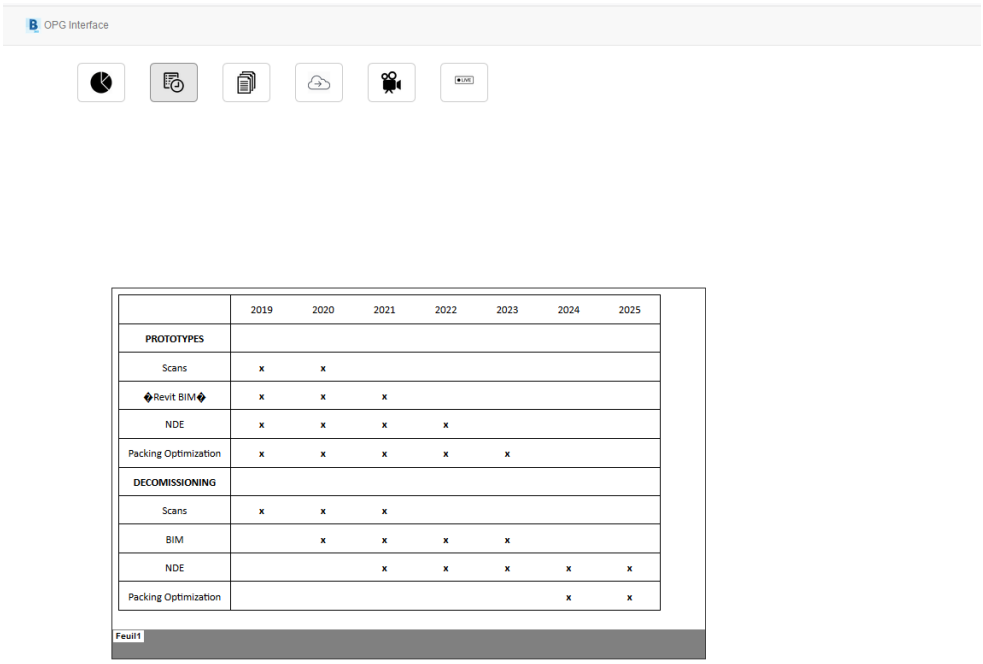


Figure 24. *Example of an Excel schedule integrated to the interface*

As presented before, other management tools such as MS Project® are much used in project management. They could also possibly be integrated into the HTML page by the same approach. First, they to be converted to HTML files and then can be referred to in order to be integrated to the web page.

IV.3.b.iii. Documents

BIM 360 was previously presented as a solution to access files from the interface by linking the account to the Forge Viewer. But what about files that cannot be shared in a cloud-based storage platform, even if it is secured? Some files in confidential projects are stored in local servers. Others do not concern other members or are just not important enough to be uploaded in a group project but can still have important information for one user.

However, for security reasons, it is not possible to access a local folder directly from the interface. The browser does not allow to upload a file on a web page. Moreover, as said earlier, the interface is just an intermediate between the user and other application. Uploading files directly n the interface is against this idea and not following security rules. To manage this issue, a link to local documents from the interface is created (**Figure 25**). A web page where the local files can be found is developed. It is coded in webix, a tool to create applications in Javascript and HTML language. This enables to create a clear environment for data storage. On this web page, documents, pictures and another kind of information can be uploaded and will be stored, even when the interface is not running. Creating this new HTML page is a good option to pass the security rules of the browser, and still assuring the privacy of the data.

Once again, a button is created. This time, it is interesting to keep the interface open at the same time as going through the user's documents. It was thus decided to open the linked environment in a new tab when the button is selected. This is done by adding the option "target:_blank" to the button HTML code.

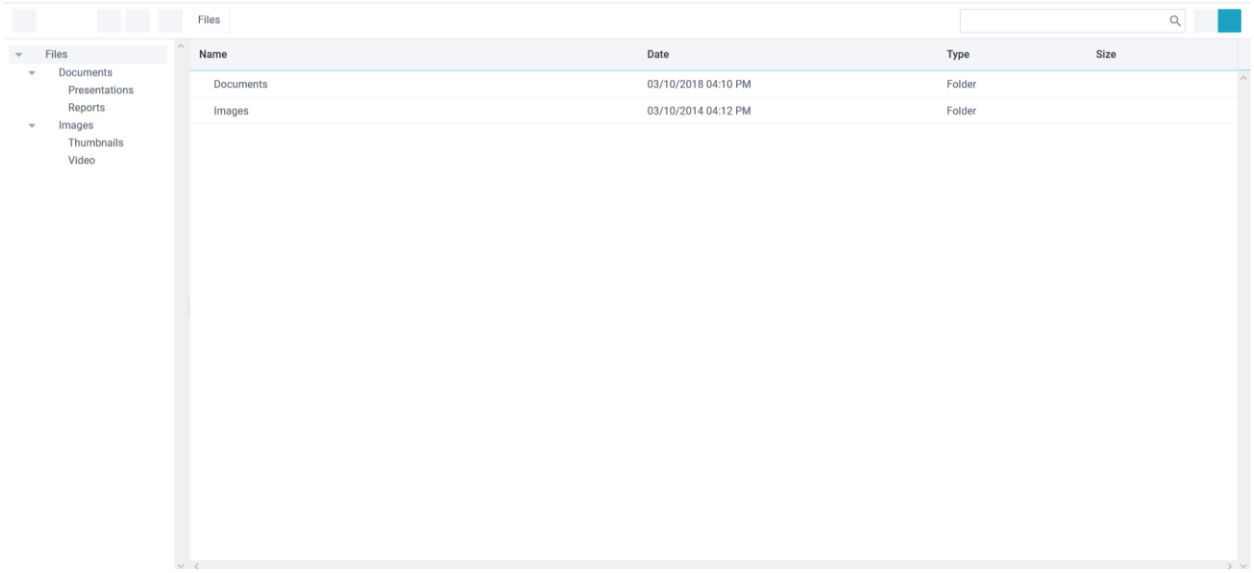


Figure 25. *Links to local documents by the interface.*

IV.3.b.iv Links to Demonstrations and scans

The interface is also a place to see more information on the project and how it evolved. It is a good opportunity to integrate demos of the scanning process and NDE approach, and also explain more about BIM. In fact, for some members of the project, the lack of knowledge on BIM or Scan-to-BIM is a problem. In **Chapter II**, it was explained that, for practitioners and specialist, a cloud-based BIM would induce the need for teaching and education. Especially in big projects where members come from various backgrounds. Therefore, tutorial videos and demos can be integrated into the interface for any members to educated themselves on different subjects (**Figure**

26). This way anyone can understand better the project directly from the interface and take part in decision making. An HTML link is created and attached to this function. This way, the user does not leave the interface while watching tutorials. A new web page is developed as a "Demonstration Page", where links to videos are attached and can be watched directly on the page. But other options can be added. As it is developed in HTML, a lot of other links to tutorials or products' descriptions can be integrated.

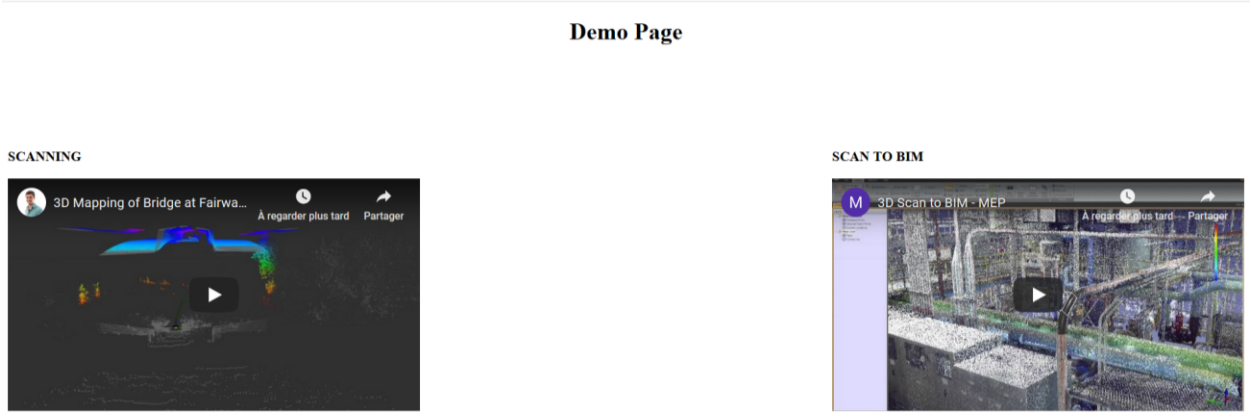


Figure 26. *Demo page accessible from the interface.*

As explained in the Introduction (**Chapter I**), this research is part of a bigger project regrouping scans, NDE, and packing optimization. The interface is an environment where all the data from the project is gathered. It should thus integrate the other parts. Unfortunately, Forge Viewer is not able to show scans. And from the BIM 360 account, it is also Impossible to get the point clouds and view them. Point clouds are voluminous files and contain a lot of information. To prevent loss of data and be able to see the scans of the project, more specific viewers have to be used. But they cannot be integrated to an HTML page. While the demo page can give the process of how to get the scans, another web page has to be created for the clients to have access to them (**Figure 27**). In our case, people working on the scanning aspect created a web page where all scans are viewable. This is what was linked to our interface. The point cloud viewer used here is POTREE, an open code that can be found on Github. It enables many options like the Forge Viewer with 3D models. Indeed, the user can walk through the scanned environment, measure, and zoom in and out on some points. As it is also opened in a new tab, the user can compare scans to the open model on the interface.

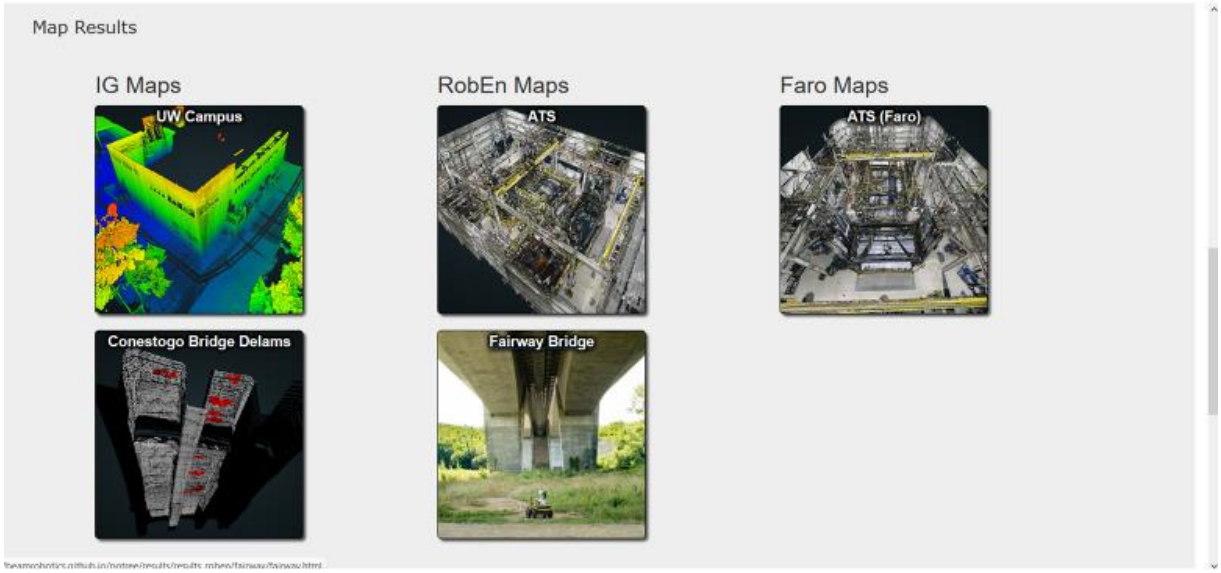


Figure 27. *Point cloud viewer page accessible from the interface.*

IV.4. Generalization of this approach to other projects

Construction projects are more and more leaning on BIM. Improvements in this sector are depending on improvements in BIM, as a designing but also an operating tool. To reach BIM Level 3, an integrated system that supports all members of a project is a necessity [**24**]. By offering a space where all information is shared, gathered and that helps all members of a project, this research tends to aim a more developed BIM aspect. In all part of the creation of this interface, the first priority along with safety was to keep it simple and customizable. As this research is related to nuclear decommissioning, all the functions developed here are done to help that cause. But it is very important to notice that each and every one of these can either be developed in another way or deleted. And that new ones can be added on the same schema.

The importance of keeping the coding part simple is also a priority. No web designers or computer science engineers are part of this thesis. And it is also the case for most of the construction/deconstruction project. Therefore, by having the already conceived interface, the additional codes will not have to be too complex. For simple functions, Github can provide codes to integrate. And a lot of them contain the most management options needed during maintenance and operation phases of construction. The languages and tool used are also reflecting the purpose of helping the future users. HTML offers also a lot of possibilities by linking easily another web page or integrating some formats to one page. Chart JS and Webix are tools for helping nonspecialist develop new applications.

For example, this interface for the construction of a building would offer a viewer, with the different objects and their properties listed. The schedule would still be useful, as well as the charts of the entire project for the providers. If scanning and NDE would probably not be part of

the project, the live data graph and the link to scans can be replaced by a table listing the elements needed for the construction as an Excel file. Or could be replaced by a link to providers, and comparison. Maybe even animations of the construction during the project with a timeline and simulation.

But it does not only stand for a construction project, and can be applied to any operations using BIM as a design tool and neglecting it in operation and maintenance phases. For instance, the infrastructure and transportation domain find an interest in BIMs but cope with the same problem than in construction. The lack of transition between design/construction and operation is significant.

This research proposes an architecture in the form of a prototype interface to gather information of a civil engineering project all in the same space. It follows the idea of having access to every useful data at once. But stays focused on keeping the security, the ownership and the confidentiality of some documents by classifying all users in three categories: administrators and editors, who can view and modify documents; viewers, who can only see without editing; and limited actors, who have access to data only concerning their work.

V. Conclusion and Future Work

V.1. Conclusion of this research

As a part of an ongoing decommissioning project, this thesis had the main objective which was creating BIMs from point cloud collected by a SLAM technology. The main purpose was to find the best solution to transcribe important details from a point cloud to a 3D model. By comparing manual and semi-automatic recognition, the choice was made easily. Indeed, the lack of precision and accuracy of the semi-automatic product, and the fact that a manual action was still predominant lead to choosing a manual approach. By using the scan as a base for the model, the work of the designer is to pick the similar BIM objects and draw them on top of it. Whereas it seems like a repetitive and hard task, it actually only asks for basic knowledge of BIM software and elements. No measures or special task is needed if a normal 3D model is wanted as a result.

The main issue that came out of this research is the need for a license to show and edit a 3D BIM model. The same problem comes for the other part of the project, like NDE or scans viewings. Viewers exist but a simple option did not yet exist. For such a huge project, a way to remedy this problem was looked at. A choice of developing an interface where all data could be gathered was made. Not only could it bring the possibility to view BIMs and their properties, but also it had other options and functions helping management for the operation and maintenance phases. With the help of existing products, this environment offers the possibility of also marking up the models and listing issues, as well as looking at the different elements and their properties individually.

Other functions were developed, keeping in mind that this is then transmitted to the client and members of the project with different backgrounds. Two main priorities thus appeared: security and facility of utilization. For the first one, a 3 legged token security was generated in the code. Meaning an authentication is necessary, as well as authorization of sharing the data. Also, no information is directly downloaded to the interface. It only plays the role of an intermediate by linking the user to other applications regrouped in the same space. Indeed, all the applications used here could be developed or reached individually, but are all gathered on the same local web page. The local aspect of the deployment is finally another security insurance. The interface cannot be found outside of the local server or computer.

Finally, to keep easy utilization and keep the interface open to future modifications, only common coding languages were used. The customizable aspect was very important for generalization and future prospects and the tools used in the interface transcript this approach. Forge Viewer is a good option since it can integrate many extensions already coded or new ones. Chart JS and Webix can help generate charts and web pages for non-specialists. Potree can

help view heavy point clouds that could not be seen otherwise. BIM 360 Docs helps the administrative and management aspect and is in constant improvement.

As a final result of this research, an environment where the BIMs created can be seen as well as all data directly related to the project can be found. One of the main objectives was to remedy the use of BIM during maintenance and operating phases. With this architecture, the user could be able to follow and manage the project which is supposed to last until 2100. The different versions of each document and model could be found and records of the decommissioning could be kept. The prototype developed in this research could be generalized to other projects that have a need for BIM in management and for operation, by adding more specific and complex functions to it in a simple way. The contribution of this thesis is not only gather data but create an environment where it can then be used for a long period. Even if edited, the former versions are kept and are accessible. Information from different technologies can be linked to a same space, and can be used together by using relationships between them.

V.2 Future Work

The first improvement that could be done is the automatic recognition process. As nuclear power plants have very similar elements composing it (mainly wall and pipes), an automatic extraction could be time-saving. In this research, only one commercial product was tested, which was an extension for Revit called "Scan-to-BIM". As it was not satisfying, it is important to note that many other products are on the market concerning these tasks. Future work could focus on trying out this other software and see their accuracy compared to the one used in this thesis. However, a quality-price ratio will also have to be taken into consideration. Indeed, these commercial products are often very expensive, and thus the results have to be convincing enough to justify the high price. Also, "automatic feature extraction" is not yet totally an automatic process. Whatever the product is, a technician has to select points from a wanted element in the point cloud. But products are still in constant improvement, and this has been a focus in research this last decade. The future expectations in this filed are high and soon could we found very convincing software.

Also, this research only furnishes a prototype of an interface that could be used for civil projects. Further development needs to be done, especially concerning the web page design. All the features here are coded in the simplest way. Indeed, the priority was to develop useful tools to build a prototype of a future application. The only interest that was made on the design was for the clarity of the data and buttons. Moreover, for projects that do not concern decommissioning, certainly, other functions will need to be coded. However, this research could be a starting point for future applications. An interesting part of this interface is to link all the data from the project. Therefore, linking the corresponding databases to the running code will need to be done. This can be developed with the help of MySQL or Excel file, for simpler cases.

An interesting improvement would also be to extend this interface to mobile devices. This way, it would be accessible on smartphones or tablets and could be brought on the field. For other construction or deconstruction projects, it is sometimes important to have access to all the data whenever needed.

