Current Topics:
AI EE Development

Current State and Ideas for AI Usage in EE Development

Author:	Lucas Calvin Mannherz
Institution:	Hochschule der Medien, Stuttgart
Lecture:	Current Topics
Lecturer:	Prof. Dr. Andreas Koch
Date:	February 28, 2021

Abstract

This paper shows the current workflow of developing steps in Electrical Engineering (EE) from the moment an idea was born to the point where it got built in hardware. Available tools are presented and set into scene of the corresponding workflow. The possibilities and advantages for current Artificial Intelligence (AI) tools are worked out and at the end ideas are given for currently missing parts in the development flow.

Contents

List of Figures

Acronyms and Abbreviations

AI	Artificial Intelligence
CAD	Computer Aided Design
EE	Electrical Engineering
PCB	Printed Circuit Board
QA	Quality Assurance
RL	Reinforcement Learning

1. Introduction

The meaning of AI increases, usage and more different products appear on the market so new ways of earning money show up. Its application area is broad, reaching from language processing to autonomous driving and it gets involved in more applications the more the technology evolves. Beside making products for end consumers smarter it can be used in usually hidden areas which are not directly visible in the end product. The way a product goes through until it gets released for the end consumer is long and usually complex, especially when engineering and technologies are used. AI tools can help out on steps between the idea generation and releasing the product. With new tools it is possible to accelerate the process of developing and save money and time by increasing the productivity of the engineers or replace certain steps by automated processes completely. This paper describes a typical EE process for making an electronic device and points out possibilities where new tools can have a huge impact regarding the product quality, time and economical aspects.

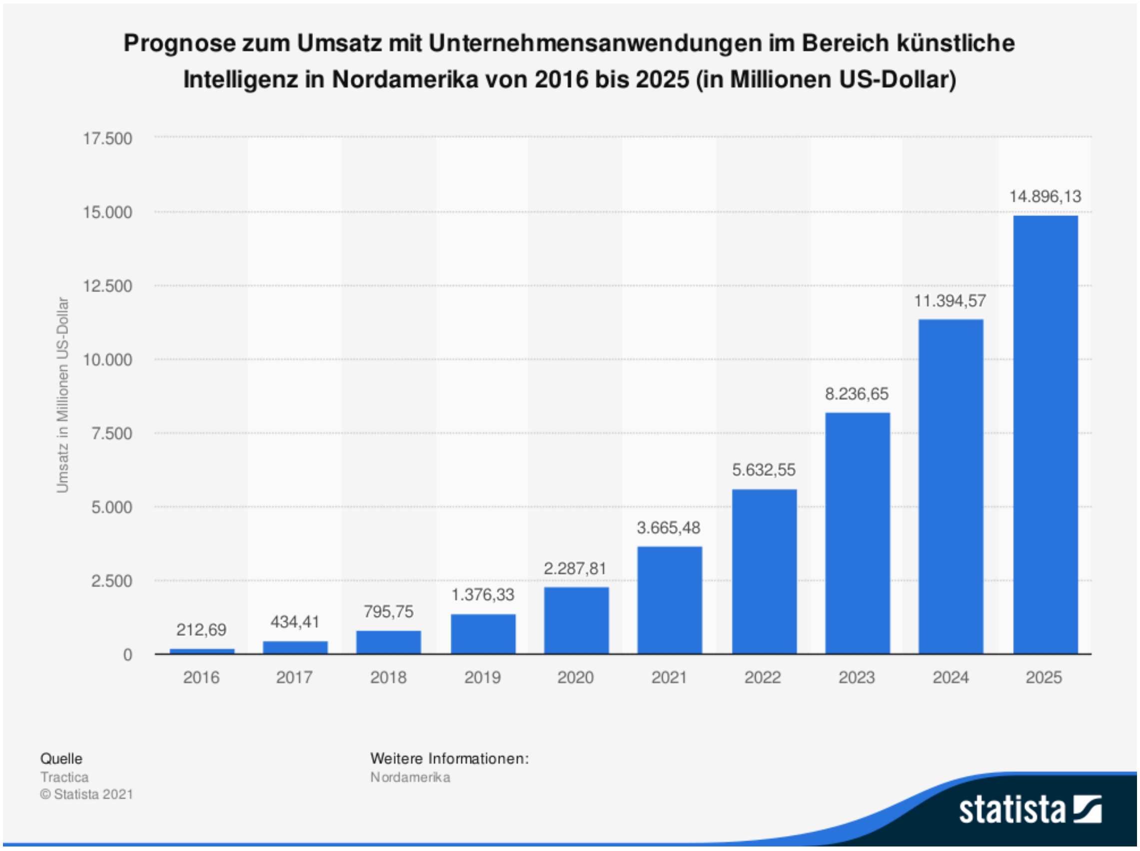

Figure 1.1: AI revenue prediction for north America
Source: Statista, [11]

