BES 468574

Combining Forge and Machine Learning to Automate Time-Consuming Tasks

Rana Zeitouny
EMDC Group

Majd Makhlouf
Building Information Researchers and Developers (BIRD) OÜ

Learning Objectives

- Describe the concept and advantages of cloud computing and the Autodesk Forge platform.
- Identify repetitive tasks that can nowadays be automated through machine learning and cloud computing.
- Explore machine learning as an automation method that analytically builds data models and identifies data correlations.
- Estimate the benefits of integrating revolutionary technologies into a firm's daily workflow.

Description

Using our daily office routines, we must accomplish several repetitive and time-consuming tasks to meet client requirements and international standards. Some of these tasks are easy to automate; others seem limited by desktop resources or require human intervention. With the evolution of machine learning and introduction of the Forge Design Automation API, EMDC Group was able to automate some of those processes, thus saving tremendous amounts of time and improving team's efficiency. This class aims to present a sample of what we've built in-house to give attendees an idea about the tasks that can nowadays be automated and the potential that artificial intelligence and cloud computing introduce into the architecture, engineering, and construction (AEC) industry.

Speaker(s)

Rana Zeitouny entered the engineering consultancy industry in 2002, after obtaining her bachelor's degree in Electrical Engineering from the Lebanese American University in Lebanon. Since then, she deeply grasped the AutoCAD software and in 2005, she standardized the AutoCAD implementation including layers, blocks, commands, shortcuts, etc. and trained the Electrical department. From 2008 till 2016, she gave university courses about Electrical Design including training sessions on AutoCAD software. Today, after taking a REVIT training in 2015, she is using both AutoCAD and REVIT software in her daily consultancy works.

Majd Makhlouf Majd is a Mechanical Engineer and Design Technologist, with a Master of Science in Mechanical Engineering. He is an Autodesk Revit Certified Professional and a member of the Autodesk Developer Network. In January 2020, he founded Building Information Researchers and Developers OÜ, a software development company based in Estonia and providing services for the AEC sector worldwide. He specializes in BIM Management, Autodesk Revit and AutoCAD Add-in development, both public and custom developed, Forge web and cloud-based apps, Dynamo Zero Touch Node Packs, and mobile VR/AR applications.

Introduction: Defining the Buzzwords

In the English dictionary, "Forge" means to create something strong and enduring. In addition, "Machine Learning" means an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.

Therefore, in the AEC world, what can be the implication of Forge and Machine Learning and especially when they are combined?

Combining Forge and Machine Learning to Automate Time Consuming Tasks?

Why is Automating Time Consuming Tasks so important?

As the projects are currently so fast paced, the most essential priority is to meet deadlines. However, time consuming and iterative tasks can be most of the time a burden on the project's planning, allowing less time for important decisions to make. When these time-consuming tasks are automated, time will be invested in more important tasks and the project's duration and cost can be considerably brought down.

However, there are Limitations for Automation

Automation is limited by many factors; mainly the desktop resources whereby in some cases the automation time is almost equal to the manual iteration time. In addition, some processes will still require manual input. Also, in some cases, the API limitations complicate automation until making it impossible in case a certain functionality is not publicly exposed.

What is Autodesk Forge?

Autodesk Forge is in fact, cloud apps are starting to dominate the digital world, and they will be the main technological focus in the 2020s, the way mobile platforms were 10 years ago, and Forge is Autodesk's introduction of Cloud Computing into the world of CAD oriented processes. It is a cloud-based developer platform enabling developers to automate tasks using cloud computing.

Something to keep in mind is that forge is not just one solution; it offers several functionalities and exposes several APIs. For example, the BIM 360 API allows developers to automate BIM 360 project setup, by assigning permissions, initiating projects, and so on. These tasks were slightly time consuming and the BIM 360 API enables the elimination of several of these tasks.

Another API worth mentioning is the Data Management API, which automates BIM 360 Docs file transfer operations, such as downloads, uploads, and so on. That opens a door to schedule document upload to BIM 360 docs for example, which often is a necessity.

Several other APIs and functionalities are featured by Forge, like the Forge Viewer which enables the viewing of the model within web browsers, the Reality Capture API, which handles point clouds, raster images, and so on, webhooks, etc.

However, the Forge API on which this class focuses mostly and the one with the most potential for EMDC group as engineering consultants is the Design Automation API.

Simply put, the Design Automation API runs Autodesk software instances and executes precompiled Add-Ins and Scripts over cloud servers. Hence, the Design Automation for Revit API, for example, enables developers to compile Revit cloud add-ins that run remotely in the cloud. Several Autodesk platforms are supported by the Design Automation API, from AutoCAD and Revit, which are the main focus of this class, to 3DS Max and Inventor.

Why use the Design Automation API?

Several advantages are offered by the Design Automation API. Using this functionality, the user can avoid wasting desktop workstations on processes with long runtimes as Forge provides an additional virtual workstation.

Moreover, the Forge add-in is running on high performing servers that no desktop workstation can even compete with, which offers a highly increased performance.

That increase in performance implies a huge decrease in runtime and thus saves tremendous amounts of time.

Another considerable advantage is that running these apps can be initiated from any platform, whether desktop, mobile, or any web browser. So, any user with an internet connection, whether at home, in the office, or on site, can access those apps and run them, which is becoming a standard feature to have nowadays.

