BabyMOD, a Collaborative Model Editor for Mastering Model Complexity in MBSE

Nicolas Hili, Patrick Farail
IRT Saint Exupéry, 3 Rue Tarfaya, CS 34436, 31400 Toulouse, France
{nicolas.hili, patrick.farail}@irt-saintexupery.com

***Abstract*—Modelling is nowadays commonly practiced by system architects. However, it remains a difficult task that requires some advanced User Interface (UI) modelling tools to ease the design of large-scale models. BabyMOD is an interactive and collaborative model editor for mastering model complexity in system engineering. It combines three objectives: visualizing models of systems imported from authoring tools ; reviewing imported models through model annotations ; and editing existing models or creating new models from scratch. In this paper, we present a work-in-progress prototype and discusses some original features, including sketch recognition and enhanced visualization through auto-layout and animation.**

***Keywords*—System Engineering, Model-Driven Engineering, Model-Based System Engineering, Interactive Whiteboards.**

I. 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editing, and annotating models of system engineering takes its root from different sources: unsuitable representations, outdated interfaces, laborious modification, and difficult collaboration [Rudin, 2019].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.

[1]Windows, mouse, and pointer interfaces.

In this paper, we present BabyMOD, a web-based model editor featuring a lighweight and intuitive interface for editing and annotating models of systems in a collaborative way. BabyMOD positions itself between interactive electronic whiteboards for sketching diagrams and model editors. As such, it shares common similarities with other academic and industrial projects, such as OctoUML [Jolak et al., 2016], [Vesin et al., 2017] and MyScript [MyScript, 2020], but it also distinguishes itself from them on various points, including its language-agnostic sketch recognition assistant and its editing and visualization capabilities.

II. BabyMOD Overview

BabyMOD (see Fig. 1) is a web-based multi-modal model editor that has been developped from the ground to adapt itself on different devices equipped with modern browsers. It can run not only on traditional devices, including laptops and PCs, but also on tablets equipped with active stylus, and large multi-touch screens. As such, it intends to cover a large spectrum of scenarios, from single-user modelling to multi-user collaborative reviewing.

BabyMOD targets three main objectives: visualizing models imported from authoring tools, editing the imported models, and reviewing them through model annotations. Minor changes to the imported models can be carried out directly in BabyMOD, but most of the cases, heavier changes will be carried out inside the authoring tools (e.g., Capella). As such, BabyMOD does not intend to replace existing authoring tools (e.g., Capella), but rather complement them with enhanced visualization and interaction features.

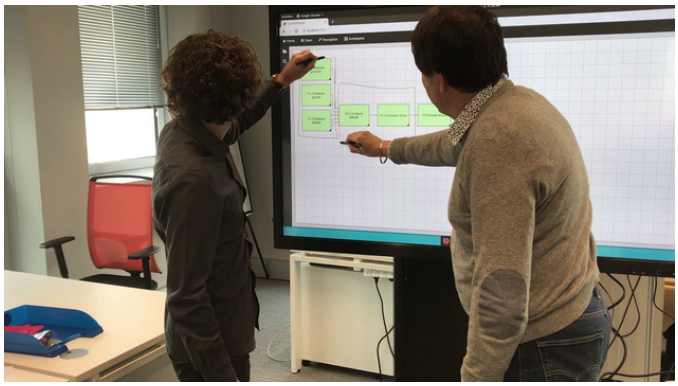

Fig. 1. BabyMOD running on a multi-touch screen where two users are collaboratively editing a functional model.

