Software and Systems Modeling (2022) 21:1907–1929
https://doi.org/10.1007/s10270-022-01007-y

SPECIAL SECTION PAPER

Fabricatable axis: an approach for modelling customized fabrication machines

Frikk H. Fossdal[1] · Rogardt Heldal[1] · Jens Dyvik[2] · Adrian Rutle[1]

Received: 1 March 2021 / Revised: 24 February 2022 / Accepted: 28 March 2022 / Published online: 7 May 2022

Abstract
Digital fabrication tools such as 3D printers, computer-numerically controlled (CNC) milling machines, and laser cutters are becoming increasingly available, ranging from consumer to industrial versions. Recent studies have shown that users, ranging from researchers, to industry professionals, to hobbyists, are interested in modifying and changing the inherit workflows these tools provide. As an answer to this, these users are increasingly modifying and customizing their machines by changing the work envelope, adding different end-effectors, and creating their own fabrication workflows in software. However, customizing, modifying and creating digital fabrication machines and the workflows they provide require extensive knowledge within multiple different engineering domains and is non-trivial. In this article we present a model-driven approach that enables users to expand their digital fabrication scope by providing a high-level tool that facilitates the customization of fabrication tools. We present The Farbicatable Axis, a model that enables users to create customized linear actuators. The model takes high-level input parameters such as length and gearing-parameters, and outputs a CAD model of a linear motion axis consisting of *fabricatable* parts. We then present how instances of the Fabricatable Axis can be combined and used to design and implement Fabricatable Machines.

Keywords Model driven engineering · Digital fabrication · Machine building · CNC · CAD/CAM

1 Introduction

Automated manufacturing and digital fabrication is changing the way we design, interact and create the physical objects we surround ourselves with. Over the last decade, these machines, ranging from 3D printers, to laser cutters, to CNC (computer numerical controlled) milling machines, have translated from industry-only tools, to consumer-friendly desktop sized tools. Through the emergence of community spaces like Makerspaces and FabLabs [1] these tools are becoming common practice in how different practitioners, ranging from hobbyists, to industry professionals to researchers, make parts and designs for application specific use. In parallel, the threshold for interacting with these tools, and creating parts to be manufactured, is being significantly lowered by modern CAD (computer aided design) programs and more user-friendly software tools. This has spurred a new era of decentralized manufacturing, enabling more and more people to take part in digital and high-precision fabrication [3].

As the digital fabrication curriculum expands, recent studies [4–7] show that there is an increasing interest in not only using these tools to make customized parts, but also to customize and extend the capability of the machines themselves. For example, in [8], the authors present a laser-cutter that is modified with a pick & place tool for placing electronic components and soldering tools for soldering circuits

Communicated by S. Abrahão, E. Syriani, H. Sahraoui, and J. de Lara.

✉ Frikk H. Fossdal
ffo@hvl.no

Rogardt Heldal
rohe@hvl.no

Jens Dyvik
jens@dyvikdesign.com

Adrian Rutle
aru@hvl.no

[1] Western Norway University of Applied Sciences, Bergen, Norway

[2] Dyvik Design, Oslo, Norway

[1] Makerspaces [1] and FabLabs [2] are community spaces where people can access and use different digital fabrication equipment.

directly on top of the cut sheet. This example show that researchers across multiple fields, without practical knowledge in machine design, benefits from access to customized machines to conduct experiments that require computer controlled motion. These researchers want to harness the precision and autonomy that computer controlled machinery offers, and they want to be able to customize it to their specific applications. This is analogous to how software users not only want to use existing software, but to also customize and extend software for their own particular needs.

However, extending, customizing and modifying these machines requires machine builders to have significant knowledge in multiple different engineering domains. Even if digital fabrication and modern CAD workflows has lowered the threshold and simplified the process of designing and fabricating parts, designing and implementing functional machines is not an easy task. It requires the machine builder to have significant knowledge about principles and best practices in machine design. Even if the user knows which machine s/he ultimately needs, it is hard to conceptualize this into something implementable. Our broad goal with this body of work is to lower the threshold for implementing and experimenting with new types of digital fabrication tools and thus enable a wider group of people to build their own fabrication infrastructure.

Model Driven Engineering (MDE) provides principles and best practices that enable developers to manage the complexity of large scale systems by representing them at higher level abstraction [9] [10]. This has shown success in not only software industry, but also in more hardware oriented industries such as the automotive industry, where large software systems are integrated with physical and electromechanical systems [11].

More specifically, CAD/CAM workflows has been subject to prior research in the MDE community. In [12] Dalibor et al proposes a method for converting abstract system descriptions into mechanical CAD models using the System Modelling Language (SysML). Similarly Scheffler et al [13] explored an approach for transforming high-level system constraints into parametric CAD models. Our approach is motivated by these efforts, but differs in its specificity towards machine design and in its practical application of converting the virtual models into parts that entail a manufacturable design for the machines in this context.

In this paper we present the *Fabricatable Axis*, a model that allows users to design and customize instances of linear actuators from a high-level abstracted interface. The model is implemented in the Grasshopper language, which is a graphical programming language for creating programs that represents and generates virtual geometry. The Fabricatable Axis takes high-level input such as length, width, and gearing parameters, and transforms this into mechanical designs that is possible to make and implement using accessible tools such as a CNC milling machine. In addition to the model of the Fabricatable Axis, we present how it can be combined into *Fabricatable Machines*.

Our work is based upon model-driven principles and aims to develop a streamlined and high-level modelling approach for machine builders with limited knowledge of machine design to implement functional and robust instances of computer controlled machines. The novelty of our implementation lies in the ability it provides users to access and customize a sound and well tested mechanical design. The model has been developed to not only provide a theoretical development tool for machine designs, but also as a practical design tool that generates blueprints for parts that can be manufactured by existing machines in the digital fabrication context. Thus, it allows less experienced machine designers to harness the expertise of an experienced machine designer, without exposing them to the complexities that normally has to be taken into account when designing such mechanisms. The research questions driving our presented research are:

- *How can we encapsulate the knowledge of a machine designer into a parametric model in Grasshopper that can be used for building the Fabricatable Axis?*
- *How can our parametric Grasshopper model support users to build digital fabrication machines?*

The main focus of our research is to investigate how models can be used to capture the domain knowledge of a machine builder and encapsulate this into a unified approach that can be used by machine builders to successfully customize, design and deploy computer controlled machines. The material which is presented in this article extends our prior work [14,15] in several ways: We have added a more detailed specification of the model of the Fabricatable Axis together with a detailed specification of how it is modelled in the Grasshopper language. Furthermore, we have included a new model, *Fabricatable Machines*, that specifies how the Fabricatable Axis can be incorporated into specific machines with different workflows.

Paper outline We have structured the rest of the article in the following way: In Sect. 2, we present the necessary context to understand the rest of the material presented in the article. In Sect. 3, we present the methodology we used in conducting our research. In Sects. 4 and 5, we give a detailed overview of our implemented model of the Fabricatable Axis and how this model can be used to design Fabricatable Machines, respectively. Moreover, in Sects. 6 and 7 we evaluate our model by firstly showcasing a constructed scenario where we ourself design and implement a specific machine using the model, and secondly, an evaluation of users who have been using the model to experiment, design and make different machines respectively. Furthermore, Sect. 8 is dedicated to discuss the obtained results with

respect to the posed research questions while in Sect. 9 we highlight the relevant findings of our study for the modelling community. Finally, we include related work in Sect. 10 and conclude our study in Sect. 11 along with a list of future work.

2 Background

In this section, we highlight the context that is necessary to understand the principles behind the implementation of our models of the Fabricatable Axis and Fabricatable Machine. We explain how the machines in this context can be designed using linear actuators as building blocks, together with a generalized overview of all the activities behind the process of implementing a machine from a machine intent as starting point. Furthermore, we explain the Grasshopper language that is used to implement our model.

2.1 Building blocks for machines

Figure 1 shows 3 different types of computer controlled machines. In the figure, A1 shows a large format CNC router and B1 shows a small milling machine and a 3D printer. A 3D printer is an *additive manufacturing* machine, where physical objects are rendered by adding material layer upon layer. The CNC router and milling machines are both *subtractive manufacturing* machines, where objects are made by removing material from an initial stock. Their workflows are different, but we can see that their operation is inherently similar; they consist of a computer controlled motion platform, that moves a tool precisely in a three-dimensional Cartesian space.

To make the tool move around, these machines consist of a combination of *linear actuators* that together constitute a motion platform. In this context, a linear actuator is a mechanical device that can move in one single direction by transforming digital control signals into mechanical movement. It consists of an electromechanical drivetrain which is controlled by a computer. Typically, these drivetrains consist of a stepper or servo motor with a gearing mechanism to regulate torque.

By reviewing the router in A1, we see that it's motion platform consists of actuators arranged in the following way: a Z axis that move the tool up and down (the tool is depicted as a red box in the figure), a Y-axis that moves the Z-axis and the tool in Y-direction, and two X-axes, that moves the Z-Y arrangements in X-direction. This is depicted in A2. Similarly, B2 shows the motion platform of the CNC milling machine and the 3D printer in B1 (their motion platform is the same).

We now see how linear actuators can be used to create a diverse range and different types of machines. By providing these actuators as abstract building blocks that can be combined into machines, we reduce the threshold for experimenting with different types of motion platforms.

2.2 How to make a machine

The activity diagram in Fig. 2 shows a generalized overview of the activities that are involved in implementing a machine. Initially a machine builder will have a requirement of what the purpose of the machine will be and what process the machine is going to be used for. (Machine Requirement). For example the machine builder might have a goal of creating a 3D printer with a large work envelope (a large printing area). Based on the requirements, the machine builder will start designing the machine (Machine Design). This activity typically takes place in a CAD tool, and the machine builder will create a digital blueprint for the entire machine design. This step will require the machine builder to have knowledge about what parts are available and knowledge about best practices for how these parts can be used and incorporated into the design. Ultimately, a machine designer can choose to either source these parts from external vendors, or to make the parts himself using existing tools. A machine design will typically consists of a mixture of these two, depending on the fluidity of the machine builder's ability in making parts. Modern digital fabrication workflows have greatly enhanced a machine builder's capability to make and incorporate custom parts into a machine design, but it still requires the builder to have significant knowledge in how these parts actually can be designed in order to accomplish a robust result. Once a machine design has solidified itself, the machine builder has to acquire or make the parts that are used in the design (Sourcing / fabrication of parts). The quality of these parts will inherently be connected to the budget the builder has for buying parts, or the skills and experience s/he have in making the parts.

Once the machine builder has acquired all of the parts, the parts are assembled into a final mechanical composition (Mechanical Assembly) according to the blueprint created in step 2. Here the builder will attach the different parts with fasteners and the mechanical composition of the machine takes final shape. The performance of the machine will be tightly coupled to the builders experience in performing this step. If the assembly is done in a poor manner, the final workflow that the machine offers will suffer from this. This is also the first time the machine builder can evaluate the complete implementation and if the design does not meet the initial requirements they can choose to iterate back from this point.

If the mechanical construct meets the initial requirements, the electronic components are integrated into the machine design (Electronics / software Integration). The motors of the different actuators are wired and connected to circuitry that drives the motors. This circuit is coupled with programmable logic that receives commands from a higher level software

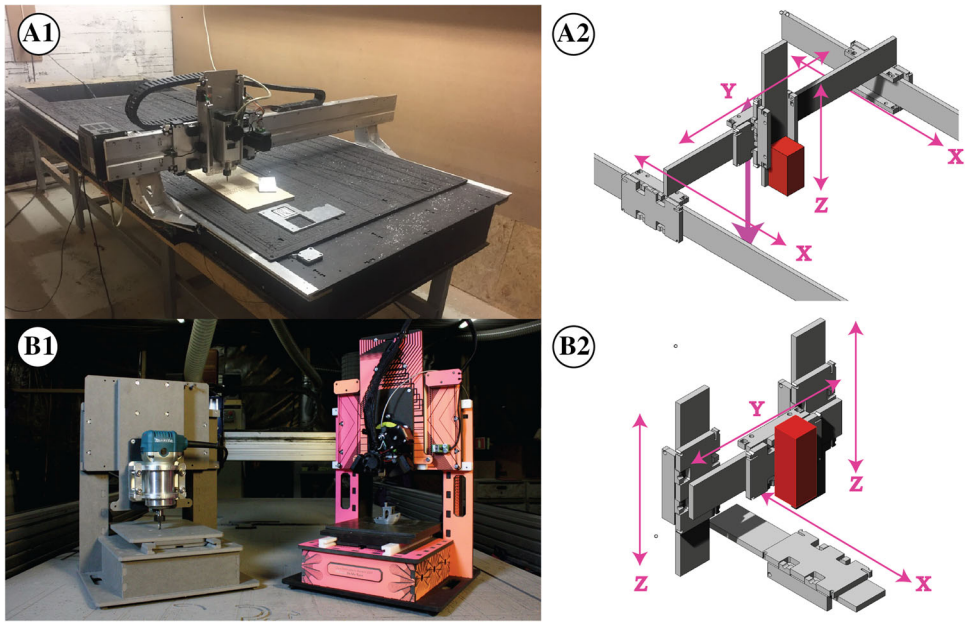


Fig. 1 A1 and B2 shows 3 types of computer controlled machines. A2 and B2 shows how the machines are made of combinations of linear actuators

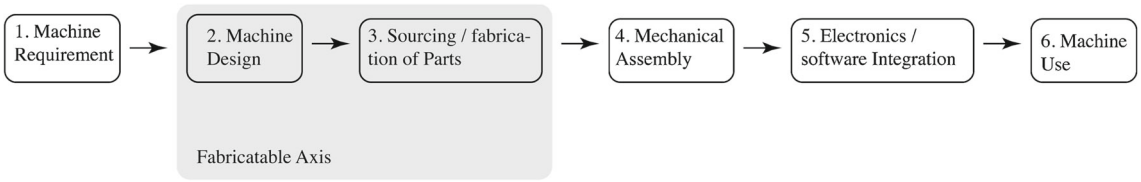


Fig. 2 The steps to go from a machine requirement to a machine implementation

interface and transforms this into electrical signals that turns the motors. At this level we will also typically see additional sensors like for example limit switches that is used to determine position of the different actuators. For traditional machine designs (like 3D printers, CNC milling machines and laser cutters) there exists a rich eco-system of controllers and drivers that can be utilized for such applications. Once this activity is completed, the machine will manifest itself as something that the machine builder can evaluate according to the initial machine requirements.