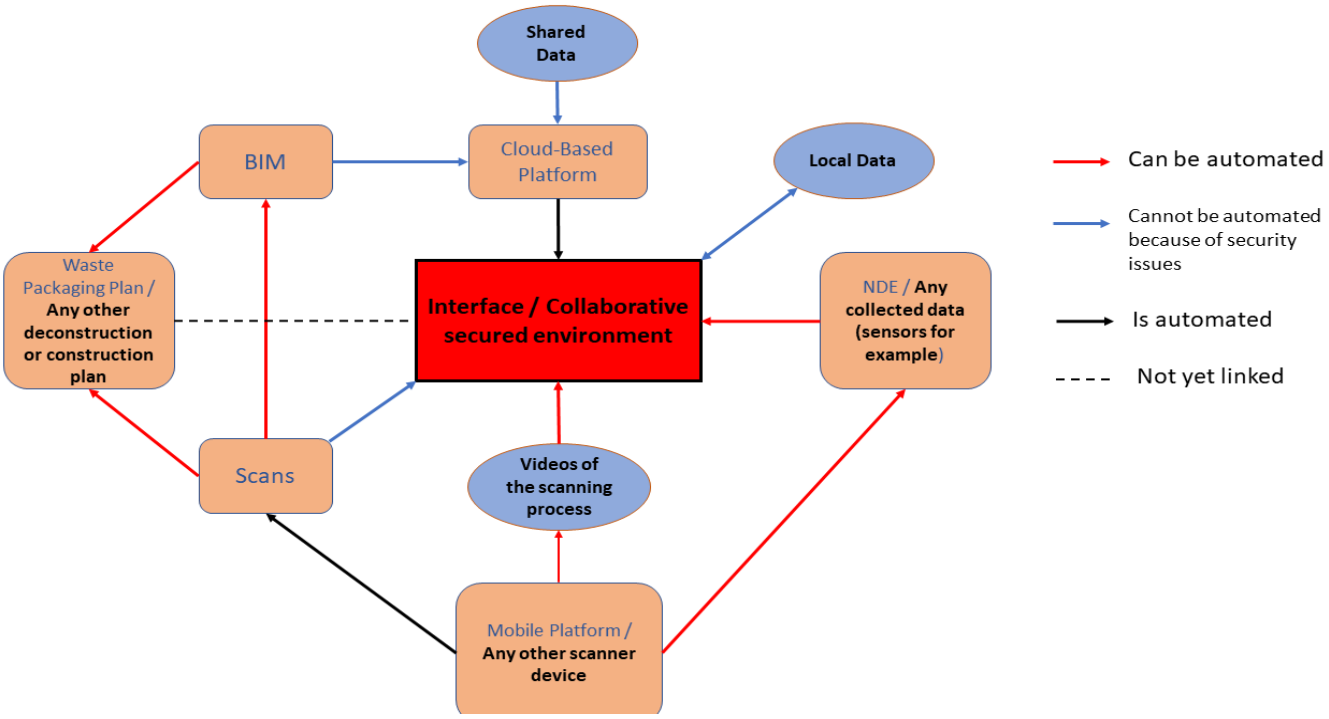


Figure 28. *Recapitulation of the research and possible Generalisation.*

References

[**1**] "State of research in automatic as-built modeling", V. Patraucean, I. Armeni, M. Nahangi, J. Yeung, I. Brilakis, C. Haas, *Advanced Engineering Informatics 29*, 2015

[**2**] "Scan to BIM for secondary building components", A. Adan, B. Quintana, S. Prieto, F. Bosché, *Advanced Engineering Informatics 37*, 2018

[**3**] "The value of integrating Scan to BIM and Scan vs BIM techniques for construction monitoring using laser scanning and BIM: The case of cylindrical MEP components", F. Bosché, M. Ahmed, Y. Turkan, C.T. Haas, R. Haas, *Automation in Construction 49*, 2015

[**4**] "Point cloud quality requirements for Scan vs BIM-based automated construction", D. Rebolj, Z. Pucko, N.C. Babic, M. Bizjak, D. Mongus, *Automation in Construction 84*, 2017

[**5**] "A 4D augmented reality model for automating construction progress monitoring data collecting, processing, and communication", M. Golparvar-Fard, F. Pena-Mora, S. Savarese, *Journal of Information Technology in Construction 13,* 2009

[**6**] "Using digital photogrammetry for pipe-works progress tracking", M. Ahmed, C. Haas, R. Haas, *Canadian Journal of Civil Engineering 39,* 2012

[**7**] "Automatic reconstruction of as-built BIM from Laser scanned point clouds: A review of related techniques", P. Tang, D. Huber, B. Akinci, R. Lipman, A. Lytle, *Automation in Construction 19,* 2010

[**8**] "Automatic surface flatness control using terrestrial laser scanning and BIM models", F. Bosché, E. Guénet, *Automation in Construction 44*, 2014

[**9**] "4-plane congruent sets for automatic registration of as-is 3D point clouds with 3D BIM models", M. Bueno, F. Bosché, H. Gonzalez-Jorge, J. Martinez-Sanchez, P. Arias, *Automation in Construction 88*, 2018

[**10**] "Method for registration of 3D shapes", P. Besl, N. Mckay, *Proceedings Volume 1611*, 1991

[**11**] "Automated progress control using laser scanning technology", C. Zang, D. Arditi, *Automation in Construction 36*, 2013

[**12**] "Automated retrieval of 3D CAD model objects in construction range images", F. Bosché, C.T. Haas, *Automation in Construction 17*, 2008

[**13**] "Parallel systems and Structural frames realignment planning and actuation strategy", M. Nahangi, P. Czerniawski, C. Haas, S. Waldbridge, J. West, *Journal of Computing in Civil Engineering 30,* 2016

[**14**] "Automated construction progress measurement using a 4D BIM and 3D data", C. Kim, H. Son, C. Kim, *Automation in Construction 31*, 2013

[**15**] "Performance Evaluation of automatically generated BIM from Laser scanner data for sustainability analysis", C. Wang, Y.K. Cho, *Procedia Engineering 118*, 2015

[**16**] "3D structural component recognition and modeling method using color and 3D data for construction progress monitoring", H. Son, C. Kim, *Automation in Construction 19*, 2010

[**17**] "Autonomous Modelling of Pipes within Point Clouds", M. Ahmed, C. Haas, R. Haas, *30th International Symposium on Automation and Robotics for Construction and Mining*, 2013

[**18**] "Automatic detection of cylindrical objects in built facilities", M. Ahmed, C. Haas, R. Haas, *Journal of Computing in Civil Engineering*, 2014

[**19**] "An approach to combine progressively captured point clouds for BIM update", T. Gao, B. Akinci, S. Ergan, J. Garrett, *Advanced Engineering Informatics 29*, 2015

[**20**] "Automatic reconstruction of parametric building models from indoor point clouds", S. Ochmann, R. Vock, R. Wessel, R. Klein, *Computers&Graphics 54*, 2016

[**21**] "Automated Removal of Planar Cutter from 3D Point clouds for improving industrial object recognition", T. Czerniawski, M. Nahangi, S. Waldbridge, C. Haas, *33rd International Symposium on Automation and Robotics for Construction*, 2016

[**22**] "Productive modeling for development of as-built BIM of existing indoor structures", J. Jung, S. Hong, S. Jeong, S. Kim, H. Cho, S. Hong, J. Heo, *Automation in Construction 42*, 2014

[**23**] "Plane based registration of construction laser scans with 3D/4D models", F. Bosché, *Advanced Engineering Informatics 26*, 2012

[**24**] "Prioritizing object types for modeling existing industrial facilities", E. Agapaki, G. Miatt, I. Brilakis, *Automation in construction 96*, 2018

[**25**] "Building information modeling for FM: are we there yet ?", R. Edirisinghe, K. London, P. Kalutara, G. Aranda-Mena, *Engineering Construction & Architectural Management 24*, 2017

[**26**] "The use of BIM-based framework to support safe Facility Management processes", E.M. Wetzel, W.Y Thabet, *Automation in Construction 60*, 2015

[**27**] "A study on software architecture for effective BIM/GIS-based facility management data integration", T.W. Kang, C.H. Hong, *Automation in Construction 54*, 2015

[**28**] "Planning and developing facility-management enabled BIM", P. Pishdad-Bozorgi, X. Gao, C. Eastman, A.P. Self, *Automation in Construction 87*, 2018

[**29**] "Issues in BIM for Facility Management from Industry's Practitioner's perspective", R. Liu, R.R.A. Issa, *Computing in Civil Engineering 2013*

[**30**] "BIM for FM literature review and future needs", M. Yalcinkaya, V. Singh, *Product Lifecycle Management for a Global Market*, 2014

[**31**] "Developing mobile-and BIM-based integrated visual facility maintenance management system", *The Scientific World Journal 7*, 2013

[**32**] "A review of cloud-based BIM technology in the construction sector", J. Wong, X. Wang, H. Li, G. Chan, Ha. Li, *Journal of Information Technology in construction*, 2014

[**33**] "Applying cloud computing technology to BIM visualization and manipulation", T-H. Chuang, B-C. Lee, I-C. Wu, *28th International Symposium on Automation and Robotics in Construction*, 2011

[**34**] "Virtual reality-based cloud BIM platform for integrated AEC projects", J. Goulding, F. Rahimian, X. Wang, *Journal of Information on Construction, 19 pp308-325 ISSN 1874-4753*, 2014

[**35**] "Cloud and open BIM-based building information interoperability research", Juan, Q. Zheng, *Journal of service science and management*, 2014

[**36**] "Towards cloud Augmented Reality for Construction application by BIM and SNS integration", Y. Jiao, S. Zhang, Y. Li, Y. Wang, B. Yang, *Automation in Construction 33*, 2012

[**37**] "Component-based engineering of a mobile BIM-based augmented reality system", S. Meza, Z. Turk, M. Dolenc, *Automation in Construction 42*, 2014

[**38**] "Exploring how information exchange can be enhanced through Cloud BIM", A. Redmond, A. Hore, M. Alshawi, R. West, *Automation for Construction 24*, 2012

[**39**] "Challenges to BIM-cloud integration: implication of security issues on secure collaboration", A-M. Mahamadu, L. Mahdjoubi, C. Booth, *IEEE International Conference on Cloud Computing Technology and Science*, 2013

[**40**] "Cloud-Based BIM Governance Platform Requirements and Specifications: Software Engineering approach using BPMN and UML", E. Alreshidi, M. Mourshed, Y. Rezgui, *Journal of Computing in Civil Engineering 30*, 2016

[**41**] "A framework of Cloud-Computing-based BIM service for building lifecycle", J.P. Yang, Q. Liu, F.Q. Yu, Z.Z. Hu, W.Z. Zhao, *Computing in Civil and Building Engineering*, 2014

[**42**] "BIMCloud: A distributed Cloud-based Social BIM Framework for Project Collaboration", M. Das, J.C.P. Cheng, S.S. Kumar, *Computing in Civil and Building Engineering*, 2014

[**43**] "Real-time progress management: Re-Engineering processes for cloud-based BIM in construction", J. Matthews, P.E.D. Love, s. Heinemann, R. Chadler, C. Rumsey, O. Olatunj, *Automation in Construction 58*, 2015

[**44**] "An intuitive robot teleoperation system for NPP decommissioning", C-H. Lee, T. Gu, K-M. Lee, S-J. Ye, Y-B. Bang, *Transactions of the Korean Nuclear Society Spring Meeting*, 2017

[**45**] "Modelling of Nuclear Power Plant Decommissioning financing", J. Bems, J. Knapek, T. Kralik, M. Hejhal, J. Kubancak, J. Vasicek, *Radiation Protection Dosimetry Vol 164 Issue 4*, 2015

[**46**] *Management of Radioactive Nuclear waste after an NPP accident*, *Chapter 7 and 8*, 2016, OECD (Organisation for Economic Co-Operation and development) and NEA (Nuclear Energy Agency, Radioactive Waste Management

[**47**] "Modeling of NPP decommissioning tasks-issues and their solution", M. Hornacek, V. Necas, *International Conference on Applied Physics of Condensed Matter*, 2016

[**48**] "Innovative NPP Building Arrangement in Consideration of Decommissioning", W-J. Choi, M-S. Roh, C-L. Kim, *Nuclear Engineering and Technology Vol 49*, 2017

[**49**] "Applying and adapting the Swedish regulatory system for decommissioning to nuclear power reactors – the regulator's perspective", M. Amft, M. Leisvik, S. Caroll, *Journal of Environmental Radioactivity 196*, 2019

[**50**] "Towards Advanced Robotics Manipulations for Nuclear Decommissioning", N. Marturi, A. Rastegarpanah, V. Rajasekaran, V. Ortenzi, *Intech*, 2017

[**51**] "Robotics for Nuclear Power Plants – Challenges and Future Perspectives", J. Iqbal, A. Mahmood Tahir, M. Razha UI Islam, *International Conference On Applied Robotics for the Power Industry*, 2012

[**52**] "Small teleoperated Robot for Nuclear Radiation and Chemical Leak detection", K. Qian, A. Song, J. Bao, H. Zhang, *International Journal of Advanced Robotics System Vol 9*, 2012

[**53**] "Intelligent radioactive waste process cloud-based system for nuclear power plant decommissioning", Y-C. Chen, H-M. Sun, R-S. Chen, J-H Yeh, X. Fan, I-H. Chou, *Journal of Ambient Intelligence and Humanized Computing*, 2018

[**54**] "Intelligent radioactive waste management platform for radioactive waste storage facilities", Y-C. Chen, Y-S. Yu, H-M. Sun, R-S. Chen, J-H Yeh, I-H. Chou, *Nuclear Technology 182*, 2013

[**55**] "Emergency Response to the Nuclear Accident at Fukushima Daiichi Nuclear Power Plant using Mobile Rescue Robots", K. Nagatani, S. Kiribayashi, Y. Okada, K. Otake, K.

Yoshida, S. Tadokoro, T. Nishimura, T. Yoshida, E. Konayagi, M. Fukushima, S. Kawatsuma, *Journal of Field Robotics 30*, 2013

[**56**] "Visual classification of waste material for nuclear decommissioning", A. Suffat, Y. Gao, J. Kuo, B. Bowen, P. Mort, *Robotics and Autonomous Systems 75*, 2016

[**57**] "Cost of decommissioning Nuclear Power Plants Chapter 3", Nuclear development 2016, *Nuclear Energy Agency and OECD report*

[**58**] "Robots application in NPP", F. Gelhaus, H. Roman, *Progress in Nuclear Energy Vol 23*, 1990

[**59**] "A robotic system to locate hazardous chemical leaks", R.A Russell, D. Thiel, R. Deveza, A. Mackay-Sim, *Proceedings of 1995 IEEE Conference on Robotics and Automation*, 1995

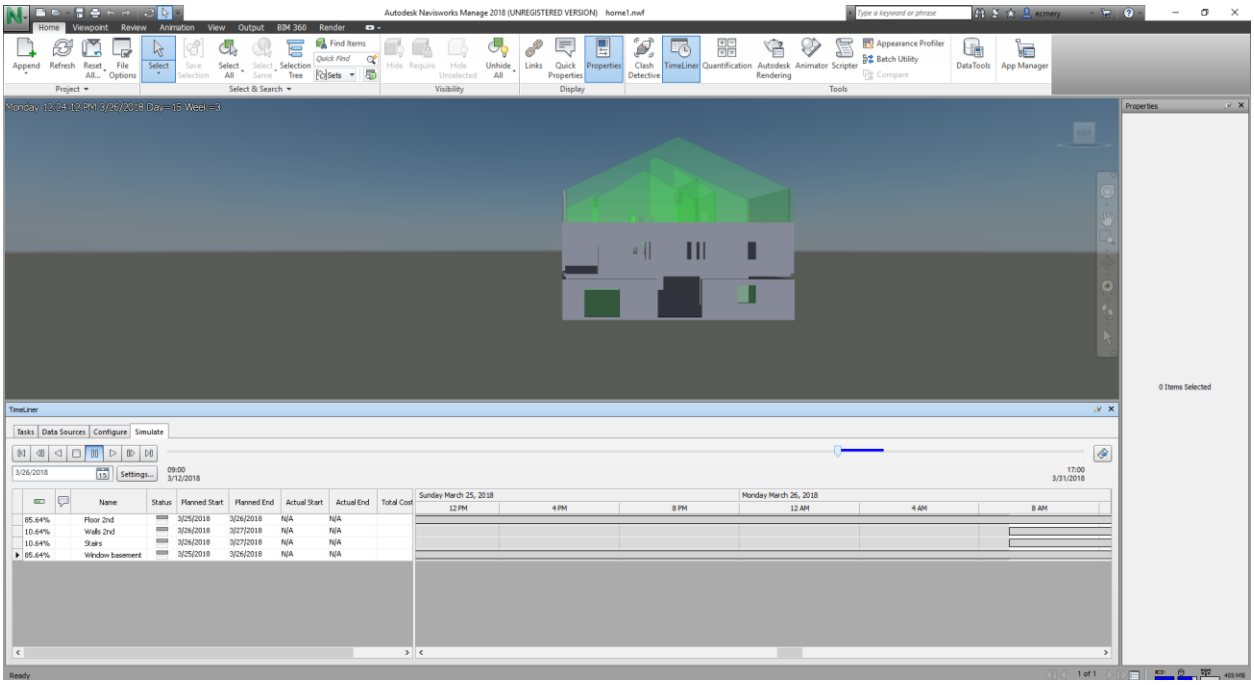
[**60**] Autodesk® Recap Website, Case studies: https://www.autodesk.com/customer-stories/case-studies?products=product--recap

[**61**] Autodesk® Forge Website : forge.autodesk.com

[**62**] Autodesk Forge RCDB, examples of applications developed with Forge's APIs: https://forge-rcdb.autodesk.io/configurator

