Enable building system simulation by deriving a digital twin for operation from 2D P&ID plans

Hermann Mayer
Siemens AG, Munich, Germany

Abstract
We propose a method for generating semantics for 2D plans of building systems, predominantly P&ID plans of HVAC systems. For most buildings, even today designing the HVAC system and other building systems is not yet an integral part of a completely centralized 3D BIM modelling process. This applies even more to legacy buildings, which are regularly renovated for better energy performance. Therefore, most of system descriptions, like for HVAC, are still 2D drawings with no semantics included and no preparation for an automated simulation pipeline. The method introduced below shows how to get a machine-readable simulation model from those 2D plans by an easy-to-use, semi-automatic process.

Key Innovations
- Pattern recognition for P&ID plans
- Knowledge graph as basis for simulations
- Versioning of system models

Research Implications
Prefer classical pattern recognition over novel AI methods, since they provide a better confidence level. Use knowledge graphs as intermediate storage and for versioning of your model. Perform only one-directional conversions and be aware of the true source of data.

Introduction
The main challenge addressed in this project is the conversion of non-annotated 2D data (like plans for the layout of HVAC systems) into a semantically enriched format (like knowledge graphs). We assume the original data consists of drawing primitives, like lines and circles, which are already vectorized. This means the drawing primitives themselves are described in a parametric format (like providing only start and end points of a line, instead of describing plain pixels). If the latter formats are used (bmp and gif files), a vectorization would have to be performed before the below-mentioned methods can be applied. Vectorization methods are not subject of this paper, since there are already many of them available for usage, e.g. generalized Hough-transform in Duda (1972) or Canny edge detector in Canny (1986).

One approach for detecting complex objects, which are formed out of drawing primitives is deep learning, an AI method. The advantage of this method is that it can be applied to all sorts of data – even unvectorized pixel data. The user just presents examples of data and manually associates them with desired outputs (training phase). Afterwards, the neural network can derive own conclusions from previously unknown instances. However, when we tested this methodology, two main disadvantages emerged, which finally prevented us from going any further with this. The first is, deep learning methods require lots of training examples to perform well. This is particularly problematic in the case of HVAC and P&ID plans, which often come with individualized elements (like the pressurized air unit in figure 1), while only a few of them can be transferred between different plans (like standardized valves). Therefore, for each new plan type, the user would have to train the network again – a tedious task, since reusage of the results is hardly feasible. The other issue is reduced reliability of AI methods. While a recognition rate of 99% is regarded as more than sufficient for many applications such as face recognition, it remains problematic for our challenge. Large HVAC plans often contain thousands of elements (typically 5000 – 10000 for highrise buildings). Therefore, a success rate of AI based recognition of 99% still means we expect >50 elements to be detected wrong. Even worse, it cannot be analyzed easily which of the elements have been classified wrong.

Material and Methods
For showing the efficiency and generality of the developed method, we apply it to the HVAC layout of a highrise building comprising more than 5000 elements. Figure 1 shows one branch of the system in one floor.

Figure 1: Part of HVAC plan.
As a first step, all elements of the system need to be detected to assign their function and parameters for later exploitation (like simulation). Apart from the system elements, also some additional drawing elements need to be identified to close the piping circuit (like the texts “CHR/CHS” overlaying the pipes in figure 1). They will be ignored once the circuit is closed.

For describing the patterns, which are used to detect all elements of the same type, the user just selects a minimal set of primitives describing the elements. For example, to
Identify the “PAU” element in figure 1, it is sufficient to select the box and the enclosed arc. Those two elements and their parameters and relations will be stored as pattern. Parameters are kept as general as possible to allow for later application of patterns to transformed instances. Up to now we support all rigid transforms, i.e. translation, rotation, and scaling. Therefore, parameters are stored in percentual units instead of absolute values (e.g. for the box we store the ratio between height and length). As relation between elements, for example the size and position of the arc compared to the enclosing box is stored. We have implemented a rule-based pattern recognition method, where the rules are provided by graphical user input - just by manually selecting single instances of the relevant features of the patterns). Rule-based pattern recognition has also been proven effective by other authors like Bellet (2011). While we can already perform reliable detection of all instances on a plan with these patterns, further generalization is still work in progress. The goal is to express the patterns by means of a domain specific language, which is automatically derived when the user clicks on the examples. Note that it is only necessary to define exactly one example before the detection can start. This is a significant advantage over AI based methods as described above.