The activity diagram in Figure 2 aims to show a generalized overview of all the steps that are taken in a machine design process. Typically this process will not be this linear. A machine builder will iterate between the different activities as the machine matures. New mechanics will be added, additional constraints can arise during the process of building the machine, control logic is modified and changed as the mechanical assembly matures, etc. Ultimately, the requirement of the machine will be inherently connected to what type of workflow the machine aims to provide. Different workflows will require different types of mechanical tolerances and rigidity of the machine construction. Understanding all of these requirements can be a tedious process for an inexperienced machine builder. These activities are each time-intensive and it takes quite a lot of time before a machine builder arrive at a point where s/he can actually evaluate a physical result. Thus, a unified and simplified approach for arriving to a tangible implementation is crucial for these users to be able to implement functional machines. The activities that are marked in gray in the activity diagram shows the activities where our model benefits our users the most. This is further detailed in the Fabricatable Axis and Fabricatable Machines sections.

2.3 Grasshopper

Grasshopper [16] is a graphical modelling language for modelling, simulating and analyzing 3D geometry. It is incorporated in the popular CAD tool Rhino [17]. Grasshoppers primary interface is a graphical block diagramming tool that includes a rich set of shelf tools to manipulate geometry. Geometry is represented as a program that receives input and transforms this into a geometrical output. Modelling geometry in such a way is a form of parametric associative modelling [13].

In Fig. 3 we show an example of how Grasshopper can be used to describe geometry parametrically. The top model (A) shows a program that generates 3-dimensional boxes

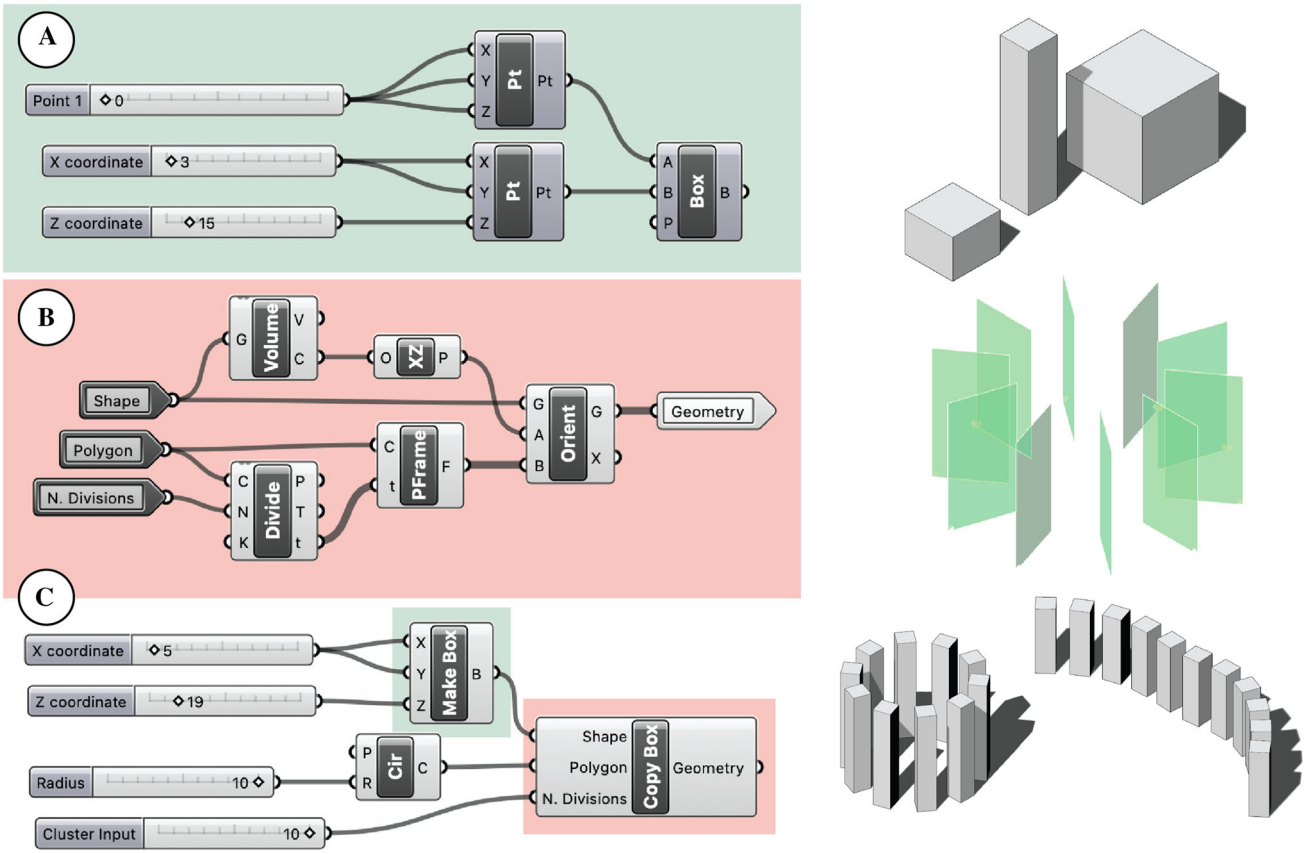


Fig. 3 Different Grasshopper models and their output

(highlighted in green). The models contains 3 sliders that manipulates a range of input parameters. These are connected to two Pt-blocks that defines two points in 3-dimensional space. The Pt-blocks are then connected to a block that generates a box between two points (A & B). We can see that by varying the sliders, the model will generate different boxes as output. This output is depicted to the right of the figure.

The model in the middle of the figure (B, highlighted in pink) is more complex. It takes Shape, Polygon and number of divisions (N.divisions) as input. It then subdivides the polygon into a range of points according to the number of divisions where each point is given a PFrame (a plane that is oriented perpendicular to the normal of the point). The input Shape is then copied to each of these points, orienting each copy to its corresponding PFrame. In the figure to the right of the model we can see a visualization of the orientation frames that are distributed over 10 points in a circle (the Polygon input is a circle).

The bottom model (C) is a combination of the two models above (A & B). Programs in Grasshopper can be grouped into "Clusters". Clusters can have user-defined names and user-defined inputs and outputs. In this model the programs highlighted in green and pink are grouped into clusters that are respectively called MakeBox (the green program) and CopyBox (the pink program). Combined, the two clusters define a logic that can generate a large amount of different variations of geometry by simply changing the value of the input sliders. The rendered output to the right in the bottom example (C) shows two examples of such variations. Here, the polygon that the boxes are copied over are changed from a circular shape to a open curve shape.

In addition to the shelf tools, Grasshopper also allows users to create custom blocks in C# and Python. This is an important part of our implementation, as we describe the constraints of our model using C#. The advantage of modelling geometry in Grasshopper lies in its flexibility. While a program initially can be developed for a specific type of geometry, for example the initial input parameter could be a circle, a user can at anytime change the initial geometry to something different. A Grasshopper model also allows to set up constraints that are inherently connected to the input parameters. Furthermore, a Grasshopper model exposes the user to not only a parametric design, but also to the total design process itself. A user can go through the block diagram and visualize how the individual blocks or clusters in the network manipulates data and thus gain understanding about both the design process and the programming concepts behind an implementation.

3 Research methodology

Our research approach follows the lineage of design science methodology. In [18], Hevner et al presents a methodology framework aimed towards design science in information systems (IS). The framework includes a guideline that is meant to aid researchers in IS to deploy high-quality design-science research. Engström et al [19] aligned this approach with research in a software engineering context. The presented work in this paper follows this framework for identifying problem relevance, deploy correct evaluation methods and contributing clear and verifiable contributions back to the knowledge-base.

Our target group for this research are users of digital fabrication tools who are situated in a FabLab / Makerspace context. This group have experience in using CAD programs to design different artefacts, and they have experience in using existing digital fabrication equipment such as CNC milling machines and 3D printers to make their designs. Furthermore they are interested in expanding their fabrication capabilities by developing custom machines of different sizes with different end-effectors, and thus broadening the scope of the workflows that their fabrication tools can provide them with. We have identified two concrete problems that this group is facing:

- Understanding the design and engineering principles behind designing robust motion platforms that can be used to implement machines.
- Being able to rapidly conceptualize, deploy and evaluate machine implementations.

The implemented artefact presented in this paper is meant to aid this group by simplifying the process of conceptualizing, designing and manufacturing digitally controlled actuators through a high-level modelling tool.

To develop this type of interaction we rely on the continuous feedback from the users as they are using the artefact to deploy different types of machines. The artefact was developed over time by extracting known theories and methods from the MDE knowledge base and using this to encapsulate a machine designers knowledge into a model and deploy it in a user environment. We tracked the use of the model through a GitHub [20] repository where the target group could access the latest version of the model and provide feedback on their use of it in different applications and scenarios.

The knowledge we want to contribute back to the knowledge-base is how this type of encapsulation can facilitate access to bespoke digital fabrication tools and thus empower our target users with new types of manufacturing workflows and usages of digital fabrication.

The evaluation presented in this article consists of two types of evaluation methods: (1) a constructed scenario where we demonstrate the utility of our artefact by using it ourselves to implement a specific machine, and (2) a series of semi-structured interviews where we interview subjects who have been using the artefact to design and implement different digital fabrication machines.

The purpose of the interview was to get information about the use of our model and if the different subjects where able to incorporate its output into useful machines. Additionally, we wanted to understand why these machine builders were building machines in the first place and how successful their implementation had been using the Fabricatable Axis. We used structured interviews over video calls with practitioners from this community. The subjects were picked from an online forum that had formed around the GitHub repository. We obtained a total of six subjects from this community. The subjects in the study were between 20–35 years old and had a diverse set of backgrounds, ranging from teachers to industry mechanics (the demographic can be seen in Table 1). Prior to the interviews, we did not have any information about how the subjects had used the model, whether they were satisfied with it or what types of machines they had implemented using the model. The interviews were structured around 13 main questions. These questions are listed below. The first author conducted the interviews, while the second author watched all of them. Subsequent to the interviews the two authors discussed the findings and summarized what the collected data indicated in Sect. 9.

- How did you become involved in fab machines?
- What did you build using the Fabricatable Axis?
- Did your implantation work? Was it robust? Was it useful?
- Did the model allow you to build what you wanted to build?
- Were there any capabilities that you missed in the model?
- Did you modify the model in any way?
- Did you understand how the different parameters worked?
- Did the Fabricatable Axis simplify your workflow?
- How robust was the machine you built, and what did you use it for?
- How did you integrate electronics and control to your setup?
- What did you make using the machine you made?
- Did you do any commissions using your machine?
- Did your machine give you capabilities you did not have access to before?

