Grafter: Remixing 3D-Printed Machines

Thijs Jan Roumen, Willi Mueller and Patrick Baudisch
Hasso Plattner Institute
Potsdam, Germany
{firstname.lastname}@hpi.de

ABSTRACT

Creating new 3D printed objects by recombining models found in hobbyist repositories has been referred to as "remixing." In this paper, we explore how to best support users in remixing a specific class of 3D printed objects, namely those that perform mechanical functions.

In our survey, we found that makers remix such machines by manually extracting *parts* from one parent model and combine them with *parts* from a different parent model. This approach often puts axles made by one maker into bearings made by another maker or combines a gear by one maker with a gear by a different maker. This approach is problematic, however, as parts from different makers tend to fit poorly, which results in long series of tweaks and test-prints until all parts finally work together.

We address this with our interactive system *grafter*. Grafter does two things. First, grafter largely automates the process of extracting and recombining mechanical elements from 3D printed machines. Second, it enforces a more efficient approach to reuse: it prevents users from extracting individual parts, but instead affords extracting groups of mechanical elements that already work together, such as axles *and* their bearings or *pairs of* gears. We call this *mechanism-based* remixing. In a final user study, all models that participants had remixed using grafter could be 3D printed without further tweaking and worked immediately.

ACM Classification: H.5.m. [Information interfaces and presentation]: Misc.

Keywords: fabrication, 3D printing, remixing.

General terms: Design, Human factors.

INTRODUCTION

A fast approach to creating new objects for 3D printing is to recombine models found in publicly available sources, such as hobbyist repositories (e.g. Thingiverse.com). This process has been referred to as "remixing" and Oehlberg et al. find it to be an increasingly popular trend [23].

In this paper, we explore how to best support users in remixing a specific class of 3D printed objects, namely those that perform mechanical functions, also known as *machines.* Figure 1a shows an example of such a remixing attempt. The user is trying to make a test tube centrifuge that could, for example, be used to extract spirulina from aquarium water. Here the user has identified a parent machine that offers a crank with gearbox, a second parent machine capable of redirecting rotary movement, and a third parent model with a test tube holder. The challenge now is to make the mechanical elements from the three parent models work together.

CHI 2018, April 21–26, 2018, Montreal, QC, Canada
© 2018 Association for Computing Machinery
.ACM ISBN 978-1-4503-5620-6/18/04…$15.00
https://doi.org/10.1145/3173574.3173637

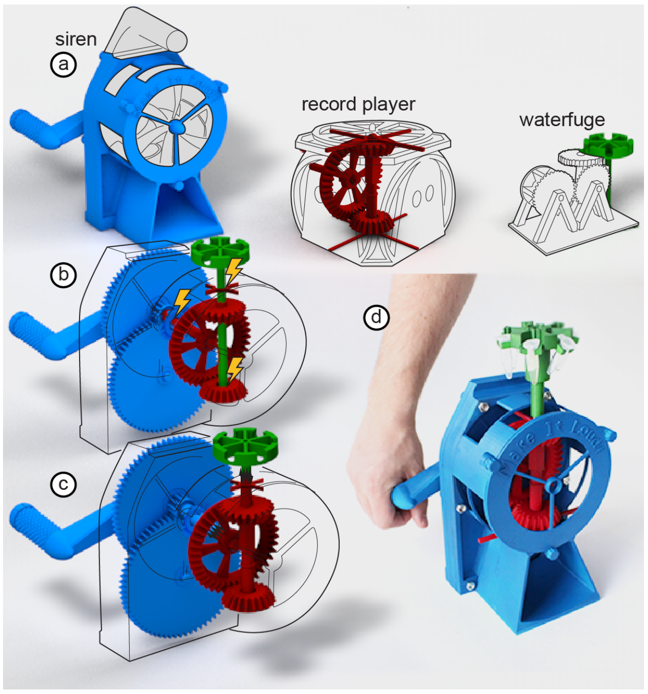

Figure 1: Our system *grafter* allows users to remix mechanical machines. Here we use it to create a centrifuge for test tubes. (a) We start by identifying three parent models from a hobbyist repository. (b) If we used the workflow commonly practiced by makers today, we would extract relevant *parts* and recombine them into the desired configuration. Making an axle spin correctly in a bearing from a different parent model, however, requires time-consuming tweaks & test-prints. (c) Grafter, in contrast, extracts groups of elements that already work well together (aka mechanisms), which it *fuses* with mechanisms from other parents. This achieves the same (d) final 3D printed object, but without the tweaking.