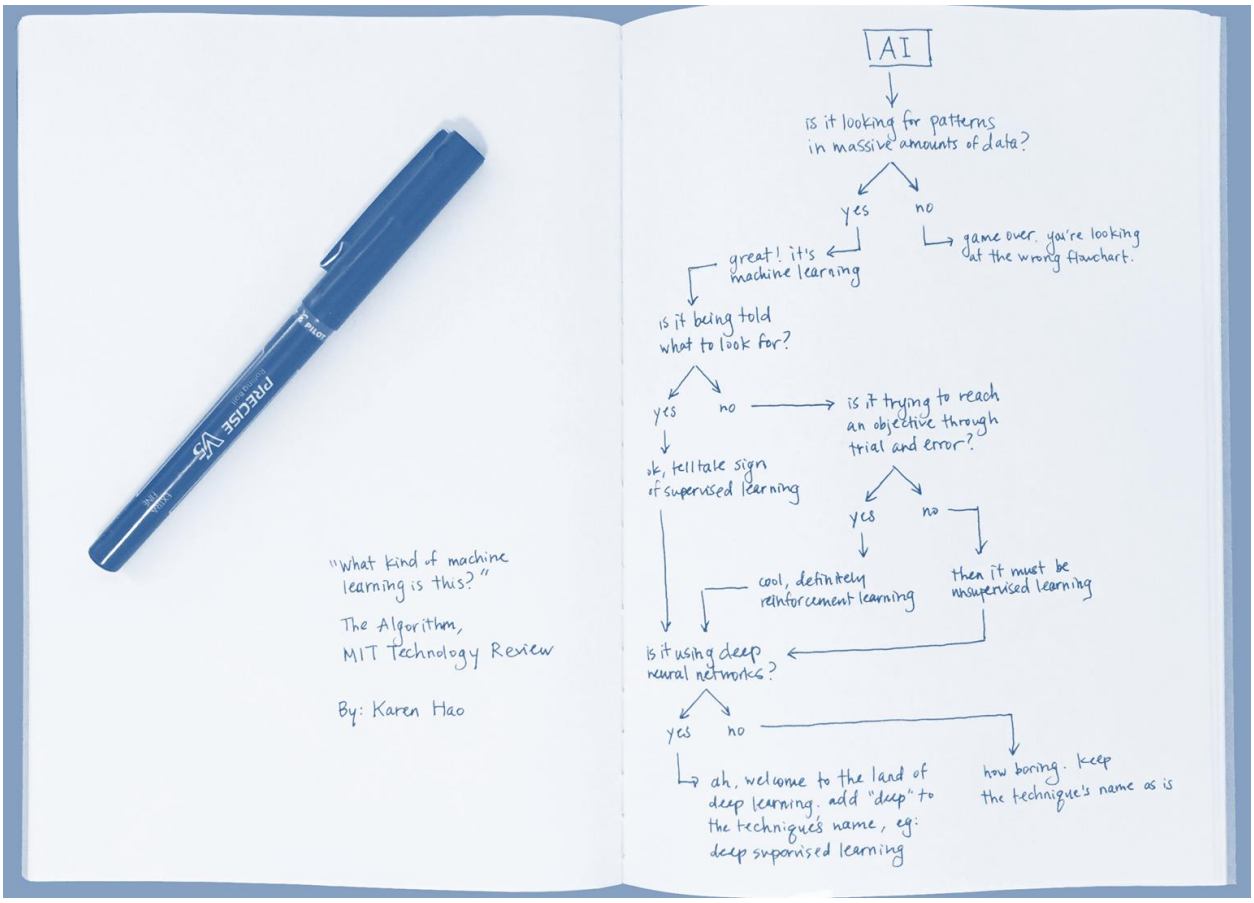
What is Machine Learning?

To make this definition simple, it is important to define what is not Machine Learning:
- Machine Learning is not a synonym for artificial intelligence
- Machine Learning does not insinuate machines gaining self-awareness and starting to read and speak
- Machine Learning does not mean that computers will gradually gain control over the world.

Then, what is Machine Learning exactly?

Machine Learning is in fact:
- A subdivision of artificial intelligence
- A combination of statistics and probability

Primarily, the first step of Machine Learning starts with automated statistical algorithms that process input data and find data relations which implies the learning term. As a second step, and after the data is gathered and the algorithm is trained, automated probabilistic processes come in play and predict new values based on previously processed data. So essentially, a probabilistic model is being built and not a perfectly accurate mathematical model. However, even with a risk of uncertainty, the decision made by the algorithms will be based on the highest “scoring”, and that makes the model useful.



This reminds us of the famous statistician George Box who said, “all models are wrong, but some are useful”.

When is Machine Learning Needed?

- When the project presents shortage of information: In other words, when dealing with non-parametric objects, like 2D CAD drawings; Fake BIM Models; Point Clouds; Raster images for example.

Or

- When the APIs have limitations: More specifically, when a functionality is not exposed by the API, or when automating a process cannot be achieved through conventional statements (For example, If statements, For loops, While loops…)

In these 2 above scenarios, 2 solutions would be available:

- Performing manual input: Which means dedicating a team of employees to do the required work manually.

Or

- Training a machine learning algorithm: Which means gathering data from previous models, finding a data relationship (being the statistical part), and using it to predict the missing parameter values and to automate the process (being the probabilistic part).

Machine learning is a buzzword, but not all automation is machine learning.

Nowadays, the term Machine Learning is a widely used buzzwords. Many active users are frequently publishing automation scripts (Dynamo scripts, Macros, Add-Ins…) and claiming that their work falls under the Machine Learning or Artificial Intelligence categories. It is important to know that not all automation, no matter how advanced it may seem, uses Machine Learning. Before dealing with real-case scenarios that actually use Machine Learning underneath, it would be useful to start with some samples that can be achieved without the use of Machine Learning.

Sample 1: Automated P-Trap Connection

The first sample is an auto-routing sample where condensate drain connection and their P-traps have been automatically connected to the main runs. This is an auto-routing sample that involves a lot of advanced mathematics, whether vector theory or linear algebra, so that the components are correctly placed and connected. Users may fall upon several auto-routing samples associated with machine learning which is often misleading as this can often be achieved through classic automation. In short, and since there is no data gathering, nor data labeling, nor algorithm training, nor statistically extracted data relations, machine learning is not in play.

Sample 2: Automated Hanger Placement

Another advanced sample is an add-in built at EMDC group that automatically places pipe or duct hangers and connects them to the nearest structural elements. Such a process for example does not involve any aspects of machine learning.

Sample 3: Automated Attached Hangers

The third sample is also a hanger connection tool that attaches hangers to the nearest hanger above it. Although this does involve some advanced mathematics to detect the bearing hanger lines, it can be achieved without the use of machine learning.

The above 3 samples illustrate explicit automations and do not use any aspect of Machine Learning. These samples can be found at www.emdc-forge.com\samples.

However, a lot of processes do involve some form of Machine Learning and are being used today in our daily life and in the AEC industry.

What cannot be “conventionally” automated?