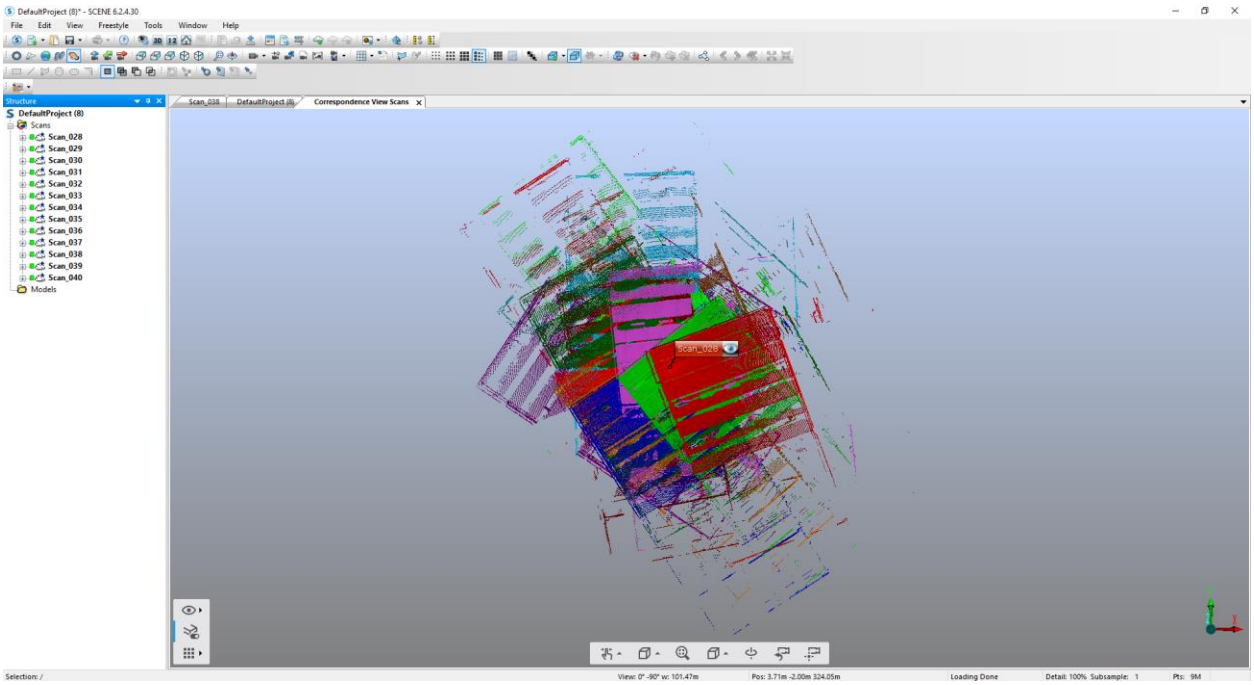
Appendix

Appendix 1: Navisworks® timeline and simulation project

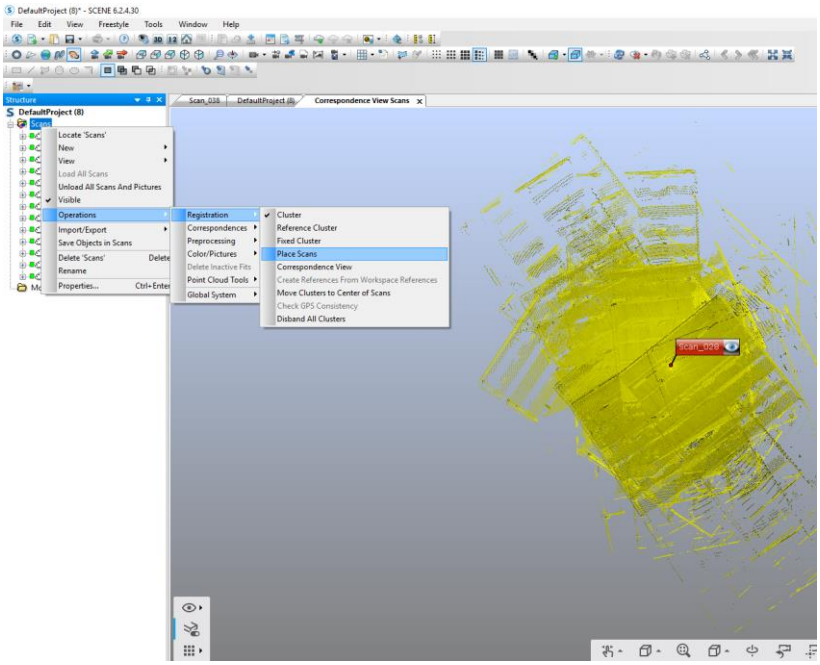


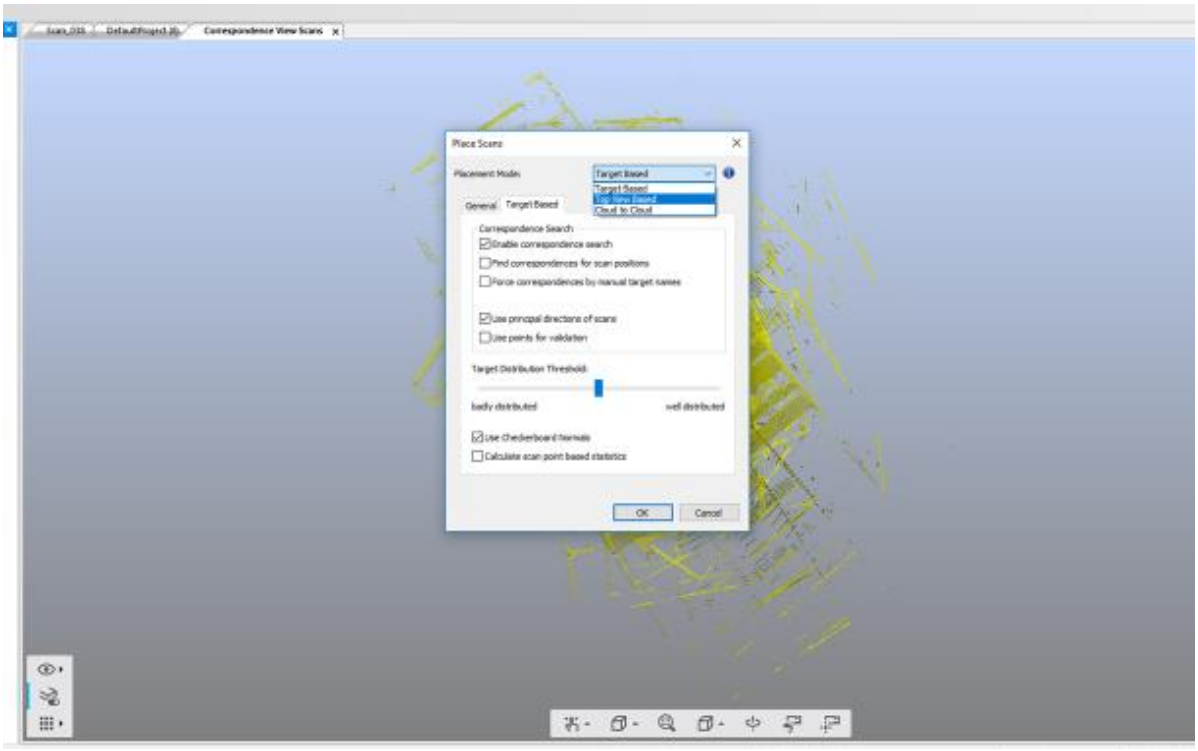
Appendix 2: Scan Registration with Scene®

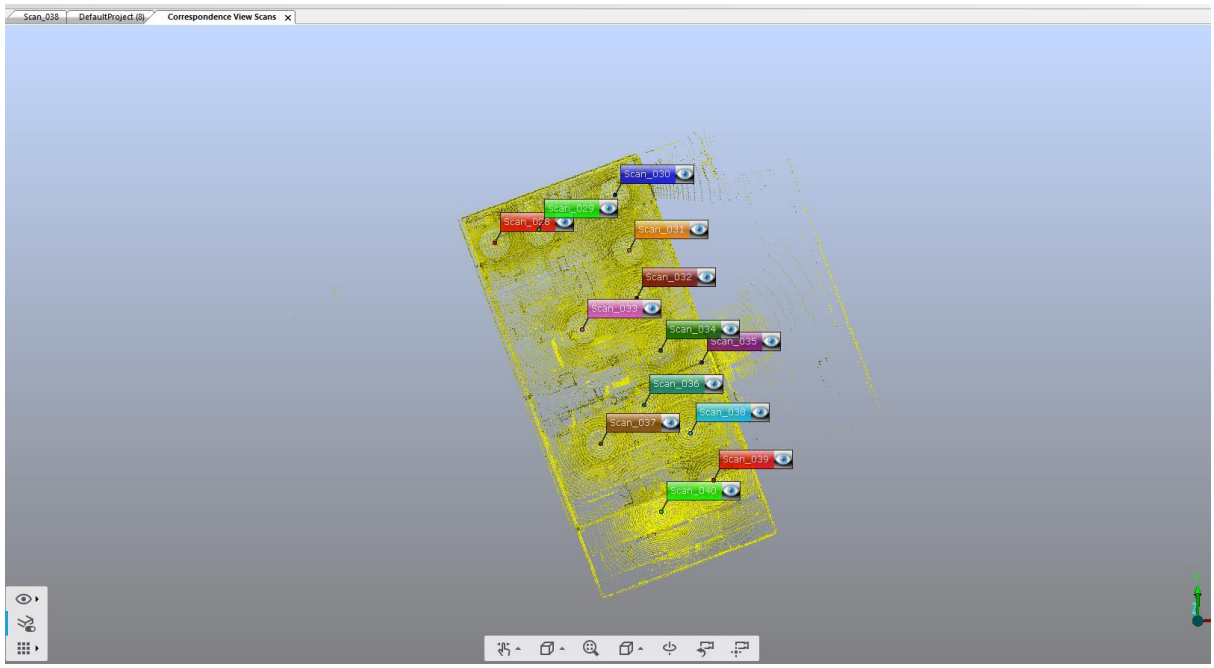
- **Laser scans opened in Scene®**



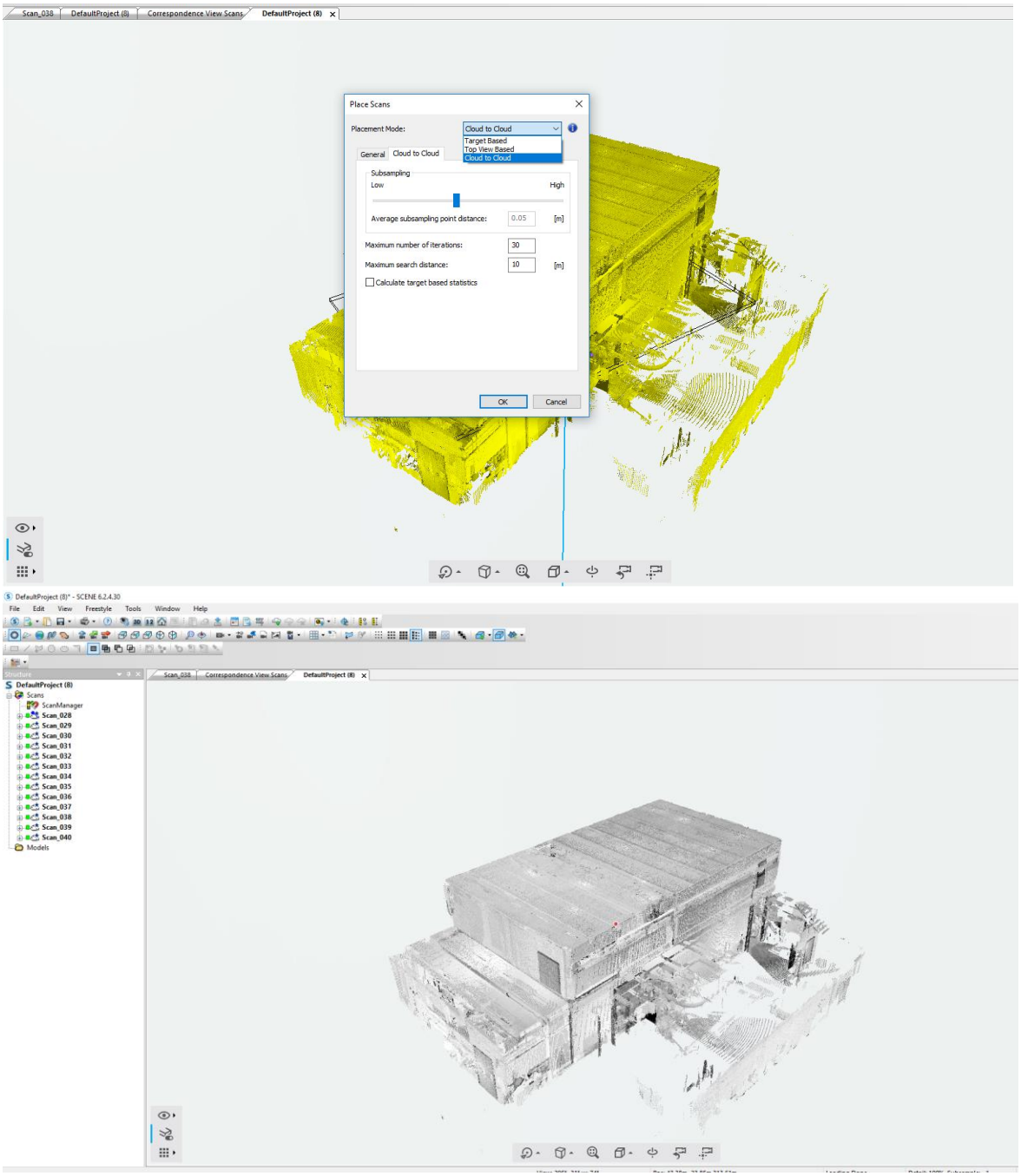
- **Top-View based Registration**



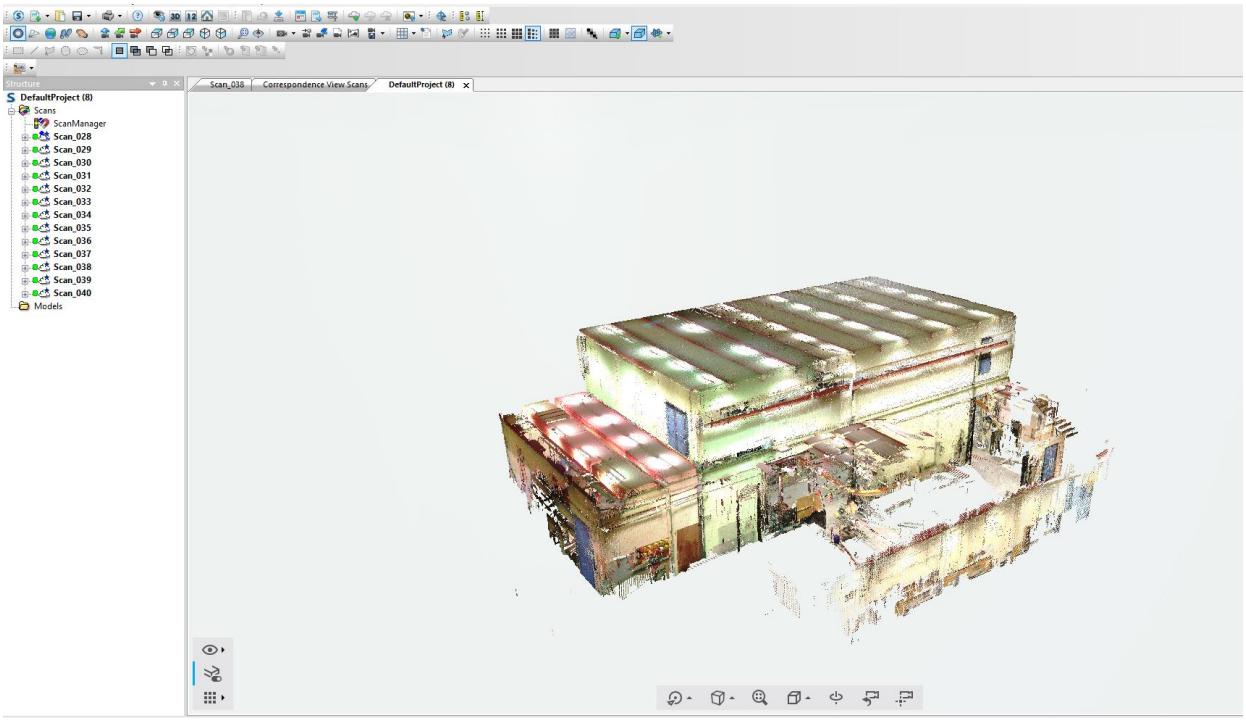




- **Cloud to Cloud-based registration**



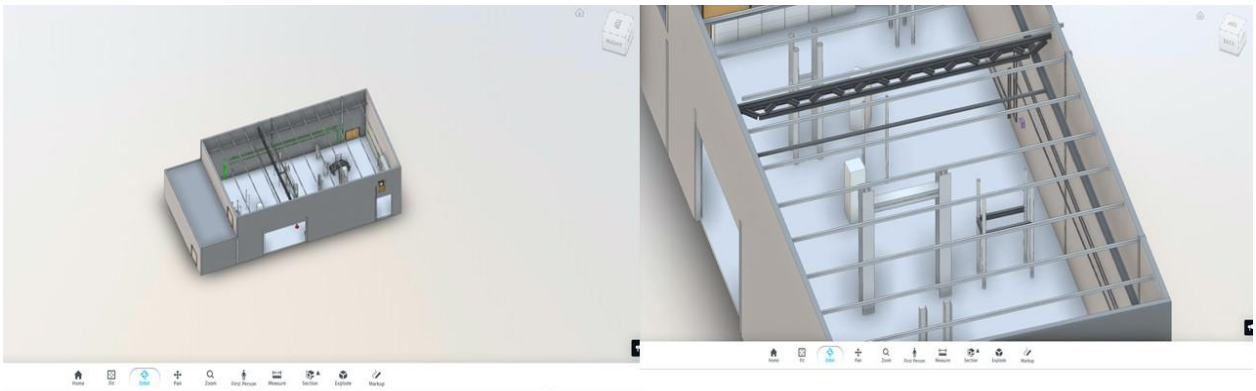
- **Colorization**


Structure
DefaultProject (8)
Scans
ScanManager
Scan_028
Scan_029
Scan_030
Scan_031
Scan_032
Scan_033
Scan_034
Scan_035
Scan_036
Scan_037
Scan_038
Scan_039
Scan_040
Models
Scan_038
Correspondence View Scans
DefaultProject (8)

Appendix 3: Basic Functions in Forge Viewer

1. Movement-based

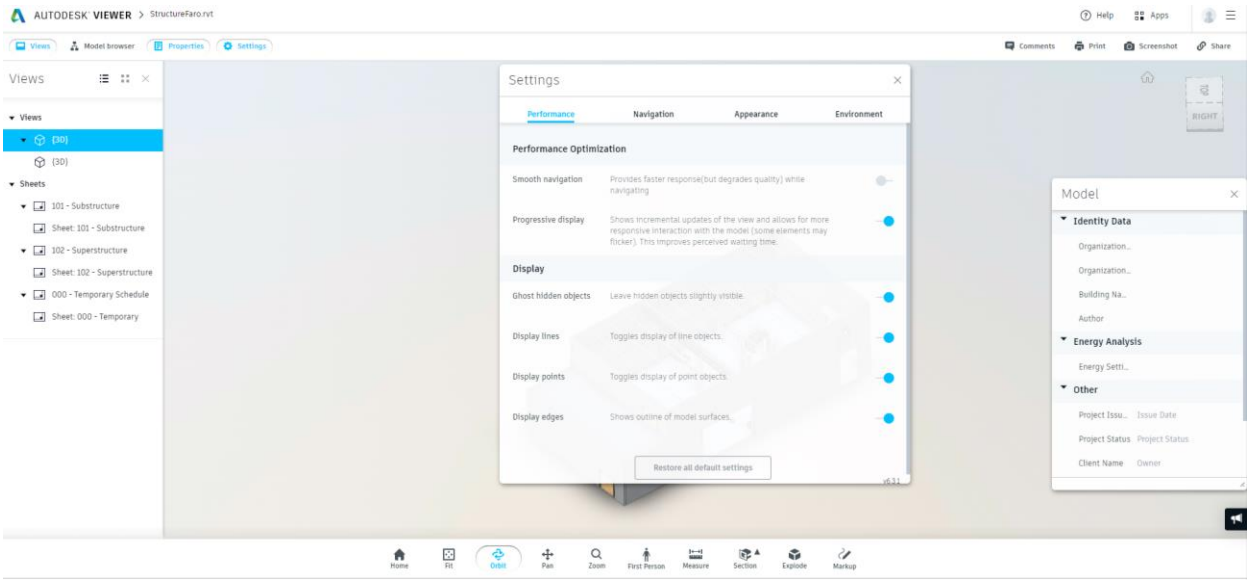
a).Rotation option



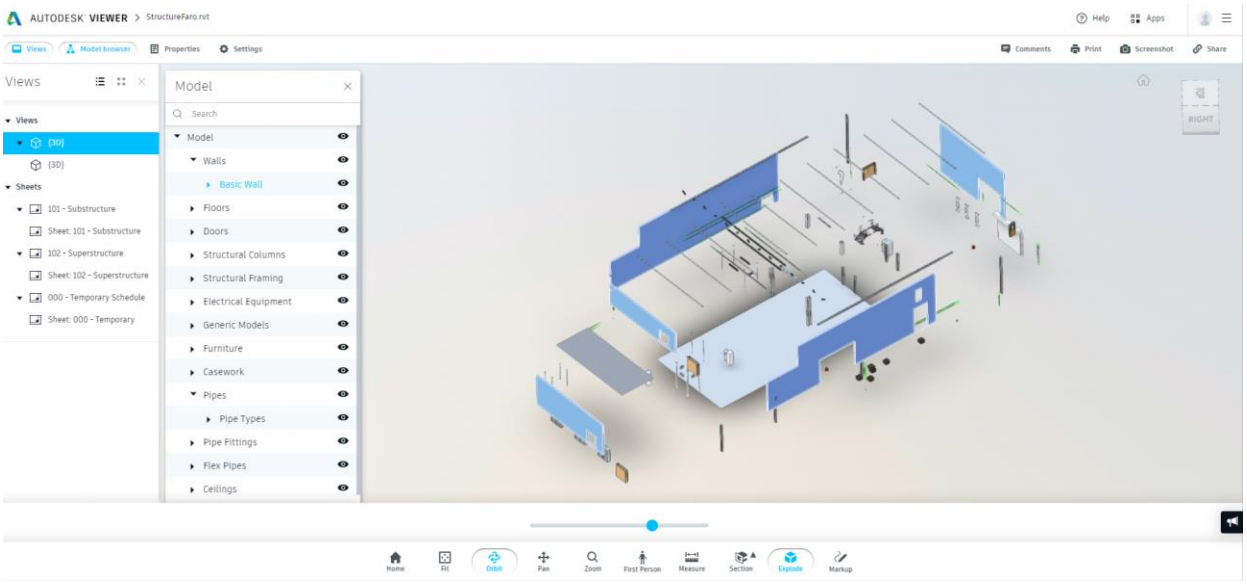
b).Zoom option

2. Placement based

3. Information based



4. Viewer based



Explode and select elements from a classified list

Appendix 4: List of viewer's extension openly available (*From Github.com)*

- Autodesk.ADN.Viewing.Extension.Basic

A basic Hello World extension that displays an alert dialog upon loading and unloading.