4 Fabricatable axis

Figure 4 shows a domain-model of the Fabricatable Axis. The model acts as a blueprint for the Grasshopper model pre-

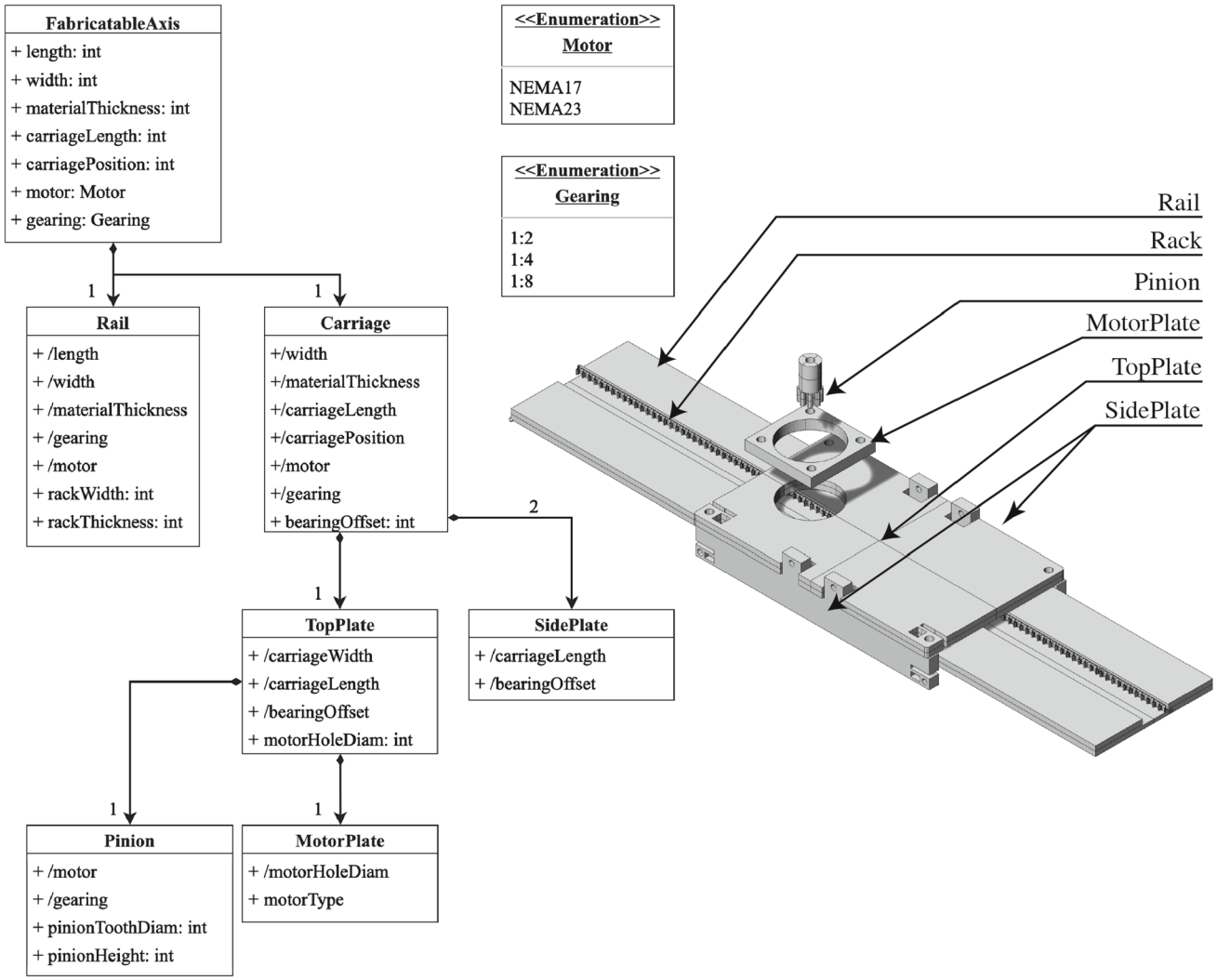


Fig. 4 Domain-model of the Fabricatable Axis

Table 1 Demographic overview of subjects in the interview

	Occupation	Machines built
Subject 1	Mechanical engineer/teacher	6
Subject 2	Machinist/teacher	3
Subject 3	Entrepreneur	1
Subject 4	Graphical designer	1
Subject 5	Physicist/computer scientist	2
Subject 6	Machine designer	6

sented in Sect. 4.1, and contains all concepts for all the parts (except screws & fasteners) of the design and how they relate to each other. The top concept FabricatableAxis contains the attributes length, width, materialThickness, carriageLength, carriagePosition, motor and gearing. length is the length of the entire actuator and width is the total width of the actuator. materialThickness sets the thickness of the material from which the parts of the actuator is fabricated. carriageLength is the length of the carriage situated on top of the rail and carriagePosition is the offset position that the Carriage will be positioned in relative to the rail. motor sets the type of motor that is used to actuate the carriage on the rail (either a NEMA17 or NEMA23 stepper motor), and the gearing attribute sets the gear-ratio of the drivetrain of the actuator. These attributes represents the parameters that a user is exposed to when using the model to generate instances of The Fabricatable Axis.

The FabricatableAxis concept contains two sub-concepts, Rail and Carriage. The concept Rail represent the rail which the actuators carriage is situated on, and the concept Carriage represents the carriage itself that actuates on top of the actuators rail.

The Rail concept contains the attributes that generates the geometry of the rail that the Carriage is actuated upon. The total lenght, width and thickness of the rail is set by the

derived attributes (as indicated by the '/' before the name of the attributes) length, width and materialThickness. Furthermore it uses the derived attributes gearing and motor to calculate the geometry of the tooth-profile of the actuators rack[2]. The attribute rackWidth and rackThickness defines how wide and deep the rack is imprinted in the rail.

Carriage is the top-concept for all the parts that are included in the assembly of the carriage that is actuated on top of the actuators rail. The main geometry of the carriage is calculated from the derived attributes width, materialThickness and carriageLength. It's position relative to the actuators rail is set by the derived attribute carriagePosition. Furthermore Carriage holds an attribute that sets the placement of the bearings, bearingOffset, on the TopPlate and the two SidePlates. This attribute is calculated relative to the derived attribute carriageLenght. The assembly of the carriage consists of 5 physical parts. These 5 parts are divided in to two assembly groups: (1) a top-plate that includes an adapter plate for attaching the stepper-motor and the pinion that is attached to the motor and engaging with the rack. (2) two identical side-plates. In the domain model these are organized into the concepts TopPlate that holds the concepts MotorPlate and Pinion, and SidePlate.

Both the concepts TopPlate and SidePlate uses the derived attributes width, materialThickness and carriageLenght to calculate it's geometry in relation to each other. TopPlate holds an additional attribute that sets the diameter of the hole where the stepper motor and the MotorPlate is placed on top of the axis. The diameter of the hole is calculated from the derived attribute Motor. Finally, the concept Pinion holds attributes pinionToothDiam and pinionToothHeight that describes the geometry of the actuators pinion. The attribute pinionToothDiam is calculated from the derived attributes gearing and motor.

The MotorPlate uses the derived attributes motorHoleDiam and motorType to calculate its geometry. The Pinion computed geometry is directly related to the gearing and motorType attributes of the top-concept. It uses these derived attributes to calculate how the geometry of the Pinion should be to match the selected gearing of the actuators drivetrain.

CAD models are derived from sets of constraints and relationships that ultimately describe an end design. For example a hole can be described as a `5 mm` hole that is perpendicular to a specified line with a distance of `15 mm` between the line and the hole. That line can then have another constraint that constrains it to be angled 45 degrees to a third line. The design of the Fabricatable Axis consists of numerous of such constraints that play a crucial role in ensuring that the parts of the Fabricatable Axis design are kept within a proportional relationship relative to each other, and that the different geometrical properties are kept within certain ranges that yields a valid geometry. For example, if the total length of the carriage is too short, the geometry of the sidePlates will fail (because it is not possible to fit the required geometry within this range). If the thickness of the material is too high, the geometry of the sidePlates will not be able to fit underneath the top-plate geometry. Further examples of these constraints are shown in more detail in Sect. 4.1. It is these proportional relationships that ensure that the final output of the model delivers a design with mechanical integrity and a design that is functional in its ability to deliver robust mechanical linear motion. It is the domain expertise of creating these constraints that the model ultimately encapsulates. The model allows non-experts to access this expertise and use it in their specific application without needing to know about the constraints that drive the design of The Fabricatable Axis.

[2] The rack is the static side of the actuators rack & pinion drivetrain. A rendered image of the Fabricatable Axis' rack can be seen in Fig. 4.

4.1 Grasshopper model

The Fabricatable Axis is modelled in the Grasshopper language and conforms to the domain-model in 4. We have chosen Grasshopper as the modelling language due to its rich set of computational tools for handling geometry data and due its widespread use in our target community. Each of the concepts in the domain-model are represented with with a Grasshopper cluster (clusters are explained in the Background section), and the clusters conforms to the same hierarchy as the domain-model. The complete grasshopper model can be accessed and dowloaded via the Fabricatable Machines repository on GitHub [20].

Initially only the top-level concept Fabricatable Axis is exposed to the user. The cluster takes the parameters Length, Width, MaterialThickness, CarriageLength, CarriagePosition, Motor and Gearing, and outputs the full geometry of the Fabricatable Axis. The input parameters are given by slider and drop-down boxes which is shelf components in Grasshopper. Both the sliders and the drop-down boxes are preset with values within specific ranges. This is to constrain the user to only input values that yield a sound design. For example (the minimum value of the Length is set to 400, while the maximum value of CarriageLength is set to 200. This ensures that the carriage can never be longer that the rail it acts upon. An image of the cluster is shown in the top of Fig. 5

By selecting a cluster its output is previewed in Rhino. For example if a user selects the top cluster, Fabricatable Axis, the entire axis is previewed in Rhino. If the cluster Carriage is selected, only the Carriage will be previewed in Rhino. By opening a cluster the next "level" of the model is shown. For example, by opening the Fabricatable Axis, the Carriage and Rail clusters are shown. The sub-cluster of Fabricatable Axis is shown in the bottom Fig. 5. Here we can see how the input

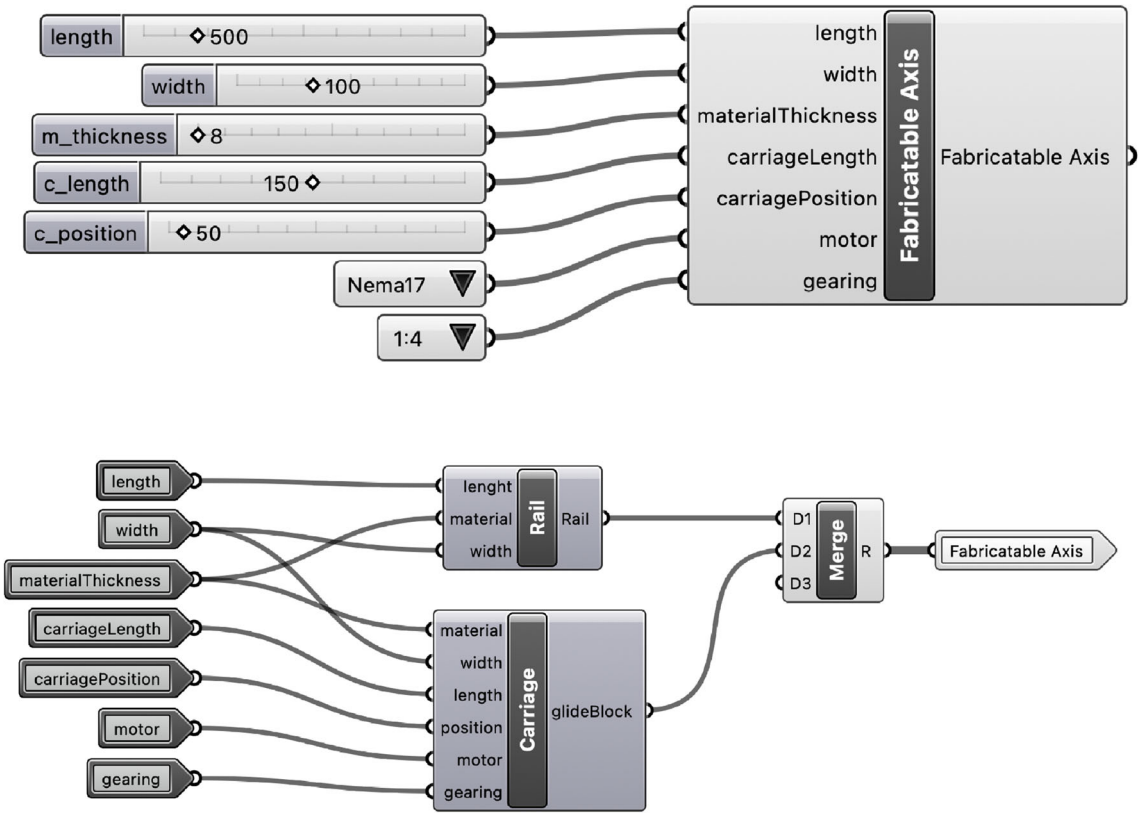


Fig. 5 Top figure shows the Fabricatable Axis top-level cluster in Grasshopper. Bottom figure shows the top-levels sub-clusters; Carriage and Rail. These clusters are merged into the output of the Fabricatable Axis

parameters of the top cluster is piped into the top-level cluster and propagated through the network of different sub-clusters. Each sub-cluster is responsible for handling how the input data is transformed into valid parts. The different constraints of the design and the constrained relationships between the input parameters are modelled here. As stated in the previous section, this is really where the domain expertise of the Machine Designer is captured. The constraints control the mechanical relationship that is essential for the designs final geometry (where holes, bearings and the motor are placed). The constraints needs to be parametrically defined in relation to the input parameters (when the width of the axis is changed the parameters needs to translate points and geometry in relation). For example, the holes that the bearings are attached to on the TopPlate is defined to be 8mm in y-direction in regards to the horizontal edges of the top of the geometry. As the width of the Carriage is changed, so does the width of the TopPlate and the placement of the holes.

The majority of these constraints are modelled using C# code blocks inside of Grasshopper. Figure 6 shows how the holes for bearings and attachment points for the Sideplates are generated inside of the TopPlate Cluster under the Carriage cluster. The code-blocks are named top_bearings and side_plate_links. The placement of these features are directly connected to the top-level parameters of the Fabricatable Axis. Figure 7 shows the code that is contained in top_bearingsblock. Here we can see how the top-level parameters material, width and CarriageLength are used to determine the placement of the holes on the TopPlate, and that by changing these parameters the placement will be changed relative to the value of these. This example also shows how the the different clusters and code-blocks can be visualized as they are selected. In Figure 6 the two code-blocks that are selected (marked in green) is being visualized in the Rhino viewport (image in the square).

An overview of a typical user interaction with the model is shown in Fig. 8. On startup the model is populated by default parameters, and the user is immediately exposed to a preview of the output of the model in Rhino. A user interacts with the model by manipulating the input parameters from the top-level cluster, Fabricatable Axis.