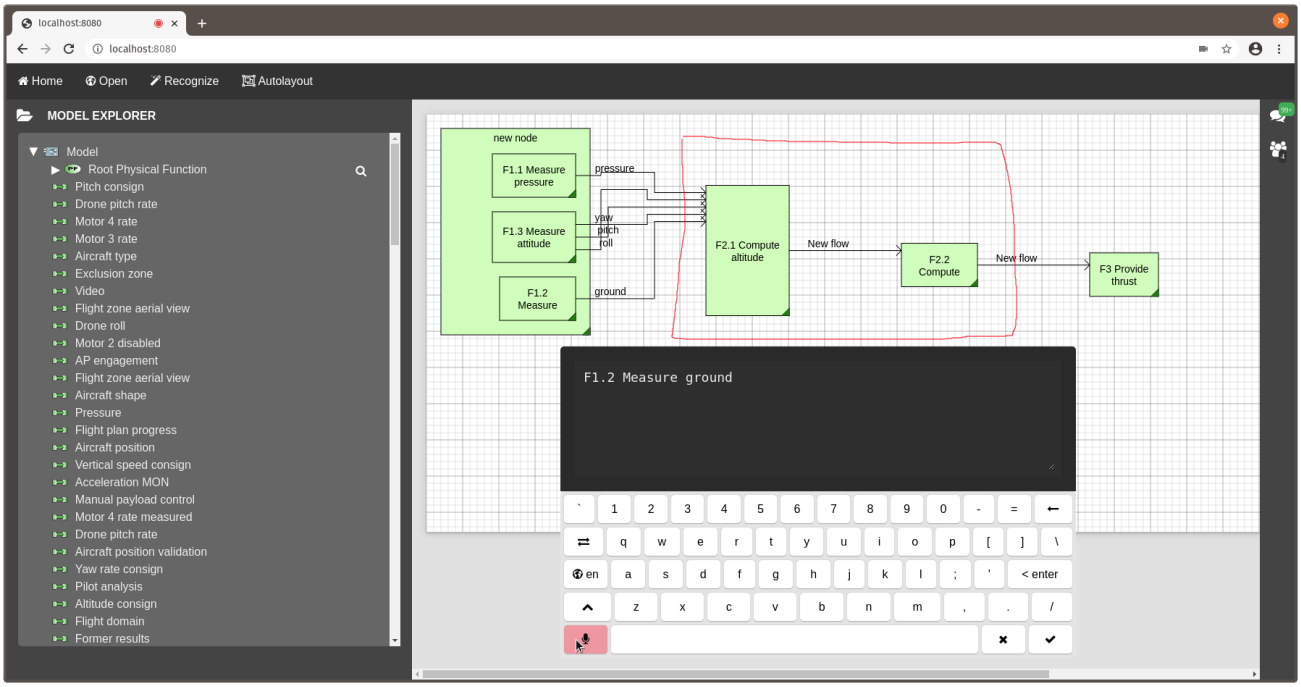

Fig. 2. Overview of the BabyMOD interface: a model explorer (left-side) allows the user to explore the model hierarchically while a graphical editor (right-side) allows him/her to visualize it and to edit it in a freeform way. Model element's properties can be edited through virtual on-screen keyboard and voice recognition.

The originality in BabyMOD lies in its sketch recognition assistant that allows multiple users to edit or annotate models in a free-form modelling way by sketching elements on an interactive whiteboard (see Fig. 1). Sketch recognition is performed in real-time or on-demand and provides the user with explicable outputs in the form of a selection of choices.

Finally, BabyMOD supports basic editing features, allowing users to add new model elements into the model, remove existing model elements, and modify model element's properties. A screen cast of our current implementation and the different features it provides is available online.[2]

III. Implementation

We implemented BabyMOD using Web technologies (JavaScript and HTML5), standardized Web APIs, and open source third-party libraries, so that it makes it easer to deploy it and to run it on different devices without any prior set-up. The core element is the interface (see Fig. 2) that mainly consists of i) a Canvas-based area for visualizing and editing models graphically ; and ii) a model explorer to display models hierarchically. Fig. 3 shows the architecture of BabyMOD. Besides the core element, BabyMOD relies on different assistants, including sketch and text recognition assistants to recognise hand drawn model elements, a virtual keyboard, an auto-layout algorithm to efficiently render models, and an assistant to import models from existing tools.