2. Current EE creation workflow

Figure 2.1 shows a strongly abstracted workflow which is usually an iteration of the process in creating an electronic device. The order might differ depending on the product and the used methods, as well as the orders of these steps are not fixed and can differ for each context. Smaller iterations inside the "big outer" iteration are possible. For example, when the simulation returns bad results it can happen that the schematics need to be adjusted and simulated again. These two steps are therefore usually being worked on at the same time in reality due to its strong dependence to each other. The figure gives an idea of common steps in the workflow and not about fixed sequences.

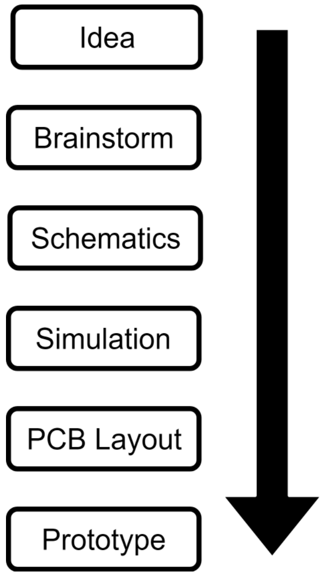

Figure 2.1: Current workflow

Idea The idea (vision) for a product is born with a basic feature set which should get implemented in the product

Brainstorm In this step technological and architectural decisions are made, the outcome can look like quick drawings, descriptions and circuits drawn on paper.

Schematics Transform the paper prototypes into a CAD format so computer tools can be used.

Simulation Use the CAD schematics for simulations in order to specify the part attributes in detail. Functionality and edge cases can be tested.

PCB Layout Use the CAD files to generate PCB files, so the layout can be produced. The logical schematics are being transformed into a "real world" circuit.

Prototype The previously generated PCB files get sent to a board manufacturer. After assembling the board with its parts the hardware can be tested, the prototype hardware is complete at this point.

3. KiCad

KiCad is an open source and therefore free tool for drawing circuits and design the PCB layout [7]. Beside economical aspects it is available for all three well known operating systems (Windows, macOS and Linux) so the user is not bound to an operating system predefined by the developers which gives another huge advantage over other commercial available products [8].

After the prototype ideas are drawn on paper the user starts to transform them into a program like KiCad. As an example for this paper a custom motor driver for the game "Looping Louie" (for more information about the game see appendix A) is taken. In order to make the game more interesting the plane should be able to act on its own by implementing new actions, for example flying in the reverse direction or stop suddenly, the actions it performs are random or controlled by another person in later versions. The different actions and parameters are realized via software which controls the motor driver.

Figure 3.1 shows the basic sheet with its three main parts, the microcontroller with the logic (STM32 on the left hand side), the power supply in the middle and the motor driver on the right hand side.