As users interact with the input parameters, the model's output geometry is constantly rendered to the Rhino viewport. A typical interaction with the model is shown in Fig. 8. Here we can see the Rhino viewport on the right side and the Grasshopper model to the right. As the Fabricatable Axis cluster is selected in Grasshopper (highligthed in green), it's visual output is rendered to the viewport (in green). This

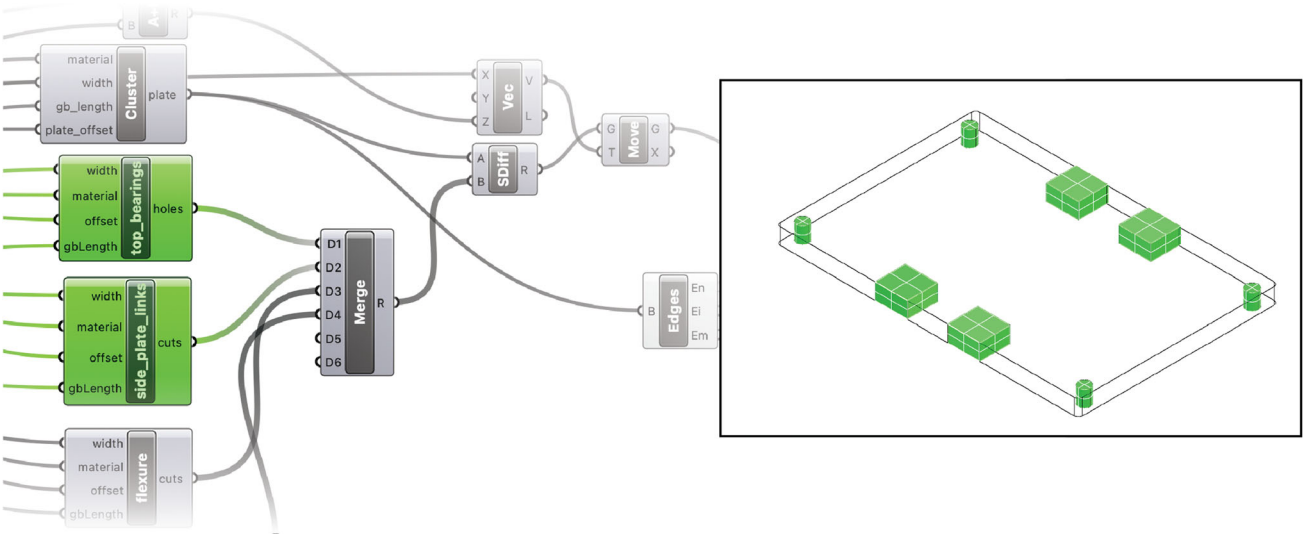


Fig. 6 The relationships between input parameters and the generated geometry are modelled using code blocks

Fig. 7 Example of the code that is contained in the top_bearings code-block shown in Fig. 6

```
int distX = 5;
int distY = 8;
int rad = 3;
Point3d[] points = new Point3d[4];

points[0] = new Point3d(distX,-offset-material+distY,material);
points[1] = new Point3d(distX,width + material - distY,material);
points[2] = new Point3d(carriageLength-distX,-offset-material+distY,material);
points[3] = new Point3d(carriageLength-distX,width + material - distY,material);

Cylinder[] c = new Cylinder[4];
c[0] = new Cylinder(new Circle(points[0],rad),-material);
c[1] = new Cylinder(new Circle(points[1],rad),-material);
c[2] = new Cylinder(new Circle(points[2],rad),-material);
c[3] = new Cylinder(new Circle(points[3],rad),-material);

//export cut holes
holes = c;
```

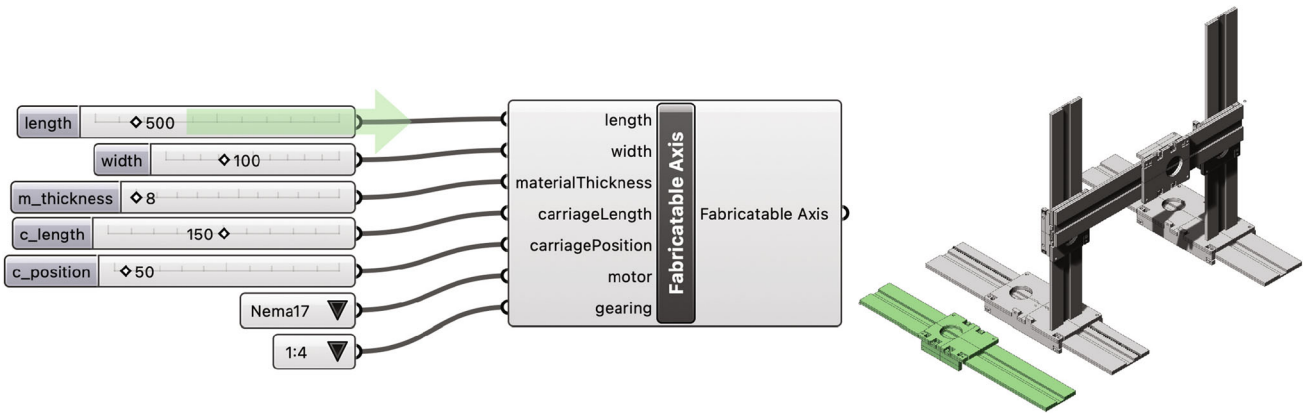


Fig. 8 Interaction with the Fabricatable Axis model. By changing the input parameters of the model, users can create instances of the model and arrange them in Rhino

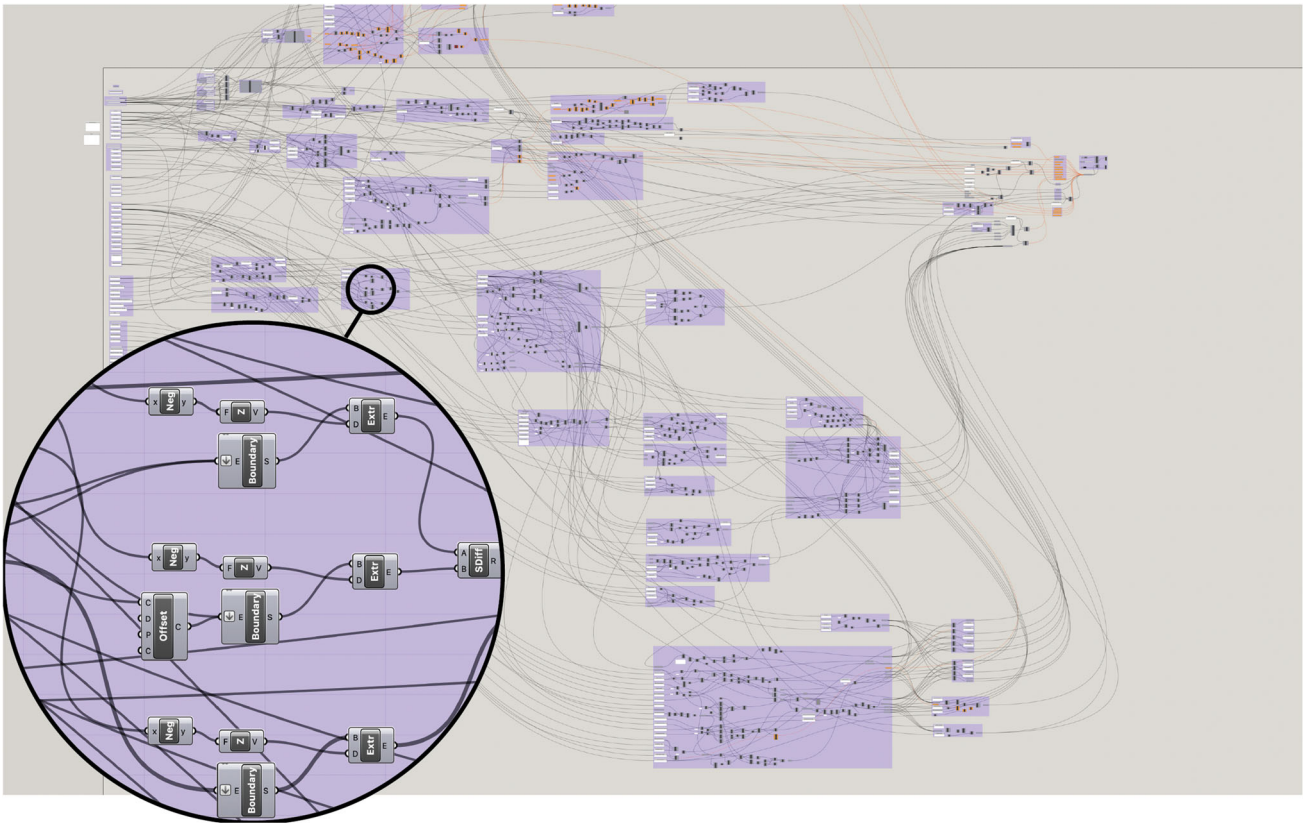


Fig. 9 The complete model in Grasshopper with all clusters expanded

type of visual feedback enables users who doesn't necessarily understand what or how the different parameters affect the output, to use the model to design machines, and even gain understanding as they use it. By interacting with these parameters users are able to design and customize a wide range of different linear actuators in a very short time. Each instance that the model output can be baked into a solid geometry in Rhino. These instances can then be translated and rotated using the native tools in Rhino, and users can quickly start conceptualizing machine constellations using the different instances. An example of this process is provided in the constructed scenario in Sect. 6.

The Grasshopper model does not limit a user in any way to alter or modify it's inherit logic or functionality. Users can move beyond the initial Fabricatable Axis cluster and manipulate how the individual sub-clusters operates on the input data freely. We find this to be an important feature of our model in regards to the usability and flexibility it provides for our users. The presented interviews later in Sect. 7 shows that several of the users had indeed further customized the functionallity of the Fabricatable Axis model. In addition we believe that it is important that users are able to access the model at this level and allowed to examine the models entirety in order to gain understanding of how the model fundamentally works and manipulates data. To the left in Fig. 9 we show how the complete model looks like with all the clusters exploded. Each purple box in the figure represents a cluster. This shows the total complexity of all the calculations and steps that take place in order to output the final design of The Fabricatable Axis, and the complexity which the model alleviates a user from.

5 Fabricatable machines

Figure 10 show a domain-model of a Fabricatable Machine. The concept FabricatableMachine contain the attribute workflow. A workflow describes what type of process the machine is going to be used for. The enumeration type Workflow currently includes 3DPrinting, SoftMilling (milling of lighter materials such as plastics, wax and lighter metals), HardMilling (milling of harder materials like hard-wood, steel), LaserCutting, PenPlotting, LiquidPrinting, Inspection (for probing parts and checking tolerances) and PickPlace (picking objects up with a gripper and moving them in space). These enumerations includes the machine workflows that have been incorporated into implemented machines up till this point in the project, but does not exclude other types of workflows in the future. In the Evaluation section, we present a machine that is built to provide the Plotter workflow and

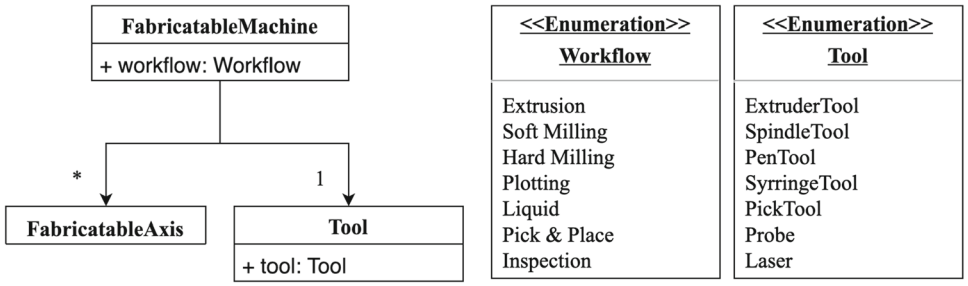


Fig. 10 Domain-model of the Fabricatable Machine

in the end of the paper we show examples of machines with other workflows.

The concept FabricatableMachines contains the concepts FabAxis and Tool. A Fabricatable Machine does not limit the amount of Fabricatable Axes, but it can only have one Tool. The concept Tool represent the tool that the Machine is using and is inherently connected to the workflow that the machine provides. The enumerations for this attribute includes ExtruderTool (for extruding materials), SpindleTool (for cutting), PenTool (for plotting), SyrringeTool (for 3D printing with liquids) and ProbingTool (for inspecting parts). These enumerations are connected to the Workflow enumeration. For example if the workflow of the machine is 3D printing, the tool will be an ExtruderTool. If the workflow is inspection, the ToolType is PropingTool.

The process of modelling machines is done by generating actuators using the Fabricatable Axis, and manually placing the actuators into a machine constellation in Rhino. The Fabricatable Axis benefits the user in generating CAD models of customized motion modules that the user can fit into their machine application. By interacting with the parameters they can quickly generate actuators with different qualities and combine them into machines. An overview of this process is shown in Fig. 8. Here we can see how the Fabricatable Axis model is used to generate different actuators, and how they are placed in a constellation consisting of a total of 5 actuators.

Once a user has a desired arrangement of the actuators they can choose to further modify the CAD model by designing and adding new parts, or by modifying the CAD output of the different instances of The Fabricatable Axis. These steps are thoroughly documented in a GitHub [3] repository. The repository holds several examples of machines that are implemented using the Fabricatable Axis and that are used to provide different workflows.