[2]https://youtu.be/VRSxZr0VjKQ

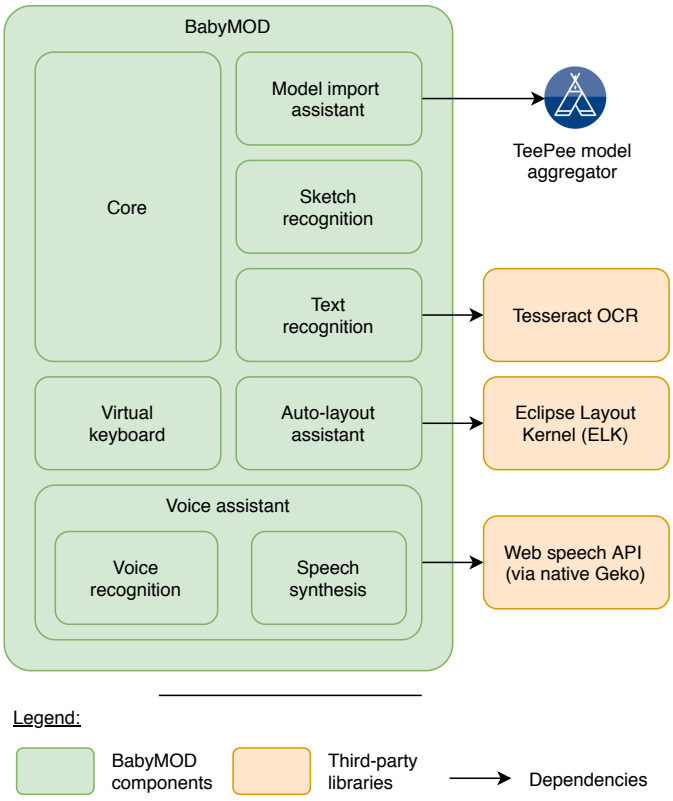

Fig. 3. Overview of the BabyMOD architecture: model import capabilities rely on the TeePee model aggregator [Baclet, 2019a].

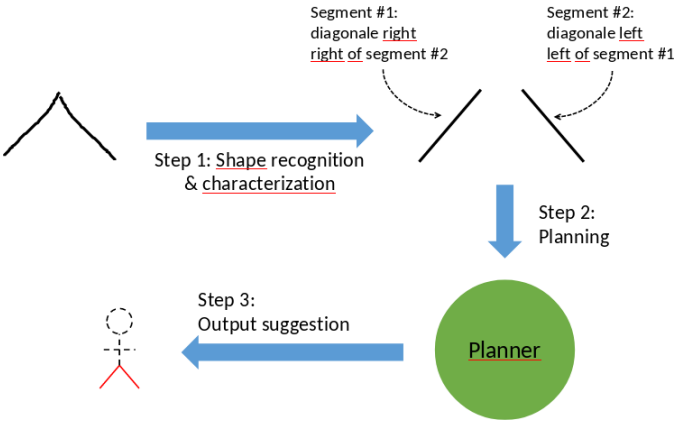

Fig. 4. Overview of the recognition approach [Albore and Hili, 2020]: first, elementary shapes from a partial draw are recognised and characterised ; Second, a planner compares the characterised elements againts a set of goals ; Third, the planner provides te user with explicable suggestions.

A. Sketch and Text Recognition Assistants

We implemented our sketch recognition assistant based on an approach of automated Artifical Intelligence (AI) planning [Ghallab et al., 2004] called *Plan Recognition* [Ramírez and Geffner, 2009], that consists, given an initial state and a plan, in recognising the most probable goals to reach. We adapted this approach to our domain where the initial state represents a partial draw on the interactive whiteboard initiated by a user, and a *goal library* describes the set of possible solutions in the form of model elements the user may want to draw. Fig. 4 illustrates the plan recognition approach. Details of the approach and an initial implementation are given in [Albore and Hili, 2020].

In addition to the sketch recognition assistant, text recognition, virtual on-screen keyboard, and voice recognition assistants are provided to the users to edit the different properties of model elements (e.g., the name of a function). The text and voice recognition assistants respectively rely on the Tesseract open source Optical Character Recognition (OCR) engine[3] and the standardized Web speech API[4] available in most recent web browsers. The virtual keyboard is available by the user after selecting a model element in the editor. Voice recognition can be used in different languages and the user may optionally modify the output of the voice recognition assistant using the virtual keyboard if the result is unsatisfactory due to environmental noise or wrong pronunciation.