- Autodesk.ADN.Viewing.Extension.BasicES2015

A Hello World extension but written using ES6/ES2015 syntax. Needs transpiling with a tool like Babel or Traceur before being used with the viewer.

- Autodesk.ADN.Viewing.Extension.BoundingBox

Displays a bounding box around the selected component.

- Autodesk.ADN.Viewing.Extension.Chart

Displays a PieChart and a dropdown menu listing all available model properties. When a pie is selected in the chart, isolates the specific components.

- Autodesk.ADN.Viewing.Extension.ContextMenu

Illustrates how to customize the viewer context menu for zero-selection menu or item specific menu.

- Autodesk.ADN.Viewing.Extension.CustomTool

A basic viewer tool that just dumps events in the console, useful for testing and debugging or using as a tool boilerplate.

- Autodesk.ADN.Viewing.Extension.DockingPanel

A basic docking panel demo.

- Autodesk.ADN.Viewing.Extension.EventWatcher

Creates a panel which lets user activates any event available in the API. Output event arguments to a section.

- Autodesk.ADN.Viewing.Extension.Explorer

Creates a camera animation using a custom tool, rotating eye position around the model.

- Autodesk.ADN.Viewing.Extension.GeometrySelector

Illustrates how to snap geometry: vertices, edges, faces and how to create selection commands to let user pick geometry on the model.

- Autodesk.ADN.Viewing.Extension.Hotkeys

Creates hotkeys to switch viewer to fullscreen.

- Autodesk.ADN.Viewing.Extension.IFramePanel

Creates a simple docking panel containing an iframe.

- Autodesk.ADN.Viewing.Extension.Layers

Iterates through layers. Valid only for 2D drawings.

- Autodesk.ADN.Viewing.Extension.Markup3D

Add 3D markups on a 3D model. Uses StateManager extension to save & restore markups.

- Autodesk.ADN.Viewing.Extension.Material

Changes material of selected component. Supports color and textures.

- Autodesk.ADN.Viewing.Extension.MeshData

Access mesh data of selected component, vertices and edges to represent them graphically.

- Autodesk.ADN.Viewing.Extension.MeshImporter

Imports custom json into the viewer and creates three.js meshes from it.

- Autodesk.ADN.Viewing.Extension.MetaProperties

Adds some extra hardcoded properties to viewer property panel.

- Autodesk.ADN.Viewing.Extension.ModelStructure

Dumps model structure to browser console.

- Autodesk.ADN.Viewing.Extension.ModelStructurePanel

Custom model structure panel behavior.

- Autodesk.ADN.Viewing.Extension.PropertyDump

Dumps properties for selected component in browser console.

- Autodesk.ADN.Viewing.Extension.PropertyListPanel

Custom panel derived from property list panel

- Autodesk.ADN.Viewing.Extension.PropertyPanel

Inserts custom data into viewer property panel.

- Autodesk.ADN.Viewing.Extension.PropertyTranslator

Uses microsoft translation API to translate property panel values on the fly.

- Autodesk.ADN.Viewing.Extension.StateManager

Save and restore states of the viewer (position, markups, rotation, zoom, etc).

- Autodesk.ADN.Viewing.Extension.ScreenShotManager

Creates a panel that lets you manage screenshots taken with the API.

- Autodesk.ADN.Viewing.Extension.Toolbar

Various toolbar controls customization example.

- Autodesk.ADN.Viewing.Extension.UIComponent

Illustrates on to create a docking panel more advanced than the basic docking panel extension.

- Autodesk.ADN.Viewing.Extension.Workshop

Appendix 5: Code lines of the cloud-based interface

- **General aspect and decoding (following the Autodesk® Forge Tutorial)**

Start.js:

```
'use strict';

var app = require('./server/server');

// start server
var server = app.listen(app.get('port'), function () {
  if (process.env.FORGE_CLIENT_ID == null || process.env.FORGE_CLIENT_SECRET ==
null)
    console.log('****************\nWARNING: Forge Client ID & Client Secret not
defined as environment variables.\n****************');

  console.log('Starting at ' + (new Date()).toString());
  console.log('Server listening on port ' + server.address().port);
})
```

App.js:

```
{
    "name": "BIM 360 Visual Reports",
    "description": "BIM 360 Visual Reports",
    "repository": "https://github.com/Autodesk-Forge/bim360appstore-viewer-
nodejs-visual.reports",
    "logo": "https://avatars0.githubusercontent.com/u/8017462?v=3&s=200",
    "keywords": [
      "express",
      "framework",
      "autodesk",
      "forge",
      "template",
      "bim360appstore"
    ],
    "env": {
      "FORGE_CLIENT_ID": {
        "description": "bMSogt6cGF8lVGFZoHMM3bcvmim4TYFc"
      },
      "FORGE_CLIENT_SECRET": {
        "description": "e22luc6kJCRL6hlv"
      },
      "FORGE_CALLBACK_URL": {
```

```
        "description": "Callback URL of your Forge app, required for 3-legged
OAuth",
        "value": "http://localhost:3000/api/forge/callback/oauth"
      }
    },
    "success_url": "/"
  }
```

Package.json:

```
{
  "name": "visualreportstest",
  "version": "1.0.0",
  "description": "",
  "dependencies": {
    "body-parser": "^1.18.3",
    "bootstrap": "3.3.7",
    "chartjs-plugin-streaming": "^1.7.1",
    "clipboard": "^1.7.1",
    "cookie-parser": "^1.4.3",
    "d3": "^5.7.0",
    "d3pie": "^0.2.1",
    "express": "^4.16.3",
    "express-session": "^1.15.6",
    "forge-apis": "0.4.3",
    "jquery": "^3.2.1",
    "jstree": "3.3.4",
    "moment": "^2.22.2",
    "notifyjs": "^3.0.0",
    "nvd3": "^1.8.6",
    "request": "^2.81.0"
  },
  "repository": {
    "type": "git",
    "url": ""
  },
  "scripts": {
    "start": "NODE_ENV=production node start.js",
    "dev": "node start.js",
    "nodemon": "nodemon start.js --ignore www/"
  },
  "author": "Jaime Rosales (Forge Partner Development)",
  "license": "MIT"
}
```

Main.css:

```
body {
  padding-top: 3%; /* space for the top nav bar */
  font-size: 14px !important;
}

#pieChart > p{
  margin-left: 50px;
  width: 40%;
  padding-top: 15px;
}


html, body {
  height: 100%;
}

svg{
  margin-left: 0%;


.data-container {
  height: calc(100vh - 55px);
  width: calc(30vw - 36px);
  overflow: auto;
}


.report-dropdowns {
  margin-left: 7.5%;
  padding-top: 2%;
  font-size: 8px !important;
  visibility: hidden;
}

.svgcanvas{
  margin-left: 2%;
}

#dropdown2dviews {
  height: 2%;
  width: 40% !important;
  display: block;
  position: absolute;
  bottom: 0;
```

```
  top: 51vh;
  left: 28%;
  right: 0;
  z-index: 2;
  visibility: hidden;
}


#forgeViewer {
  height: 47% !important;
  width: 40vw !important;
  display: block;
  position: absolute;
  bottom: 0;
  top: 4%;
  left: 28%;
  right: 0;

}

#viewerSecondary {
  height: 47% !important;
  width: 40vw !important;
  display: block;
  position: absolute;
  bottom: 0;
  top: 53vh;
  left: 28%;
  right: 0;

}

#positionCharts {
  height: 94vh !important;
  width: 45vw !important;
  display: block;
  position: absolute;
  bottom: 0;
  left: 67%;
  top: 6%;
  right: 0;
  overflow: auto;
}

#Schedule {
```

```
  margin-left:30px;
  margin-top:100px;
}

#Charts {
  margin-left: 100px;
  margin-top:100px;
}

#Documents {
  margin-left:30px;
  margin-top:100px;
}

#Scans {
  margin-left:30px;
  margin-top:100px;
}

#myChart1{
  margin-top:150px;
}
#myChart2 {
  float:right;
  margin-top:150px;
}

#Demos {
  margin-left:30px;
  margin-top:100px;

}

#Live {
  margin-left:30px;
  margin-top:100px;
}
#ScheduleExcel {
  display:none;
  margin-bottom:100px;
  margin-top:-50px;
}
```

Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Visual Reports</title>
  <!-- Common packages: JQuery & Bootstrap -->
  <link rel="stylesheet" href="/css/bootstrap.min.css">
  <link rel="stylesheet" href="/css/jstree/style.css"/>
  <link rel="stylesheet" href="/css/nv.d3.css">
  <link rel="shortcut icon" href=/img/favicon.ico type=image/x-icon>
  <link rel="stylesheet" href="/css/main.css"/>
  <script src="/js/jquery.min.js"></script>
  <script src="/js/bootstrap.min.js"></script>
  <script src="/js/jstree.min.js"></script>
  <script src="/js/moment.min.js"></script>
  <script src="/js/clipboard.min.js"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/canvasjs/1.7.0/canvasjs.min.js"></scr
ipt>
  <script
src="https://developer.api.autodesk.com/modelderivative/v2/viewers/three.min.js">
</script>
<script
src="https://developer.api.autodesk.com/modelderivative/v2/viewers/viewer3D.min.j
s"></script>
  <!-- Notification ballons -->
  <script src="js/libraries/notify.min.js"></script>

  <!-- Autodesk Forge Viewer files -->
  <link rel="stylesheet"
href="https://developer.api.autodesk.com//modelderivative/v2/viewers/style.min.cs
s?v=v6.4" type="text/css" />
  <script
src="https://developer.api.autodesk.com//modelderivative/v2/viewers/viewer3D.min.
js?v=v6.4"></script>
  <script src="https://d3js.org/d3.v3.min.js"></script>
  <script src="/js/libraries/d3.min.js"></script>
  <script src="/js/libraries/d3pie.min.js"></script>
  <script src="/js/libraries/nv.d3.min.js"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.7.3/Chart.min.js"></script
>
  <script
src="https://autodeskviewer.com/viewers/2.2/extensions/MarkupsCore.min.js"></scri
pt>
```

```
  <!-- BIM360 packages: login and authentication -->
  <script src="/js/oauth.js"></script>
  <script src="/js/forge.data.management.js"></script>
  <script src="/js/forge.viewer.js"></script>
  <script src="/js/reportData.js"></script>
  <script src="/js/barchart.js"></script>
  <script src="/js/piechart.js"></script>
  <script src="/js/ProjectCharts.js"></script>
  <script src="/js/schedule.js"></script>
  <script src="/js/markupExt.js"></script>

</head>
<body>
<nav class="navbar navbar-default navbar-fixed-top">
  <div class="container-fluid">
    <ul class="nav navbar-nav left">
      <li>
        <a href="http://developer.autodesk.com" target="_blank">
          <img alt="Autodesk Forge" src="/img/bim360.png" height="22" />
          OPG Interface
        </a>
      </li>
    </ul>

    <ul class="nav navbar-nav navbar-right">

      <li>
        <a href="#" id="signInButton"><span class="glyphicon glyphicon-user"
id="signInProfileImage"></span>
          <span id="signInButtonText" title="Sign in with your Autodesk
Account">Sign in</span></a>
      </li>
    </ul>


  </div>
</nav>
<!-- End of navbar -->

<div class="container-fluid">
      <div class="row data-container">
        <div class="col-sm-3 col-md-2 sidebar " id="dataManagementHubs">
```

```
          <Button class="btn btn-lg btn-default button-ipad"
onclick="forgeSignIn()"><img src="img/autodesk_text.png" height="20px"/> Sign
in</Button>
        </div>
      </div>
      <div class="row" >
        <div id="forgeViewer" >
        </div>
      </div>
      <div class="row" >
        <div id="dropdown2dviews">
          2D Viewer  <select id="list2dviews"></select>
        </div>
        <div id="viewerSecondary" >
        </div>
      </div>
      <div class="row" >
        <div id="positionCharts">
          <div class="tab-panel active report-dropdowns" id="tab_panel_1">
            <label for="pu_reportToRun" class="report-dropdowns">Report</label>
              <select id="pu_reportToRun" class="select-light" name="reportToRun"
disabled>
              <!-- Options filled in by function in Report_PieChart.js -->
              </select>
            <label for="pu_sortOrder" class="sort-dropdowns">Sort Order</label>
            <select id="pu_sortOrder" class="select-light" name="sortOrder"
disabled>
                <option value="value-asc">Value - Ascending</option>
                <option value="value-desc" selected="selected">Value -
Descending</option>
                <option value="label-asc">Label - Ascending</option>
                <option value="label-desc">Label - Descending</option>
                <option value="random">Random</option>
            </select>
            </div>
            <div class="svgcanvas">
              <div id="pieChart">
              </div>
              <div id="barChart"></div>
            </div>
        </div>
      </div>

</div>
```

```
<!-- Testing environment -->
<div class="modal fade" id="testinEnvWarning" tabindex="-1" role="dialog" aria-
labelledby="myModalLabel">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal" aria-
label="Cancel"><span
                aria-hidden="true">&times;</span></button>
        <h4 class="modal-title">Warning!</h4>
      </div>
      <div class="modal-body">
        <p>You are accessing a testing version of this app, which is not
recommended.</p>
        <p>Please <a id="testEnvRedirectPage">go to live version</a> and update
your bookmark.</p>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-
dismiss="modal">Continue anyway</button>
      </div>
    </div>
  </div>
</div>

<!-- Charts of the project-->
<div class="container-fluid">
<div class="row">

<button class="btn btn-lg btn-default" id="Charts">
  <img src="/img/ChartIcon.png" height="30">
</button>
<button class="btn btn-lg btn-default" id="Schedule">
  <img src="/img/schedule.png" height="30">
</button>
<a href="doc.html" target="_blank">
<button class="btn btn-lg btn-default"  id="Documents">
  <img src="/img/documents.jpg" height="30">
</button>
</a>
<a href="http://beamrobotics.github.io/" target="_blank">
<button class="btn btn-lg btn-default" id="Scans">
```

```
  <img src="/img/scans.png" height="30">
</button>
  </a>
  <a href="demo.html" target="_blank">
    <button class="btn btn-lg btn-default" id="Demos">
      <img src="/img/demo.png" height="30">
    </button>
  </a>
  <a href="live.html" target="_blank">
  <button class="btn btn-lg btn-default" id="Live">
    <img src="/img/live.png" height="30">
  </button>
</a>
</div>

<div class="row">
<div style="width:50%; float:right">
  <canvas id="myChart1" width="400" height="100"></canvas>
  </div>
  <div style="width:50%; float:left;">
    <canvas id="myChart2" width="400" height="100"></canvas>
    </div>
    <div style="width:400; height:400" id="ScheduleExcel">
      <center>
      <iframe  src="ScheduleTest.htm" width="800" height="500"></iframe>
      </center>
    </div>
    </div>
</div>
</body>
</html>
```

Oauth.js:

```
$(document).ready(function () {
    var currentToken = getForgeToken();

    if (window.location.href.indexOf('herokuapp') > 0) {
          $("a").attr("href", 'https://bim360reports.autodesk.io');
          $('#testinEnvWarning').modal('toggle');
    }

    if (currentToken === '')
      $('#signInButton').click(forgeSignIn);
    else {
```

```
      getForgeUserProfile(function (profile) {
        $('#signInProfileImage').removeClass(); // remove glyphicon-user
        $('#signInProfileImage').html('<img src="' + profile.picture + '"/>')
        $('#signInButtonText').text(profile.name);
        $('#signInButtonText').attr('title', 'Click to Sign Out');
        $('#signInButton').click(forgeLogoff);
      });
    }
  });

  function forgeSignIn() {
    jQuery.ajax({
      url: '/user/authenticate',
      success: function (rootUrl) {
        location.href = rootUrl;
      }
    });
  }

  function forgeLogoff() {
    jQuery.ajax({
      url: '/user/logoff',
      success: function (oauthUrl) {
        location.href = oauthUrl;
      }
    });
  }

  function getForgeToken() {
    var token = '';
    jQuery.ajax({
      url: '/user/token',
      success: function (res) {
        token = res;
      },
      async: false // this request must be synchronous for the Forge Viewer
    });
    if (token != '') console.log('3 legged token (User Authorization): ' +
token); // debug
    return token;
  }

  function getForgeUserProfile(onsuccess) {
    var profile = '';
    jQuery.ajax({
```

```
      url: '/user/profile',
      success: function (profile) {
        onsuccess(profile);
      }
    });
  }
```

- **Forge Viewer (following Autodesk® Forge Tutorial)**

Forge.viewer.js:

```
var viewer = {};
viewer['3d'] = null;
viewer['2d'] = null;
var flagModel = false;

function blankOutReportPane() {
  $("#pieChart").empty();
  $("#barChart").empty();
  $("#list2dviews").empty();
  $("#viewerSecondary").empty();
  $("#forgeViewer").empty();
}

function blankOutReportPane3d() {
  $("#pieChart").empty();
  $("#barChart").empty();
  $("#forgeViewer").empty();
  $(".report-dropdowns").css('visibility', 'hidden');
  $("#dropdown2dviews").css('visibility', 'hidden');
}

function blankOutReportPane2d() {
  $("#pieChart").empty();
  $("#barChart").empty();
  $("#list2dviews").empty();
  $("#viewerSecondary").empty();
  $(".report-dropdowns").css('visibility', 'hidden');
  $("#dropdown2dviews").css('visibility', 'hidden');
}

function launchViewer(urn, div3d, div2d) {
  blankOutReportPane()
  var options = {
    env: 'AutodeskProduction',
```

```
    getAccessToken: getForgeToken
  };
  var documentId = 'urn:' + urn;
  Autodesk.Viewing.Initializer(options, function onInitialized() {
    Autodesk.Viewing.Document.load(documentId, function(doc) {
      // clear both viewers
      showModel(doc, '3d', div3d);
      showModel(doc, '2d', div2d, function(viewables) {
        var options = $("#list2dviews");
        viewables.forEach(function(view) {
          options.append($("<option />").val(view.guid).text(view.name));
        });
        options.change(function() {
          // destroy and recreate the 2d view
          viewer['2d'].impl.unloadCurrentModel();
          viewer['2d'].tearDown();
          viewer['2d'].finish();
          var viewerDiv = document.getElementById(div2d);
          viewer['2d'] = new Autodesk.Viewing.Private.GuiViewer3D(viewerDiv);
          var selected = this.value;
          viewables.forEach(function(view) {
            if (view.guid === selected)
              showSvf(doc, view, '2d');
          })
        });
      });

    }, onDocumentLoadFailure);
  });
}

function showModel(doc, role, div, callback) {
  // A document contains references to 3D and 2D viewables.
  var viewables =
Autodesk.Viewing.Document.getSubItemsWithProperties(doc.getRootItem(), {
    'type': 'geometry',
    'role': role
  }, true);
  if (viewables.length === 0) {

    if (role === '3d'){
      blankOutReportPane3d()
      flagModel = true;
      $("#forgeViewer").append("<h2><em>There is no viewables available for 3D
models</em></h2>")
```

```
      $("#pieChart").append("<p><em>No data could be retrieved for charts.  This
report is probably not applicable for the given model.  As an example, Revit
models can be sorted by Type or Level, but Fusion models cannot.  Fusion models
are more appropriate for reports sorted by Mass, Volume, or Material.  Try
switching to a different report or a different model.</em></p>");