Before discussing the real case Machine Learning based custom apps at EMDC group, it is important to start by enumerating general samples that cannot be explicitly automated. For example:

- Scanned PDF Data Extraction:
  Example: A scanned PDF drawing from which information is needed like handwritten comments.
- Documentation of Exploded CAD drawings:
  Example: An AutoCAD drawing from where sheet names and numbers need to be extracted and listed in a spreadsheet.
- Raster Image Detail Conversion:
  Example: An installation detail which needs to be translated in an AutoCAD drawing.
- Point Cloud to BIM model conversion:
  Example: A Point Cloud model which needs to be translated into an as-built model.

This impossibility of automation will therefore lead to:

- Requiring Manual Input
- Performing iterative tasks which are most of the time out of scope
- Leading to Overpayment to cater for employees' overtime and additional tasks.

How can these tasks be accomplished?

Solutions to the above are few. One solution would be to train any competent (or not) operator to do this time-consuming job or… train a machine!

Based on the above, automated solutions for each of the issues can be identified:

- Scanned PDF Data Extraction

Solution: Use of Automated Optical Character Recognition, which applies aspects of Machine Learning to recognize typical schematics. In this case, the letters of the handwritten comments are being recognized.

- Documentation of Exploded CAD drawings

Solution: Use of Machine Learning to find sheet information and document it, based on previous cases. In this case, finding sheets names and numbers.

- Raster Image Detail Conversion:

Solution: Use of a Vectorization software which also applies Machine Learning to detect lines, circles and forms. In this case, the pixels are converted into AutoCAD elements.

- Point Cloud to BIM model conversion:

Solution: Use of Machine Learning to automatically detect elements. In this case, connecting the group of cloud points and translating them into building elements.

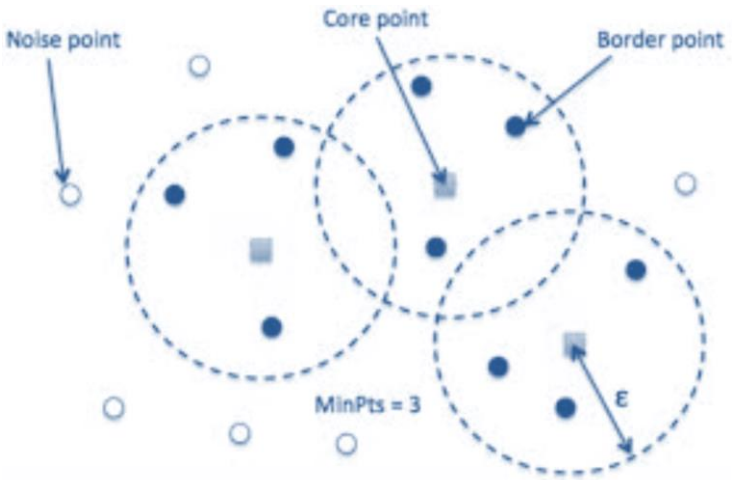
Machine Learning Subfields Involved in EMDC Group's customized workflow

Machine Learning is a diverse field, spanning from clustering to supervised learning, unsupervised learning, classification, regression, decision trees… It includes complex subfields such as deep learning and neural networks, which operate on several automation layers and open opportunities to achieve impressive results, such as an algorithm that learns how to complete a Console Game level by itself and so on.

Therefore, before going through the EMDC Group real case scenarios in detail, it is important to define which Machine Learning subfields were involved in the development of these customized apps.

Clustering

Clustering is the grouping of non-labeled data based on a certain similar property. Clustering is a form of unsupervised machine learning. It is unsupervised because the data processed has not been labeled before, nor has a training set been generated before based on labeled data. So the elements are raw, unsorted, unlabeled and need to be grouped into wider blocks.



The main challenge that clustering involves is the grouping definition. In simpler terms, the challenge lies in determining what makes a group of elements different than another, and what defines the beginning of a group, when a group should be trimmed, and when another group of elements should start to be gathered.

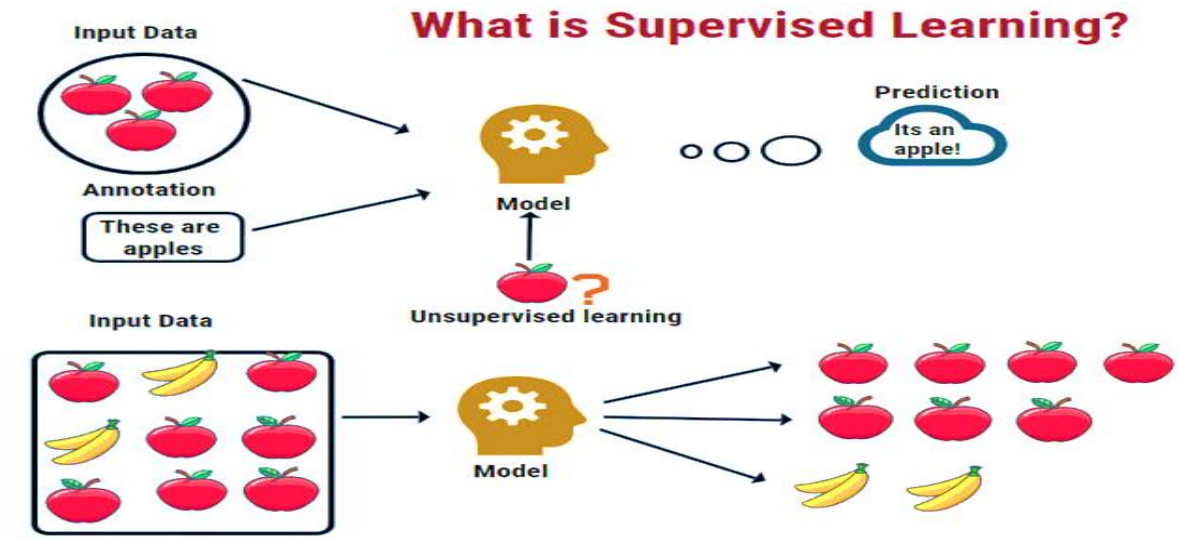
That is defined using statistical processes that automatically define a group of similar elements. Clustering includes several algorithms, of which we'll be mentioning two: the first is based on the mean value of the data, so that the mean values are close to or match a chosen number (K) of kernel elements. It is called a K-Means clustering algorithm. The second is based on data density, where similar elements are gathered until their density distribution is lower than a certain threshold, after which the group is limited. That is called a DBSCAN algorithm.

