From Informal Sketches to System Engineering Models using AI Plan Recognition

Opportunities and Challenges

Alexandre Albore and Nicolas Hili

IRT Saint-Exupéry, 3 Rue Tarfaya, CS 34436, 31400 Toulouse, France
first.last@irt-saintexupery.com

Abstract. Despite its proved modeling value in many engineering domains, Computer-Aided Design (CAD) tools fail to convince and to be accepted by system engineers and architects to assist them in their day-to-day tasks. The complexity of creating, editing, and annotating models of system engineering takes its root from different sources: unsuitable representations, outdated interfaces, laborious modifications, and difficult collaborations. As a result, system architects still heavily rely on traditional methods (whiteboards, papers, and pens) to outline a problem and its solution, and then they use modeling expert users to digitize informal data into modeling tools. In this paper, we present an approach based on Artifical Neural Network (ANN) and automated plan recognition to informally capturing sketches of system engineering models and to incrementally formalizing them using specific representations. We present a first implementation of our approach and we discuss the opportunities and challenges of applying plan recognition to system engineering.

Keywords: System Engineering · Model-Based System Engineering · Artificial Intelligence · Plan Recognition.

1 Motivation

Despite its proved modeling value in many engineering domains, Computer-Aided Design (CAD) tools have only a moderate acceptance by system engineers and architects to assist them in their day-to-day tasks [Robertson and Radcliffe, 2009]. The complexity of creating, editing, and annotating models of system engineering takes its root from different sources: unsuitable representations, outdated interfaces, laborious modification, and difficult collaboration [Rudin, 2019].

As a result, especially in the early development phases, system architects tend to favor more traditional tools, such as whiteboards, paper, and pencils, over CAD tools to quickly and easily sketch a problem and its solution. Among the different benefits of remaining with traditional tools, whiteboards foster collaboration and creativity as the users do not need to strictly conform to a formal notation.

A common pitfall for using traditional tools, however, is that human users are required to reproduce any sketched solutions inside formal tools when it comes to

formalizing them. Modern post-WIMP[1] interfaces (e.g., electronic whiteboards) could help to automatize this task by allowing users working on a digital representation of the model that can be directly exported to be modified via modelling tools. Bridging the informality of the working sketches captured on interactive whiteboards with formal notations and representations, has the potential to lower the barrier of acceptance of CAD tools by the industry [Botre and Sandbhor, 2013, Alblawi et al., 2019]. This acceptance can be obtained by automatically or semi-automatically translating informal sketches into their corresponding elements using a specified formal notation.

One trend is to rely on Artifical Intelligence (AI) and more specifically on Machine Learning (ML) techniques to recognize modeling elements based on Artifical Neural Network (ANN) algorithms. Such algorithms typically involve two phases. During the training phase, algorithms are trained to recognize elements based on pre-existing libraries. During the recognition phase, they can identify elements with a certain degree of confidence.

While broadly used for recognizing medical images, this technique is inappropriate for system engineering for two main reasons. Namely, the similarity between graphical symbols representing model elements in standard modelling languages, e.g., UML [Moody, 2010], and the lack of explicability of the ML solution [Rudin, 2019].

"Explicability" is the property of a system that provides an output that makes understandable to the human user the reasons of an algorithm's choice. This is a barrier to several ML applications, but it is a condition needed by any process-directed tool that allows the users to evaluate the criteria behind a choice to use the tool efficaciously [Rosenfeld and Richardson, 2019].

In this paper, we propose an approach to CAD tools for system engineering that makes use of AI solutions that quickens model building by providing an automated assistant that performs shapes completion, while providing a certain level of explicability. This approach combines traditional shape recognition algorithms and *planning recognition* (to forecast the final shape and to provide the with user a selection of recognition choices).

The remainder of this paper is structured as follows: Section 2 presents the related work; Section 3 presents our approach and the details of a preliminary implementation; Section 4 identifies some challenges to address; and Section 5 concludes this paper and details future work.

2 Related Work

Incremental formalization aims at bridging the gap between free-form modeling and formal representation using dedicated graphical notations. In software engineering, modern post-WIMP interfaces, such as interactive whiteboards, interactive walls, and large multi-touch screens, have been used to capture models in software engineering during the first stages of the design process.

[1] Windows, mouse, and pointer interfaces.