**Figure 2: After object detection and connection.**

As a next step, one of the detected instances is displayed for the user to check correct application. Optionally, the user can now define in- and outports for the element. Those will be used later to determine the flow direction in the pipes. While the flow direction can be also detected afterwards by a computer simulation, it is helpful for some services to have it already in this moment (e.g. for visualization and commissioning). Figure 2 shows in- and outports (green and red circles) for the detected elements. Afterwards all junctions in the connections between elements are detected fully automatically (in figure 2 the junctions are displayed as tiny red boxes). Junctions are important to derive the correct connectivity and flow direction in a later step. Also, for some disciplines like HVAC, junctions might be treated as separate elements, which have their own physical representation. In other disciplines, like electric planning, junctions are resolved later into n:m connections.

Given the correctly identified elements and ports, the system model can now be completed by introducing connections. This step still needs some user interaction: the user must click on an example of a line style, which is used for connections. In most cases this step could also be done fully automated, but optional user interaction provides more stability across different plan types. In principle, the connections are detected by a breadth first search, starting at an arbitrary collection of elements.

Figure 2 shows all elements of the system are connected correctly (blue color). However, sometimes the connection algorithm fails, due to errors in the original drawing. In this case, uncolored sections of the system remain and can be easily reviewed by the user. In general, the method follows an all or nothing principle (another advantage over AI methods, which only provide certain probabilities for detection confidence).

**Figure 3: Error detection capability.**

Typical errors, which can be detected by the method are shown in figure 3, from left to right: contradictory direction indicators (cyan arrows), copy & paste errors (same elements inserted several times), misconnected pipes, incorrect piping (green and red pipes interconnected).

After the system has been correctly detected, its elements can be transferred to an intermediate storage like a building twin, where additional services and applications can be built on top.

In general, elements of the HVAC system are assigned to a dedicated room or site. By doing so, they can be combined with other systems for the same entity (e.g. the HVAC system and the lighting system can be combined via the electrical system, which provides power for both of them). However, if the elements of a system cannot be assigned to an embracing building structure, a combination with other systems will hardly be possible and only stand-alone solutions for the underlying domain are feasible.

The intermediate representation in a digital building twin can also be used for versioning. For our example, we have used a knowledge graph as a backbone for the digital twin, which already comes with some versioning capabilities.

**Figure 4: Versioning of an HVAC system.**

Usually a new version is deployed in the original format (in this case 2D CAD drawings). The new version can now be converted again into the target format (knowledge graph DB). By recognizing identical elements by their parameters, the new version can be merged into the existing one. Figure 4 shows the situation of the HVAC system after one element was added. Note that adding one functional element induces the creation of two additional junctions, which are treated as separate nodes. There are still limitations in this generic versioning approach, as the right side of figure 4 reveals additional transitive connections, which should have been removed by a proper merging process.
Once a cleaned-up version of the knowledge graph of a building system is available, it can be converted into a 2D simulation model (like Simulink or Amesim) by directly converting the nodes of the graph into entities of the simulation system. This is done by an application-specific mapping exploiting the API or file format of the underlying simulation system. Converting via the knowledge graph as common data model also avoids n:m conversions between specific models.

Conclusion and Outlook
We have presented a method to derive a semantically enriched model out of standard 2D system drawings. The model can be used for versioning the system or for later system simulation. Additional challenges have been identified for further research (like defining a domain-specific language for pattern definition or providing context-aware merging of different system versions.

In the future, this procedure will be used to digitize legacy plans of buildings, which is a high-potential market (today it’s estimated that more than 90% of all buildings do not come with digitized plans in a sense that those can be directly processed by computers). Digital plans of buildings are particularly helpful if the asset is renovated to reduce its energy consumption. Simulation models based on digital plans will help to identify the most effective measures to reach requirements for energy savings. Reducing the energy consumption of existing buildings is a major contribution to a carbon-neutral society in the future.

References