    } else {
      blankOutReportPane2d()
      flagModel = true;
      $("#viewerSecondary").append("<h2><em>There is no viewables available for
2D models</em></h2>")
      $("#pieChart").append("<p><em>No data could be retrieved for charts.  This
report is probably not applicable for the given model.  As an example, Revit
models can be sorted by Type or Level, but Fusion models cannot.  Fusion models
are more appropriate for reports sorted by Mass, Volume, or Material.  Try
switching to a different report or a different model.</em></p>");

    }
    //console.error('Document contains no viewables.');
    return;
  }

  flagModel = false;
  var viewerDiv = document.getElementById(div);
  viewer[role] = new Autodesk.Viewing.Private.GuiViewer3D(viewerDiv);

  showSvf(doc, viewables[0], role);

  if (callback) callback(viewables);
}

function showSvf(doc, viewable, role) {
  // Choose any of the avialble viewables
  var svfUrl = doc.getViewablePath(viewable);
  var modelOptions = {
    sharedPropertyDbPath: doc.getPropertyDbPath()
  };

  viewer[role].start(svfUrl, modelOptions, onLoadModelSuccess, onLoadModelError);
}

function onDocumentLoadFailure(viewerErrorCode) {}

var blockEvent = false;
```

```
function onLoadModelSuccess(model) {
  viewer[(model.is3d() ? '3d' :
'2d')].addEventListener(Autodesk.Viewing.SELECTION_CHANGED_EVENT,
function(selection) {
    if (blockEvent) return;
    if (selection.dbIdArray.length == 0) return;
    var role = (model.is3d() ? '2d' : '3d');
    blockEvent = true;
    viewer[role].select(selection.dbIdArray);
    viewer[role].fitToView(selection.dbIdArray);
    blockEvent = false;

  });

  // when the geometry is loaded, automatically run the first report

  if (model.is3d() && flagModel === false) {
    disableReportMenu();
    viewer['3d'].addEventListener(Autodesk.Viewing.GEOMETRY_LOADED_EVENT,
function(event) {
      enableReportMenu();
      //runReport(-1);   // run the currently selected report (the first one if
this is the first model loaded, current one if loading a subsequent model)
      $("#tab_button_1").click();
      startReportDataLoader(viewer['3d'], viewer['2d'], runReport);
    });

  }


}



function onLoadModelError(viewerErrorCode) {}

function getForgeToken() {
  jQuery.ajax({
    url: '/user/token',
    success: function(res) {
      token = res;
    },
    async: false
  });
  return token;
```

```
}
```

Forge.tree.js:

```
$(document).ready(function () {
    prepareAppBucketTree();
    $('#refreshBuckets').click(function () {
      $('#appBuckets').jstree(true).refresh();
    });

    $('#createNewBucket').click(function () {
      createNewBucket();
    });

    $('#createBucketModal').on('shown.bs.modal', function () {
      $("#newBucketKey").focus();
    })
  });

  function createNewBucket() {
    var bucketKey = $('#newBucketKey').val();
    var policyKey = $('#newBucketPolicyKey').val();
    jQuery.post({
      url: '/api/forge/oss/buckets',
      contentType: 'application/json',
      data: JSON.stringify({ 'bucketKey': bucketKey, 'policyKey': policyKey }),
      success: function (res) {
        $('#appBuckets').jstree(true).refresh();
        $('#createBucketModal').modal('toggle');
      },
      error: function (err) {
        if (err.status == 409)
          alert('Bucket already exists - 409: Duplicated')
        console.log(err);
      }
    });
  }

  function prepareAppBucketTree() {
    $('#appBuckets').jstree({
      'core': {
        'themes': { "icons": true },
        'data': {
          "url": '/api/forge/oss/buckets',
          "dataType": "json",
```

```
          'multiple': false,
          "data": function (node) {
            return { "id": node.id };
          }
        }
      },
      'types': {
        'default': {
          'icon': 'glyphicon glyphicon-question-sign'
        },
        '#': {
          'icon': 'glyphicon glyphicon-cloud'
        },
        'bucket': {
          'icon': 'glyphicon glyphicon-folder-open'
        },
        'object': {
          'icon': 'glyphicon glyphicon-file'
        }
      },
      "plugins": ["types", "state", "sort", "contextmenu"],
      contextmenu: { items: autodeskCustomMenu }
    }).on('loaded.jstree', function () {
      $('#appBuckets').jstree('open_all');
    }).bind("activate_node.jstree", function (evt, data) {
      if (data != null && data.node != null && data.node.type == 'object') {
        $("#forgeViewer").empty();
        var urn = data.node.id;
        getForgeToken(function (access_token) {
          jQuery.ajax({
            url:
'https://developer.api.autodesk.com/modelderivative/v2/designdata/' + urn +
'/manifest',
            headers: { 'Authorization': 'Bearer ' + access_token },
            success: function (res) {
              if (res.status === 'success') launchViewer(urn);
              else $("#forgeViewer").html('The translation job still running: ' +
res.progress + '. Please try again in a moment.');
            },
            error: function (err) {
              var msgButton = 'This file is not translated yet! ' +
                '<button class="btn btn-xs btn-info"
onclick="translateObject()"><span class="glyphicon glyphicon-eye-open"></span> '
+
                'Start translation</button>'
```

```
            $("#forgeViewer").html(msgButton);
          }
        });
      })
    }
  });
}

function autodeskCustomMenu(autodeskNode) {
  var items;

  switch (autodeskNode.type) {
    case "bucket":
      items = {
        uploadFile: {
          label: "Upload file",
          action: function () {
            var treeNode = $('#appBuckets').jstree(true).get_selected(true)[0];
            uploadFile(treeNode);
          },
          icon: 'glyphicon glyphicon-cloud-upload'
        }
      };
      break;
    case "object":
      items = {
        translateFile: {
          label: "Translate",
          action: function () {
            var treeNode = $('#appBuckets').jstree(true).get_selected(true)[0];
            translateObject(treeNode);
          },
          icon: 'glyphicon glyphicon-eye-open'
        }
      };
      break;
  }

  return items;
}

function uploadFile(node) {
  $('#hiddenUploadField').click();
  $('#hiddenUploadField').change(function () {
    if (this.files.length == 0) return;
```

```
      var file = this.files[0];
      switch (node.type) {
        case 'bucket':
          var formData = new FormData();
          formData.append('fileToUpload', file);
          formData.append('bucketKey', node.id);

          $.ajax({
            url: '/api/forge/oss/objects',
            data: formData,
            processData: false,
            contentType: false,
            type: 'POST',
            success: function (data) {
              $('#appBuckets').jstree(true).refresh_node(node);
            }
          });
          break;
      }
    });
  }

  function translateObject(node) {
    $("#forgeViewer").empty();
    if (node == null) node = $('#appBuckets').jstree(true).get_selected(true)[0];
    var bucketKey = node.parents[0];
    var objectKey = node.id;
    jQuery.post({
      url: '/api/forge/modelderivative/jobs',
      contentType: 'application/json',
      data: JSON.stringify({ 'bucketKey': bucketKey, 'objectName': objectKey }),
      success: function (res) {
        $("#forgeViewer").html('Translation started! Please try again in a
moment.');
      },
    });
  }
```

Forge.data.management.js:

```
$(document).ready(function () {
  $('#refreshAutodeskTree').hide();
  if (getForgeToken() != '') {
    prepareDataManagementTree();
    $('#refreshAutodeskTree').show();
```

```
    $('#refreshAutodeskTree').click(function(){
      $('#dataManagementHubs').jstree(true).refresh();
    });
  }

    $.getJSON("/api/forge/clientID", function (res) {
        $("#ClientID").val(res.ForgeClientId);
    });

    $("#provisionAccountSave").click(function () {
        $('#provisionAccountModal').modal('toggle');
        $('#dataManagementHubs').jstree(true).refresh();
    });

});

var haveBIM360Hub = false;

function prepareDataManagementTree() {
  $('#dataManagementHubs').jstree({
    'core': {
      'themes': {"icons": true},
      'data': {
        "url": '/dm/getTreeNode',
        "dataType": "json",
        "multiple": false,
        "cache": false,
        "data": function (node) {
          $('#dataManagementHubs').jstree(true).toggle_node(node);
          return {"id": node.id};
        },
        "success": function (nodes) {
          nodes.forEach(function (n) {
            if (n.type === 'bim360Hubs' && n.id.indexOf('b.') > 0)
              haveBIM360Hub = true;
          });
            if (!haveBIM360Hub) {
                $("#provisionAccountModal").modal();
                haveBIM360Hub = true;
            }
        }
      }
    },
    'types': {
      'default': {
```

```
        'icon': 'glyphicon glyphicon-question-sign'
      },
      '#': {
        'icon': 'glyphicon glyphicon-user'
      },
      'hubs': {
        'icon': '/img/a360hub.png'
      },
      'personalHub': {
        'icon': '/img/a360hub.png'
      },
      'bim360Hubs': {
        'icon': '/img/bim360hub.png'
      },
      'bim360projects': {
        'icon': '/img/bim360project.png'
      },
      'a360projects': {
        'icon': '/img/a360project.png'
      },
      'items': {
        'icon': 'glyphicon glyphicon-file'
      },
      'folders': {
        'icon': 'glyphicon glyphicon-folder-open'
      },
      'versions': {
        'icon': 'glyphicon glyphicon-time'
      }
    },
    "plugins":
      ["types", "state", "sort"]
  }).bind("activate_node.jstree", function (evt, data) {
    if (data != null && data.node != null && data.node.type == 'versions') {
      if (data.node.id === 'not_available') { alert('No viewable available for
this version'); return; }
      var parent_node =
$('#dataManagementHubs').jstree(true).get_node(data.node.parent);
      $(".report-dropdowns").css('visibility', 'visible');
      $("#dropdown2dviews").css('visibility', 'visible');
      launchViewer(data.node.id, 'forgeViewer', 'viewerSecondary');
      $.notify("loading... " + parent_node.text, { className: "info",
position:"bottom right" });

    }
```

```
  });
}
```

Forge.Viewer.js:

```
var viewerApp;

function launchViewer(urn) {
  var options = {
    env: 'AutodeskProduction',
  };
  var documentId = 'urn:' + urn;
  Autodesk.Viewing.Initializer(options, function onInitialized() {
    viewerApp = new Autodesk.Viewing.ViewingApplication('forgeViewer');
    viewerApp.registerViewer(viewerApp.k3D,
Autodesk.Viewing.Private.GuiViewer3D);
    viewerApp.loadDocument(documentId, onDocumentLoadSuccess,
onDocumentLoadFailure);
  });
}

function onDocumentLoadSuccess(doc) {
  // We could still make use of Document.getSubItemsWithProperties()
  // However, when using a ViewingApplication, we have access to the **bubble**
attribute,
  // which references the root node of a graph that wraps each object from the
Manifest JSON.
  var viewables = viewerApp.bubble.search({ 'type': 'geometry' });
  if (viewables.length === 0) {
    console.error('Document contains no viewables.');
    return;
  }

  // Choose any of the available viewables
  viewerApp.selectItem(viewables[0].data, onItemLoadSuccess, onItemLoadFail);
}

function onDocumentLoadFailure(viewerErrorCode) {
  console.error('onDocumentLoadFailure() - errorCode:' + viewerErrorCode);
}

function onItemLoadSuccess(viewer, item) {
  // item loaded, any custom action?
}
```

```
function onItemLoadFail(errorCode) {
  console.error('onItemLoadFail() - errorCode:' + errorCode);
}
```

- **Visual Reports (following Autodesk® Github Code)**

Report.data.js:

```
var _modelLeafNodes;
var _root;
var _viewerMain = null;
var _viewerSecondary = null;
var instanceTree;

    // Preloading steps
    // called by LoadModel.js to preload all the leaf nodes whenever a new model
is loaded,
    // get and keep all the leaf nodes of this model for future use.
function startReportDataLoader(viewer3d, viewer2d, callback) {
    _modelLeafNodes = [];

    _viewerMain = viewer3d;
    _viewerSecondary = viewer2d;

    instanceTree = _viewerMain.model.getData().instanceTree;
    var modelRoot = instanceTree.getRootId();

    getModelLeafNodes(modelRoot, _modelLeafNodes);

        if (callback)
            callback();

}

    // recursively add all the leaf nodes
function getModelLeafNodes(rootId, leafNodes, callback) {

    instanceTree.enumNodeChildren(rootId, function(childId){
        leafNodes.push(childId)
    }, true);
}


//*************************************************************************
*********
```

```
//
//      Group data according to node model types
//
//*************************************************************************
*********


    // group the children of a treeNode according to their types
function groupDataByType(treeNode) {

    var subTypes = {};

    if (!treeNode) {
      treeNode = {};
      treeNode.id = _viewerMain.model.getData().instanceTree.getRootId();
    }
    console.log(treeNode);

    treeNode.children = [];

    // add all the types into subTypes
    instanceTree.enumNodeChildren(treeNode.id, function (childId){
      treeNode.children.push(childId);
    });

        // if the treeNode contains only one child, dig deeper to see if there're
more branches
    while (treeNode.children && treeNode.children.length === 1) {
        treeNode = treeNode.children[0];
    }

    $.each(treeNode.children, function(i, childNode) {
        var leafNodes = [];
        getModelLeafNodes(childNode, leafNodes);
        subTypes[instanceTree.getNodeName(childNode)] = leafNodes;
    });

    return subTypes;

}


//*************************************************************************
*********
//
```

```
//      Group data according to specific property value
//
//*************************************************************************
*********


    // group the data by a property string, two things are returned
    // a buckets object that uses the property value as the key and dbIds as the
value
    // and a misCount array which contains all the nodes that doesn't have such
property
function groupDataByProperty(propertyName, callback) {
    var buckets = {};
    var misCount = [];

        // iterate through the preloaded leafNodes to check their properties
    $.each(_modelLeafNodes, function(index, dbId) {
        _viewerMain.getProperties(dbId, function(propObj) {
            for (var i = 0; i < propObj.properties.length; i++) {
                if (propObj.properties[i].displayName === propertyName &&
(!propObj.properties[i].hidden)) {
                    var formatVal =
Autodesk.Viewing.Private.formatValueWithUnits(propObj.properties[i].displayValue,
propObj.properties[i].units, propObj.properties[i].type);
                    if (!(formatVal in buckets)) {
                        buckets[formatVal] = [];
                    }
                    buckets[formatVal].push(dbId);
                    break;
                } else if (i == propObj.properties.length - 1) {
                    misCount.push(dbId);
                }
            };

            if (index === _modelLeafNodes.length - 1) {
                if (callback) {
                    callback(buckets, misCount);
                }
            }
        });
    });
}
```

```
//**************************************************************************
*********
//
//      Group quantity data according to a certain range
//
//**************************************************************************
*********


    // first get all the quantity data and their dbIds by propertyName, and put
them in an array
    // in the meantime, find the min and max value to prepare for range grouping
in next step
function getQtyDataByProperty(propertyName, callback) {
    var qtyArr = [];
    var bound = {"min":-0.1, "max":-0.1};
    var misCount = [];

    // console.time("getQtyByProperty");
    $.each(_modelLeafNodes, function(index, dbId) {
        _viewerMain.getProperties(dbId, function(propObj) {

            for(var i = 0; i < propObj.properties.length; i++) {
                if (propObj.properties[i].displayName === propertyName) {
                    var propValue =
parseFloat(propObj.properties[i].displayValue);
                    if (propValue < bound.min || bound.min < 0)
                        bound.min = propValue;
                    if (propValue > bound.max || bound.max < 0)
                        bound.max = propValue;
                    var formatVal =
Autodesk.Viewing.Private.formatValueWithUnits(propObj.properties[i].displayValue,
propObj.properties[i].units, propObj.properties[i].type);
                    qtyArr.push({"dbId":dbId, "val":propValue, "label":formatVal,
"units":propObj.properties[i].units});
                    break;
                } else if (i == propObj.properties.length - 1) {
                    misCount.push(dbId);
                }
            }

            if ( index === _modelLeafNodes.length - 1) {
                // console.timeEnd("getQtyByProperty");
                if (callback) {
                    callback(qtyArr, misCount, bound);
```

```
                }
            }
        });
    });
}


    // group the quantity array by range, returned is a buckets object that uses
the range label
    // as key and the dbIds as value, range is  calculated based on the min value
and the range value,
    // the array does not has to be sorted
function groupQtyDataByRange(qtyArr, bound, range, callback) {
    var buckets = {};
    var min = bound.min;
    range = parseFloat(range);

    for (var i = qtyArr.length - 1; i >= 0; i--) {
        var key;
        if (range === 0) {
            key = qtyArr[i].label;
        } else {
                // calculate the range label according to min and range
            var floor = min + range * Math.floor((qtyArr[i].val-min)/range);
            var ceil = floor + range;
            key = Math.round(floor*1000)/1000 + " - " +
Math.round(ceil*1000)/1000 + " " + qtyArr[i].units;
        }
            // check if this range label is already in the buckets
        if (!buckets.hasOwnProperty(key))
            buckets[key] = [];

        buckets[key].push(qtyArr[i].dbId);
    };

    if (callback)
        callback(buckets);
}
```

pieChart.js:

```
var _pieChart = null;
var _sortOrder = "value-desc";

var _reportOptions = [
```

```
    { label : "Qty - Type",             fieldName: "",                  fieldType
: "ModelType"},
    { label : "Qty - Level",            fieldName: "Level",             fieldType
: "Properties"},
    { label : "Qty - Base Constraint",  fieldName: "Base Constraint",   fieldType
: "Properties"},
    { label : "Qty - System Type",      fieldName: "System Type",       fieldType
: "Properties"},
    { label : "Qty - Assembly Code",    fieldName: "Assembly Code",     fieldType
: "Properties"},
    { label : "Qty - Material",         fieldName: "Material",          fieldType
: "Properties"}
];

    // populate the popup menu with the avaialable models to load (from the array
above)
function loadReportMenuOptions() {
        // add the new options for models
    var sel = $("#pu_reportToRun");
    $.each(_reportOptions, function(i, item) {
        sel.append($("<option>", {
            value: i,
            text : item.label
        }));
    });
}

function enableReportMenu() {
    $('#pu_reportToRun').attr("disabled", false);
    $('#pu_sortOrder').attr("disabled", false);
}

function disableReportMenu() {
    $('#pu_reportToRun').attr("disabled", true);
    $('#pu_sortOrder').attr("disabled", true);
}

function runReport(index) {

        // if they pass in a negative index, look up the current one
    if (typeof (index) === "undefined" || index === -1)
        index = parseInt($("#pu_reportToRun option:selected").val(), 10);

    var reportObj = _reportOptions[index];
    console.log("Running report: " + reportObj.label);
```

```
    $("#reportinput").empty();
    _currentQty = null;
    _currentBound = null;

    if (reportObj.fieldName === "") {
        var modelTypes = groupDataByType();
        wrapDataForPieChart(modelTypes);
    }
    else if (reportObj.fieldType === "Quantity") {

        getQtyDataByProperty(reportObj.fieldName, function(Qty, misCount, bound){
            var initrange = 100;

            createReportUserInput(bound, initrange);
            _currentQty = Qty;
            _currentBound = bound;

            groupQtyDataByRange(Qty, bound, initrange, wrapDataForPieChart);
        });
    }
    else {
        groupDataByProperty(reportObj.fieldName, wrapDataForPieChart);
     }
}

var _currentQty = null;
var _currentBound = null;