OctoUML [Jolak et al., 2016, Vesin et al., 2017] is a prototype of a modeling environment that captures UML models in a free-form modeling fashion and in a collaborative way. It can be used on various devices, including desktop computers and large interactive whiteboards. Sketches are then converted into a graphical UML notation. OctoUML supports class and sequence diagrams. It uses a *selective recognition* algorithm to support an incremental formalization, although no detail about the implementation of an algorithm is provided.

MyScript [MyScript, 2020] is a leading company in the domain of handwriting recognition that developed a prototype of a diagramming tool[2] to formally recognize graphical elements in structural and flow diagrams. Recognition is performed remotely on the full diagram all at once after a few seconds to provide a result. Incremental formalization is not supported and the user has to switch between an editable, informal representation and a non-editable, formal one.

AI planning has been used to perform activity recognition in the context of a system managed by with human operators, whose currently pursued goal has to be determine [Hollnagel, 1988]. Several plan recognition [Kautz and Allen, 1986] fields of application surfaced, including "operator modelling" to improve the efficiency of man-machine systems. Early applications of the approach failed because of the complexity of plans, and issues due to evaluating actions that did not fit any plan, or even from interleaving planning and execution. Moreover, the work in plan recognition has historically proceeded independently from the planning community, using handcrafted libraries rather than planners [Avrahami-Zilberbrand et al., 2005].

To quicken the development of models of system engineering, we have implemented a web-based modeling environment (see Fig. 1) that easily captures functional models of systems on large interactive screens. At first, we attempted to rely on ANN algorithms to define a library of model elements that our tool

[2] A web demo is available here: https://webdemo.myscript.com/views/diagram/

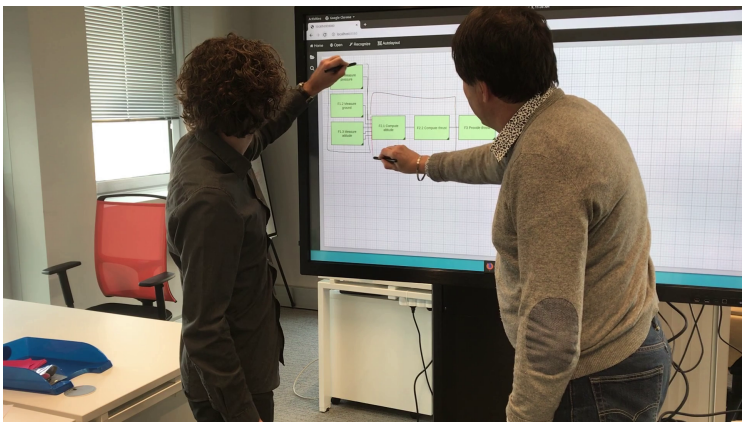

Fig. 1. Overview of our modelling environment.

could recognize based on user inputs. While the approach worked for few sets of model elements, the number of errors increased proportionally with the number of model elements that had to be recognizes because of the similarity between the different graphical symbols. Besides, since the ANN provided no way to interpret the result, the user was often clueless when attempting to understand why a shape had not been correctly identified.

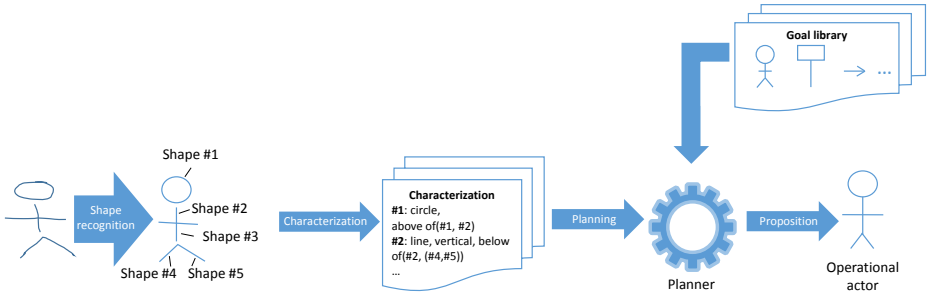

Fig. 2. Recognition process overview.

3 Approach Overview

Fig. 2 gives an overview of our recognition process. It involves four automatic phases: *shape recognition*, *characterization*, *planning*, and *recognition*. The entry point of the process is a sketch manually drawn by human users on a digital whiteboard. This sketch will constitute the drawing of a model element to be recognized by the process.