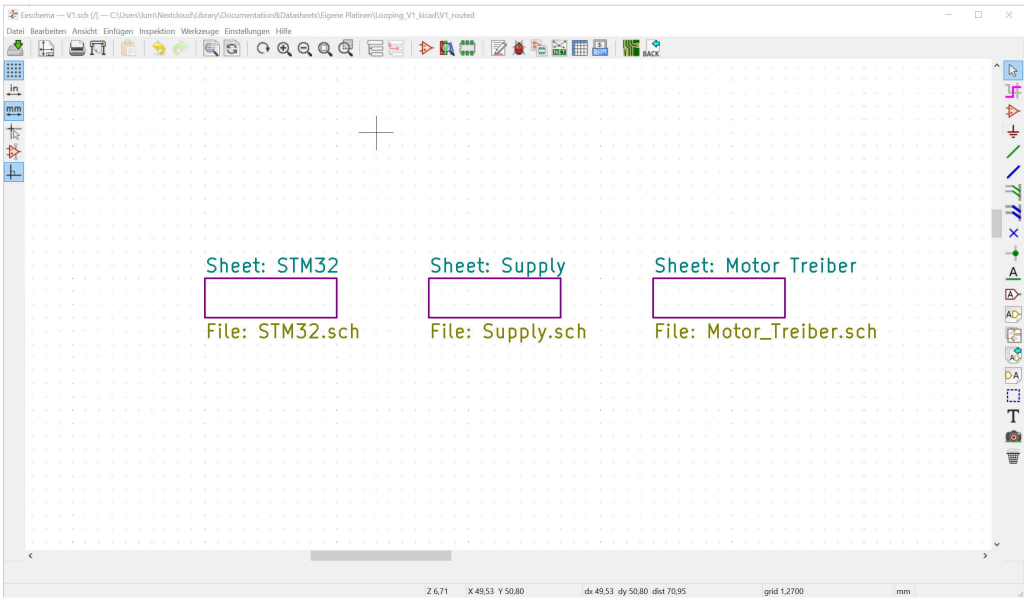

Figure 3.1: KiCad board sheet

The power supply sheet is shown in figure 3.2 in detail. The labels connected to the lines result in a net, so the components in the three split sheets are logically connected through the nets. Figure 3.3 shows the PCB viewer of KiCad. The white lines show the nets from the previous picture and are not physically connected yet. After placing the parts manually (that already happened in this picture) the engineer usually starts with routing the nets so the parts get connected. This step is rather time consuming and repetitive so the following section presents the first possible usage of AI tools.