    // Create user input div for quantity type
function createReportUserInput(bound, initVal) {

    var slider = document.createElement("input");
    slider.id = "qtyslider";
    slider.type = "range";
    slider.style.height = "12px";
    slider.min = 0;
    slider.max = Math.round(bound.max - bound.min);
    slider.value = initVal;
    slider.onchange = function() {
        $("#qtyfield").val(slider.value);
        groupQtyDataByRange(_currentQty, _currentBound, slider.value,
wrapDataForPieChart);
    };
```

```
    var preLabel = document.createElement("label");
    preLabel.htmlFor = slider.id;
    preLabel.innerHTML = slider.min;
    preLabel.style.marginRight = "10px";
    var postLabel = document.createElement("label");
    postLabel.htmlFor = slider.id;
    postLabel.innerHTML = slider.max;
    postLabel.style.marginLeft = "10px";

    var textField = document.createElement("input");
    textField.id = "qtyfield";
    textField.type = "text";
    textField.style.width = "40px";
    textField.placeholder = slider.value;
    textField.onkeydown = function(e) {
        if (e.keyCode == 13) {
            var inputVal = parseFloat(this.value);
            if (inputVal <= parseFloat(slider.max) && inputVal >= 0) {
                $("#qtyslider").val(this.value);
                groupQtyDataByRange(_currentQty, _currentBound, this.value,
wrapDataForPieChart);
            }
        }
    };

    var fieldLabel = document.createElement("label");
    fieldLabel.htmlFor = textField.id;
    fieldLabel.innerHTML = "Range: "
    fieldLabel.style.marginLeft = "25px";

    var inputDiv = document.getElementById("reportinput");
    inputDiv.style.margin = "20px";
    inputDiv.appendChild(preLabel);
    inputDiv.appendChild(slider);
    inputDiv.appendChild(postLabel);
    inputDiv.appendChild(fieldLabel);
    inputDiv.appendChild(textField);
}

function wrapDataForPieChart(buckets, misCount) {
    var reportIdx = parseInt($("#pu_reportToRun").val());
    var fieldName = (_reportOptions[reportIdx].fieldName === "") ? "Object Type"
: _reportOptions[reportIdx].fieldName;
    var pieOpts = initPieOpts(fieldName, reportIdx);
```

```
    for (var valueKey in buckets) {
        var pieObject = {};
        pieObject.label = valueKey;
        pieObject.value = buckets[valueKey].length;
        pieObject.lmvIds = buckets[valueKey];
        pieOpts.data.content.push(pieObject);
    }

    loadReportDataPieChart(pieOpts);
}

$(document).ready(function() {

    console.log("Document Ready: excuting func in pieChart.js");

    loadReportMenuOptions();

        // user selected a new model to load
    $("#pu_reportToRun").change(function(evt) {

        // Only calls when user selection changes

        evt.preventDefault(); // The event.preventDefault() method stops the
default action of an element from happening

        var index = parseInt($("#pu_reportToRun option:selected").val(), 10);

        runReport(index);
    });

    $("#pu_sortOrder").change(function(evt) {
        evt.preventDefault();

        _sortOrder = $("#pu_sortOrder option:selected").val();
            // rebuild the report with the new sort order
        if (_pieChart)
            runReport(_pieChart.options.reportIndex);   // re-run same report
    });

});

    // callback function that fills the pieChart up with the data retrieved from
LMV Object Properties
function loadReportDataPieChart(pieOpts) {
        // free up anything that is already there
```

```
    if (_pieChart)
        _pieChart.destroy();

    $("#barChart").empty();

    if (pieOpts.data.content.length === 0) {
        $("#pieChart").append("<p><em>No data could be retrieved for charts.
This report is probably not applicable for the given model.  As an example, Revit
models can be sorted by Type or Level, but Fusion models cannot.  Fusion models
are more appropriate for reports sorted by Mass, Volume, or Material.  Try
switching to a different report or a different model.</em></p>");
    }
    else {
        // if we have a lot of buckets, don't let the pie chart get out of
control, condense anything with 2 or less
        // into an "Other" wedge.

        //pieOpts.data.sortOrder = "value-desc";
        pieOpts.data.content.sort(function (a, b) {
            if (a.value < b.value) return 1;
            else if (a.value > b.value) return -1;
            return 0;
        });

        if (pieOpts.data.content.length < 10) {
            pieOpts.data.smallSegmentGrouping.enabled = false;
        } else if (pieOpts.data.content.length > 20) {
            pieOpts.labels.truncation.enabled = true;
            var thresholdObj = pieOpts.data.content[19];
            pieOpts.data.smallSegmentGrouping.value = thresholdObj.value;
        }

        _pieChart = new d3pie("pieChart", pieOpts);
         console.log('my pie chart', _pieChart)

         // Responsiveness for PIECHART Prototype

         var width = $("#pieChart").width();
         var height = $("#pieChart").height();

         _pieChart.svg.attr('viewBox','0 0 '+width+' '+height)

        loadBarChart(pieOpts.data);
    }
}
```

```
    // initialize
function initPieOpts(fieldName, reportIndex) {
    var pieOpts = initPieDefaults(fieldName);
    pieOpts.reportIndex = reportIndex;

    pieOpts.data = {
        "sortOrder": _sortOrder,
        "content": [],
        "smallSegmentGrouping": {
            "enabled": true,
            "value": 1,
            "valueType": "value"   // percentage or value
        },
    };

    return pieOpts;
}

function initPieDefaults(fieldName) {
    var strSubTitle = "Quantities in model (" + fieldName + ")";

    var pieDefaults = {
        "header": {
            "title": {
                "text": fieldName,
                "fontSize": 34,
                "font": "courier"
            },
            "subtitle": {
                "text": strSubTitle,
                "color": "#999999",
                "fontSize": 10,
                "font": "courier"
            },
            "titleSubtitlePadding": 9
        },
        "footer": {
            "color": "#999999",
            "fontSize": 10,
            "font": "open sans",
            "location": "bottom-left"
        },
        "size": {
            "canvasWidth": 590,
```

```
        "pieInnerRadius": "43%",
        "pieOuterRadius": "71%"
    },
    "labels": {
        "outer": {
            "pieDistance": 32
        },
        "inner": {
            "hideWhenLessThanPercentage": 3
        },
        "mainLabel": {
            "fontSize": 11
        },
        "percentage": {
            "color": "#ffffff",
            "decimalPlaces": 0
        },
        "value": {
            "color": "#adadad",
            "fontSize": 11
        },
        "lines": {
            "enabled": true
        },
        "truncation": {
            "enabled": false,
            "truncateLength": 30
        }
    },
    "effects": {
        "pullOutSegmentOnClick": {
            "effect": "linear",
            "speed": 400,
            "size": 8
        }
    },
    "misc": {
        "gradient": {
            "enabled": true,
            "percentage": 100
        }
    },
    "callbacks": {
        onClickSegment: clickPieWedge
    }
```

```
    };
    return pieDefaults;
}


var _selectedWedge;

function clickPieWedge(evt) {

    if (_selectedWedge !== evt.data.label) {
        ids = [];
        if (evt.data.isGrouped === true) {  // "Other" bucket will group things
together
            for (i=0; i<evt.data.groupedData.length; i++)
                ids = ids.concat(evt.data.groupedData[i].lmvIds);
        }
        else {
            ids = evt.data.lmvIds;
        }

        _viewerMain.isolate(ids);
        _viewerSecondary.select(ids);
        _selectedWedge = evt.data.label;
    } else {
        _selectedWedge = null;

        _viewerMain.showAll();
        _viewerSecondary.clearSelection();
    }

}
```

- **Project Charts**

projectChart.js:

```
$(document).ready(function(){
    $("#Charts").on("click",function(){

        var MyChart = document.getElementById("myChart1").getContext('2d');
        var MyBarChart = new Chart(MyChart, {
            type:'bar',
            options:{
              title:{
                display:true,
```

```
            "text":"Bar Chart"
            }
          },
          data:{
            labels:['Steel','Concrete','Undefined','Contaminated Concrete'],
            datasets:[{
              label:'Quantity in km3',
              data:[
                23,
                48,
                56,
                39,
              ],
              backgroundColor:[
                'rgba(0, 0, 128, 0.6)',
                'rgba(255, 99, 132, 0.6)',
                'rgba(54, 162, 235, 0.6)',
                'rgba(255, 159, 64, 0.6)',
                ]
            },
          {

            label:'Area in km2',
            data:[
              1,
              12,
              24,
              8.6,
            ],
            backgroundColor:[
                'rgba(0, 0, 128, 0.6)',
                'rgba(255, 99, 132, 0.6)',
                'rgba(54, 162, 235, 0.6)',
                'rgba(255, 159, 64, 0.6)',
                ]},
              ]}
      });

      var MyChart = document.getElementById("myChart2").getContext('2d');
var MyPieChart = new Chart(MyChart, {
          type:'pie',
          options:{
            title:{
              display:true,
            "text":"Pie Chart"
```

```
            },
            layout:{
              left:400,

            }
          },
          data:{
            labels:['Steel','Concrete','Undefined','Contaminated Concrete'],
            datasets:[{
              label:'Material',
              data:[
                23,
                48,
                56,
                39,
              ],
              backgroundColor:[
                'rgba(0, 0, 128, 0.6)',
                'rgba(255, 99, 132, 0.6)',
                'rgba(54, 162, 235, 0.6)',
                'rgba(255, 159, 64, 0.6)',
                ]
            }]
          }
                      });

    })
})
```

- **Schedules**

Schedules.js:

```
$(document).ready(function(){
         $("#Schedule").on("click",function(){
             if(document.getElementById("ScheduleExcel").style.display=="none") {
document.getElementById("ScheduleExcel").style.display="block";
 }
else
{
    document.getElementById("ScheduleExcel").style.display="none";
}
    });
```

```
}
,)
```

- **Local Documents**

doc.html:

```
<link href="//netdna.bootstrapcdn.com/bootstrap/3.0.0/css/bootstrap.min.css"
rel="stylesheet" id="bootstrap-css">
<script
src="//netdna.bootstrapcdn.com/bootstrap/3.0.0/js/bootstrap.min.js"></script>
<script src="//code.jquery.com/jquery-1.11.1.min.js"></script>
<!------ Include the above in your HEAD tag ---------->


<!DOCTYPE html><html lang='en' class=''>
<head><script src='//production-
assets.codepen.io/assets/editor/live/console_runner-
079c09a0e3b9ff743e39ee2d5637b9216b3545af0de366d4b9aad9dc87e26bfd.js'></script><sc
ript src='//production-assets.codepen.io/assets/editor/live/events_runner-
73716630c22bbc8cff4bd0f07b135f00a0bdc5d14629260c3ec49e5606f98fdd.js'></script><sc
ript src='//production-assets.codepen.io/assets/editor/live/css_live_reload_init-
2c0dc5167d60a5af3ee189d570b1835129687ea2a61bee3513dee3a50c115a77.js'></script><me
ta charset='UTF-8'><meta name="robots" content="noindex"><link rel="shortcut
icon" type="image/x-icon" href="//production-
assets.codepen.io/assets/favicon/favicon-
8ea04875e70c4b0bb41da869e81236e54394d63638a1ef12fa558a4a835f1164.ico" /><link
rel="mask-icon" type="" href="//production-assets.codepen.io/assets/favicon/logo-
pin-f2d2b6d2c61838f7e76325261b7195c27224080bc099486ddd6dccb469b8e8e6.svg"
color="#111" /><link rel="canonical"
href="https://codepen.io/vaidyasr/pen/waboEY?depth=everything&order=popularity&pa
ge=36&q=editable&show_forks=false" />


<style class="cp-pen-styles">undefined</style></head><body>
<script src="//cdn.webix.com/edge/webix.js" type="text/javascript" charset="utf-
8"></script>
<link rel="stylesheet" type="text/css" href="//cdn.webix.com/edge/webix.css">
<link rel="stylesheet" type="text/css"
href="https://docs.webix.com/codebase/filemanager/filemanager.css">
<style>
    .my_style{
      background-color:#f5f5f5;
    }
  </style>
```

```
<script src='//production-assets.codepen.io/assets/common/stopExecutionOnTimeout-b2a7b3fe212eaa732349046d8416e00a9dec26eb7fd347590fbced3ab38af52e.js'></script>
<script >webix.type(webix.ui.tree, {
  name: "FileTree",
  css: "webix_fmanager_tree",
  folder: function(t) {
    return t.$count && t.open ? "<div class='webix_icon icon fa-folder-open'></div>" : "<div class='webix_icon icon fa-folder'></div>"
  }
}), webix.type(webix.ui.dataview, {
  name: "FileView",
  css: "webix_fmanager_files",
  height: 110,
  margin: 10,
  width: 150,
  template: function(t, e) {
    var i = t.type || "file";
    i = e.icons[i] || e.icons.file;
    var s = "webix_fmanager_data_icon",
      n = e.templateName(t, e);
    return "<div class='webix_fmanager_file'><div class='" + s + "'>" + e.templateIcon(t, e) + "</div>" + n + "</div>"
  }
}), webix.i18n.filemanager = {
  name: "Name",
  size: "Size",
  type: "Type",
  date: "Date",
  copy: "Copy",
  cut: "Cut",
  paste: "Paste",
  upload: "Upload",
  remove: "Delete",
  create: "Create Folder",
  rename: "Rename",
  location: "Location",
  select: "Select Files",
  sizeLabels: ["B", "KB", "MB", "GB"],
  saving: "Saving...",
  errorResponse: "Error: changes were not saved!",
  replaceConfirmation: "The folder already contains files with such names. Would you like to replace existing files ?",
  createConfirmation: "The folder with such a name already exists. Would you like to replace it ?",
```

```
  renameConfirmation: "The file with such a name already exists. Would you like
to replace it ?",
  yes: "Yes",
  no: "No",
  types: {
    folder: "Folder",
    doc: "Document",
    excel: "Excel",
    pdf: "PDF",
    pp: "PowerPoint",
    text: "Text File",
    video: "Video File",
    image: "Image",
    code: "Code",
    audio: "Audio",
    archive: "Archive",
    file: "File"
  }
}, webix.protoUI({
  name: "filetree"
}, webix.EditAbility, webix.ui.tree), webix.protoUI({
  name: "fileview"
}, webix.EditAbility, webix.ui.dataview), webix.protoUI({
  name: "filetable",
  $dragHTML: function(t) {
    var e = "<div class='webix_dd_drag webix_fmanager_drag' >",
      i = this.getColumnIndex("value");
    return e += "<div style='width:auto'>" + this.config.columns[i].template(t,
this.type) + "</div>", e + "</div>"
  }
}, webix.ui.datatable), webix.protoUI({
  name: "path",
  defaults: {
    layout: "x",
    separator: ",",
    scroll: !1
  },
  $skin: function() {
    this.type.height = webix.skin.$active.buttonHeight ||
webix.skin.$active.inputHeight
  },
  $init: function() {
    this.$view.className += " webix_path"
  },
  value_setter: function(t) {
```

```
    return this.setValue(), t
  },
  setValue: function(t) {
    this.clearAll(), t && ("string" == typeof t && (t =
t.split(this.config.separator)), this.parse(webix.copy(t)))
  },
  getValue: function() {
    return this.serialize()
  }
}, webix.ui.list), webix.FileManagerStructure = {
  structure: {
    actions: {
      config: function() {
        var t = this.config.templateName;
        return {
          view: "contextmenu",
          width: 200,
          padding: 0,
          autofocus: !1,
          css: "webix_fmanager_actions",
          template: function(e, i) {
            var s = t(e, i);
            return "<span class='webix_icon fa-" + e.icon + "'></span>" + s
          },
          data: "actionsData"
        }
      },
      oninit: function() {
        var t = this.getMenu();
        t.$q = !1, t && (this.getMenu().attachEvent("onItemClick",
webix.bind(function(e, i) {
          var s = this.getMenu().getItem(e),
            n = this[s.method] || this[e];
          if (n) {
            var a = this.getActive();
            if (this.callEvent("onbefore" + (s.method || e), [a])) {
              ("upload" != e || !webix.isUndefined(XMLHttpRequest) &&
!webix.isUndefined((new XMLHttpRequest).upload)) && (t.Uq(!0), t.hide());
              var r = [a];
              "upload" == e && (i = webix.html.pos(i), r.push(i)),
webix.delay(function() {
                n.apply(this, r), this.callEvent("onafter" + (s.method || e), [])
              }, this)
            }
          }
```

```
      }, this)), this.getMenu().attachEvent("onBeforeShow", function(t) {
        var e = this.getContext();
        return e && e.obj ? e.obj.callEvent("onBeforeMenuShow", [e.id, t]) : !0
      }))
    }
  },
  actionsData: {
    config: function() {
      return [{
        id: "copy",
        method: "markCopy",
        icon: "copy",
        value: webix.i18n.filemanager.copy
      }, {
        id: "cut",
        method: "markCut",
        icon: "cut",
        value: webix.i18n.filemanager.cut
      }, {
        id: "paste",
        method: "pasteFile",
        icon: "paste",
        value: webix.i18n.filemanager.paste
      }, {
        $template: "Separator"
      }, {
        id: "create",
        method: "createFolder",
        icon: "folder-o",
        value: webix.i18n.filemanager.create
      }, {
        id: "remove",
        method: "deleteFile",
        icon: "times",
        value: webix.i18n.filemanager.remove
      }, {
        id: "edit",
        method: "editFile",
        icon: "edit",
        value: webix.i18n.filemanager.rename
      }, {
        id: "upload",
        method: "uploadFile",
        icon: "upload",
        value: webix.i18n.filemanager.upload
```

```
        }]
      }
    },
    mainLayout: {
      type: "clean",
      rows: "mainRows"
    },
    mainRows: ["toolbar", "bodyLayout"],
    toolbar: {
      css: "webix_fmanager_toolbar",
      paddingX: 10,
      paddingY: 5,
      margin: 7,
      cols: "toolbarElements"
    },
    toolbarElements: ["menu", {
      id: "menuSpacer",
      width: 65
    }, {
      margin: 2,
      cols: ["back", "forward"]
    }, "up", "path", "search", "modes"],
    menu: {
      config: {
        view: "button",
        type: "iconButton",
        css: "webix_fmanager_back",
        icon: "bars",
        width: 37
      },
      oninit: function() {
        this.$$("menu") && (this.$$("menu").attachEvent("onItemClick",
webix.bind(function() {
          this.callEvent("onBeforeMenu", []) && (this.getMenu().nh = null,
this.getMenu().show(this.$$("menu").$view), this.callEvent("onAfterMenu", []))
        }, this)), this.config.readonly && (this.$$("menu").hide(),
this.$$("menuSpacer") && this.$$("menuSpacer").hide()))
      }
    },
    back: {
      config: {
        view: "button",
        type: "iconButton",
        css: "webix_fmanager_back",
        icon: "angle-left",
```

```
        width: 37
      },
      oninit: function() {
        this.$$("back") && this.$$("back").attachEvent("onItemClick",
webix.bind(function() {
          this.callEvent("onBeforeBack", []) && (this.goBack(),
this.callEvent("onAfterBack", []))
        }, this))
      }
    },
    forward: {
      config: {
        view: "button",
        type: "iconButton",
        css: "webix_fmanager_forward",
        icon: "angle-right",
        width: 37
      },
      oninit: function() {
        this.$$("forward") && this.$$("forward").attachEvent("onItemClick",
webix.bind(function() {
          this.callEvent("onBeforeForward", []) && (this.goForward(),
this.callEvent("onAfterForward", []))
        }, this))
      }
    },
    up: {
      config: {
        view: "button",
        type: "iconButton",
        css: "webix_fmanager_up",
        icon: "level-up",
        disable: !0,
        width: 37
      },
      oninit: function() {
        this.$$("up") && this.$$("up").attachEvent("onItemClick",
webix.bind(function() {
          this.callEvent("onBeforeLevelUp", []) && (this.levelUp(),
this.callEvent("onAfterLevelUp", []))
        }, this))
      }
    },
    path: {
      config: {
```