During the first step (*shape recognition*), the process extracts elementary shapes out of the sketch using traditional shape recognition algorithms. Currently, our algorithms recognize circles and polylines. A polyline consists of a series of connected straight line segments. To facilitate the characterization and planning phases, a polyline is not characterized as a whole, but instead, each segment composing the polyline is characterized individually and independently of the others.

Once a sketch is split into a set of elementary shapes, the *characterization* phase consists in characterizing each shape individually. Additionally, every shape (being a circle or a straight line) is characterized by its position with respect to all of the other shapes composing a sketch. For example, in Fig. 2, `Shape #1` is characterized as being a *circle*, located *above* `Shape #2`, a polyline.

After this phase, our task is to identify 1) the model element being drawn and, 2) the relation between this element and the rest of the drawn model. These tasks do not appear to be easily solvable by ML techniques based on an ANN. This is performed during the *plan recognition* phase.

Automated AI planning [Ghallab et al., 2004] is a model-based approach to the task of selecting and organizing actions in a sequence called *plan*, with the aim of fulfilling high-level objectives called *goals*. Here the task is to identify the shapes yet to be drawn, and their placement to create a meaningful system-engineering sketch from an initial shape. For our sketching tool (see Fig. 1), we adopt a planning-based approach that uses the planning framework to perform the converse task of automated planning, i.e., recognizing the most probable goal given an initial state and a plan. We thus follow an approach to plan recognition using a *goal library* to describe the possible solutions of a plan [Ramírez and Geffner, 2009], given a model of the domain that a hypothetical drawing-agent would evolve. Using plan recognition to characterize how far along a drawn set of simple shapes is from a complete sketch of a model element used in system-engineering design gives us the ability to provide users with an interpretation of the planner's results.

More formally, a STRIPS planning problem (the model of the environment and the planning task) is described by a tuple $P = (F, I, A, G)$, where F is a set of fluents, where $I \subseteq F$ and $G \subseteq F$ are sets of fluents describing the initial and goal situations respectively.

In our approach, each simple shape drawn with the CAD tool on the digital whiteboard – e.g., line, circle – corresponds to a fluent in F. More fluents are added to F to describe the spatial relationships between simple shapes (e.g., *above(x,y)*, *left_to(x,y)*). The set of actions A in P represents the addition (or removal) of a shape plus its relative position to the shapes present in a whiteboard situation. A situation (or a state in the graph where actions represent the edges) is a set of such fluents representing a sketch. A plan π is then a solution plan when, starting from I, the transformations induced by the sequence of actions in π lead to a goal state G.

A plan recognition problem is given by a tuple $R = (P, \mathcal{G}, O)$, where $P = (F, I, A)$ is the planning problem stripped of its goals, $\mathcal{G}$ is a *set* of possible goals $G \subseteq F$, and $O = \{o_i\}_n$ is a sequence of n observations such that $o_i \in A$. $\mathcal{G}$ then corresponds to a pre-compiled set of meaningful system-engineering sketches.

Without entering into the details of the plan recognition algorithm, found in [Ramírez and Geffner, 2009], the planning setting is used to calculate the probability of having a planner reach the goals in $\mathcal{G}$, given a plan recognition problem R. The implicit information here is that a planner will more likely calculate shorter plans, avoiding plans with a length greater than needed.

The planning phase outputs a set of reachable goals from the current state and the distance between the state and the goal. Elements described by goals are recognized in ascending order of distance from the current state. The decision can be taken automatically by the planner or semi-automatically by proposing that the user selects among the three most closest goals computed. This second solution is similar to the decision taken by Google Autodraw ; it increases explicability as the user comes to understand which subsets of each of the three graphical elements have been recognized that led the planner to propose the different solutions. Graphical feedback could also be provided to the user.

4 Challenges

We have currently implemented part of the system described above. More specifically, we have implemented the first two phases, along with a handcrafted implementation, of the plan recognition algorithm as a proof of concept. Future work includes adapting the plan recognition algorithm and goal library to our task, and addressing the following challenges identified below.

Classes of Equivalence Existing planning recognition approaches consider a finite set of states leading from an initial state to a goal. In the case of model recognition, some graphical elements have potentially an unbound set of states. For example, a polyline may be sketched to model a connector between two distant node elements in a diagram the same semantic value (i.e., connecting two elements) ; the possible representations with polylines can be multiple.

These scenarios may require users to define classes of equivalence, and do provide reasoning at a semantic level.