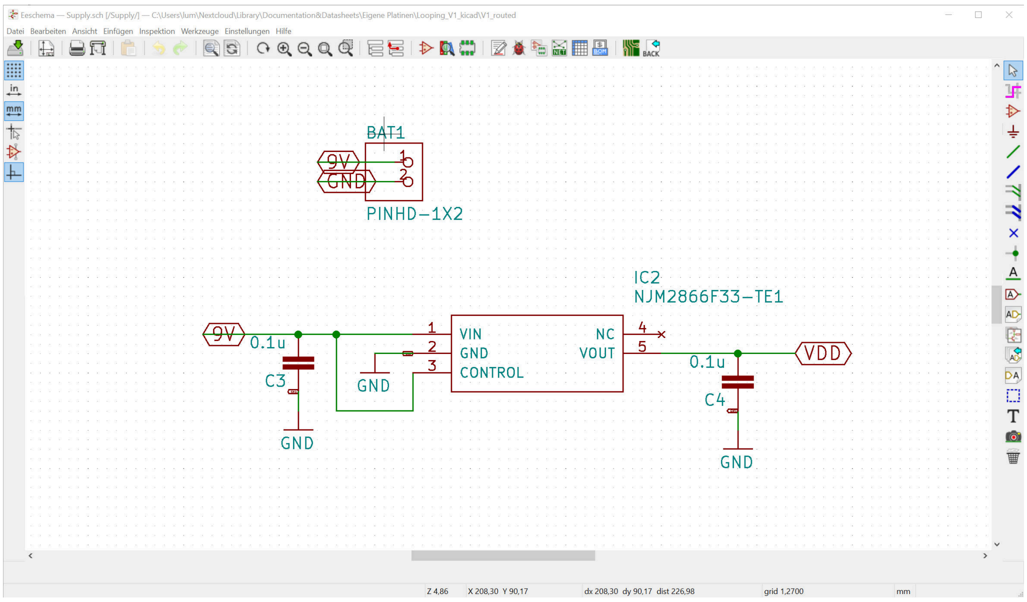

Figure 3.2: KiCad power supply sheet

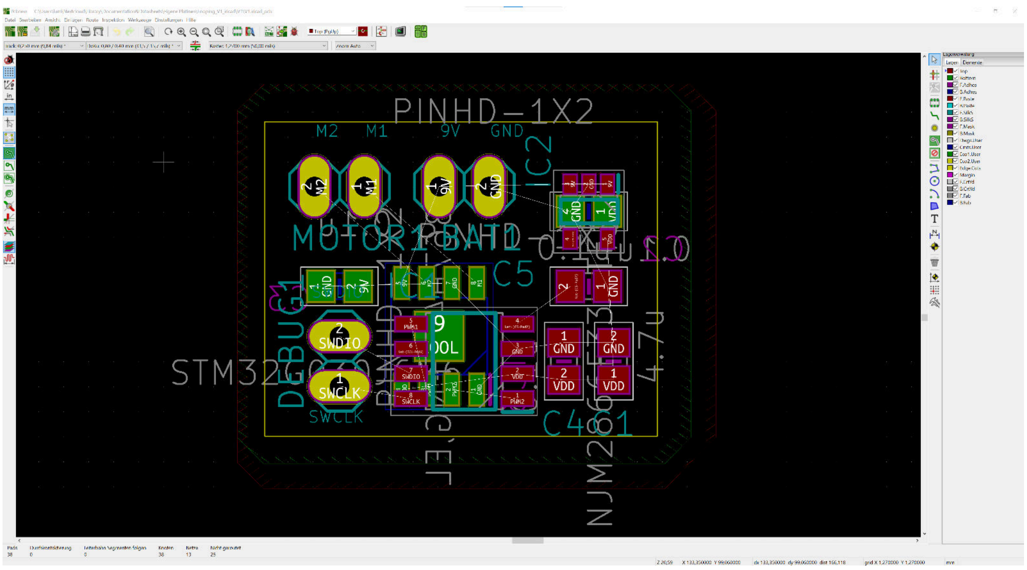

Figure 3.3: KiCad PCB view

In terms of simulation KiCad offers the integration of NGSPICE so "classical" simulation is possible in this program [9]. Section 6 shows new approaches with the help of AI possibilities regarding simulations so this kind of tooling in KiCad wont be discussed further.

4. DeepPCB

DeepPCB is a beta tool which takes care of the repetitive routing described in the previous section [2]. It is one of several AI tools of the company named InstaDeep [6].

Routing needs to satisfy a couple of constraints otherwise the hardware might not work as intended. To show a few of them, when a track is to thin and cannot provide the consuming part with enough power the board gets hot and might get damaged. When interconnections for high speed communication get integrated they must be realized with special geometric shapes in order to work properly. All of these case specific constraints need to be satisfied so the engineer draws the tracks by hand and needs to resolve all checks afterwards by hand too. DeepPCB automates this process by automatically routing the board with the usage of Reinforcement Learning (RL) [1].

4.1. Usage

When talking about complex CAD programs and other engineering tools there is usually the impression that those kind of tools are hard to learn and to get used to. DeepPCB is the opposite of learning a new CAD tool and the user does not even recognize the usage of AI. This subsection shows the ease of the usage of DeepPCB.

At first the user creates an account. After registering he lands in the dashboard of his profile (figure 4.1). When the user clicks on new project he gets asked to upload a file (figure 4.2). This file can be exported in KiCads PCB viewer. After clicking on submit the file gets uploaded and the processing begins.