This can be illustrated through the most common clustering-based process encountered every day: facial recognition. A face is recognized, knowing that its image is usually a group of pixels, by the detection of its components, which in turn are recognized based on a clustering algorithm. The eye for example, and if we were to simplify the process which in reality is way more complicated, is recognized by having the pixels of the eye color grouped until the color's density, which at some point starts to decrease gradually until it fades away, is lower than a certain limit. That is an eye detected. The same process happens for other face components and that way the whole face is recognized.

Supervised Learning

The other Machine learning subfield involved is Supervised Learning. We'll be introducing the concepts behind Supervised Learning by a simple example involving apples and bananas.

- First, data (in this case pictures of apples and bananas) is labeled and fed into a learning algorithm.
- Then, a training set is generated, and the relation between the labels, which in this case is the fruit type, and the data, which is the fruit pictures, is statistically found (for example, a high number of red images corresponds to a high number of apples).
- When an unlabeled item is fed into the algorithm, the generated dataset is used and the type of the fruit is detected based on how probable it is matching with the labeled data (for example, if it's red, there's a 99% probability that it's an apple.) Of course, and as it is based on probability, there is a chance for outliers or, in other terms, wrong results, in case for example, the user feeds in the picture of a rotten, yellowish apple.



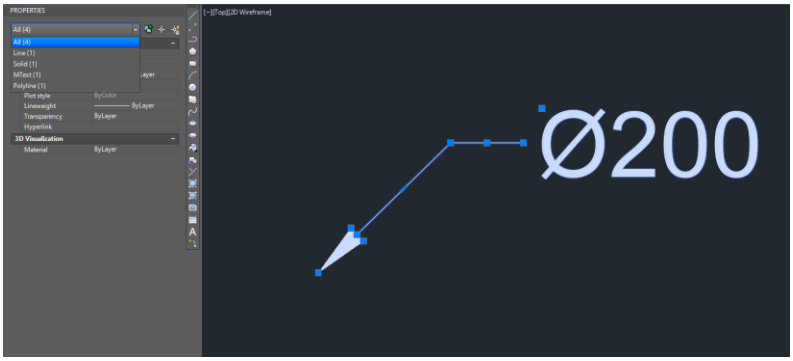
Why is it called supervised? Because the data entered to generate the set and statistically find the relation has previously been labeled by a user.

Apple and banana detection might seem silly, but a slightly more advanced process is nowadays used for groundbreaking advancements, such as cancer prediction algorithms, where a software can give, to a certain accuracy, a prediction on whether someone may or may not have cancer five years later, which can save millions of lives. Labeled mammographic data for examples with whether a patient had cancer 5 years after the corresponding mammography exam are fed into the system and the particular aspects of the cancer that were not previously known to pathologists are being found due to such processes.

Real Case Scenario 1: Using Clustering to Automate Multileader Grouping

Defining the Problem

- A client wants to produce detailed AutoCAD drawings based on a BIM model.
- The first step would be to export the Revit model into AutoCAD drawings as this saves 50% of the work.
- However, a problem occurs: Tags are exported into exploded multileaders. In this case, multileaders were required by the client, and even in a determined style.



Classical Solution

The classical solution is to create an AutoLISP routine to group text elements and exploded leaders into multileaders.

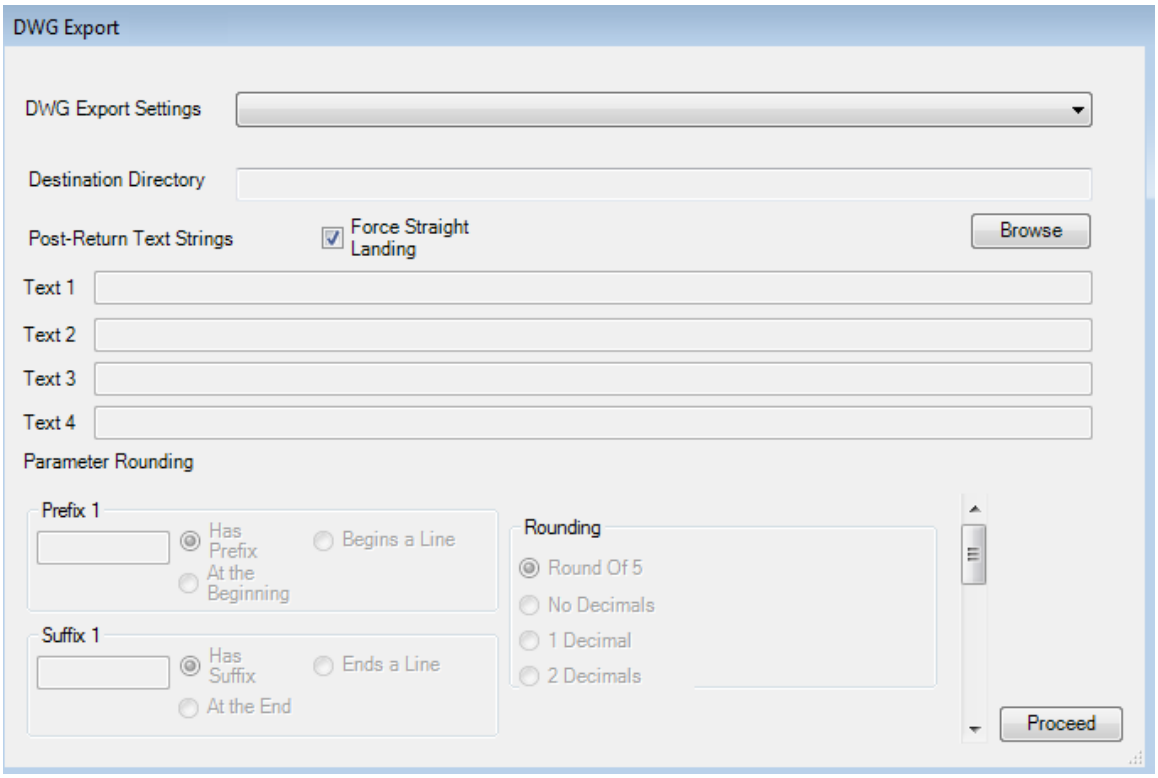
- For this purpose, the user selects the elements to be grouped, one group at a time.
- The problem is that this routine still requires manual input, by selecting the text related to the multileader, which presents a big margin of errors if automated.
- The solution is the use of the DBSCAN Clustering algorithm which groups multileader elements into multileaders based on their distribution density and similarity around each text, sorting them out without needing user input. The similarity parameter is the distance separating each line within a text paragraph.

```
(defun C:myfunc ( / I ITEM SS)
  (setq i 0)
  (princ "Select Text Objects")
  (setq ss (ssget '((0 . "TEXT"))))
  (repeat (sslength ss)
    (setq item (ssname ss i))
    (princ (cdr (assoc 7 (entget item))))
  )
)
```

How is this solution implemented?