6 Evaluation: constructed scenario

To show the process of using the Fabricatable Axis to design a Fabricatable Machine, we present a constructed scenario where one of the authors of this article used the model to implement a 2-axis pen-plotter. The pen-plotter was designed with specific requirements: it needed to fit in a suitcase so that it could be brought on an airplane to a demo at a conference. As part of the demo, it also needed to be able to draw different plots with the marker with relatively high precision. The author who implemented the machine was involved in the development of the model itself, and had fluent knowledge about how to use Rhino as a CAD tool and in using CNC milling machines to fabricate parts.

To meet the requirements, the machine needed to have a specific size (it needed to fit in a suitcase). It also needed to move a pen-tool around in 2 dimensions quickly and precise. The requirements for rigidity and stiffness were not that high since a pen-tool is relatively light-weight and requires little torque to be moved around. The dimensions of the machine were decided to be 400 x 500mm. We used a 3mm round milling bit to cut the parts, and the properties of the drivetrain were optimized to be made using this bit-diameter. The parts were made from a 1x1m sheet of 8mm POM (Polyoxymethylene). An object-diagram of the Pen-Plotter is shown in Fig. 11. It shows how the constraints of the machine and its workflow is modelled into the different Fabricatable Axes and Tool that constitutes the machine. The diagram only shows the top concepts of the Fabricatable Axis, but their instantiation will follow the meta-model presented in Fig. 4.

Based on the constraints of the object diagram, the Fabricatable Axis model was used to generate the components needed to make the motion platform. An overview of this process is shown in Fig. 12. As the different instances were generated with the model and baked into Rhino, we used the native tools of Rhino to move the actuators around in space and position them into a desired composition (A). If we at anytime needed to change or modify one of the instances, we simply regenerated the instances using the model. This iterative process continued until we were contempt with the layout of the machine.

Once the layout of the actuators was completed, an additional frame was modelled and added to the composition to increase the stiffness of the machine and to simplify the process of assembling the parts into a rigid construction (B).

[3] https://github.com/fellesverkstedet/fabricatable-machines/wiki.

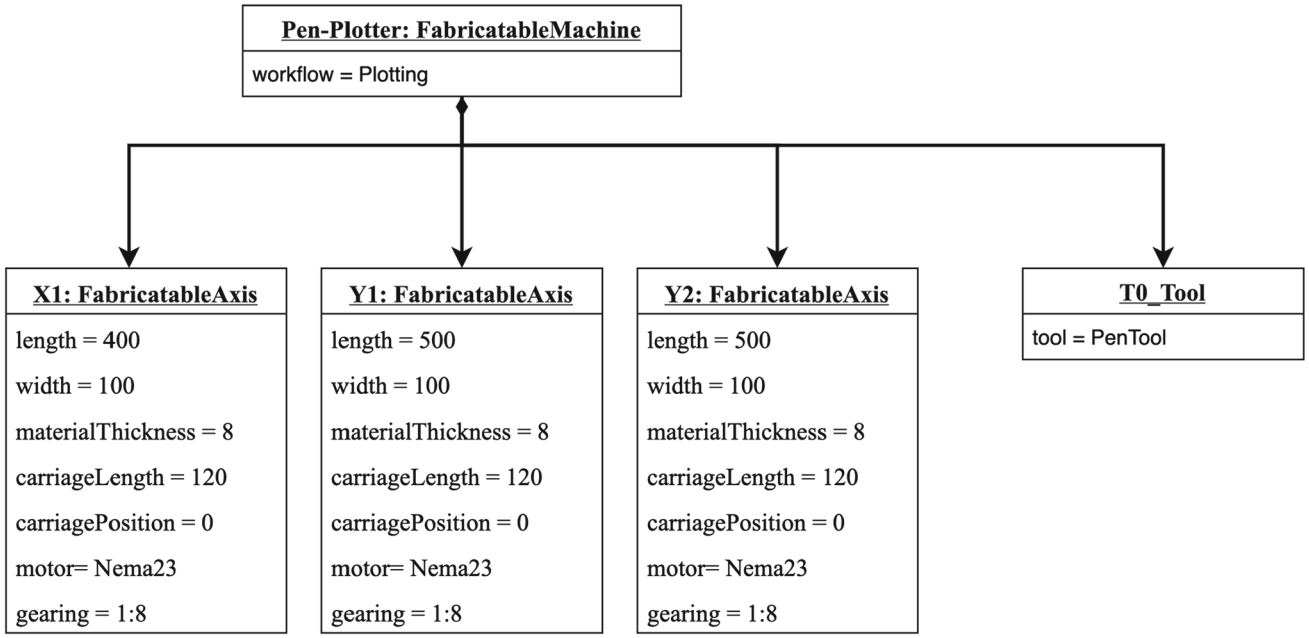


Fig. 11 Object diagram of the Pen-Plotter

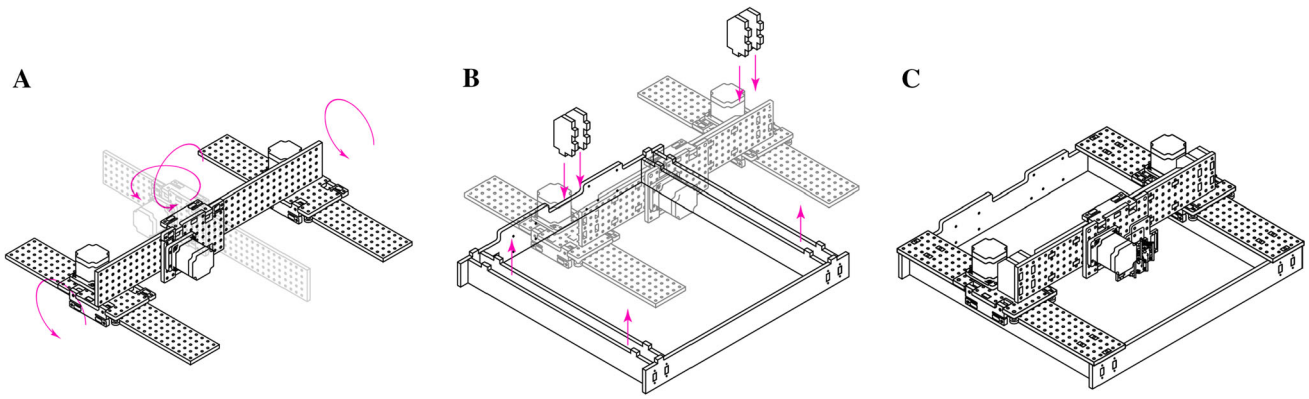


Fig. 12 Process of designing 2-axis pen-plotter using the Fabricatable Axis. The model is used to generate 3 different actuators. The actuators is then put in a machine composition by translating and rotating them in Rhino, which can be seen in **A**. In **B**, an additional frame structure and blocks for joining are added to the machine. The final machine design is shown in **C**

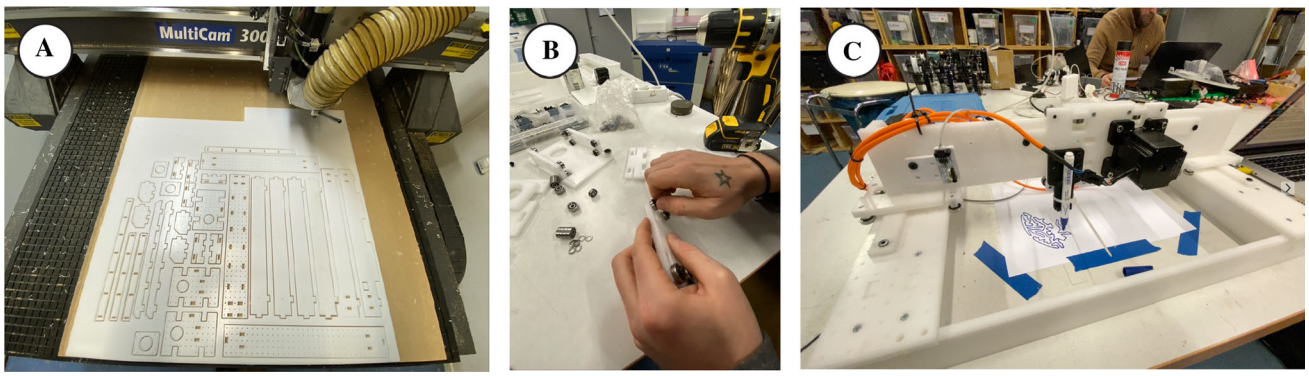


Fig. 13 In **A** the parts for the 2-axis pen-plotter is cut using a CNC milling machine at a local Makerspace. **B** shows the assembly process of the machine. In **C** the completed pen-plotter is being used to draw line art on paper

This part was done manually in Rhino on top of the already generated linear actuators. Finally a CNC milling machine was used to cut out the required parts. The complete milling operation was approximately completed in 1 hour.

All of the milled parts were then assembled using fasteners and screws from a local hardware store. First each carriage of each individual axis was assembled. These were then attached on top of the rail of each corresponding axis and finally assembled as a fully assembled axis. Then each axis was combined into the final machine layout. In the last steps the manually modelled frame and attachment blocks were used. Once this is done the mechanical design is complete and ready for use.

Apart from conceptualizing, fabricating and assembling the mechanical design (this is the activities 2,3 and 4 in the activity diagram in Fig. 2), the process also involved a final step were the mechanical platform was further modified with wiring, electronics and a machine controller. This was a relatively trivial step for a simple 2-axis machine since there exists a quite rich ecosystem for different off-the-shelf controllers for controlling machines with 2-3 degrees of freedom. However, as the degrees of freedom increases this is no longer the case, and implementing the control is often left to the user.

The machine was designed and built over the course of one week and thereby shows the efficiency of both the tool and the advantage of having the design fabricatable. The complete bill of materials of the machine is included in Table 2. The demonstration shows how a machine designer, who is already fluent in machine design and in the use of digital fabrication tools, benefits from a high-level model for generating motion modules. By generating the required modules, the machine designer was able to conceptualize and manifest a design quickly, both digitally and physically. However, it is important to note that the machine designer, based on his experience, already had a clear image of what a sound machine design would look like. This might not be the case for users who are inexperienced in machine design.

7 Evaluation: feedback from community

We conducted a series of interviews with 6 participants from the projects Github community where the model is published and maintained. In the following subsection we summarize the subjects insights and reflections on 7 of these questions. During the first interview we found that one of the subjects had ported a version of the model to a different CAD tool, namely Autodesk Fusion360 [21], and that two of the other subjects in the interviews had used this version of the model rather than the original. The user experience with this model is the same as with the Grasshopper model, so for the purpose of the interview, whether a subject used the Fusion360 model or the Grasshopper model is not crucial. We discuss this revelation in more depth in Sect. 8.

7.1 Interview results

What type of machine did you build using The Fabricatable Axis? Some of the subjects had generated several different machines using the model, ranging from more common machine types like CNC milling machines and 3D printers, to more unique machine designs like ping-pong shooting machines and light-painting machines. All subjects in the study had interacted and used the model to generate different designs of The Fabricatable Axis that they in turn fabricated and assembled. We found that subjects who were inexperienced in machine design typically used the model to generate one single instance of an actuator, rather than combining multiple axes into a complete machine design. They then proceeded to fabricate the parts for the actuators, assemble it using the instructions found in the repositories wiki, and finally test it. Once this group were comfortable using the model to generate instances of the axis, they were also more comfortable with using the model to create more complex machine compositions. Here, several subjects underlined the importance of how the model allowed them to move quickly from conceptualising and designing the actuator, to actually producing the parts of it, assembling it, and finally testing it physically.

Table 2 Bill of materials Pen-Plotter. NOK 3709,- is the equivalent to $420

Part	Amount	Price per item (nok)	Total (nok)
M5 Screw	50	10,-	500,-
625 Bearing	36	10,-	360,-
Nema23 Stepper Motor	3	250,-	750,-
1 × 1 m POM sheet 8mm	1	1200,-	1200,-
Wiring	–	200	200,-
Arduino Mega	1	499,-	499,-
RAMPS 1.4 Motion Controller	1	300,-	300,-
TOTAL	**–**	**–**	**3709,- ($420)**

Bold indicates the sum of (total) cost of parts for the machine in the prior example

Although all subjects had interacted with the model, we found that the majority of the complete machine designs found in the repository were designed by a core group of 3 subjects. The rest of the subjects were reusing these machine designs rather than using the model to create new machine designs. When asked about why, these subjects answered that the existing machine designs solved the requirements that they had and that they did not see the need to design an entirely new machine. Even though they used these predefined machine designs, the majority of the subjects reported that they had done further modifications to the existing machine designs manually in CAD. The modifications typically consisted of changing diameter of screw holes, creating new attachment types for different tools or encasing the machine design in some sort of casing.

Did you understand the different parameters of the model? The majority of the subjects answered that they understood the main parameters such as axis type and axis length, but struggled to understand how the properties controlling the drivetrain worked. They speculated over how it was hard to differentiate between what parameters would yield a drivetrain with high resolution, or what parameters would yield a drive train with high torque. The subjects commented that it was hard to know what type of drivetrain would be appropriate for different applications; e.g what should the parameters be for actuators that are to be used for a 3D printing application, or what are appropriate parameters for a machine that is to be used for a milling application. One of the subjects made comments about how this could be solved by abstracting these parameters to a higher level. For example, the axis generator could "high-torque, "fast", "slow" or "high precision", or even, parameters representing application specific information such as 3D printing, milling or laser cutting. However, we found that even if the subjects did not fully understand how the different parameters inflicted the model, they were able to gain understanding of this by changing the parameters and observing the visual CAD results, or in some cases, fabricating the actuator to evaluate the how the different parameters inflicted the design.