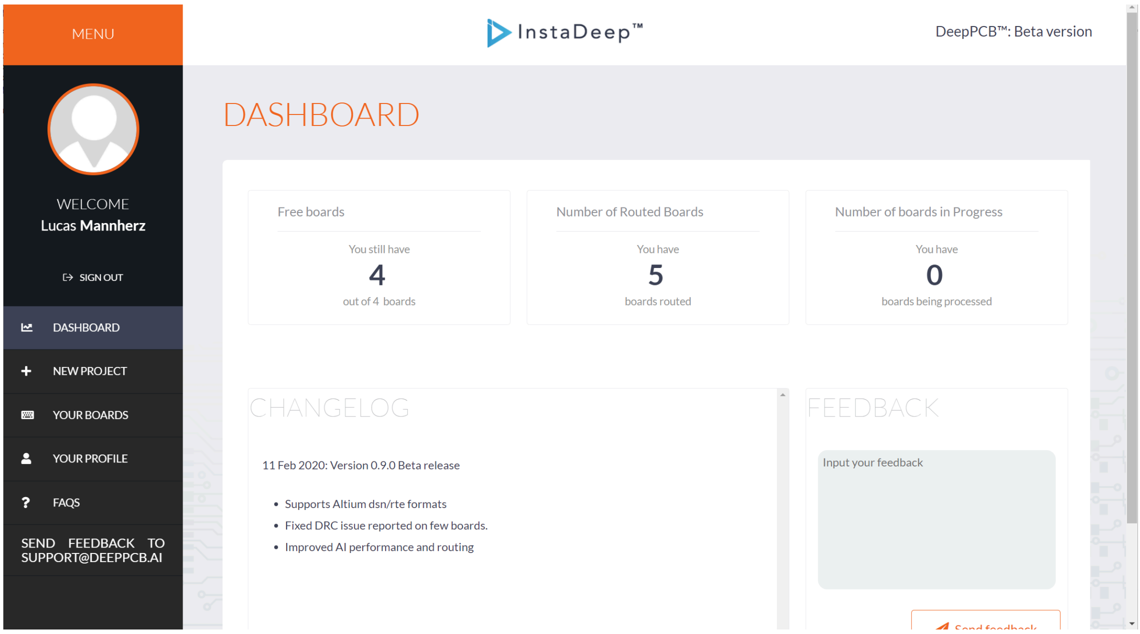

Figure 4.1: DeepPCB dashboard

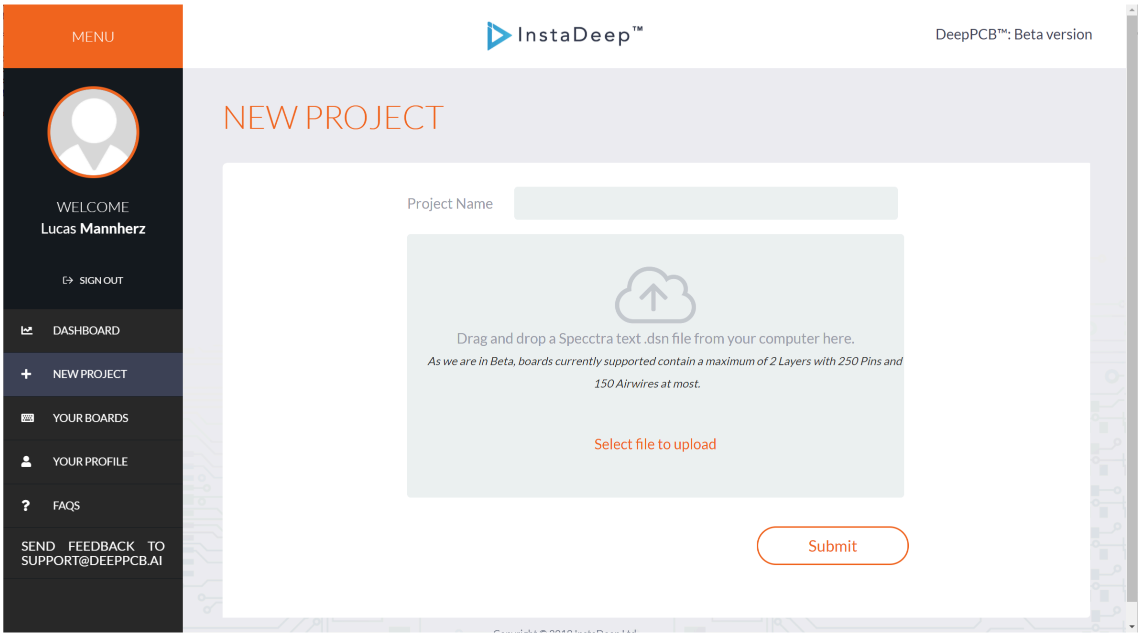

Figure 4.2: DeepPCB create new project

4.2. Results

DeepPCB guarantees a solution within 24 hours, but in this test here with the rather simple motor driver board the solution was available after around 15 minutes. In the beta phase the number of pins are strongly limited so it was not possible to test the tool with more complex boards.

The tool created three solutions, the first one being alike the second one. Figure 4.3 and 4.4 present two different solutions. The uploaded file had no strong constraints regarding the nets/tracks so this feature was not tested in detail. By downloading the file and importing it into KiCad the previously not routed PCB (figure 3.3) is now routed in KiCad as well.

The tool could be used for free in the beta at the time of writing this paper for a limited amount of boards.