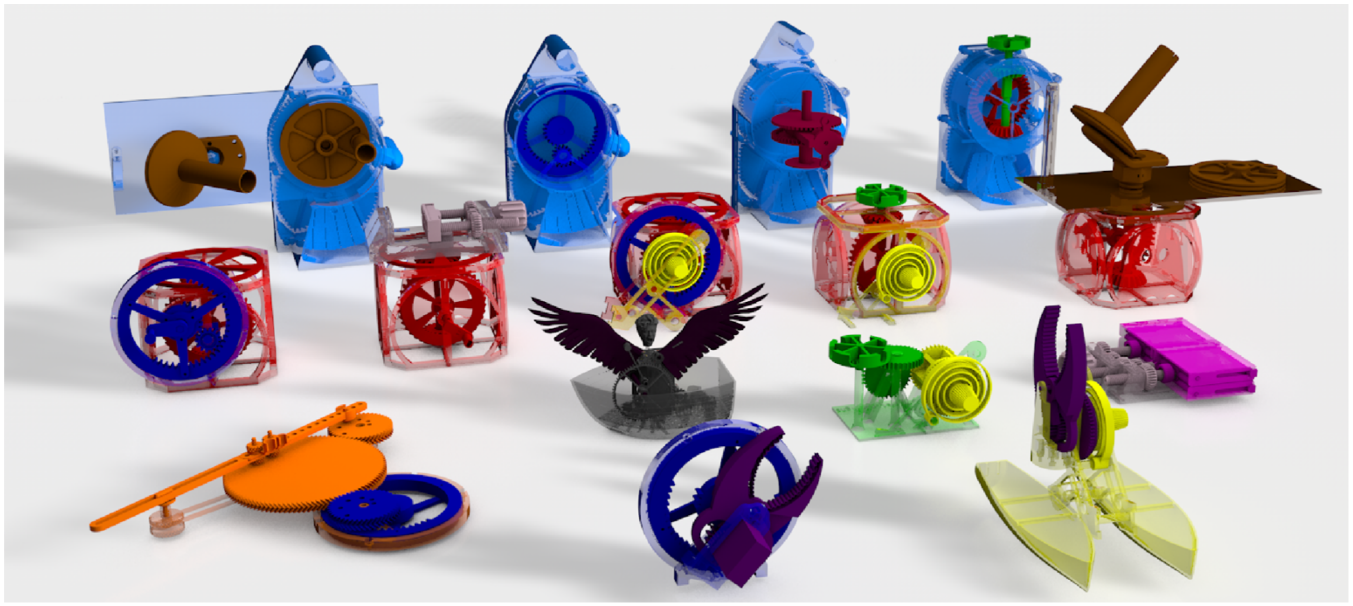

Figure 2: Some of the models we remixed using grafter, i.e., using *mechanism-based* remixing. Colors indicate the parent model from which the respective mechanism was taken. (a) All models in the top row are variations of the same actuation mechanism (a hand-cranked siren). (b) All models in the center row remix the same record player parent model, but replace different elements, i.e., actuation, transmission, or end effector. (c) Models in the front row sample a wider range of parent models.

There is no particular software support for remixing machines (and one of our two main contributions is to demonstrate such a system). In fact, in our survey, presented below, we found that makers have developed best practices for remixing machines. As illustrated by Figure 1b, this approach entails extracting relevant *parts* from each model and recombining them into the target object, here the test tube centrifuge. We will refer to this approach as *part-based reuse.*

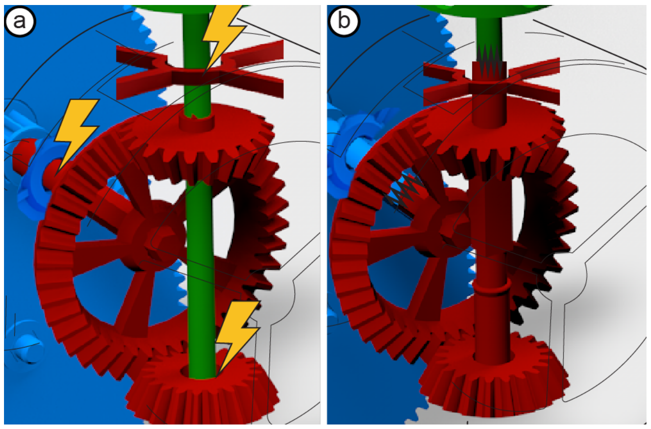

Figure 3: Close-up of Figure 1 (a) part-based remixing puts the green axles into the red bearing, which was made by a different maker. (b) Grafter, in contrast, extracts groups of elements that already work well together.