Did your machine give you capabilities you did not have access to before? Here, almost all of the subjects made remarks about the upside of having the output of the model being fabricatable. Some of the subjects had experience with making machines prior to using the model and explained how they felt overwhelmed about finding correct parts for a machine design and going through the time intensive process of sourcing them. They explained how the model constrained them to a design space where they were limited in these choices and how they felt empowered by being able to go into a Makerspace, use the model to generate the parts they needed, and use a CNC milling machine to fabricate the parts.

Was there any capabilities you missed in the model?

Following the thread of understanding the different parameters of the model, some of the subjects found it difficult understanding all of the parameters that the model exposed to them, and wanted a more high-level user interface for creating instances of actuators. Some of the subjects also found it difficult to integrate the actuators they generated into robust machine compositions, and as an answer to this, they wanted the model to have a suggestive feature where they could input the type of machine they wanted, for example a 3D printer with a given work envelope, and have the model generate a suggestive composition of such a machine.

Finally several of the subjects also made comments about how they found it challenging to integrate electronics and control to the machines they designed. They suggested that when modelling the machine in Rhino, all the information needed to model the kinematic representation of it is available and that it should be utilized into a fully integrated tool where this information was used to generate the control infrastructure dynamically as the machine is being designed.

Did you do any modifications to the model? Several of the subjects had either modified the model directly or by giving feedback through the repository. As discussed in the previous subsection, one of the users in the study had also ported the model to another CAD tool. When asked about his motivation for doing this, the subject commented that this was due to him having more proficiency and knowledge about using this particular tools. The subject explained that he had studied the Grasshopper models data-structure, and implement a Fusion360-version based on this structure. As the interviews proceeded, we found that several users had used the Fusion360-model to design and implement different machines, rather than the original Grasshopper model. When asked about why they preferred the one over the other, they answered that this was due to either the licensing cost of Rhino (Fusion360 is free for hobbyists), or because they were more comfortable or had more experience in using Fusion360. One of the subjects also explained that he wanted to port the model to a browser-based application, freeing the model from a licensed software-paradigm and enabling it as a cross-platform open-source browser application.

Additionally, two of the subjects made remarks about the robustness of the output design. They commented on how the original carriage design did not suffice in high-torque application. Based on this they modified the model to generate a carriage design that used a different bearing-pattern to compensate for friction. They had contributed this design back to the original repository as an alternative for applications where more torque is required.

How robust was the machine you built, and what did you use it for? We asked several questions about how useful the machines the different subjects implemented were, together with questions about what they were using their machines. Several of the subjects responded that the machines they

built were implemented with a end-goal of experimentation and that they built the machines to familiarize themselves with digital fabrication tools. Furthermore, we also found several examples of machines being put to commercial use. Two of the subjects had used the Fabricatable Axis to design large-format CNC milling machines that were actively being used in several Makerspaces. Two of the subjects had been commissioned to design and build CNC mills for different companies. When asked about how these companies in turn had used the machines in question, or weather the subjects looked upon their implementation as successes, they commented that the machines were now being used on a daily basis my the companies, but that they had performed several maintenance runs to further improve the machines robustness. They further commented that building machines from modularized machine components comes at a price in terms of rigidity. They speculated over how the requirements of precision and rigidity is not necessarily always the most important end-goal in a machine building process, and that a hobbyist building a machine will often have very different requirements for repeatability and precision then an industrial endeavour.

Why are you interested in building machines? A common answer presented itself throughout the group of subjects: to them building machines was a learning curriculum and a practical access point to gain knowledge about how to both use digital fabrication machines and to gain knowledge about the driving engineering principles behind machine design. The Fabricatable Axis worked as an onboarding artefact that enabled them to learn about these topics. Two of the subjects worked as teachers in vocation schools where several machines were built and implemented in the schools workshop, and used on a daily basis by the students. The subjects underlined the importance of fabricatable design as a learning curriculum. The machines being used at the schools were both made and maintained by the students. The subjects reflected over how valuable it was for the students to be in control of almost the entire supply chain of the parts that were used in the machines, and how, by having the machines as fabricatable designs, the students could constantly add new parts, fix broken parts and make modifications to the machines.

Apart from building machines to gain knowledge, several subjects made comments about how Fabricatable Axis was a cheap and easy access point for them to expand their digital manufacturing capabilities. This was in particular important for the users that needed machines for commercial use. They saw the Fabricatable Axis as a mean to gain access to machines that they previously did not have access to. These users also reflected over what types of machines they saw as valuable in this context. They saw Makerspaces, and the machines that Makerspaces provide, as an access point to common digital fabrication machines (laser cutters, CNC mills and 3D printers), whilst they could use the Fabricatable Axis to create more specialized machines that they needed in their practice. For example, one of the subject was working on creating an automated anodizing platform based on a linear axis created by our model.

7.2 Interview summary

The interviews shows that all the subjects of the study were able to use the model to implement a working mechanical actuator. The subjects made positive remarks about the tools ability to generate parts that could be assembled into working electromechanical actuators. They underlined the importance of fabricatable designs and how this released them from the otherwise complicated process of choosing and sourcing parts. As some of the subjects speculated over, the tool has unreleased potential when it comes to aiding them in being able to create fully functional machine compositions. The subjects especially pointed out that suggestive machine compositions and automatic control system generation are features that could significantly broaden their ability to design and implement customized machine designs.

8 Results & interpretations

In this section we will interpret the results of our study in light of our original research questions:

RQ1

The collected data indicates that our implementation of The Fabricatable Axis in Grasshopper is a success in light of our first research question. It does encapsulate domain expertize of a machine designer, and it enables less experienced machine builders to use this knowledge to implement computer controlled linear motion and machines with different properties. Our constructed scenario show how the model can be used to implement a machine that in turn can be used for a specific digital fabrication workflow. The subjects from our interview study also show that users at different levels of domain expertise were able to use the model to design and implement a mechanical linear actuator, and in some cases integrate this actuator into complete machine configurations.

When we approached the subjects about whether they understood all of the input parameters of the model, the majority of the users answered that they understood the parameters, but not always what parameters would yield best results for particular machine applications. This was especially the case with the drivetrain properties, and how these properties could be used to generate gear ratios that were best suited for specific machine applications. However, due to the fast iteration cycle that the model provides, they were able

to physically test different parameters by designing actuators with different gear ratios, fabricating them, and testing them in a specific application.

We believe that one of the critical parts of our success lies in the Grasshopper modelling language's ability to make models using block diagrams. Several of the developers of the model did not have domain expertise in software development, but through this language they where able to create models for this application. This type of modelling is similar to other graphical modelling tools such as Simulink [22] or LabView [23]. Our earlier research [24] shows that Simulink, another graphical programming language, can speed up Agile development within the automotive industry. Why these types of modelling tools seem to give a good level of abstraction, and thus making them popular for users with limited domain knowledge, can be an attractive direction for further modelling research.

In addition we also see the role of visualization as vital in not only the user interaction that the model provides, but also in the development of the model itself. As the model was developed, its output was constantly rendered as a CAD model in the Rhino environment. We could rely on this visualization for debugging how the geometry was calculated and in understanding how the data was modified and changed as it was piped through the different clusters of the model. This type of development is similar to other visualization-dependent programming tools such as Scratch [25], that enables children to implement program logic, or Houdini [26], which is a programming tool for visual designers. We believe that this type of visual feedback in programming can play an important role in programming and debugging of complex software systems where the developers of the programs might be domain experts in other fields, like machine design, and still be able to implement and develop relatively complex software solutions like The Fabricatable Axis.

RQ2

In our interview study, all of the subjects stated that they where able to use the model to customize and generate a version of The Fabricatable Axis.

Not all of the subjects had used the model to conceptualize and design new machine designs. When approached about why they had not done so, the subjects provided two insights: (1) They where able to find completed machine designs of the specific machine they were interested in the Github Repository and thereby they did not see the need for implementing new types of machine designs. (2) they still found it too challenging to incorporate the instances of the a Fabricatable Axis into a complete machine design. In light of this finding we conclude that further development of the model is necessary to enable this group to be able to successfully use The Fabricatable Axis to implement complete machines. As some of the subjects suggested an interesting direction of this future work would be to implement a higher level model that aims to encapsulate a complete machine design into a model. This model could then potentially be used to derive machines based on desired workflow of a machine.

The subjects who had used the model to design and implement complete machines gave valuable insight about a key feature of the model: By constraining a machine designer to a constrained vocabulary of parts and a constrained design language where different machines can be expressed by using a parameterized modular component, it reduces the threshold for designing new machines. An important facet in machine design, and mechanical design in general, is having prior knowledge about different mechanical principles and what types of parts are available to be used when developing a novel design. These subjects reported on how the model relieved them from the tedious process of finding and sourcing appropriate parts and rather letting them focus on designing machine compositions consisting of variations of The Fabricatable Axis.

The interviews show that the model aids in different ways for different types of users. For users who already have experience in machine building, the model functions as a tool for quickly conceptualizing and prototyping different types of motion platforms. For less experienced practitioners, the model acts as a gateway for learning about machine building practices, and to familiarize themselves with how such machines can be implemented. We observed that users who use our model to generate their own machines are more likely to further customize and expand the features of the machine throughout its life cycle. By lowering the threshold for implementing customized motion platforms, we see that our users are more free to experiment and customize their specific fabrication workflows, rather than spending time on implementing the motion platform.

The model follows the lineage of other visual modelling tools. Through our study we have seen the importance of how visual results are aiding our users in using the model. Several of the subjects in the interviews had little to no experience in either using Grasshopper as a modelling tool, or in machine design best practices. They did not necessarily understand what the different input parameters of our model did, or how they worked. However, by adjusting the parameters, and seeing visually how the output CAD model was changed and adapted, they where able to gain understanding as they where interacting with the model. We see this as an interesting insight in how graphical modelling tools can aid users in understanding and utilizing complex systems.

Finally, an important design feature of our implementation is to keep the "source-code" of the model transparent and accessible to the user, thus allowing the user to go beyond the high-level interface and to modify and change the output and inherent function of the model. We consider this impor-

tant for our user group since the majority of them are fluent CAD users and they often have specific requirements on top of what The Fabricatable Axis provides. As we found in our interviews, several of the subjects had modified the models original design to further facilitate their needs. One of the subjects had even ported the model to a different CAD tool using the Grasshopper model as a blueprint for his new design. We see this as an interesting insight in how high-level abstraction certainly can be useful, but how it also is important to have the lower-level "logic" of the abstraction accessible, allowing users to both study and edit its logic and functionality.

In light of our second research question we conclude that our model plays a nuanced role in how users adapt it and use it to create new machines. For users who have some prior knowledge in machine design the model acts as a good starting ground that allows them to express and conceptualize machines rapidly. For less experienced users, it serves as a learning curriculum that allows them to explore and learn about best practices in machine design, but not necessarily as a tool that lets them easily deploy customized digital fabrication tools. As the interview summary concludes, an abstraction layer one level above the Fabricatable Axis where the composition of a machine is suggested or derived from a workflow, could potentially benefit these users in doing so.

8.1 Threats to validity

To classify potential threats to validity, and reason about our corresponding mitigation strategies, we follow the scheme proposed by Runeson and Höst [27].

(a) *Internal* To build a machine using a framework that one has built is a potential thread of internal validity since one might be able to build the machine only because one has built the framework. Therefore, the interviews play an essential role where others are using the framework. In terms of threats to internal validity regarding the interview, we followed a systematic approach in setting up the study and best practice guidelines for both data collection and analysis [27]. Moreover, the interview questions might have influenced the participants in how good they thought the model was. To mitigate this risk, we spent time discussing how to phrase the questions and types of questions to avoid.

(b) *External* Generalizability is inherently limited for case studies. All interviews were done within the FabLab community. The reason for this is that this community is an early adapter of building Fabricatable Machines. They are used to play and build; this might not be the case in general. However, the group of people who want to build their own machines are increasing as the number of fab-labs around the world is being built.

(c) *Construct* Two of the authors have prior experience with the building machines and been key contributors of the model, which we leverage to ensure construct validity. Thus, the interview situation was informal and characterized by mutual trust. Furthermore, the interview guide was refined through multiple iterations.

(d) *Reliability* To ensure reliability, we used observer triangulation during the interviews. The first author conducted the interviews via Zoom, and the second watched the interview and discussed them with the first author. What we missed here was the ability for the second author to ask follow-up questions for additional clarifications; however, that was not a problem in our case, since the interview was easy to understand.

The first author did the coding and discussed it with the second author. We got a good understanding by listening to the interviews, so transcriptions did not appear to offer a positive return on time we would have to invest. Even with the small number of participants, we feel we got a good saturation [28] due to the relatively homogeneous group of people building machines.

9 Discussion

Our study indicates that our model enables a diverse set of practitioners to implement digitally controlled motion through a high-level design tool. Figure 14 show a selection of some of the machines that have been built throughout the project. Throughout the project we have observed examples of that not only conventional machines were built, but newer and unexpected machines like multi-headed 3D printers, light painting machines, machines fitted to rooftops of cars, drawing robots and robots for extruding pancake-batter. This paints a picture of how the users in this context have a significantly different ambition then what we typically see in their industrial counterpart. As the threshold for accessing and utilizing computer controlled motion lowers, we expect that the field of digital fabrication will see more of this type of diversity and personalization. The project show that there is certainly an interest in not only in using digital fabrication machines to make parts, but that it is also an interest in further customizing these machines for new and diverse types of workflows. As the field of fabrication engages more diverse users, we need tools that reflect that diversity. We believe that the concepts and methods of model-driven engineering can play an important role in this.