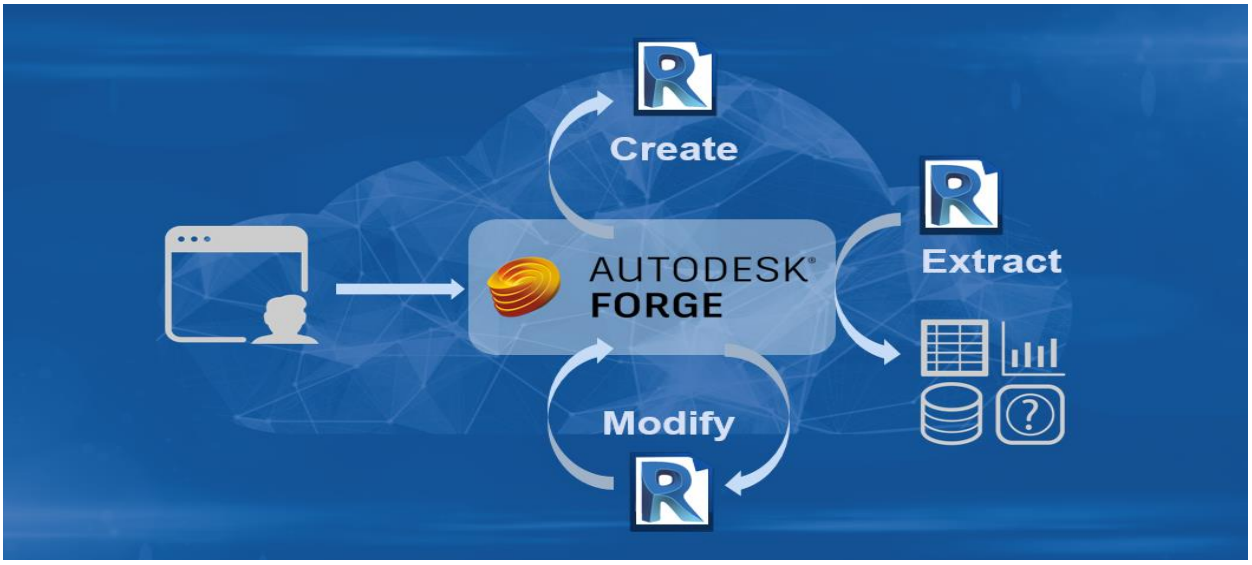
The automated solution would be a Revit API based add-in used to execute the DBSCAN algorithm over exported AutoCAD drawings.

- This add-in uses AutoCAD Interoperability to process drawings as soon as they are exported. In other words, it exports the Revit sheets to AutoCAD drawings, opens the exported drawings and runs the solution directly in AutoCAD.
- This add-in saves drawing opening time and eliminates manual input.
- However, this solution has a problem: It is a resource heavy algorithm, implying longer execution time and requiring dedicated workstations.



How is this scenario resolved? By using FORGE Design Automation for Revit.

The desktop add-in previously described is converted into a Forge add-in with lower execution time and less user interaction. As a result, the desktop workstations are saved for other tasks.



HTML Interface

As in any software, there is the user interface part that should be user friendly and that the user sees and uses to operate an app, and the algorithm, hidden underneath, with all the detailed processes involved.

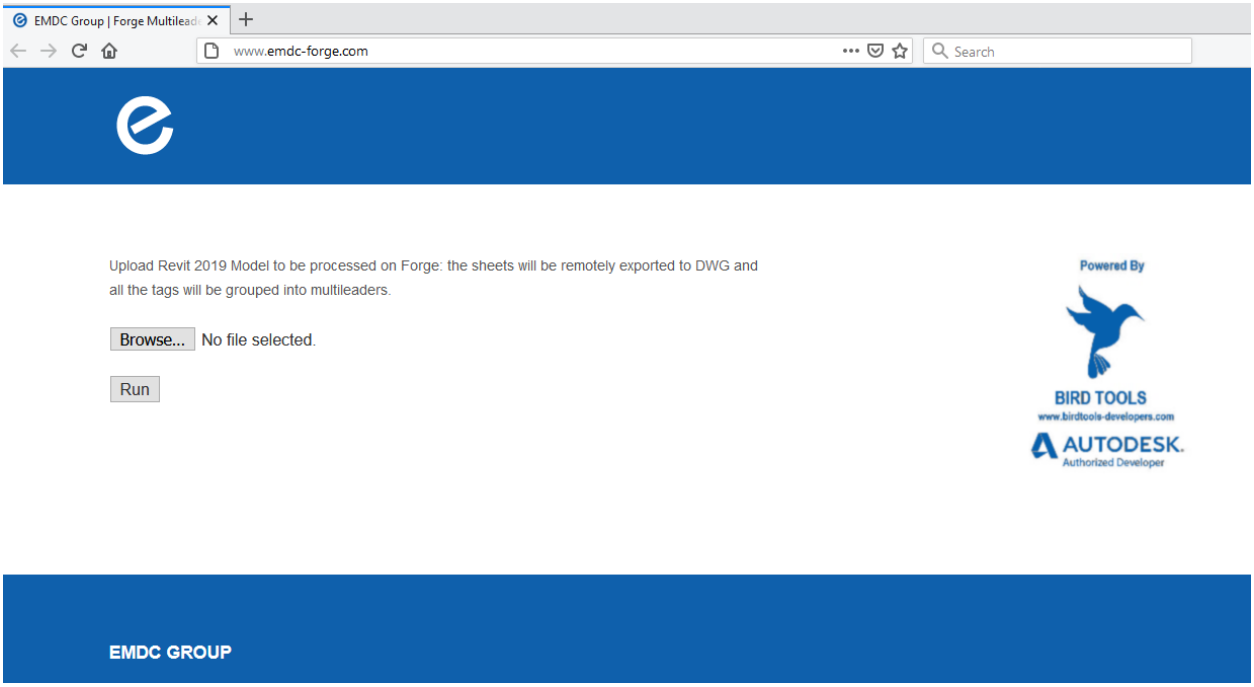
In a Design Automation for Revit app, the app is controlled through HTML requests, which are what happens underneath whenever a user executes anything online, whether login, logout…

To upload, run a Forge app, download the resulting files, and so on, a request should be posted from the user side to the server side, telling the server to initiate that operation.