Unfortunately, part-based reuse is surprisingly inefficient. As illustrated by Figure 1b and magnified in Figure 3a, part-based reuse tries to connect moving parts made by one maker with parts made by another maker. Here it is the red axle in the blue bearing and a green axle in two red bearings. This is problematic. While part-based reuse makes sense when applied to professional mechanical engineering, where parts are standardized and thus interchangeable, the absence of standards in the maker community (see below) causes this type of connection to generally *not* fit. Getting the resulting model to work thus requires a series of tweaks to the diameters of axles/bearings and consequential test-prints, making the remixing of machines very time-consuming.

In this paper, we challenge the best practice of *part-based* reuse and instead propose what we call *mechanism-based* reuse. Rather than extracting parts, our software system *grafter* extracts groups of elements that work together (aka *mechanisms*) and recombines these into the target machine. As magnified in Figure 3b, this always puts red axles into red bearings, eliminating those configurations that traditionally require tweaking and test printing. Instead, grafter cuts up parts and fuses them together with other parts. In Figure 3b, for example, grafter spliced part of the green axle with part of the red axle (visualized as a black squiggly line between the two parts), allowing the red bearing to hold a red axle, rather than a green axle—a connection that is known to work without further tweaking.

The main benefit of such mechanism-based remixing is that it produces the same target machines as part-based remixing, such as the test tube centrifuge shown in Figure 1d. This allows the resulting models to work immediately upon the first 3D print—without the need for tweaking or test printing. This allows creating a wide range of machines quickly (Figure 2). The models need to be prepared for use in grafter, which we will discuss in the “annotating machines” section.

SURVEYS OF EXISTING REMIXING PRACTICES

The observations about "best practices" in remixing referred to above are the result of two simple surveys we conducted.

Survey 1: part-based reuse

The first survey we conducted on a set of 349 remixes, from a hobbyist repository (all models submitted to the *thingiverse remixing challenge* [27]). One of our expert engineers classified all models and found that machines accounted for only 26 (7%) of all remixes; this is expected as remixing machines is harder than remixing decorative objects. Out of the remixed machines, 13 (50%) of them were remixed from multiple parents, which is the type of remixing that is particularly difficult (which is why we support it with grafter).

The main result of our survey is that the makers of all 13 multi-parent machines had indeed remixed by means of *parts-based* reuse. Figure 4a shows such a remix. This model combines elements from three parents (red = thingiverse id:452248, green = 272132, blue = 1037341) with fresh contents (yellow). As indicated by the yellow flash, this remix makes the green propeller axle run inside of a blue bearing; the red flash indicates where the remix tries to mesh a red gear with a blue gear. These are two of the points where this maker had to tweak axles, bearings, and gears to make the resulting model work; especially the gear designs have many degrees of freedom—a lot to tweak. (b) For comparison, this is the corresponding remix we made using grafter. By fusing part of the blue axle with part of the green axle and by importing the red spring motor in its entirety grafter eliminates the problem and allows this model to print in one attempt.

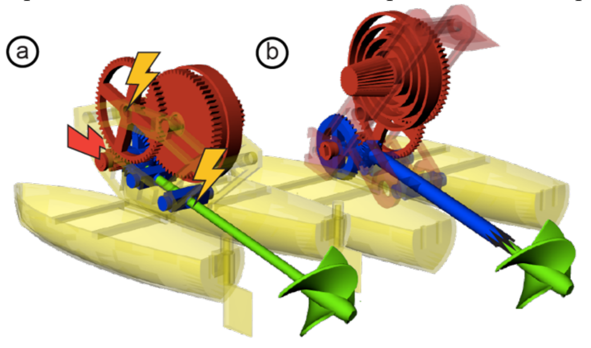

Figure 4: (a) Remix of a spring-powered catamaran. The green shaft in the blue bearing and the mismatched gears required tweaking. (b) The same model remixed using grafter works instantly without tweaking.

Furthermore, the comments that makers had uploaded to document their models suggest that part-based reuse was indeed time-consuming. Four makers (out of the eight with a full process description) stated that they had iterated multiple times to get parts to fit. One commented "After *5 tries and 5 hours* waiting on the printer to finish, I popped it out removed the supports and I was happy to see it fit perfectly" (emphasis added). Several makers commented that parts in the model that they had shared *still* did not fit properly. One wrote: "Parts are a bit tight when fitting together. Will upload an updated version."