```
        view: "path",
        borderless: !0
      },
      oninit: function() {
        this.$$("path") && (this.attachEvent("onFolderSelect",
webix.bind(function(t) {
          this.$$("path").setValue(this.getPathNames(t))
        }, this)), this.$$("path").attachEvent("onItemClick",
webix.bind(function(t) {
          var e = this.$$("path").getIndexById(t),
            i = this.$$("path").count() - e - 1;
          if (this.$searchResults && this.hideSearchResults(), i) {
            for (t = this.getCursor(); i;) {if
(window.CP.shouldStopExecution(1)){break;}if
(window.CP.shouldStopExecution(1)){break;}t = this.getParentId(t), i--;
window.CP.exitedLoop(1);
}
window.CP.exitedLoop(1);

            this.setCursor(t)
          }
          this.callEvent("onAfterPathClick", [t])
        }, this)), this.data.attachEvent("onClearAll", webix.bind(function() {
          this.clearAll()
        }, this.$$("path"))))
      }
    },
    search: {
      config: {
        view: "search",
        gravity: .3,
        css: "webix_fmanager_search"
      },
      oninit: function() {
        var t = this.$$("search");
        t && (t.attachEvent("onTimedKeyPress", webix.bind(function() {
          if (9 != this.cx) {
            var e = t.getValue();
            e ? this.callEvent("onBeforeSearch", [e]) &&
(this.showSearchResults(e), this.callEvent("onAfterSearch", [e])) :
this.$searchResults && this.hideSearchResults()
          }
        }, this)), t.attachEvent("onKeyPress", function(t) {
          this.cx = t
        }), this.attachEvent("onAfterModeChange", function() {
```

```
        this.$searchResults && this.showSearchResults(t.getValue())
      }))
    }
  },
  bodyLayout: {
    css: "webix_fmanager_body",
    cols: "bodyCols"
  },
  bodyCols: ["tree", {
    view: "resizer",
    width: 2
  }, "modeViews"],
  tree: {
    config: {
      width: 251,
      view: "filetree",
      id: "tree",
      select: !0,
      filterMode: {
        showSubItems: !1,
        openParents: !1
      },
      type: "FileTree",
      navigation: !0,
      scroll: !0,
      editor: "text",
      editable: !0,
      editaction: !1,
      drag: !0,
      tabFocus: !0,
      onContext: {}
    },
    oninit: function() {
      var t = this.$$("tree");
      if (t) {
        t.type.icons = this.config.icons, t.sync(this, function() {
          this.filter(function(t) {
            return t.$count || "folder" == t.type
          })
        }), t.attachEvent("onAfterSelect", webix.bind(function(t) {
          this.callEvent("onFolderSelect", [t])
        }, this)), this.attachEvent("onAfterCursorChange", function(e) {
          e && (t.select(e), t.open(this.getParentId(e)))
        }), t.attachEvent("onItemClick", webix.bind(function() {
          this.$searchResults && this.hideSearchResults()
```

```
          }, this)), t.attachEvent("onItemDblClick", function(t) {
            this.isBranchOpen(t) ? this.close(t) : this.open(t)
          }), t.attachEvent("onBlur", function() {
            var t = this.getTopParentView();
            t.getMenu() && t.getMenu().isVisible() ||
webix.html.addCss(this.$view, "webix_blur")
          }), t.attachEvent("onFocus", webix.bind(function() {
            this.dx = t, webix.html.removeCss(t.$view, "webix_blur"),
this.$$(this.config.mode).unselect()
          }, this)), this.attachEvent("onPathLevel", function(e) {
            t.open(e)
          }), this.attachEvent("onPathComplete", function(e) {
            t.showItem(e)
          }), this.config.readonly || (this.getMenu() &&
this.getMenu().attachTo(t), t.attachEvent("onBeforeMenuShow", function(t) {
            this.select(t), webix.UIManager.setFocus(this)
          })), t.attachEvent("onBeforeEditStop", webix.bind(function(e, i) {
            return this.callEvent("onBeforeEditStop", [i.id, e, i, t])
          }, this)), t.attachEvent("onAfterEditStop", webix.bind(function(e, i) {
            this.callEvent("onAfterEditStop", [i.id, e, i, t]) &&
this.renameFile(i.id, e.value)
          }, this)), t.attachEvent("onBeforeDrag", function(t, e) {
            var i = this.getTopParentView();
            return !i.config.readonly && i.callEvent("onBeforeDrag", [t, e])
          }), t.attachEvent("onBeforeDragIn", function(t, e) {
            var i = this.getTopParentView();
            return !i.config.readonly && i.callEvent("onBeforeDragIn", [t, e])
          }), t.attachEvent("onBeforeDrop", function(t, e) {
            var i = this.getTopParentView();
            return i.callEvent("onBeforeDrop", [t, e]) && t.from &&
(i.moveFile(t.source, t.target), i.callEvent("onAfterDrop", [t, e])), !1
          });
          var e = function() {
            t && webix.UIManager.setFocus(t)
          };
          this.attachEvent("onAfterBack", e), this.attachEvent("onAfterForward",
e), this.attachEvent("onAfterLevelUp", e), this.attachEvent("onAfterPathClick",
e), this.config.readonly && (t.define("drag", !1), t.define("editable", !1))
        }
      }
    },
    modeViews: {
      config: function(t) {
        var e = [];
        if (t.modes)
```

```
          for (var i = 0; i < t.modes.length; i++) {if
(window.CP.shouldStopExecution(2)){break;}if
(window.CP.shouldStopExecution(2)){break;}e.push(t.modes[i]);}
window.CP.exitedLoop(2);

window.CP.exitedLoop(2);

        return {
          animate: !1,
          cells: e
        }
      },
      oninit: function() {
        this.$$(this.config.mode) && this.$$(this.config.mode).show(),
this.attachEvent("onBeforeCursorChange", function() {
          return this.$$(this.config.mode).unselect(), !0
        });
        var t = this.config.modes;
        if (t)
          for (var e = 0; e < t.length; e++) {if
(window.CP.shouldStopExecution(3)){break;}if
(window.CP.shouldStopExecution(3)){break;}this.$$(t[e]) && this.$$(t[e]).filter
&& this.ex(t[e])}
window.CP.exitedLoop(3);

window.CP.exitedLoop(3);

      }
    },
    modes: {
      config: function(t) {
        var e = 0,
          i = this.structure.modeOptions;
        if (i)
          for (var s = 0; s < i.length; s++) {if
(window.CP.shouldStopExecution(4)){break;}if
(window.CP.shouldStopExecution(4)){break;}i[s].width && (e += i[s].width +
(i.length ? 1 : 0));}
window.CP.exitedLoop(4);

window.CP.exitedLoop(4);

        var n = {
          view: "segmented",
          options: "modeOptions",
```

```
          css: "webix_fmanager_modes",
          value: t.mode
        };
        return e && (n.width = e + 4), n
      },
      oninit: function() {
        this.$$("modes") && this.$$("modes").attachEvent("onBeforeTabClick",
webix.bind(function(t) {
          var e = this.$$("modes").getValue();
          return this.callEvent("onBeforeModeChange", [e, t]) && this.$$(t) ?
(this.config.mode = t, this.$$(t).show(), this.callEvent("onAfterModeChange", [e,
t]), !0) : !1
        }, this))
      }
    },
    modeOptions: [{
      id: "files",
      width: 32,
      value: '<span class="webix_fmanager_mode_option webix_icon fa-th"></span>'
    }, {
      id: "table",
      width: 32,
      value: '<span class="webix_fmanager_mode_option webix_icon fa-list-
ul"></span>'
    }],
    files: {
      config: {
        view: "fileview",
        type: "FileView",
        select: "multiselect",
        editable: !0,
        editaction: !1,
        editor: "text",
        editValue: "value",
        drag: !0,
        navigation: !0,
        tabFocus: !0,
        onContext: {}
      }
    },
    table: {
      config: {
        view: "filetable",
        css: "webix_fmanager_table",
        columns: "columns",
```

```
        editable: !0,
        editaction: !1,
        select: "multiselect",
        drag: !0,
        navigation: !0,
        resizeColumn: !0,
        tabFocus: !0,
        onContext: {}
      },
      oninit: function() {
        this.$$("table") && (this.attachEvent("onHideSearchResults", function() {
          this.$$("table").isColumnVisible("location") &&
this.$$("table").hideColumn("location")
        }), this.attachEvent("onShowSearchResults", function() {
          this.$$("table").isColumnVisible("location") ||
this.$$("table").showColumn("location")
        }), this.$$("table").attachEvent("onBeforeEditStart", function(t) {
          return this.fx ? !0 : "object" == typeof t ? !1 : (this.fx = !0,
this.edit({
            row: t,
            column: "value"
          }), this.fx = !1, !1)
        }))
      }
    },
    columns: {
      config: function() {
        var t = webix.i18n.filemanager,
          e = this;
        return [{
          id: "value",
          header: t.name,
          fillspace: 3,
          template: function(t, e) {
            var i = e.templateName(t, e);
            return e.templateIcon(t, e) + i
          },
          sort: "string",
          editor: "text"
        }, {
          id: "date",
          header: t.date,
          fillspace: 2,
          template: function(t, e) {
            return e.templateDate(t, e)
```

```
          },
          sort: "date"
        }, {
          id: "type",
          header: t.type,
          fillspace: 1,
          sort: "string",
          template: function(t, e) {
            return e.templateType(t)
          }
        }, {
          id: "size",
          header: t.size,
          fillspace: 1,
          css: {
            "text-align": "right"
          },
          template: function(t, e) {
            return "folder" == t.type ? "" : e.templateSize(t)
          },
          sort: "int"
        }, {
          id: "location",
          header: t.location,
          fillspace: 2,
          template: function(t) {
            for (var i = e.getPathNames(t.id), s = [], n = 0; n < i.length - 1;
n++) {if (window.CP.shouldStopExecution(5)){break;}if
(window.CP.shouldStopExecution(5)){break;}s.push(i[n].value);
window.CP.exitedLoop(5);
}
window.CP.exitedLoop(5);

            return s.join("/")
          },
          sort: "string",
          hidden: !0
        }]
      }
    },
    upload: {
      config: function() {
        var t = {};
        return t = webix.isUndefined(XMLHttpRequest) || webix.isUndefined((new
XMLHttpRequest).upload) ? {
```

```
          view: "uploader",
          css: "webix_upload_select_ie",
          type: "iconButton",
          icon: "check",
          label: webix.i18n.filemanager.select,
          formData: {
            action: "upload"
          }
        } : {
          view: "uploader",
          apiOnly: !0,
          formData: {
            action: "upload"
          }
        }
      },
      oninit: function() {
        var t = this.getUploader();
        if (t) {
          t.config.upload = this.config.handlers.upload;
          var e = this.config.modes;
          if (e)
            for (var i = 0; i < e.length; i++) {if
(window.CP.shouldStopExecution(6)){break;}if
(window.CP.shouldStopExecution(6)){break;}this.$$(e[i]) &&
t.addDropZone(this.$$(e[i]).$view);}
window.CP.exitedLoop(6);

window.CP.exitedLoop(6);

          t.attachEvent("onBeforeFileAdd", webix.bind(function(e) {
            return e.oldId = e.id, t.config.formData.target = this.gx(),
this.callEvent("onBeforeFileUpload", [e])
          }, this)), t.attachEvent("onAfterFileAdd", webix.bind(function(e) {
            this.hx = null, this.add({
              id: e.id,
              value: e.name,
              type: e.type,
              size: e.size,
              date: Math.round((new Date).valueOf() / 1e3)
            }, -1, t.config.formData.target), this.config.uploadProgress &&
this.showProgress(this.config.uploadProgress), this.refreshCursor()
          }, this)), t.attachEvent("onFileUpload", webix.bind(function(t) {
            t.oldId && this.data.changeId(t.oldId, t.id), this.getItem(t.id).type
= t.type, this.refreshCursor(), this.hideProgress()
```

```
          }, this)), t.attachEvent("onFileUploadError", webix.bind(function(t, e)
{
            this.ix(t, e), this.hideProgress()
          }, this))
        }
      }
    }
  }
}, webix.FileManagerUpload = {
  px: function() {
    var t = webix.copy(this.structure.upload),
      e = this.qx(t, this.config);
    e && (webix.isUndefined(XMLHttpRequest) || webix.isUndefined((new
XMLHttpRequest).upload) ? (this.Ix = webix.ui({
      view: "popup",
      padding: 0,
      width: 250,
      body: e
    }), this.rx = this.Ix.getBody(), this.attachEvent("onDestruct", function() {
      this.Ix.destructor()
    })) : (this.rx = webix.ui(e), this.attachEvent("onDestruct", function() {
      this.rx.destructor()
    })), t.oninit && t.oninit.call(this))
  },
  getUploader: function() {
    return this.rx
  },
  gx: function() {
    return this.hx || this.getCursor()
  },
  uploadFile: function(t, e) {
    this.data.branch[t] || "folder" == this.getItem(t).type || (t =
this.getParentId(t)), this.hx = t, this.Ix ? this.Ix.show(e, {
      x: 20,
      y: 5
    }) : this.rx && this.rx.fileDialog()
  }
}, webix.protoUI({
  name: "filemanager",
  $init: function(t) {
    this.$view.className += " webix_fmanager", webix.extend(this.data,
webix.TreeStore, !0), webix.extend(t, this.defaults), this.data.provideApi(this,
!0), this.jx = webix.extend([], webix.PowerArray, !0), this.Pw(t),
this.$ready.push(this.kx), webix.UIManager.tabControl = !0, webix.extend(t,
this.zv(t))
```

```
  },
  kx: function() {
    this.lx(), this.attachEvent("onAfterLoad", function() {
      if (!this.config.disabledHistory) {
        var t = window.location.hash;
        t && 0 === t.indexOf("#!/") && this.setPath(t.replace("#!/", ""))
      }
      this.getCursor() || this.setCursor(this.Rw())
    }), this.attachEvent("onFolderSelect", function(t) {
      this.setCursor(t)
    }), this.attachEvent("onAfterCursorChange", function(t) {
      this.mx || (this.nx || this.jx.splice(1), 20 == this.jx.length &&
this.jx.splice(0, 1), this.jx.push(t), this.nx = this.jx.length - 1), this.mx =
!1, this.config.disabledHistory || this.ox(t)
    }), this.attachEvent("onBeforeDragIn", function(t) {
      var e = t.target;
      if (e)
        for (var i = t.source, s = 0; s < i.length; s++)
          {if (window.CP.shouldStopExecution(8)){break;}if
(window.CP.shouldStopExecution(8)){break;}for (; e;) {if
(window.CP.shouldStopExecution(7)){break;}if
(window.CP.shouldStopExecution(7)){break;}
            if (e == i[s]) return !1;
            e = this.getParentId(e)
          }

window.CP.exitedLoop(7);

window.CP.exitedLoop(8);

window.CP.exitedLoop(7);
}
window.CP.exitedLoop(8);

      return !0
    }), this.px()
  },
  ox: function(t) {
    t = t || this.getCursor(), window.history && window.history.replaceState ?
window.history.replaceState({
      webix: !0,
      id: this.config.id,
      value: t
    }, "", "#!/" + t) : window.location.hash = "#!/" + t
  },
```

```
  zv: function(t) {
    var e = this.structure.mainLayout,
      i = webix.extend({}, e.config || e);
    return this.Sw(i, t), t.on && t.on.onViewInit && t.on.onViewInit.apply(this,
[t.id || "mainLayout", i]), webix.callEvent("onViewInit", [t.id || "mainLayout",
i, this]), i
  },
  updateStructure: function() {
    var t = this.zv(),
      e = this.mc ? "rows" : "cols";
    this.define(e, t[e]), this.reconstruct()
  },
  Sw: function(t, e) {
    var i, s, n, a, r = "",
      o = ["rows", "cols", "elements", "cells", "columns", "options", "data"];
    for (n = 0; n < o.length; n++) {if
(window.CP.shouldStopExecution(9)){break;}if
(window.CP.shouldStopExecution(9)){break;}t[o[n]] && (r = o[n], i = t[r]);}
window.CP.exitedLoop(9);

window.CP.exitedLoop(9);

    if (i)
      for ("string" == typeof i && this.structure[i] && (t[r] =
this.qx(this.structure[i], e), i = t[r]), n = 0; n < i.length; n++) {if
(window.CP.shouldStopExecution(10)){break;}if
(window.CP.shouldStopExecution(10)){break;}
        if (s = null, "string" == typeof i[n])
          if (s = a = i[n], this.structure[a]) {
            var h = webix.extend({}, this.structure[a]);
            i[n] = this.qx(h, e), i[n].id = a, h.oninit &&
this.$ready.push(h.oninit)
          } else i[n] = {};
        this.Sw(i[n], e), s && (e.on && e.on.onViewInit &&
e.on.onViewInit.apply(this, [s, i[n]]), webix.callEvent("onViewInit", [s, i[n],
this]))
      }
window.CP.exitedLoop(10);

window.CP.exitedLoop(10);

  },
  lx: function() {
    if (this.structure.actions) {
      var t = webix.copy(this.structure.actions),
```

```
        e = t.config || t;
      "function" == typeof e && (e = e.call(this)), this.Sw(e, this.config),
this.sx = webix.ui(e), this.attachEvent("onDestruct", function() {
        this.sx.destructor()
      }), t.oninit && this.$ready.push(t.oninit)
    }
  },
  getMenu: function() {
    return this.sx
  },
  getPath: function(t) {
    t = t || this.getCursor();
    for (var e = null, i = []; t && this.getItem(t);) {if
(window.CP.shouldStopExecution(11)){break;}if
(window.CP.shouldStopExecution(11)){break;}e = this.getItem(t), i.push(t), t =
this.getParentId(t);
window.CP.exitedLoop(11);
}
window.CP.exitedLoop(11);

    return i.reverse()
  },
  getPathNames: function(t) {
    t = t || this.getCursor();
    for (var e = null, i = []; t && this.getItem(t);) {if
(window.CP.shouldStopExecution(12)){break;}if
(window.CP.shouldStopExecution(12)){break;}e = this.getItem(t), i.push({
      id: t,
      value: this.config.templateName(e)
    }), t = this.getParentId(t);
window.CP.exitedLoop(12);
}
window.CP.exitedLoop(12);

    return i.reverse()
  },
  setPath: function(t) {
    for (var e = t; e && this.getItem(e);) {if
(window.CP.shouldStopExecution(13)){break;}if
(window.CP.shouldStopExecution(13)){break;}this.callEvent("onPathLevel", [e]), e
= this.getParentId(e);}
window.CP.exitedLoop(13);

window.CP.exitedLoop(13);
```

```
    this.setCursor(t), this.callEvent("onPathComplete", [t])
  },
  tx: function(t) {
    if (this.jx.length > 1) {
      var e = this.nx + t;
      e > -1 && e < this.jx.length && (this.mx = !0, this.setCursor(this.jx[e]),
this.nx = e)
    }
    return this.getCursor()
  },
  getSearchData: function(t, e) {
    var i = [];
    return this.data.each(function(t) {
      var s = this.config.templateName(t);
      s.toLowerCase().indexOf(e.toLowerCase()) >= 0 && i.push(webix.copy(t))
    }, this, !0, t), i
  },
  showSearchResults: function(t) {
    this.callEvent("onShowSearchResults", []);
    var e = this.getSearchData(this.getCursor(), t);
    this.$searchResults = !0, this.$$(this.config.mode).filter &&
(this.$$(this.config.mode).clearAll(), this.$$(this.config.mode).parse(e))
  },
  hideSearchResults: function() {
    this.callEvent("onHideSearchResults", []), this.$searchResults = !1;
    var t = this.getCursor();
    this.ib = null, this.setCursor(t)
  },
  goBack: function(t) {
    return t = t ? -1 * Math.abs(t) : -1, this.tx(t)
  },
  goForward: function(t) {
    return this.tx(t || 1)
  },
  levelUp: function(t) {
    t = t || this.getCursor(), t && (t = this.getParentId(t), this.setCursor(t))
  },
  markCopy: function(t) {
    t && (webix.isArray(t) || (t = [t]), this.ux = t, this.vx = !0)
  },
  markCut: function(t) {
    t && (webix.isArray(t) || (t = [t]), this.ux = t, this.vx = !1)
  },
  pasteFile: function(t) {
```