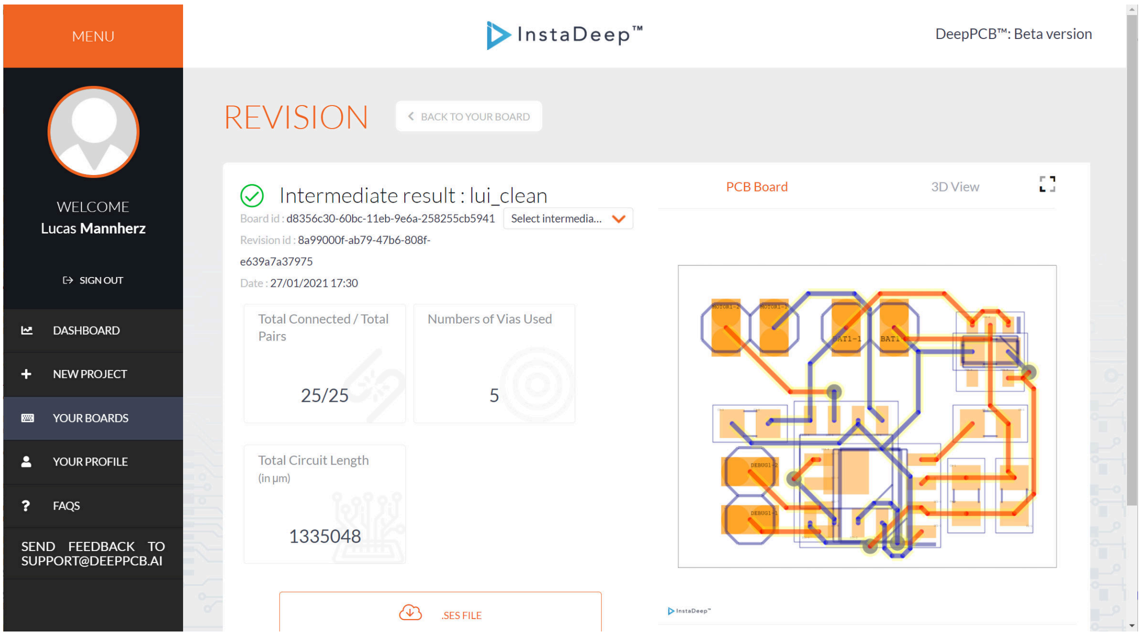

Figure 4.3: DeepPCB solution 1

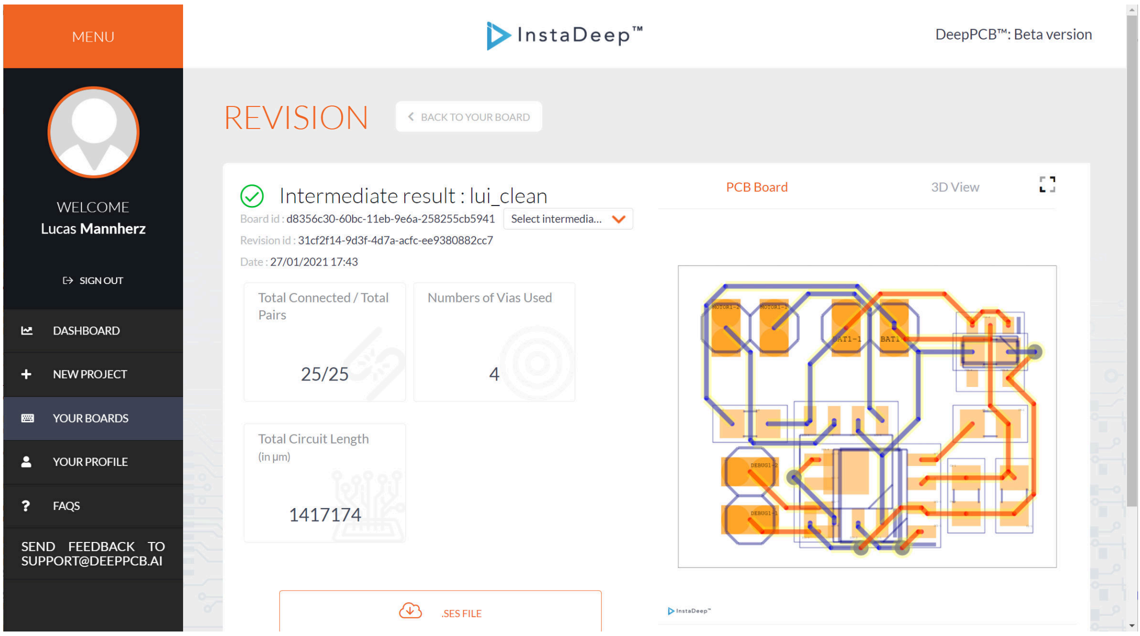

Figure 4.4: DeepPCB solution 3

5. Quality Assurance and verification

Process automation requires Quality Assurance (QA) as manual work does too. When working manually there are always humans involved so the QA is sometimes "built in" and happens while working on certain steps. This changes with process automation because there is no person who can notice any abnormity which makes additional QA necessary. By automating this step time can be saved and the amount of assurances is scalable. Especially when previous steps are automated it is possible that compared to the manual work multiple solutions come up. Scalability is therefore an important factor in terms of process automation and speed.

When it comes to special kind of services like checking the hardware outcome and its quality, ScanCAD for example offers services like "Automated Inspection & Measurement" and "Stencil & Screen Inspection" [10]. Their tools can also be used to reverse engineer unknown hardware.

Figure 5.1 shows the resulting loop. The previously developed and produced prototype needs to be checked. Before the prototype got build some features and constraints were defined. The QA checks if these requirements are met and returns a report. When this report is similar to the requirements and the tolerances are in the defined range the prototype satisfies the earlier defined criteria.

At this point the potential of using highly automated and integrated tools is visible. Prototype and QA are general terms and can be replaced by any step in a workflow. Regarding figure 2.1 therefore multiple iterations show up, each box has an arrow to the next step and vice versa.