Despite the small sample size, our first survey suggests that part-based remixing is indeed the commonly used practice among makers and that the tweaking resulting from part-based reuse is indeed time-consuming.

Survey 2: non-adherence to standards

We conducted a second survey with the objective of testing our assumption that makers do not adhere to standards. More specifically, we checked whether makers were adhering to standard diameters for axles and bearings.

We started by downloading another set of models from thingiverse.com. To obtain actual machines, we filtered for the term "mechanism", which gave us the models of 324 3D-printable machines.

We then ran a simple custom script to locate axles and bearings on these models, i.e., cylinder-detection algorithm based on RANSAC [5]. Our script found 5203 axles and bearings overall. As we were interested in standards *across* models, not within a model, we combined multiple axles of the same diameter within the same model into one. This left us with 1782 axles and bearings.

We then analyzed these axles and bearings by clustering them by diameter. If makers adhered to standards we would see only very few buckets with lots of models; if makers did not adhere to standards, we would see more of a spread.

Figure 5 shows our results. Overall the histogram is spread out, which suggests that makers do not adhere to standards.

The histogram does feature a few peaks, which we felt required closer inspection. Analysis of the objects, revealed that most of the peaks (highlighted red in Figure 5) had either resulted from the integration of industrially manufactured metal parts, such as metal bolts and steel axles (Figure 6) or from inheriting geometry from the same parent model.

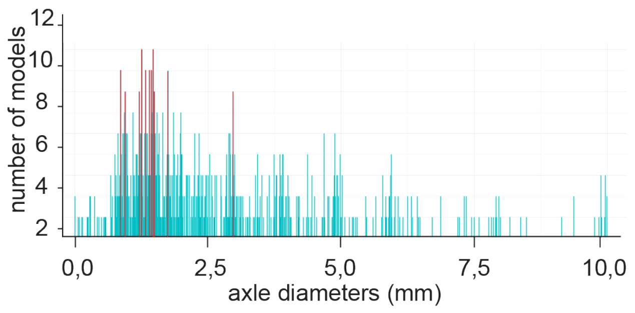

Figure 5: Histogram of axle and bearing diameters from 324 3D printed machines found online. Only 12 diameters appear in more than 8 models (marked in red).

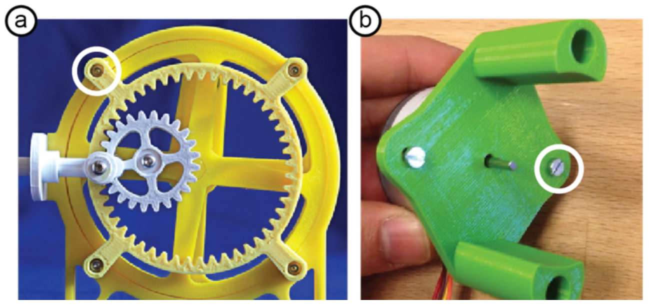

Figure 6: Two example models the geometry of which was designed to fit standardized parts, such as screws/bolts/nuts.

After removing these special cases, the most popular diameter accounted for only 0.5% of all models, which certainly does not suggest the user of standard sizes. Our second survey thus suggests that makers do not adhere to standards, unless their geometry interacts with industrially produced components.

Based on the insights from these two studies, we designed our system, *grafter*.

GRAFTER

grafter is a software system. Its first main contribution is that it supports users in remixing a specific challenging class of 3D printed objects, namely those that perform mechanical functions. Its second main contribution is that it does *not* follow common maker practice of extracting and re-assembling *parts*. Instead, it uses *mechanisms* as its main unit of reuse, i.e., it allows users to extract and recombine only entire mechanisms. Since mechanisms are self-contained units, essentially *independent* of the context they were designed for, they are easier to integrate into a new context of use than parts.

Mechanism-based remixing requires cutting up parts. As illustrated by Figure 7, machines generally consist of a chain of mechanisms, with each of the axles being part of two mechanisms. Given that grafter does not allow cutting up mechanisms, this means grafter has to cut *somewhere else*. And, as illustrated by Figure 7, cutting up somewhere else means to cut *through parts*, here the axles. This is exactly what grafter did in Figure 1c, where it extracted the bevel gear mechanism by cutting the two axles and fusing the remaining axle stumps with the axle stumps of the other two parent objects. Fortunately, grafter is designed to work with 3D printers, and so cutting and fusing parts are inexpensive operations.