Furthermore, our interviews indicates that users who make their own machine are likely to further customize and modify the machine throughout its life cycle. It seems that the ability to design, manufacture and replace parts on the fly, gives machine designers a large esteem of confidence. We speculate that many of the more unique designs seen in the project is a direct consequence of the rapid prototyping iterations and the quick turn around that our model provides.

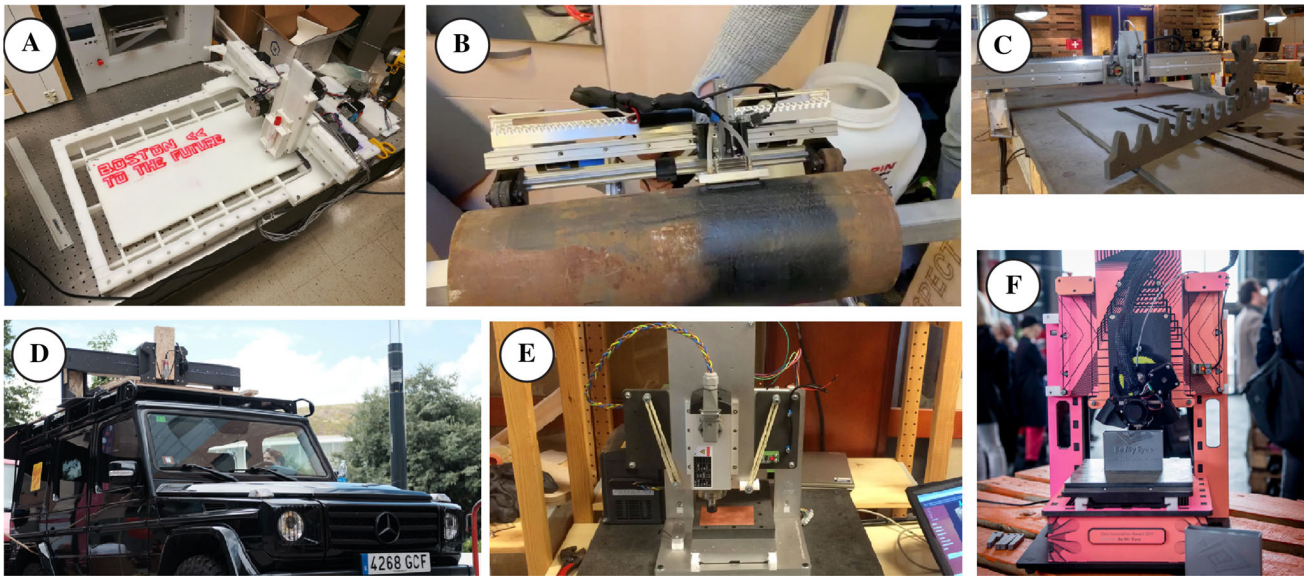


Fig. 14 Examples of machines built with the Fabricatable Axis. **a** Pen-Plotter, **b** Pipe-inspection robot, **c** Large-Format CNC mill, **d** CNC mill on top of a Gelendewagen, **e** PCB milling machine, F. 3D printer with silk screen decals

To enable more end-users to experiment with digital fabrication tools there is a need for high-level infrastructure for creating these new types of machines. We see the body of work presented in this article as an important first step towards this goal. The Fabricatable Axis provides the means to generate customized motion platforms. However, as we saw in the interviews, there is still challenges to be resolved in order to enable more types of end-users to experiment with machine building. Understanding what types of platforms and compositions are suited for different fabrication workflow is a difficult task for less experienced machine designers, and thus there is a need for a tool that aids these users in this task. We also see that when combining different actuators into compositions in a CAD tool, we have all the information necessary to extract the kinematic representation of a motion platform and utilize this to automatically generate the control infrastructure necessary to control the machine. We see this as an important future goal of our work.

10 Related work

As automated tools and fabrication technology are becoming widespread, aiding how users can more readily adapt and use these tools have become a salient topic in computer science. Literature shows the benefits and the need for allowing end-users to customize and tailor their own machine interactions. For example, Tian modified a customized lathe with its own graphical user interface to explore how novel users could learn about machining through haptic feedback. Teibrich et al. [29] added an additional end-effector to an existing 3D printer to be able to use the same platform for subtractive and additive processes. Adding functionality to 3D printers for interactive or hybrid fabrication has also been achieved [6,7]. Yet 'hacking' limits researchers to modifications that can be made to only existing platforms.

10.1 End-user CAD tools

We are indebted to Hofman et al.'s framing of the need for CAD tools that use an end-user program perspective [30]. In their framework PARTs, they provide users with familiarity but not expertise in CAD with tools to more easily express and reuse 3D design intent. In 1992, Gantt and Nardi wrote about the different roles different levels of users of CAD may take on within a single organization and how 'local developers' provide support [31]. Since then, the role of 'local developer' has perhaps mostly moved online, but we believe the 'gardener' role now has also taken on the development and maintenance of third-party CAD plugins (E.g. for Autodesk's app store or Food4Rhino [32]). Grafter is another example of an end-user CAD tool; it encapsulates the complexity of mechanisms to ease their reuse [33]. Drawing from this related work, the work presented in this paper aims to encapsulate machine design expertise in an abstraction that is more accessible to an end-user.

10.2 Fabrication toolkits

Encapsulating subject expertise into toolkits is an active topic in fabrication and robotics. In [34], the authors presented a toolkit for modularizing machines into linear and rotary

actuators made out of cardboard, a modular control infrastructure for controlling the actuators implemented in Python. However, cardboard is a limiting material to build functional machines, and our approach differs in its ability to create parts made out of high-performance materials. We are inspired by the open-source work by Holland et al. who released the Soft Robotics Toolkit (SRT) [35]. Oh et al. encapsulate information on mechanisms into their toolkit FoldMecha [36], enabling an end-user approach to adding motion to papercraft. Mehta et al. enable the design of robots from functional specifications in their work on end-to-end systems for printable robots [37]. In [38], the authors provide an open-source motion platform for user defined end-effectors. Our implementation differs from this related work in its application to digital fabrication machines and its ability to enable customized motion platforms.

10.3 Fabrication in MDSE

CAD tools have been subject to prior work in the modelling community. Specifically, [13] presented a MDE driven approach for generating meta-models for capturing designer intent and transforming it into parametric models. [39] presented a feature language for sculptured modelling. In [40], the authors present a UML approach for modelling part assemblies. In [41], Royo et al demonstrates models intended for integrating processes for form generation, digital fabrication and material computation. The body of work presented in this paper is motivated by these efforts but focuses on machine design rather then creating a generalized abstraction of intent-to-CAD creation. We argue that this specificity is necessary for a complex mechanism like a linear actuator. Furthermore in [42] the authors presented a framework for going from conceptual code detailing how a integrated circuit design should function, to generate the necessary PCB-design files to manufacture and implement the complete circuit design. Our body of work is motivated by the way this work uses high-level parameters and block-based modelling to transform user intent into fabricatable designs. However, milling circuits is a significantly different process then fabricating parts for a mechanical actuator and our work differs in the way our model optimizes the fabrication files for this process.

10.4 Component-based design

The Grasshopper community is rich of plugins and 3rd-party utilities that can be utilized for different purposes. For example, Firefly [43] is a set of tools that can be used in Grasshopper to communicate with low-level hardware devices like Arduinos and Raspberry Pi. Taco ABB [44] includes tools that encapsulates ways for programming and visualizing toolpaths for ABB robots. Robot Components [45]is a high-level design and simulation tool for robot programming that runs in Grasshopper. The Fabricatable Axis utilizes the same type of encapsulation as these tools, but differs in its ability to both provide a design tool and a tool that prepares and optimizes the design files for fabrication with a CNC milling machine.

Furthermore, component-based design has been utilized and explored in several different fields. In particular, Koo et al. [46] contributed a system for building mechanical prototypes where users can specify high-level relationships between components such as hinges or sliders. In [47] the authors contribute a system for generating generative furniture. The system defines a grammar for fabrication rules and adds a lexical analysis for checking the feasibility of the generated parts-assemblies. Ultimately our body of work wish to provide these same techniques to machine design.

Finally, Vention [48] is an online platform that utilizes component-based tecniques to enable users to create customized machine designs. Machines in vention are created by dragging-and-dropping different machine components and assemble them into machines. These machine compositions are then shipped to the user for assembly. However, the target audience of Vention is technical users in industrial settings which differs from our goal of rapid iteration and machine implementation by users situated in less established institutions.

10.5 Open-source hardware

Our work is closely related to and builds upon other open-source machine building efforts. RepRap and the Fab@Home are foundations in the open-source machine space [49–51]. New machines such as Maslow CNC [52] or the Prusa 3D printer embody what we believe to be best practices for online and distributed community development of machines. Companies who sell machine-building parts in low-volume such as Openbuilds [53] are reducing the need for open-source designs to not rely on external supply chains (this is in contrast to their industrial counterparts such as RexRoth [54] which are difficult for individuals to source or buy). However, we differ from most of these existing machine building efforts in that Fabricatable Machines is focused on motion platforms rather than specific end-effectors.

11 Conclusion and future work

In this paper we presented the Fabricatable Axis, a high-level model that can be utilized to design and implement customized digital fabrication machines. Additionally we presented how this model can in turn be used to create Fabricatable Machines. Through a community of machine builders we have seen how they were able to successfully implement

bespoke digital fabrication tools. As a future road map in our research we propose the following work lines:

Automatic Control system generation As users are building up the machine composition visually in the CAD environment, we have access to all the information needed to generate the necessary control topology for controlling the machine itself. As discussed earlier in the paper, it is easy for users to use off-the-shelf controllers for common 2- and 3-axis machine configurations. However, creating more complex kinematic chains is non trivial. An interesting future direction of this work would be to generate the control topology simultaneously from the information that is available as the machine is generated by our model.

Automatic machine composition generation As mentioned in the discussion, an interesting future direction would be to incorporate automatic machine compositions based on high-level parameters. For example, users could express what kind of workflow they want to create a machine for (3D printing, CNC cutting or turning, laser cutter, etc). Based on this information, we could be able to suggest what type of machine configuration would be best suited for the users application in terms of rigidity, speed and layout.

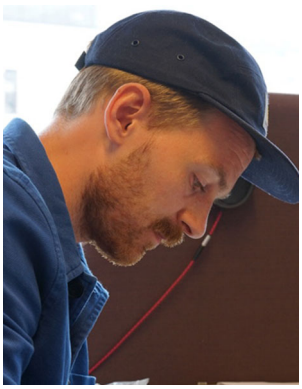
Funding Open access funding provided by Western Norway University Of Applied Sciences.

References

1. Educause ELI. 7 Things You Should Know About Makerspaces. en. (2013) https://library.educause.edu/resources/2013/4/7-things-you-should-know-about-makerspaces
2. Fablab https://www.fablabs.io/ (visited on 02/28/2021)
3. Srai, J.S., et al.: Distributed manufacturing: scope, challenges and opportunities. en. In: International Journal of Production Research 54.23, pp. 6917–6935 (2016) ISSN: 0020-7543, 1366-588X. https://doi.org/10.1080/00207543.2016.1192302. https://www.tandfonline.com/doi/full/10.1080/00207543.2016.1192302 (visited on 04/02/2020)
4. Peng, H., et al.: On-the-fly print: incremental printing while modelling'. en. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16. Santa Clara, California, USA: ACM Press, pp. 887–896 (2016). ISBN: 978-1-4503-3362-7. https://doi.org/10.1145/2858036.2858106. http://dl.acm.org/citation.cfm?doid=2858036.2858106 (visited on 07/19/2019)
5. Wang, G., et al.: xPrint: a modularized liquid printer for smart materials deposition. en. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16. Santa Clara, California, USA: ACM Press, pp. 5743–5752 (2016). ISBN: 978-1-4503-3362-7. https://doi.org/10.1145/2858036.2858281. http://dl.acm.org/citation.cfm?doid=2858036.2858281 (visited on 07/19/2019)
6. Peng, H., et al.: A 3D printer for interactive electromagnetic devices. en. In: Proceedings of the 29th Annual Symposium on User Interface Software and Technology - UIST '16. Tokyo, Japan: ACM Press, pp. 553–562 (2016). ISBN: 978-1-4503-4189-9. https://doi.org/10.1145/2984511.2984523. http://dl.acm.org/citation.cfm?doid=2984511.2984523 (visited on 07/19/2019)
7. Gao, W., et al.: RevoMaker: enabling multi-directional and functionally-embedded 3D printing using a Rotational Cuboidal Platform. en. In: Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology - UIST '15. Daegu, Kyungpook, Republic of Korea: ACM Press, pp. 437–446 (2015). ISBN: 978-1-4503-3779-3. https://doi.org/10.1145/2807442.2807476. http://dl.acm.org/citation.cfm?doid=2807442.2807476 (visited on 07/19/2019)
8. Yildirim, N., McCann, J., Zimmerman, J.: "Digital fabrication tools at work: probing professionals' current needs and desired futures". en. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. Honolulu HI USA: ACM, pp. 1–13 (Apr. 2020). ISBN: 978-1-4503-6708-0. https://doi.org/10.1145/3313831.3376621 https://dl.acm.org/doi/10.1145/3313831.3376621 (visited on 02/09/2021)
9. Brambilla, M., Cabot, J., Wimmer, M.: Model-driven software engineering in practice, Second Edition. Synthesis Lectures on Software Engineering. Morgan & Claypool Publishers (2017). https://doi.org/10.2200/S00751ED2V01Y201701SWE004
10. Whittle, J., Hutchinson, J., Rouncefield, M.: The state of practice in model-driven engineering. IEEE Softw. **31**(3), 79–85 (2014). (**10.1109/MS.2013.65**)
11. Mohagheghi, P. et al.: "MDE adoption in industry: challenges and success criteria". In: Models in Software Engineering, Workshops and Symposia at MODELS 2008, Toulouse, France, September 28 - October 3, 2008. Reports and Revised Selected Papers. Ed. by Michel R. V. Chaudron. Vol. 5421. Lecture Notes in Computer Science. Springer, pp. 54–59 (2008) https://doi.org/10.1007/978-3-642-01648-6_6
12. Dalibor, M. et al.: "Model-driven systems engineering for virtual product design". en. In: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C). Munich, Germany: IEEE, pp. 431–436 (Sept. 2019). ISBN: 978-1-72815-125-0. https://doi.org/10.1109/MODELS-C.2019.00069. https://ieeexplore.ieee.org/document/8904696/ (visited on 12/15/2020)
13. Scheffler, R. et al.: "Modelling CAD models - method for the model driven design of cad models for deep drawing tools:" en. In: Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development. Rome, Italy: SCITEPRESS - Science, pp. 377–383 (2016). ISBN: 978-989-758-168-7. https://doi.org/10.5220/0005799403770383. http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0005799403770383 (visited on 03/31/2020)
14. Fossdal, F.H. et al.: "A parametric model for creating customized fabrication machines". In: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems. MODELS '20. event-place: Virtual Event, Canada. New York, NY, USA: Association for Computing Machinery, pp. 143–153 (2020). ISBN: 978-1-4503-7019-6. https://doi.org/10.1145/3365438.3410960
15. Fossdal, F.H. et al.: "Fabricatable machines: a toolkit for building digital fabrication machines". en. In: Proceedings of the

Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction. Sydney NSW Australia: ACM, pp. 411–422 (2020). ISBN: 978-1-4503-6107-1. https://doi.org/10.1145/3374920.3374929. http://dl.acm.org/doi/10.1145/3374920.3374929 (visited on 03/03/2020)
16. McNeel. Grasshopper. en. https://www.grasshopper3d.com/ (visited on 02/23/2021)
17. Rhino 6 for Windows and Mac https://www.rhino3d.com/ (visited on 05/06/2020)
18. Hevner, A.R, March, S. T., Park, J., Ram, S.: Design science in information systems research. MIS quarterly, 75–105 (2004)
19. Engström, E. et al.: How software engineering research aligns with design science: a review. In: arXiv:1904.12742 [cs] (2019). arXiv:1904.12742. (visited on 05/22/2020)
20. Fabricatable machines repository https://github.com/fellesverkstedet/fabricatable-machines
21. Autodesk. Fusion360. en-US. Library Catalog: www.autodesk.com. https://www.autodesk.com/products/fusion-360/overview (visited on 05/22/2020)
22. MatLab. Simulink - simulation and model-based design. https://www.mathworks.com/products/simulink.html (visited on 03/01/2021)
23. National Instruments. LabView. https://www.ni.com/en-no/shop/labview.html (visited on 03/01/2021)
24. Eliasson, U et al.: "Agile model-driven engineering in mechatronic systems - an industrial case study". In: Model-driven engineering languages and systems. Ed. by Juergen Dingel et al. Vol. 8767. Series Title: Lecture Notes in Computer Science. Cham: Springer, pp. 433–449 (2014). ISBN: 978-3-319-11652-5 978-3-319-11653-2. https://doi.org/10.1007/978-3-319-11653-2_27. http://link.springer.com/10.1007/978-3-319-11653-2_27 (visited on 02/28/2021)
25. Lifelong Kindergarten Group Scratch - Imagine, Program, Share US. https://scratch.mit.edu/ (visited on 03/01/2021)
26. SideFx. Houdini https://www.sidefx.com/products/houdini/ (visited on 03/01/2021)
27. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. In: Empirical Software Engineering 14(2), 131–164 (2009). ISSN: 1382-3256, 1573-7616. https://doi.org/10.1007/s10664-008-9102-8. http://link.springer.com/10.1007/s10664-008-9102-8 (visited on 05/21/2020)
28. Saunders, B. et al.: Saturation in qualitative research: exploring its conceptualization and operationalization In: Quality & Quantity 52.4, pp. 1893–1907 (2018). ISSN: 0033-5177, 1573-7845. https://doi.org/10.1007/s11135-017-0574-8. http://link.springer.com/10.1007/s11135-017-0574-8 (visited on 05/22/2020)
29. Teibrich, A. et al.: Patching Physical Objects. In: Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology - UIST '15 Daegu, Kyungpook, Republic of Korea: ACM Press, pp. 83–91 (2015). ISBN: 978-1-4503-3779-3. https://doi.org/10.1145/2807442.2807467. http://dl.acm.org/citation.cfm?doid=2807442.2807467 (visited on 07/24/2019)
30. Hofmann, M. et al.: "Greater than the Sum of its PARTs: expressing and reusing design intent in 3D models In: Proceedings of the 2018 CHI conference on human factors in computing systems - CHI '18. Montreal QC, Canada: ACM Press, pp. 1–12 (2018). ISBN: 978-1-4503-5620-6. https://doi.org/10.1145/3173574.3173875. http://dl.acm.org/citation.cfm?doid=3173574.3173875 (visited on 07/19/2019)
31. Gantt, M., Nardi, B.A.: Gardeners and gurus: patterns of cooperation among CAD users. In: Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '92. Monterey, California, United States: ACM Press, pp. 107–117 (1992). ISBN: 978-0-89791-513-7. https://doi.org/10.1145/142750.142767. http://portal.acm.org/citation.cfm?doid=142750.142767 (visited on 07/24/2019)
32. food4rhino. Food4Rhino. Text. (2019) https://www.food4rhino.com/ (visited on 07/29/2019)
33. Roumen, T.J., Müller, W., Baudisch, P.: Grafter: remixing 3D-printed machines. In: Proceedings of the 2018 CHI conference on human factors in computing systems. CHI '18. New York, NY, USA: ACM, 63:1–63:12 (2018). ISBN: 978-1-4503-5620-6. https://doi.org/10.1145/3173574.3173637. (visited on 01/08/2019)
34. Peek, N. et al.: Cardboard machine kit: modules for the rapid prototyping of rapid prototyping machines. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. CHI '17. New York, NY, USA: ACM, pp. 3657–3668 (2017). ISBN: 978-1-4503-4655-9. https://doi.org/10.1145/3025453.3025491
35. Holland, D.P. et al.: The soft robotics toolkit: strategies for overcoming obstacles to the wide dissemination of soft-robotic hardware. In: IEEE Robotics & Automation Magazine 24.1, pp. 57–64 (Mar. 2017). ISSN: 1070-9932. https://doi.org/10.1109/MRA.2016.2639067. http://ieeexplore.ieee.org/document/7862175/ (visited on 07/25/2019)
36. Oh, H. et al.: FoldMecha: exploratory design and engineering of mechanical papercraft. In: Proceedings of the Tenth International Conference on Tangible, Embedded, and Embodied Interaction - TEI '17. Yokohama, Japan: ACM Press, pp. 131–139 (2017). ISBN: 978-1-4503-4676-4. https://doi.org/10.1145/3024969.3024991. http://dl.acm.org/citation.cfm?doid=3024969.3024991 (visited on 07/25/2019)
37. Mehta, A.M. et al.: Robot creation from functional specifications. In: Robotics Research. Ed. by Antonio Bicchi and Wolfram Bur-gard. Vol. 3. Cham: Springer (2018) pp. 631-648. ISBN: 978-3-319-60915-7 978-3-319-60916-4. https://doi.org/10.1007/978-3-319-60916-4_36. http://link.springer.com/10.1007/978-3-319-60916-4_36 (visited on 07/25/2019)
38. Vasquez, J. et al.: Jubilee: an extensible machine for multi-tool fabrication. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. CHI '20. event-place: Honolulu, HI, USA. New York, NY, USA: Association for Computing Machinery, pp. 1–13 (2020). ISBN: 978-1-4503-6708-0. https://doi.org/10.1145/3313831.3376425
39. Au, C.K., Yuen, M.M.F.: A semantic feature language for sculptured object modelling. In: Computer-Aided Design 32.1, pp. 63–74 (2000). ISSN: 00104485. https://doi.org/10.1016/S0010-4485(99)00085-8. https://linkinghub.elsevier.com/retrieve/pii/S0010448599000858 (visited on 04/01/2020)
40. Hochgeladen, R., Vyas, P.: Entwurf komplexer Zusammenbauten Rainer Hochgeladen Prachi Vyas Senden mit UML. CAD-CAM REPORT
41. Duro-Royo, J., Oxman, N.: Towards fabrication information modeling (FIM): four Case models to derive designs informed by multi-scale trans-disciplinary data. In: MRS Proceedings 1800 (2015), mrss15–2138549. ISSN: 0272-9172, 1946-4274. https://doi.org/10.1557/opl.2015.647. https://www.cambridge.org/core/product/identifier/S1946427415006478/type/journal_article (visited on 04/28/2020)
42. Ramesh, R. et al.: Turning coders into makers: the promise of embedded design generation. In: Proceedings of the 1st Annual ACM Symposium on Computational Fabrication - SCF '17. Cambridge, Massachusetts: ACM Press, pp. 1–10 (2017). ISBN: 978-1-4503-4999-4. https://doi.org/10.1145/3083157.3083159. http://dl.acm.org/citation.cfm?doid=3083157.3083159 (visited on 03/13/2020)
43. Firefly. Text. (2010). https://www.food4rhino.com/en/app/firefly (visited on 07/06/2021)
44. Taco ABB. Text. (2016). https://www.food4rhino.com/en/app/taco-abb (visited on 11/03/2021)
45. Robot Components. (2020). https://www.food4rhino.com/en/app/robot-components (visited on 11/03/2021)

46. Koo, B. et al.: Creating works-like prototypes of mechanical objects. In: ACM Trans. Graph. 33.6 (2014). Place: New York, NY, USA Publisher: Association for Computing Machinery. ISSN: 0730-0301. https://doi.org/10.1145/2661229.2661289
47. Lau, M. et al.: Converting 3D furniture models to fabricatable parts and connectors. In: ACM Trans. Graph. 30(4) (2011). Place: New York, NY, USA Publisher: Association for Computing Machinery. ISSN: 0730-0301. https://doi.org/10.1145/2010324.1964980
48. Manufacturing automation, simplified — vention. https://vention.io/ (visited on 11/03/2021)
49. Reprap. RepRap. (2019) https://reprap.org/wiki/RepRap (visited on 07/25/2019)
50. Malone, E., Lipson, H.O.D.: Fab@Home: the personal desktop fabricator kit. Rapid Prototyp. J. **13**(4), 245–255 (2007)
51. Vilbrandt, T. et al.: Universal desktop fabrication. In: Heterogeneous objects modelling and applications. Ed. by Alexander Pasko, Valery Adzhiev, and Peter Comninos. Berlin, Heidelberg: Springer, pp. 259–284 (2008). ISBN: 3-540-68441-7 978-3-540-68441-1. http://dl.acm.org/citation.cfm?id=1806158.1806169
52. Maslow. Maslow. US. (2019). https://www.maslowcnc.com/ (visited on 07/25/2019)
53. OpenBuilds. OpenBuilds. (2019). https://openbuilds.com/ (visited on 07/25/2019)
54. Bosch. Rexroth. (2019). https://www.boschrexroth.com/en/xc/ (visited on 07/25/2019)



Frikk H. Fossdal is an engineer and PhD candidate at the Western Norway University of Applied Sciences. His research focuses on unconventional digital fabrication tools, open-source hardware and human–computer interactions. Prior to his PhD studies, Frikk ran his own industrial design practice in Bergen, Norway.

Rogardt Heldal is a professor of Software Engineering at the Western Norway University of Applied Sciences. Heldal holds an honours degree in Computer Science from Glasgow University, Scotland, and a PhD in Computer Science from Chalmers University of Technology, Sweden. His research interests include requirements engineering, software processes, software architecture, cyber-physical systems, machine learning, and empirical research. Many of his research projects are performed in collaboration with the industry. Currently, he is part of a large national project to monitor the ocean, to build a smart ocean platform.

Jens Dyvik is a designer specialized in global collaboration and local manufacturing. He works with emotional connections between people and products and aims to create services and products that help make those connections meaningful.

Adrian Rutle is professor at the department of computer science, electrical engineering and mathematical sciences at the Western Norway University of Applied Sciences. His research focuses on the application of theoretical results from the field of model-driven software engineering. His work has recently focused on modelling and simulation for smart robotics, multi-level modeling, patient workflows and their verification, as well as modeling and simulation of interleaving business processes incorporated with decision-making assistance. For more information, please visit https://www.hvl.no/person/?user=Adrian.Rutle.