That is a very important aspect, because HTML requests are supported by several platforms and programming languages, so the user can build an app, or preferably a web page (which is what has been done in EMDC group's case) based on whatever language is preferred, whether cURL, Javascript, ASP.NET, Ruby, Python…

The language used in this case is Javascript. In fact, it is important that such a process is supported across as much devices and platforms and web browsers as possible, and currently JavaScript is supported by almost all web browsers out there. So, it makes sense the most. Python is definitely a runner up in that field. Note that the UI can also be a Desktop app showing a classical Windows form, or an interactive mobile app…

Therefore, posting the command should give the order to execute a script or an add-in. The current example is a Revit cloud add-in that in turn runs an AutoCAD cloud script to group the multileaders.



Whenever the Revit cloud add-in receives the corresponding request, it runs the scripts and generates the output files. The add-ins themselves, both the Revit and AutoCAD ones, are DLL Class libraries compiled to operate on top of the .NET environment. These classes can be written

in any .NET language, whether C#, F#, VB.NET, and so on. The compiled class libraries are then bundled and pre-uploaded to the Forge servers, where the server constantly listens to an HTML requests in order to run the app in question.

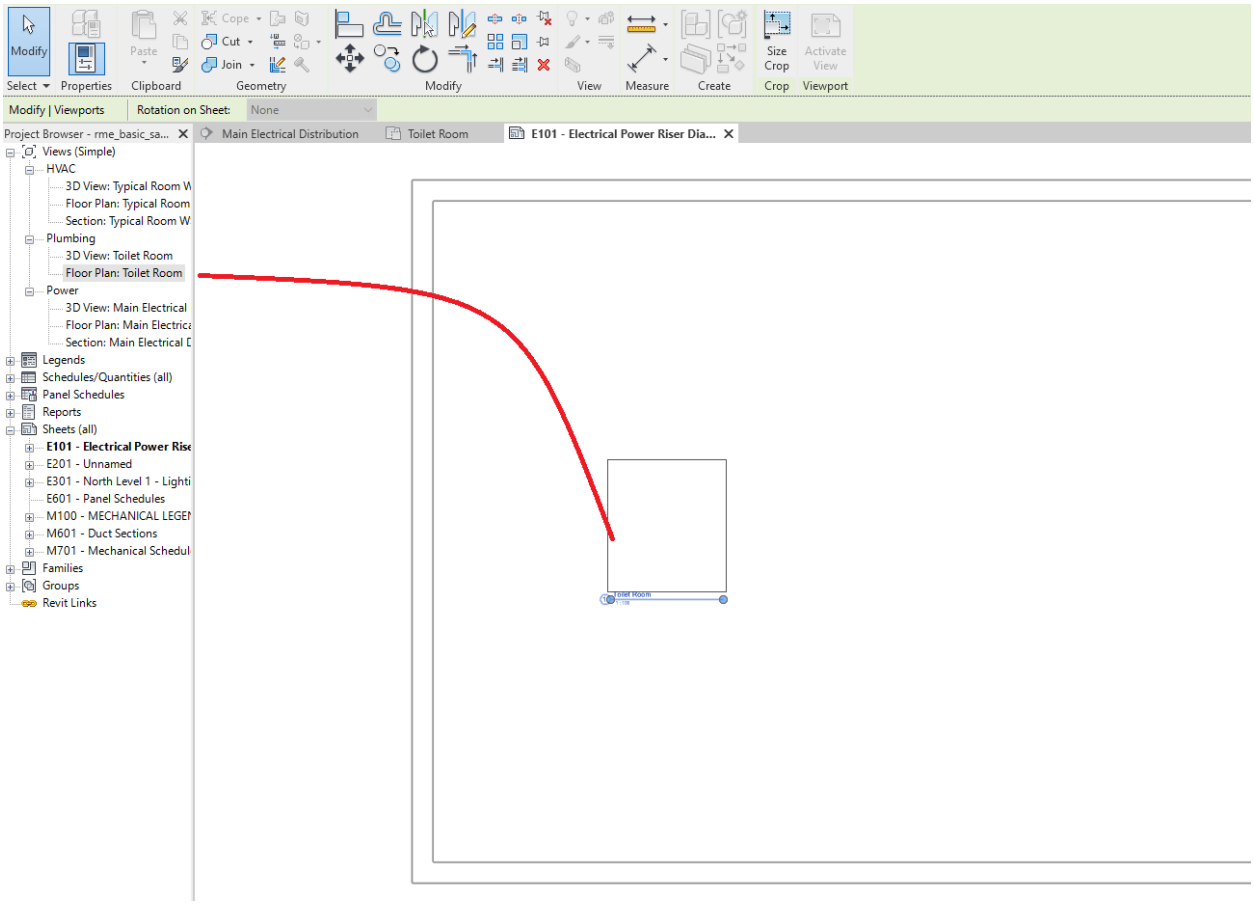
Note that any API references to the Revit or AutoCAD UIs are completely removed when a Desktop add-in is converted into a cloud one, as now the operation is driven by HTML requests and the interface is completely separate.