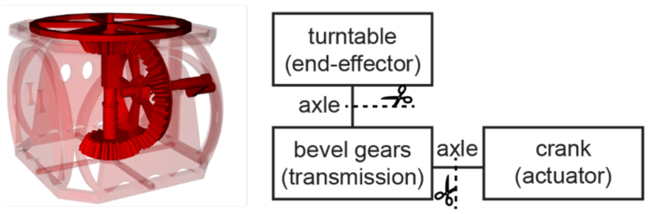

Figure 7: The record player from Figure 1, consists of (b) a simple chain of three mechanisms. Since cutting up mechanisms is problematic, grafter cuts up parts instead.

We now demonstrate how grafter allows users to remix machines. Users proceed in two steps. (1) First users annotate the mechanisms of parent models. (2) Then users remix. Grafter supports each of these steps with separate tools and both tools are implemented as plug-in to the 3D modeling software *Rhino*. In the interest of clarity, we begin with the remixing step and postpone our explanation of the annotation tool until after we have explained the inner workings of the grafter. We again use the test tube centrifuge from Figure 1 as an example.

Remixing in grafter

Figure 8a shows a user looking for parent models by searching for "test tube". As shown in Figure 8b, a model labeled *waterfuge* comes up. It looks interesting, but closer inspection reveals that it cannot spin the vials fast enough.

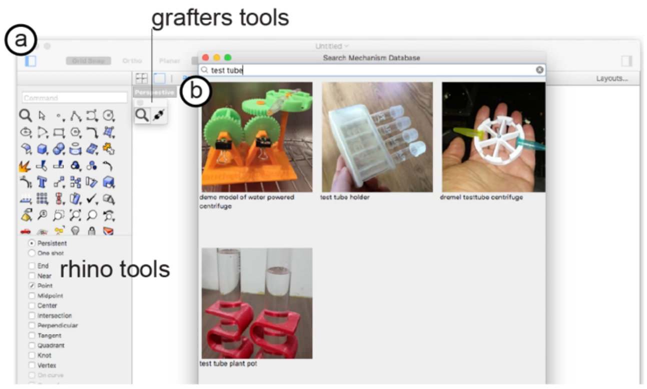
grafters tools
a
b
Search Mechanism Database
test tube
demo model of water powered centrifuge
test tube holder
dremel testtube centrifuge
test tube plant pot
rhino tools

Figure 8: (a) The user starts by clicking the search tool, which brings up the search dialog. (b) Typing in "test tube"

The test tube holder on top of the *waterfuge* looks viable, so our user clicks the model, which imports it into the current document (Figure 9a).

In search for a faster actuation mechanism, the user runs another search for "hand cranked". Among other models, this returns the *hand-cranked siren*. Its built-in gearbox allows the device to produce very fast rotation, making it well suited for the task. The user adds the siren to the document. The user has no use for the fan blades, thus removes them, resulting in the stripped-down version shown in Figure 9b.

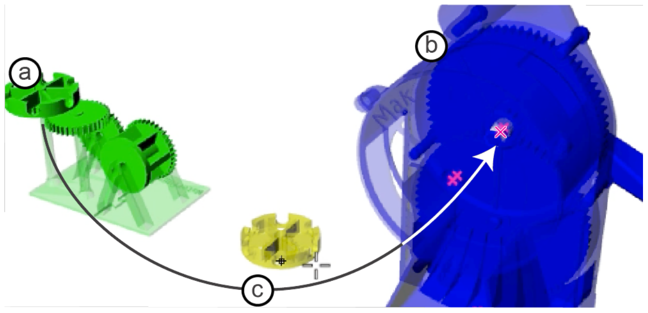

Figure 9: (a) The user imports the *waterfuge*, (b) then the *hand-cranked siren*, then (c) mounts the *waterfuge*'s test tube holder onto the *siren*.

The user now makes a first attempt at remixing. As shown in Figure 9c, the user selects the connect tool, rips the test tube holder out of its parent model and drags it towards the siren. Grafter responds by highlighting all 3 possible docking locations (red *x*): two for the bearing which used to hold the blades and one for the axle inside the machine. The user docks the test tube holder on the axle which is driven by the siren's gearbox.

Grafter performs a series of geometric computations (clearing of sweep volumes, merging of grounds, see section implementation) allowing it to merge the geometry of the test tube holder with the siren.