Fault tolerance recognition Plan recognition is sensitive to errors due to the interaction with physical devices. For example, the user may unintentionally lift the pen when drawing on an interactive touch screen. Our process would then detect two distinct shapes and incorrectly match the drawn sketch to the wrong model element. ANN techniques are less prone to this type of error since only the final result (a 2-dimensional array of pixels) is used for recognition, and not the series of user actions that lead to a result. Fault tolerance is limited in existing plan recognition approaches, while being central for human–agent systems.

5 Conclusion and Future Work

In this paper, we have proposed an approach to translating informal sketches into model elements using a formal notation. We have decomposed our approach into two main components. First, simple algorithms are used to detect elementary shapes (circles, lines) from a sketch drawn by human users; second, plan recognition techniques are used to recognize model elements based on preset libraries of goals. Compared to existing techniques based on ML only, this approach increases the explicability and interpretability of the results by end users.

One major objective of this work is to provide explainable results to the users, which is a central property in human–agent systems [Rosenfeld and Richardson, 2019]. In fact, the order of the selection for the choices has to be motivated for a better usage of the tool. Splitting the approach in two – on one side, the shape recognition, and on the other side, the symbolic planning application – permits us to provide users with the knowledge of a system through the elicitation, for a specific goal, of the causal structure behind the performed task. The human decision is then supported by the interpretability of the algorithm results.

We have presented a preliminary implementation; it will be integrated in our existing web-based modelling environment for system engineering, and discussed potential challenges will be addressed.

As future work, we plan to fully implement our approach described in this paper for our tool. The plan recognition algorithm we have proposed in this paper is *hand-crafted*; it should be replaced with the algorithm described in [Ramírez and Geffner, 2009]. The next step is to consolidate the recognition phase with respect to the challenges we have identified.

References

[Alblawi et al., 2019] Alblawi, A., Nawab, M., and Alsayyari, A. (2019). A system engineering approach in orienting traditional engineering towards modern engineering. In *2019 IEEE Global Engineering Education Conference (EDUCON)*, pages 1559–1567. IEEE.

[Avrahami-Zilberbrand et al., 2005] Avrahami-Zilberbrand, D., Kaminka, G., and Zarosim, H. (2005). Fast and complete symbolic plan recognition: Allowing for duration, interleaved execution, and lossy observations. In *Proc. of the AAAI Workshop on Modeling Others from Observations, MOO*.

[Botre and Sandbhor, 2013] Botre, R. and Sandbhor, S. (2013). Using Interactive Workspaces for Construction Data Utilization and Coordination. *International Journal of Construction Engineering and Management*, 2(3):62–69.

[Ghallab et al., 2004] Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: theory and practice*. Elsevier.

[Hollnagel, 1988] Hollnagel, E. (1988). Plan recognition in modelling of users. *Reliability Engineering & System Safety*, 22(1-4):129–136.

[Jolak et al., 2016] Jolak, R., Vesin, B., Isaksson, M., and Chaudron, M. R. (2016). Towards a new generation of software design environments: Supporting the use of informal and formal notations with octouml. In *HuFaMo@ MoDELS*, pages 3–10.

[Kautz and Allen, 1986] Kautz, H. A. and Allen, J. F. (1986). Generalized plan recognition. In *AAAI*, volume 86, page 5.

[Moody, 2010] Moody, D. L. (2010). The "Physics" of Notations: A Scientific Approach to Designing Visual Notations in Software Engineering. In *2010 ACM/IEEE 32nd International Conference on Software Engineering*, volume 2, pages 485–486. IEEE.

[MyScript, 2020] MyScript (2020). Myscript home page. https://www.myscript.com/. Accessed: 2020-01-15.

[Ramírez and Geffner, 2009] Ramírez, M. and Geffner, H. (2009). Plan recognition as planning. In *Twenty-First International Joint Conference on Artificial Intelligence*.

[Robertson and Radcliffe, 2009] Robertson, B. and Radcliffe, D. (2009). Impact of CAD tools on creative problem solving in engineering design. *Computer-Aided Design*, 41(3):136–146.

[Rosenfeld and Richardson, 2019] Rosenfeld, A. and Richardson, A. (2019). Explainability in human–agent systems. *Autonomous Agents and Multi-Agent Systems*.

[Rudin, 2019] Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206.

[Vesin et al., 2017] Vesin, B., Jolak, R., and Chaudron, M. R. (2017). Octouml: an environment for exploratory and collaborative software design. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 7–10. IEEE.