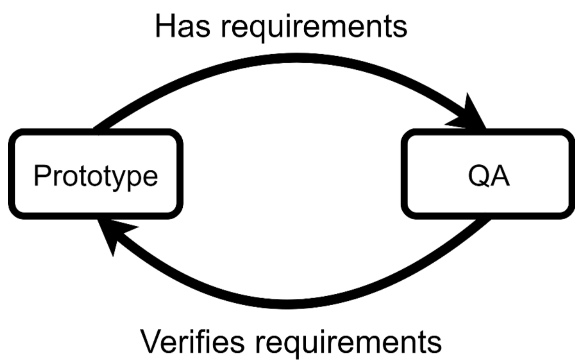

Figure 5.1: QA Iteration loop

6. Case study: Simulation with AI

Simulations can get complex especially when the whole system with strong inner and outer dependencies needs to be optimized. In a case study Dragicevic, Wheeler, and Blaabjerg [3] show a possible approach for optimizing a switching power supply, where multiple factors like costs, efficiency and lifetime are all coming together and models get rather complex if not impossible. When no simulation models are available for the specific scenario trial and error is a common approach [3, p. 1]. As previously described the needed time for each iteration is crucial to the products development time and its development costs. In the case study, where design for reliability approach is used, the designer would need to redo all the simulations again after changing design parameters [3, p. 2].

The forward ANN method was used in the case study in order to build the networks [3, p. 5]. At the end the case study reveals that the choosen approach has multiple advantages over using classical simulation models. One of them is that the evaluation speed is really fast so changing design parameters is no issue anymore. Another advantage is that classical simulations usually give certain results for fixed models. With the neural network it is possible to get different results for different models as well [3, p. 10].

7. Ideas

Figure 7.1 shows the resulting loops for each step from the steps described in section 2 after including the presented aspects from previous sections. Three points stand out when taking a closer look at the figure:

1. The steps left hand side which represent the verification of a result are still "classical" tools or processes and do not include much AI (yet). It is important as described in section 5 to automate those steps as much as possible. Interpreting results in a context and therefore evaluating them might be another aspect of integrating AI tools.
2. The creation/engineering steps (right hand side) consists of much manual work with humans involved so AI tools can have a huge impact on these kind of tasks. They can be used to automate processes here so AI can be involved in every step.
3. On the right hand side are two steps left where no tools are widely known yet (the red colored steps).

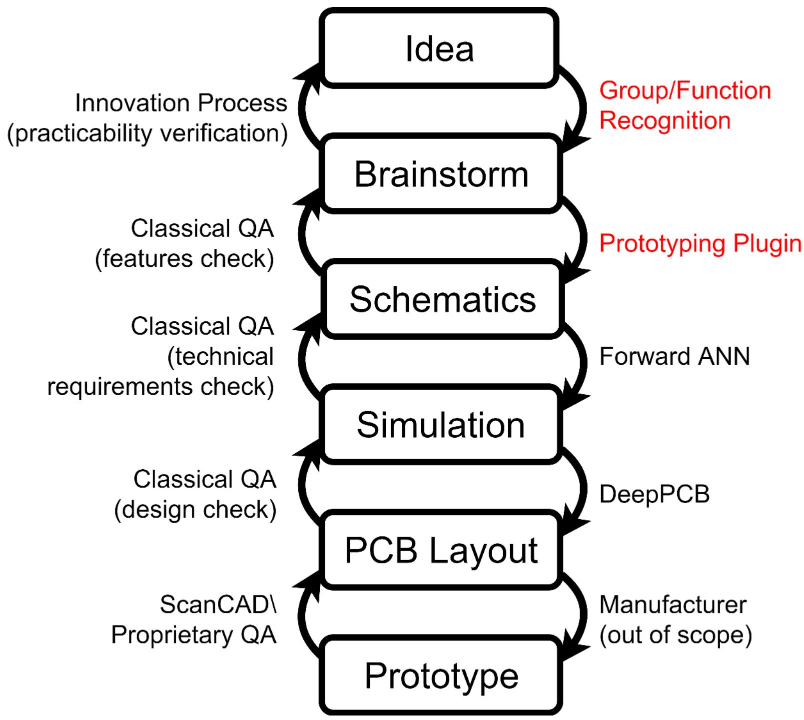

Figure 7.1: EE process loops

To close all the loops the following two subsections describe possible ideas. Plugins written in Python can be integrated into KiCad, and since KiCad is widely spread and used for creating schematics the integration of these steps into KiCad seems like to be a good solution. Additionally KiCad is open source so it is possible to make further integrations directly without being dependent on a company.

7.1. Group and function recognition

The first plugin should be able to identify groups of components. A component can either be an electrical component (resistor, capacitor, transistor, etc.) or a group of components which create a new component. Each component has a special function and combining functions result in new functions (and new components obviously). A resistor for example acts as a resistor and a capacitor as a capacitor, they are basic components and cannot be graduated into subcomponents. When a resistor gets combined with a capacitor in a certain constellation it is possible to create new properties like a low pass filter, so a low pass can be used as a component from now on. Changing the subcomponents in a component (e.g the capacitor in a low pass filter) obviously changes the components behavior. As the case study in section 6 shows AI models have big advantages over fixed models.