The resulting model is functional. Grafter could export and fabricate it immediately and, given that it is based on mechanism-based remixing, the user would be confident that it would work without any tweaking or test prints, because all of mechanisms in this new machine were *already tested*—by their original creators. However, after a moment of reflection the user realizes that making the test tubes spin up and down, will cause the sediment to mix back with the liquid—this is not a good design yet. The user undoes the connection.

In order to spin the test tubes horizontally, the user realizes that a 90-degree redirection mechanism is needed. A search for "bevel gears" brings up several devices that contain bevel gears. As shown in Figure 10a, the user picks the record player which adds it to the document. (b) Using grafter's connect tool, the user now drags the bevel gear mechanism out of the record player and docks it to the same docking location on the siren as the test tube holder. (c) Then the user docks the test tube holder to the top end of the bevel gear mechanism. This produces the model already shown in Figure 1c—this is what the user wanted.

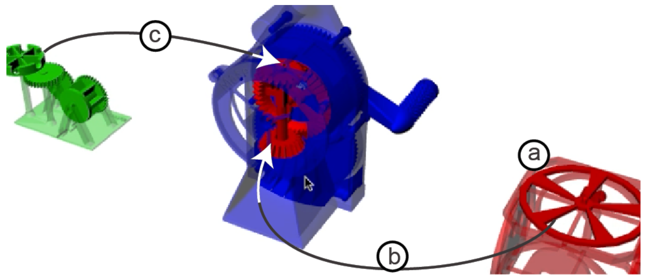

Figure 10: (a) The user imports the record player. (b) With a single drag operation each, the user connects its bevel gear mechanism to the siren, and (c) the test tube holder to the bevel gear mechanism.

The user now 3D prints the model, resulting in the physical object shown in Figure 1d. The user turns the crank and the mechanisms work *right away* without tuning.

CONTRIBUTION

This paper makes two main contributions. First, *grafter* is the first system that supports users in remixing machines. We present the remixing tool, as well as a specialized annotation tool that allows preparing models for remixing. Second, with grafter we introduce the concept of *mechanism-based* remixing. While the time required for mechanism-based remixing is roughly of the same order of magnitude as part-based remixing (if we add 23min per parent model for annotation and 5min for remixing) the main benefit of mechanism-based remixing is that it eliminates the necessity to tweak and test-print, so that remixed machines print correctly on the first attempt, which tends to save many hours to days of task time.

Grafter is currently limited to machines based on rotating axles. While this is the dominant type of machine design today, handling machines with other mechanisms would require extending grafter. For example, grafter currently fails when a translation mechanism is inserted between two rotational mechanisms. Finally, grafter will allow combining mechanisms in any technically feasible way, even if this practically does not make sense (see Figure 25e for an example of this remixed during our study).

RELATED WORK

This paper builds on work in reuse, encapsulating mechanism functionality, remixing of 3D models, and studies on the remixing behavior in hobbyist model repositories.

Reuse by professional users across domains

Designers and creative workers draw from existing examples to produce new creations [10].

In the software industry, reuse found wide adoption once processor architectures had been unified [8] and good models for reusable components could be developed [12]. Remixing is also very common in the context of computer game development—more so for functional elements than for artistic elements [11]. Cheliotis et al [4] found similar results when studying online remixing of music.

Several software tools support reuse in the field of web-based systems. Hartman et al. [9] studies web developers and hardware developers who create mash-ups by "shopping" for elements made by others. Interactive design galleries [15] help developers find and draw elements from web pages. Bricolage [14] allows users to remix web pages by combining the content of one webpage with the style of another.

As discussed earlier, professional mechanical engineers use and reuse parts from various suppliers, e.g., through *Solidwork's 3DContentCentral* [32]. Professionally engineered parts are standardized and users can integrate them into their models with confidence, regardless of the supplier.

Remixing of 3D objects

3D objects are typically remixed from standardized components. Design and Fabrication by Example [29] allows users to select and modify parts from templates. The system generates the required connector parts, such as screws, automatically. Pan et al. [5] allow users to create mechanically more advanced models.

Parametric models [30] as the extended version of Fabrication by Example, enables more customization of existing parts, basic shape remixing, and verifies whether models are physically attainable, i.e., whether they can be 3D printed. Modeling by example [29] similarly enables remixing of specific parts from different models into new configurations. Zhang et al. [33] use similar mechanical templates and allow them to be included in user-defined shapes. The challenge here is that parametric modelling does not solve the tweaking issue. It makes it easier because the user only changes the critical parameters, but one would still test print and tweak to get the functionality right, especially when dealing with different models made by others.