Real Case Scenario 2: Automating View Placement on Revit Sheets Using Supervised Learning

Defining the Problem

This task included View Placement for a huge number of sheets which is every BIM Manager's nightmare. It consisted of the following steps:

- Expand the sheetlist and look for the required sheet
- Open each sheet
- Expand the views
- Drag and drop each view on top of the sheet



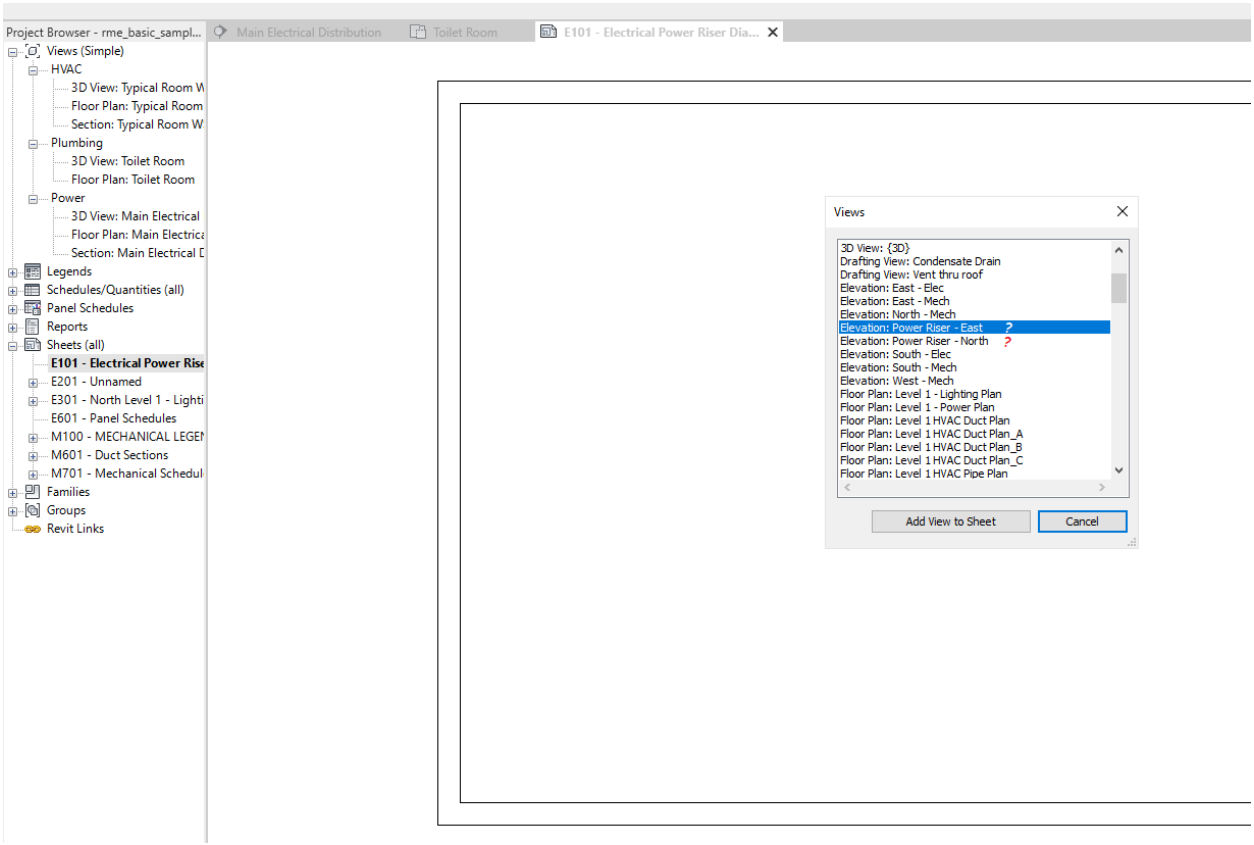
All the above is done manually and is a time-consuming process.

There are some solutions which perform the above task, already implemented by other companies:

- A first solution is an add-in that creates both views and sheets at the same time. This solution is not always an option because in some cases, many views should be shown on the same sheet; in other cases, the sheetlist is not always ready by the time we create the views.
- Another solution is an add-in with simpler user interfaces (for example: two opposed lists for views and sheets, droplists, etc.). This eliminates the manual dragging of views but still requires manual labor to match the views with the sheets.

What is the Main Problem in this scenario?

Matching a view to a sheet still requires human intervention, as the decision of matching is based on random parametric data being common for both the view and its sheet.



So, what could be the solution? To be able to match the random parameters of both sides, a process was inspired from the ranking, or more technically, scoring algorithm used by search engines: any separate keyword is extracted from all parameters on both the view side and the sheet side. When we find a match for a given sheet keyword, the score of the sheet increases, and thus, the view is placed on the sheet with the highest score, and human intervention to determine the matching sheet is minimized.

Initial tests were all positive.

However, and on a long term basis, some other flaws were found with this method: flaw number one is that sometimes technical words used in views do not match the ones in the sheet, where

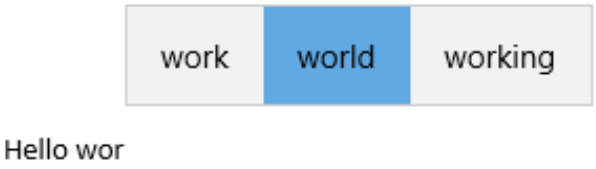
a synonym is used instead. The simplest example is related to the terms Drainage and Sanitary, which happens a lot, as usually views are created by users while they work on their Model, before sheets are created.

Another problem found, is that prepositions, such as to, of… and while they have no useful meaning, do affect this process as they have the same score that meaningful words do.

That is also applicable to meaningless parameters that should not affect the placement process, such as sheet revisions for example. Those also have the same score that other more meaningful parameters do.

So, the scoring algorithm is problematic, and the scores are not fairly assigned. Synonyms should involve higher scores, and less meaningful parameters and keywords should involve lower ones.

So, here is where supervised learning came to the rescue!