B. Model Representation and Auto-Layout

BabyMOD supports one type of graphical representation of models as graphs, i.e., graphical representation composed of nodes (possibly hierarchical) and links connecting nodes (directly or through their ports). To efficiently display graph models on the screen, BabyMOD uses Eclipse Layout Kernel (ELK) to automatically position model elements in an optimal way preventing model elements from overlapping and efficiently routing the different links between the model elements. Auto-layout is complemented with animation to progressively show the result of the application of a new layout so that users are not confused by a sudden change of their models.

C. Model Import Assistant

BabyMOD relies on prior results obtained in the MOISE project conducted at IRT Saint Exupéry [Baclet, 2019a]. *TeePee* is a model aggregator with import capabilities to import fragments of models from various off-the-shelf modelling editors (Capella, Cameo Systems Modeler, . . .) and documents (Excel spreadsheets) and aggregate them in the context of Extended Enterprises (EEs) [Baclet, 2019b]. TeePee relies on an internal data representation called *SEIM*[5] to provide a unified representation of models imported from various sources. Fig. 5 illustrates the model import capabilities using TeePee. First, TeePee converts the model into SEIM, then the model can be visualized and edited within BabyMOD. Exporting back the model into the authoring tools is currently not implemented and is a planned activity.

As only an abstract representation of a model is imported into BabyMOD, BabyMOD cannot carry specific changes requiring to have the full knowledge of the internal modelling language of every off-the-shelf editor. One benefit, however, is that users can efficiently focus on editing and reviewing tasks without being distracted by manipulating intricate concepts from the source modelling languages.

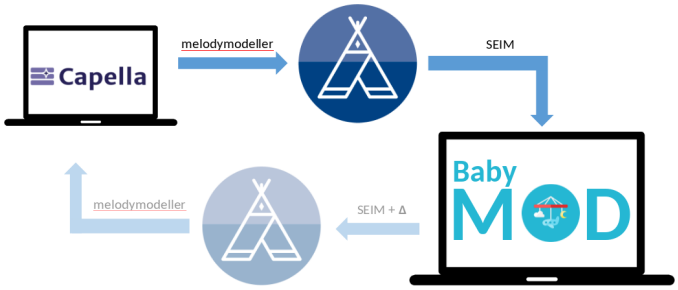

Fig. 5. Illustration of the model import facility using the TeePee model aggregator [Baclet, 2019a] for a Capella model. TeePee supports models from different sources, including Capella (melodymodeller files), Cameo Systems Modeler (Teamwork Cloud), Excel (spreadsheet files), etc.

IV. Conclusion and Future Work

In this paper, we presented BabyMOD, a work-in-progress tool for collaboratively visualizing, editing, and annotating models of system engineering. BabyMOD is a preparatory work for EasyMOD, an IRT Saint Exupéry project planned to start in 2021. EasyMOD extends the perimeter of BabyMOD with: i) multi-site collaboration; ii) support for different types of model representations ; iii) a better integration of modelling assistants – including text and voice recognition –, multi-language modelling, incremental formalization ; and iv) workflow management system integration for model review.

[3]https://github.com/tesseract-ocr/tesseract

[4]https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API

[5]Systems Engineering Information Model

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.

[Albore and Hili, 2020] Albore, A. and Hili, N. (2020). From Informal Sketches to System Engineering Models using AI Plan Recognition. In *AAAI 2020 Spring Symposium Series*.

[Baclet, 2019a] Baclet, J. (2019a). Digital Continuity for MBSE – MOISE Project. In *13th European Conference on Software Architecture (ECSA)*.

[Baclet, 2019b] Baclet, J. (2019b). Model-Based Systems Engineering in an Extended Enterprise. In *SAE 2019 AeroTech Europe*.

[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.

[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.

[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.

[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.