Encapsulating mechanism functionality

Model formats are only starting to describe functionality. 3D models, such as those shared on thingiverse, are typically stored using the STL format, which only describes the *shape* of models, but not how they work. Researchers have thus proposed including extended functionality in new file formats. MetaMorphe [31] composes 3D models from HTML for shape and structure, CSS for design details and parameter configuration, and JavaScript for functionality and key framing. The STP file extension [28], is another format to include additional meta-information about models, mostly adopted by industry.

Mitra et al. [21] demonstrate how to automatically animate illustrations of mechanisms. They use symmetry detection [20] to identify the functionality of a mechanism from drawings. More recently, the authors extended their approach to 3D scans [16].

Remixing in Hobbyist Repositories

Recent work by Flath et al. [6] investigated remixing practice in the hobbyist repository *thingiverse*. They identified remixing strategies, patterns and essentially conclude that reducing the barriers for remixing is key to maintaining a creative community. Their key implications are directly addressed by grafter.

Papadimitriou et al [24, 26] identified a trend towards remixing in hobbyist communities. The most common form of remixing on Thingiverse is by means of the *Thingiverse Customizer* [18]. This tool remixes only single parent models, hobbyists remix multiple parent models using 3D editors, such as MeshMixer.com. Flath et al. points out that individual users *either* always use Customizer *or* always use MeshMixer [6].

Alcock et al. [1] found that users often do not know how to remix models. Their second most common comment (23.9%) to models was related to the theme of customizing or remixing models. Successful remixes are most likely to be *assemblies of models* and *primitive mechanisms* [20]. They also identified two projects of highly remixed machines: the engineering of DIY 3D printers and quadcopters. These big thingiverse projects attract large amounts of hobbyists to remix (parts) of the models. It is such projects, which drive the maker community according to Anderson in his book *Makers* [2].

THE INNER WORKINGS OF GRAFTER

To allow readers to reproduce our result, we now present the algorithms behind grafter.

The model graph is grafter's main data structure

In order to allow grafter to support remixing, it internally represents models in a format we call *model graph*. Grafter performs all its operations on this data structure, including mechanism selection and the fusing of mechanisms. As an example, Figure 11 shows the model graph of the record player from Figure 1.

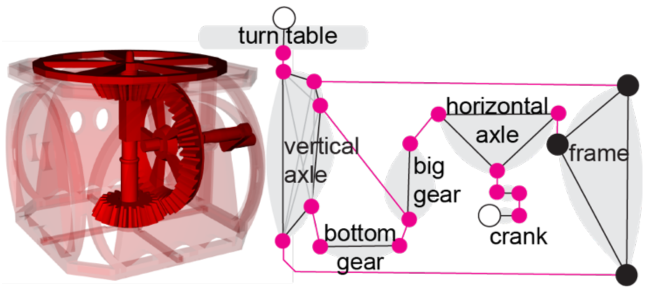

Figure 11: The record player including the bevel gear mechanism. The model is annotated with a model graph.

Grafter's model graphs are somewhat unusual. The nodes do not relate to parts, but to interfaces e.g. the surfaces of parts that touch other parts. This better depicts the core idea of *mechanism-based* remixing, as it allows easy cutting up of parts. The edges define how the interfaces are related to each other.

Pink nodes denote *interfaces:* In Figure 11, the interfaces of the three bevel gears are the surfaces of the gears' teeth.

Black lines denote *parts*. These connect the *interfaces* that are located on that part.

Pink lines denote *"mechanisms" aka kinematic pair.* They connect interfaces located on two or more different parts. An example of a mechanism in Figure 11 is an axle and the bearing that holds it.

White nodes denote *external interfaces*. These are interfaces that connect a part in the model with a part in the outside world. Objects that are not otherwise represented in the model graph, such as the hand cranking the record player.

Black dots denote *ground geometry*. Ground geometry is a crucial part of a model's geometry: it forms the "frame", i.e., typically one large part that keeps the mechanisms in their specific configuration [22]. During remixing, Grafter not only reassembles mechanisms but also "unites" the grounds of all parents.

Figure 11 shows an example of the model graph of a record player, note how the clusters of nodes connected with black lines represent what is typically considered as parts. The pink lines are the interactions between these parts. It is for that reason that the vertical axle (which connects two gears, two bearings and the rotary disc on top) contains 5 nodes, whereas the horizontal axle contains only three (the big gear, a handle and a bearing).