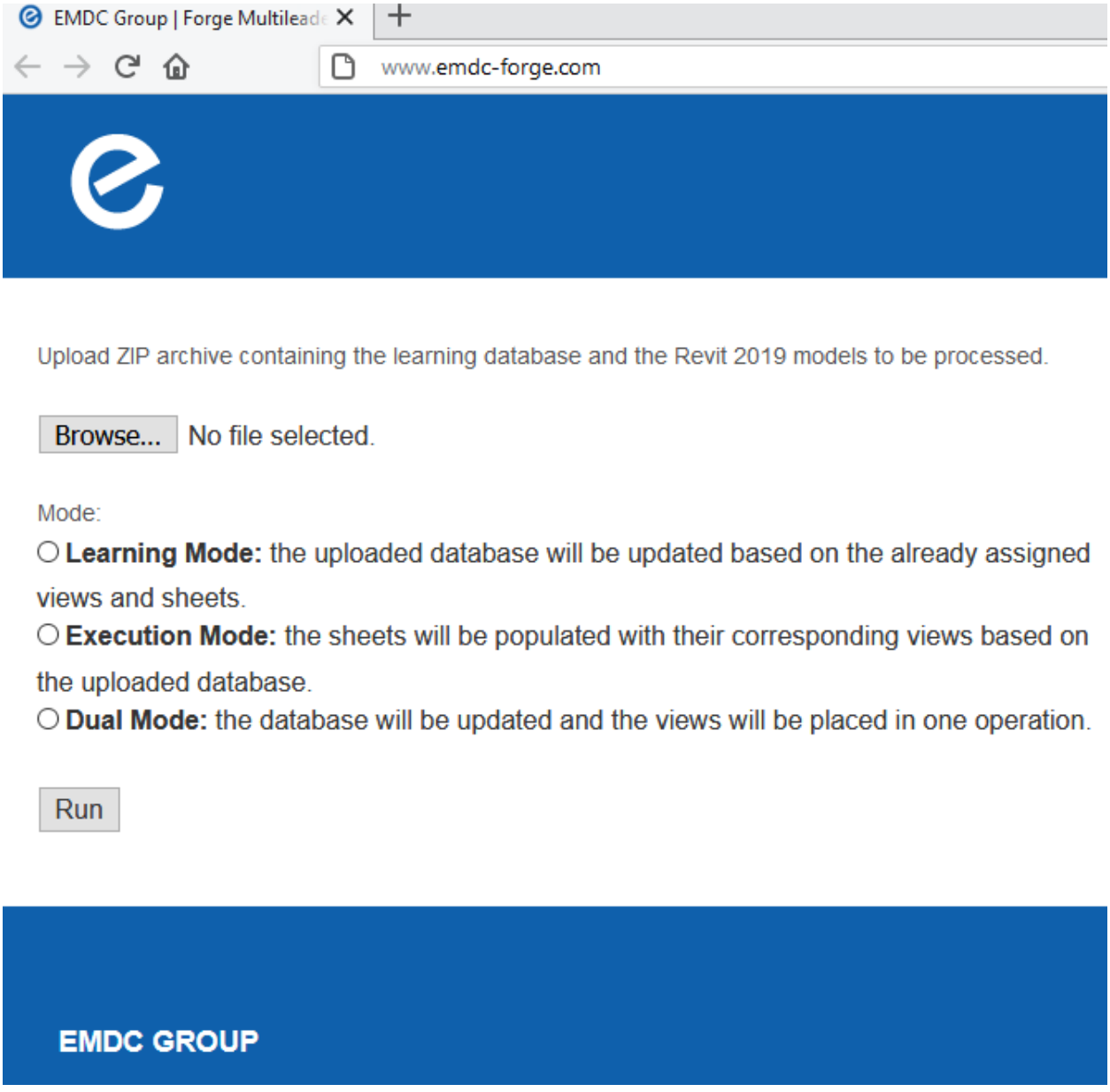


Inspired by our smartphones and the predictive text functionality that predicts the next word based on its meaning, which is due to a previously generated training set, a supervised learning algorithm was implemented, where previous models in which users had placed views on sheets before, are fed in. Keywords are automatically paired based on occurrence frequency and a training set has been generated for these keyword pairs. Now the score of each pair is no longer the same and depends on the probability of occurrence, which solves the scoring issues mentioned above.

Once again, why Forge?

Models contain thousands of sheets, especially for big projects, and for a useful database that can be reliable, it is expected to include 100s of models. Imagine doing that over a Desktop system, having to open and close each model manually. Not to mention how much computing power or runtime is involved with the training part.

Thankfully, with Forge, a user can package several models, upload them at once and have them run remotely without worrying about all that. When the resulting training set is generated, the user can access it from anywhere and at any time, as the web page is universally accessible. So considerable time-saving is involved and so many options reveal themselves.



Facts and Figures: The Realistic Benefits of AI-Powered Cloud Automation

The below is a comparison between Methods of Work and their impacts on the project's operations and hence time and cost:

- Method 1 is the Manual Method
  It wastes a lot of Time
  And it wastes human resources on repetitive tasks

- Method 2 is the Automated Method
  This method eliminates repetitive tasks.
  Reduces project's costs
  And Improves the quality of work
  But still requires manual intervention

- Method 3 is the AI Powered Method
  This method eliminates manual work further
  Saves even more time
  And Reduces cost even further
  But Requires more resources

- Method 4 is the FORGE Powered Method
  This method Automates file access operations
  And Provides a more powerful runtime environment
  With this method, the Results are also universally accessible

This progress in the methods of the works is the core idea of this class and is the best way to tailor for the accelerating environment we work in.

FORGE + Machine Learning: Stats & Figures

At EMDC group, using Forge with Machine Learning had and is still having considerable impact on the projects time and cost. As a summary, taking 2019 as an example, we are showing these figures.

35 projects were submitted during the entire year with some projects being bigger and more complex than others. On average, 2775 hours were expected to finalize each project without the use of automation, based on the time consumed during previous years when automation was not yet introduced.

However, as a real case scenario using Forge with Machine Learning, each project took around 1250 hours to be developed, leading to a time reduction of 1525 hours per project and hence 53,375 hours per year!

PROJECTS/YEAR

On average are executed by EMDC group

HOURS/PROJECT

On average is required when estimated with a full manual execution in mind

HOURS/PROJECT

Required when combining all automation tools built inhouse, machine learning powered automation processes to minimize manual, and forge cloud computing to minimize runtime

HOURS/YEAR

Of wasted time eliminated and used to handle more work

The above time gain led to an improved overall performance of the projects, both on the technical and financial levels, summarized with the following results:

- Considerable cost reduction
- And increased capability for more projects per year
- Hence, employees are now focused on decision-making tasks, improving their quality of work

And finally, the time saved is now reinvested in more research and development to cater for the continuous need to be improving and progressing and meet the market needs.

In fact, the previous decade was about mobile platforms, the current one seems to be about cloud computing and machine learning. Quantum computing started to become heard of and might be a common technology for the next decade.

Our job is to keep up and make the best out of these technologies, knowing that “if you always do what you've always done, you'll always get what you've always got” (Henry Ford).