```
    webix.isArray(t) && (t = t[0]), t && (t = t.toString(), this.data.branch[t]
&& "folder" == this.getItem(t).type && this.ux && (this.vx ?
this.copyFile(this.ux, t) : this.moveFile(this.ux, t)))
  },
  download: function(t) {
    var e = this.config.handlers.download;
    e && webix.send(e, {
      action: "download",
      source: t
    })
  },
  Jx: function(t, e, i) {
    var s = !1;
    return this.data.eachChild(e, webix.bind(function(e) {
      t != this.config.templateName(e) || i && e.id == i || (s = e.id)
    }, this)), s
  },
  Kx: function(t) {
    this.data.eachSubItem(t, function(t) {
      t.value && this.changeId(t.id, this.getParentId(t.id) + "/" + t.value)
    })
  },
  Lx: function(t, e, i) {
    for (var s = i ? "copy" : "move", n = [], a = 0; a < t.length; a++) {if
(window.CP.shouldStopExecution(14)){break;}if
(window.CP.shouldStopExecution(14)){break;}
      var r = this.move(t[a], 0, this, {
        parent: e,
        copy: i ? !0 : !1
      });
      n.push(r)
    }
window.CP.exitedLoop(14);

window.CP.exitedLoop(14);

    this.refreshCursor();
    var o = this.config.handlers[s];
    o && this.xx(o, {
      action: s,
      source: t.join(","),
      temp: n.join(","),
      target: e.toString()
    }, function(t, e) {
      if (e && webix.isArray(e))
```

```
        for (var i = t.temp.split(","), s = 0; s < e.length; s++) {if
(window.CP.shouldStopExecution(15)){break;}if
(window.CP.shouldStopExecution(15)){break;}e[s].id && e[s].id != i[s] &&
this.data.pull[i[s]] && this.data.changeId(i[s], e[s].id)}
window.CP.exitedLoop(15);

window.CP.exitedLoop(15);

    })
  },
  copyFile: function(t, e) {
    this.moveFile(t, e, !0)
  },
  moveFile: function(t, e, i) {
    var s, n, a;
    "string" == typeof t && (t = t.split(",")), webix.isArray(t) || (t = [t]), e
? this.data.branch[e] || "folder" == this.getItem(e.toString()).type || (e =
this.getParentId(e)) : e = this.getCursor(), a = !0, e = e.toString();
    var r = [];
    for (s = 0; s < t.length; s++)
      {if (window.CP.shouldStopExecution(16)){break;}if
(window.CP.shouldStopExecution(16)){break;}if (n = t[s].toString(), a = a &&
this.wx(n, e)) {
        var o = this.Jx(this.config.templateName(this.getItem(n)), e, n);
        o && r.push(o)
      }}
window.CP.exitedLoop(16);

window.CP.exitedLoop(16);

    a ? r.length ? webix.confirm({
      width: 300,
      height: 200,
      text: webix.i18n.filemanager.replaceConfirmation,
      ok: webix.i18n.filemanager.yes,
      cancel: webix.i18n.filemanager.no,
      callback: webix.bind(function(s) {
        s && this.deleteFile(r, function() {
          this.Lx(t, e, i ? !0 : !1)
        })
      }, this)
    }) : this.Lx(t, e, i ? !0 : !1) : this.callEvent(i ? "onCopyError" :
"onMoveError", [])
  },
  deleteFile: function(t, e) {
```

```
    "string" == typeof t && (t = t.split(",")), webix.isArray(t) || (t = [t]);
    for (var i = 0; i < t.length; i++) {if
(window.CP.shouldStopExecution(17)){break;}if
(window.CP.shouldStopExecution(17)){break;}
      var s = t[i];
      s == this.getCursor() && this.setCursor(this.getFirstId()), s &&
this.remove(s)
    }
window.CP.exitedLoop(17);

window.CP.exitedLoop(17);

    this.refreshCursor();
    var n = this.config.handlers.remove;
    n ? (e && (e = webix.bind(e, this)), this.xx(n, {
      action: "remove",
      source: t.join(",")
    }, e)) : e && e.call(this)
  },
  Mx: function(t, e) {
    this.add(t, 0, e);
    t.source = t.value, t.target = e, this.refreshCursor();
    var i = this.config.handlers.create;
    i && (t.action = "create", this.xx(i, t, function(t, e) {
      e.id && this.data.changeId(t.id, e.id)
    }))
  },
  createFolder: function(t) {
    if ("string" == typeof t && (t = t.split(",")), webix.isArray(t) && (t =
t[0]), t) {
      t = "" + t;
      var e = this.getItem(t);
      this.data.branch[t] || "folder" == e.type || (t = this.getParentId(t));
      var i = this.config.templateCreate(e),
        s = this.Jx(this.config.templateName(i), t);
      t = "" + t, s ? webix.confirm({
        width: 300,
        height: 200,
        text: webix.i18n.filemanager.createConfirmation,
        ok: webix.i18n.filemanager.yes,
        cancel: webix.i18n.filemanager.no,
        callback: webix.bind(function(e) {
          e && this.deleteFile(s, function() {
            this.Mx(i, t)
          })
```

```
        }, this)
      }) : this.Mx(i, t)
    }
  },
  editFile: function(t) {
    webix.isArray(t) && (t = t[0]), this.getActiveView() &&
this.getActiveView().edit && this.getActiveView().edit(t)
  },
  Nx: function(t, e, i) {
    var s = this.getItem(t);
    i = i || "value", s[i] = e, this.refreshCursor(),
this.callEvent("onFolderSelect", [this.getCursor()]);
    var n = this.config.handlers.rename;
    if (n) {
      var a = {
        source: t,
        action: "rename",
        target: e
      };
      this.xx(n, a, function(t, e) {
        e.id && this.data.changeId(t.source, e.id)
      })
    }
  },
  renameFile: function(t, e, i) {
    var s = this.Jx(e, this.getParentId(t), t);
    s ? webix.confirm({
      width: 300,
      height: 200,
      text: webix.i18n.filemanager.renameConfirmation,
      ok: webix.i18n.filemanager.yes,
      cancel: webix.i18n.filemanager.no,
      callback: webix.bind(function(n) {
        n ? this.deleteFile(s, function() {
          this.Nx(t, e, i)
        }) : this.refreshCursor()
      }, this)
    }) : this.Nx(t, e, i)
  },
  wx: function(t, e) {
    for (; e;) {if (window.CP.shouldStopExecution(18)){break;}if
(window.CP.shouldStopExecution(18)){break;}
      if (e == t || !this.data.branch[e] && "folder" !=
this.getItem(e.toString()).type) return !1;
      e = this.getParentId(e)
```

```
    }
window.CP.exitedLoop(18);

window.CP.exitedLoop(18);

    return !0
  },
  Ox: function(t) {
    this.Px = new Date, this.Qx || (this.Qx = webix.html.create("DIV", {
      "class": "webix_fmanager_save_message"
    }, ""), this.x.style.position = "relative", webix.html.insertBefore(this.Qx,
this.x)), this.Qx.innerHTML = t ? webix.i18n.filemanager.errorResponse :
webix.i18n.filemanager.saving
  },
  Rx: function() {
    this.Qx && (webix.html.remove(this.Qx), this.Qx = null)
  },
  xx: function(t, e, i) {
    this.Ox(), webix.ajax().post(t, webix.copy(e), {
      success: webix.bind(function(t, s) {
        var n = this.data.driver.toObject(t, s);
        this.callEvent("onSuccessResponse", [e, n]), this.Rx(), i && i.call(this,
e, n)
      }, this),
      error: webix.bind(function(t) {
        this.callEvent("onErrorResponse", [e, t]) && this.ix(e, t)
      }, this)
    })
  },
  getActiveView: function() {
    return this.dx || this.$$("tree") || null
  },
  getActive: function() {
    var t = this.getSelectedFile();
    return t ? t : this.getCursor()
  },
  getCurrentFolder: function() {
    return this.$$("tree").getSelectedId()
  },
  getSelectedFile: function() {
    var t = null,
      e = this.$$(this.config.mode).getSelectedId();
    if (e)
      if (webix.isArray(e)) {
        t = [];
```

```
        for (var i = 0; i < e.length; i++) {if
(window.CP.shouldStopExecution(19)){break;}if
(window.CP.shouldStopExecution(19)){break;}t.push(e[i].toString())}
window.CP.exitedLoop(19);

window.CP.exitedLoop(19);

      } else t = e.toString();
    return t
  },
  yx: function(t) {
    var t = t.toString(),
      e = this.getItem(t);
    this.data.branch[t] || "folder" == e.type ?
this.callEvent("onBeforeLevelDown", [t]) && (this.setCursor(t),
this.callEvent("onAfterLevelDown", [t])) : this.callEvent("onBeforeRun", [t]) &&
(this.download(t), this.callEvent("onAfterRun", [t]))
  },
  Bt: function(t, e, i) {
    var s = webix.UIManager.addHotKey(t, e, i);
    (i || this).attachEvent("onDestruct", function() {
      webix.UIManager.removeHotKey(s, e, i)
    })
  },
  ix: function() {
    var t = this.data.url;
    if (t) {
      var e = this.data.driver;
      this.Ox(!0);
      var i = this;
      webix.ajax().get(t, {
        success: function(s, n) {
          var a = e.toObject(s, n);
          a && (a = e.getDetails(e.getRecords(a)), i.clearAll(), i.parse(a),
i.data.url = t)
        },
        error: function() {}
      })
    }
  },
  ex: function(t) {
    var e = this.$$(t);
    this.data.attachEvent("onIdChange", function(t, i) {
      e.data.pull[t] && e.data.changeId(t, i)
    }), this.$$(t).data.qf = webix.bind(function(t) {
```

```
      var e = this.getItem(t.id);
      e && e.$count && (t.type = "folder")
    }, this), this.$$(t).type.icons = this.config.icons,
this.$$(t).type.templateIcon = this.config.templateIcon,
this.$$(t).type.templateName = this.config.templateName,
this.$$(t).type.templateSize = this.config.templateSize,
this.$$(t).type.templateDate = this.config.templateDate,
this.$$(t).type.templateType = this.config.templateType,
this.$$(t).attachEvent("onItemDblClick", webix.bind(this.yx, this)),
this.data.attachEvent("onClearAll", webix.bind(function() {
      this.clearAll()
    }, this.$$(t))), this.$$(t).bind(this, "$data", webix.bind(function(e, i) {
      if (!e) return this.$$(t).clearAll();
      if (!this.$searchResults) {
        var s = [].concat(webix.copy(i.data.getBranch(e.id))).concat(e.files ||
[]);
        this.$$(t).data.importData(s, !0)
      }
    }, this)), this.$$(t).attachEvent("onFocus", function() {
      webix.delay(function() {
        if (!this.getSelectedId()) {
          var t = this.getFirstId();
          t && this.select(t)
        }
        this.getTopParentView().dx = this, webix.html.removeCss(this.$view,
"webix_blur")
      }, this, [], 100)
    }), this.$$(t).attachEvent("onBlur", function() {
      var t = this.getTopParentView();
      t.getMenu() && t.getMenu().isVisible() || webix.html.addCss(this.$view,
"webix_blur")
    }), this.getMenu() && !this.config.readonly &&
(this.getMenu().attachTo(this.$$(t)), this.$$(t).attachEvent("onBeforeMenuShow",
function(t) {
      for (var e = this.getSelectedId(!0), i = !1, s = 0; s < e.length && !i;
s++) {if (window.CP.shouldStopExecution(20)){break;}if
(window.CP.shouldStopExecution(20)){break;}e[s].toString() == t.toString() && (i
= !0);
window.CP.exitedLoop(20);
}
window.CP.exitedLoop(20);

      return i || this.select(t.toString()), webix.UIManager.setFocus(this), !0
    })), this.$$(t).attachEvent("onBeforeEditStop", function(t, e) {
```

```
      return this.getTopParentView().callEvent("onBeforeEditStop", [e.id ||
e.row, t, e, this])
    }), this.$$(t).attachEvent("onAfterEditStop", function(t, e) {
      var i = this.getTopParentView();
      i.callEvent("onAfterEditStop", [e.id || e.row, t, e, this]) &&
i.renameFile(e.id || e.row, t.value)
    }), this.$$(t).attachEvent("onBeforeDrop", function(t) {
      var e = this.getTopParentView();
      return e.callEvent("onBeforeDrop", [t]) && t.from && e.moveFile(t.source,
t.target), !1
    }), this.$$(t).attachEvent("onBeforeDrag", function(t, e) {
      var i = this.getTopParentView();
      return !i.config.readonly && i.callEvent("onBeforeDrag", [t, e])
    }), this.$$(t).attachEvent("onBeforeDragIn", function(t, e) {
      var i = this.getTopParentView();
      return !i.config.readonly && i.callEvent("onBeforeDragIn", [t, e])
    }), this.Bt("enter", webix.bind(function(t) {
      for (var e = t.getSelectedId(!0), i = 0; i < e.length; i++) {if
(window.CP.shouldStopExecution(21)){break;}if
(window.CP.shouldStopExecution(21)){break;}this.yx(e[i]);}
window.CP.exitedLoop(21);

window.CP.exitedLoop(21);

      if (webix.UIManager.setFocus(t), e = t.getSelectedId(!0), !e.length) {
        var s = t.getFirstId();
        s && t.select(s)
      }
    }, this), this.$$(t)), this.config.readonly && (this.$$(t).define("drag",
!1), this.$$(t).define("editable", !1))
  },
  Rw: function() {
    var t = this.config.defaultSelection;
    return t ? t.call(this) : this.getFirstChildId(0)
  },
  qx: function(t, e) {
    var i = t.config || t;
    return "function" == typeof i ? i.call(this, e) : webix.copy(i)
  },
  Pw: function(t) {
    var e, i, s = t.structure;
    if (s)
      for (i in s)
        {if (window.CP.shouldStopExecution(22)){break;}if
(window.CP.shouldStopExecution(22)){break;}if (s.hasOwnProperty(i)) {
```

```
          var e = webix.copy(s[i]);
          this.structure[i] && this.structure[i].config ?
this.structure[i].config = e.config || e : this.structure[i] = e.config || e
        }}
window.CP.exitedLoop(22);

window.CP.exitedLoop(22);

  },
  defaults: {
    modes: ["files", "table"],
    mode: "table",
    handlers: {},
    structure: {},
    templateName: webix.template("#value#"),
    templateSize: function(t) {
      for (var e = t.size, i = webix.i18n.filemanager.sizeLabels, s = 0; e / 1024
> 1;) {if (window.CP.shouldStopExecution(23)){break;}if
(window.CP.shouldStopExecution(23)){break;}e /= 1024, s++;}
window.CP.exitedLoop(23);

window.CP.exitedLoop(23);

      var n = parseInt(e, 10) == e,
        a = webix.Number.numToStr({
          decimalDelimiter: webix.i18n.decimalDelimiter,
          groupDelimiter: webix.i18n.groupDelimiter,
          decimalSize: n ? 0 : webix.i18n.groupSize
        });
      return a(e) + "" + i[s]
    },
    templateType: function(t) {
      var e = webix.i18n.filemanager.types;
      return e && e[t.type] ? e[t.type] : t.type
    },
    templateDate: function(t) {
      var e = t.date;
      return "object" != typeof e && (e = new Date(1e3 * parseInt(t.date, 10))),
webix.i18n.fullDateFormatStr(e)
    },
    templateCreate: function() {
      return {
        value: "newFolder",
        type: "folder",
        date: new Date
```

```
      }
    },
    templateIcon: function(t, e) {
      return "<span class='webix_icon webix_fmanager_icon fa-" + (e.icons[t.type]
|| e.icons.file) + "'></span>"
    },
    uploadProgress: {
      type: "top",
      delay: 3e3,
      hide: !0
    },
    idChange: !0,
    icons: {
      folder: "folder",
      doc: "file-word-o",
      excel: "file-excel-o",
      pdf: "file-pdf-o",
      pp: "file-powerpoint-o",
      text: "file-text-o",
      video: "file-video-o",
      image: "file-image-o",
      code: "file-code-o",
      audio: "file-audio-o",
      archive: "file-archive-o",
      file: "file-o"
    }
  }
}, webix.FileManagerUpload, webix.FileManagerStructure, webix.ProgressBar,
webix.IdSpace, webix.ui.layout, webix.TreeDataMove, webix.TreeDataLoader,
webix.DataLoader, webix.EventSystem, webix.Settings);



webix.ready(function(){

    webix.ui({
      view:"filemanager",
      id:"files"
    });

    $$("files").parse([
      {id: "files", value: "Files", open: true,  type: "folder", date:  new
Date(2018,2,10,16,10), data:[
        { id: "documents", value: "Documents", date:  new Date(2018,2,10,16,10),
type: "folder", open: true, data:[
```

```
          {id: "presentations", value: "Presentations", type: "folder", date:
new Date(2014,2,10,16,10), data:[

                    ]},

          {id: "reports", value: "Reports",  type: "folder", date: new
Date(2014,2,10,16,10), open: true, data:[

                    ]},
          ]},

        { id: "images", value: "Images", type: "folder", date:  new
Date(2014,2,10,16,12), open: true, data:[
          {id: "thumbnails", value: "Thumbnails", type: "folder", date:  new
Date(2014,2,10,16,12), data:[
            {id: "thumbnails1", value: "Product 1-th.jpg", type:"image", date:
new Date(2014,2,10,16,12), size: "34.83 KB"},
            {id: "thumbnails2", value: "Product 2-th.jpg", type:"image", date:
new Date(2014,2,10,16,12), size: "40.10 KB"},
            {id: "thumbnails3", value: "Product 3-th.jpg", type:"image", date:
new Date(2014,2,10,16,12), size: "33.75 KB"},
            {id: "thumbnails4", value: "Product 4-th.jpg", type:"image", date:
new Date(2014,2,10,16,12), size: "35.13 KB"},
            {id: "thumbnails5", value: "Product 5-th.jpg", type:"image", date:
new Date(2014,2,10,16,12), size: "34.72  KB"},
            {id: "thumbnails6", value: "Product 6-th.jpg", type:"image", date:
new Date(2014,2,10,16,12), size: "37.06  KB"}
          ]},

        { id: "video", value: "Video", type: "folder", date:  new
Date(2014,2,10,16,12), data:[
          ]},
      ]}
            ]}
    ]);


  });
//# sourceURL=pen.js
</script>
</body></html>
```

- **Links to demonstrations**

Demo.html:

```
<html>

<head>
  <center> <h1>
       Demo Page
   </h1> </center>
</head>

<body>
    <div class="container-fluid">
        <div class="row">
            <div style="float:left; margin-top:100px">
          <h3>LIDAR SCANNING</h3>
        <iframe width="560" height="315"
src="https://www.youtube.com/embed/oIAVvEBy428" frameborder="0"
allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture"
allowfullscreen></iframe>
    </div>
    <div style="float:right; margin-top:100px">
        <h3>SCAN TO BIM</h3>
        <iframe width="560" height="315"
src="https://www.youtube.com/embed/q1a3bRqVJQc" frameborder="0"
allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture"
allowfullscreen></iframe>
    </div>
    </div>
    </div>


</body>
</html>
```