This plugin enables two scenarios:

1. Understand parts in a finished schematic or reverse engineered product. This allows fast troubleshooting and faster interpretations of complex or obfuscated systems. To give an example, if someone wants to repair a really old radio and the schematics are available, they are sometimes hard to understand due to complex analog dependencies or bad drawing. When the plugin is able to interpret functions it can assist in splitting the whole schematic into its component. When the repairing person knows where (in which component) he has to start searching for broken parts this process can be accelerated.
2. The other way around. When someone has an idea he could for example draw down a block diagram and annotate it with certain properties, the plugin needs to be able to interpret those values. The plugin then tries to solve the given constraints and behavior and automatically creates a prototype schematic as output.

This plugin is rather complex to realize and involves a lot of research regarding solving the so called "four pole theory" with AI models, at least for analog parts.

7.2. Prototyping plugin

To transform the paper drawn schematics into a CAD program like KiCad an engineer needs a more basic plugin. This enables engineers to quickly draw ideas on paper and by automatically transforming them into a machine readable format he can evaluate them faster because he can use tool which require machine readable formats.

This could be implemented by a plugin which recognizes images and is able to transform pictures into the KiCad format. The Engineer would just need to take a picture or scan the schematic. As a nice side effect this plugin could be used to digitalize older schematics and use modern tools (for example simulating them or improving old circuits).

8. Conclusion

Even with complex and sometimes expensive tools for EE it shows up that AI tools can increase the development speed significantly. By integrating the QA into the processes it could be possible to automate steps completely where humans needed to be involved before. By automatically checking if the requirements are met and automatically simulating multiple solutions it is possible to reduce the probability to make mistakes in the process which reduces costs and can increase the product quality.

It is not only possible to increase the development speed but to solve issues which were not possible to solve within a realistic amount of work and time as the case study shows in section 6. By going one step further and creating additional tools presented in section 7 the whole chain from the idea to the final prototype can nearly be automated completely.

A. Appendix Looping Louie

Looping Louie is a game made by Hasbro. It consists of a plane which moves in a circle above the players places. Each player has three chips and he needs to lift the plane up by pressing on his teeter board, otherwise the plane pushes down a chip. The player who helds the last chip wins [5, p. 2].

Figure A.1: Looping Louie game
Source: Hasbro, [4]

References

[1] DeepPCB. *FAQ*. Retrieved on 18.03.2021. https://deeppcb.ai/.

[2] DeepPCB. *Homepage*. Retrieved on 18.03.2021. https://deeppcb.ai/faqs.

[3] Tomislav Dragicevic, P Wheeler, and F Blaabjerg. *Artificial Intelligence Aided Automated Design for Reliability of Power Electronic Systems*. 2018. https://www.researchgate.net/publication/329245481_Artificial_Intelligence_Aided_Automated_Design_for_Reliability_of_Power_Electronic_Systems.

[4] Hasbro. *Hasbro products*. (German) Retrieved on 18.03.2021. https://products.hasbro.com/de-de/product/looping-louie-game:93A1A3DD-6D40-1014-8BF0-9EFBF894F9D4.

[5] Hasbro. *Looping Louie manual*. (German) Retrieved on 18.03.2021. https://www.hasbro.com/common/documents/5b96f7161d3711ddbd0b0800200c9a66/652E60D450569047F58CA83853EF297F.pdf.

[6] InstaDEEP. *Homepage*. Retrieved on 18.03.2021. https://www.instadeep.com/.

[7] KiCad. *GitLab repo*. Retrieved on 18.03.2021. https://gitlab.com/kicad.

[8] KiCad. *Homepage*. Retrieved on 18.03.2021. https://kicad.org/.

[9] NGSPICE. *Homepage*. Retrieved on 18.03.2021. http://ngspice.sourceforge.net/index.html.

[10] ScanCAD. *Homepage*. Retrieved on 18.03.2021. https://scancad.net/.

[11] Statista. *Prognose zum Umsatz mit Unternehmensanwendungen im Bereich künstliche Intelligenz in Nordamerika von 2016 bis 2025*. (German). Sept. 2016. https://de.statista.com/statistik/daten/studie/620494/umfrage/umsatz-mit-unternehmensanwendungen-im-bereich-kuenstliche-intelligenz-in-nordamerika/.