Remixing

The model graph plays a key role in remixing. As discussed earlier, grafter users remix a model by selecting a mechanism, picking the *connect* tool from grafter's menu, picking that mechanism up by a "snapping point", and dropping it onto a "snapping point" on target machine (Figure 12). When the dragged mechanism gets close enough to a snapping point, grafter snaps the model to the target and aligns their axes of rotation.

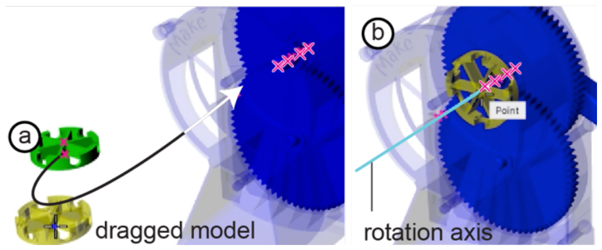

Figure 12: (a) A user drags a mechanism using the connect tool, grafter highlights target snapping points. (b) the user has dragged this tube holder close to a snapping point, grafter aligns the rotation axis of the tube holder with the siren.

How grafter makes any selection a mechanism

Any selection begins with the user clicking somewhere on a model. While such an event must inherently land on a *part*, grafter always expands this selection into a mechanism that contains this part.

There are typically multiple ways how grafter *could* extend the selection to a mechanism, as Figure 13 illustrates at the example of the record player. When the user clicks the large bevel gear, grafter responds by also selecting the axle it sits on, the small bevel gear it meshes with, and any geometry needed to reach a ground node on both sides of the selected part. This subgraph is shown in (a).

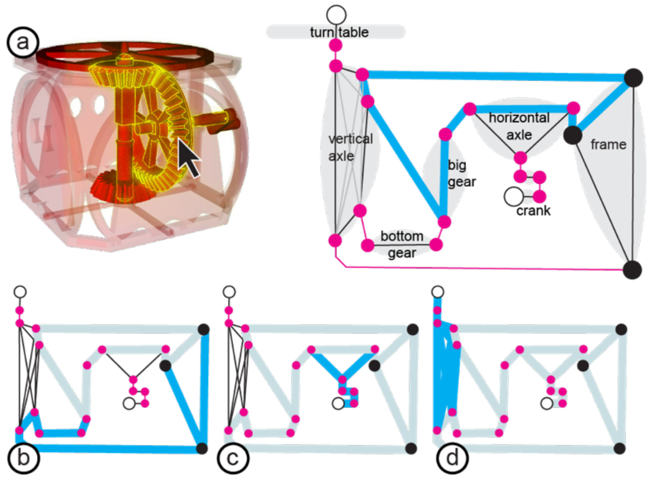

Figure 13: (a) When the user clicks the large bevel gear, grafter automatically expands the selection to a mechanism. Subsequent clicks extend that selection by adding more mechanisms (selections highlighted in blue). (b) Grafter considers subgraphs that include ground as "cheap", (c, d) while it considers those that include an external interface as "expensive".

The minimal selection in Figure 13 includes the big gear which is clicked and all interfaces until a ground node is reached from both sides of the big gear. This results in the selection of (a). When the user clicks again on the big gear, grafter uses a cost function to determine which subgraph to add to the selection.

The assumption is that the user wants to select the mechanisms connected around the selected object rather than ones that connect externally. Each subgraph either ends with a ground node or an external interface. The grounded graphs extend the selection by adding internal mechanisms and thus have a smaller cost. Furthermore, a shorter subgraph is preferred over a long one.

Grafter allows users to further expand the selection by clicking repeatedly. With every click, grafter extends the current selection with the cheapest reachable subgraph. As there is always a ground or external interface at the end of a subgraph, grafter only selects fully supported mechanisms.

When the entire machine is selected and the user clicks again, grafter will deselect everything and make the selection process start over again.

Snapping and aligning, only to leaf nodes

Once the user has selected a mechanism, grafter allows the user to drag this mechanism into another machine. During dragging, grafter highlights "snapping points" capable of accepting the dragged mechanism and prevents the user from docking anywhere else but at these snapping points.

As shown in Figure 14, snapping points are created at the ends of fully supported kinematic chains. Grafter locates such kinematic chains by searching the model graph for "leaf" nodes, i.e., nodes connected to a single other interface (besides a connection to ground or external interface) For the selection of Figure 14, leaf nodes are highlighted in blue. For these, grafter uses the intersection of the axis of rotation with the interfaces as snapping points. An axle, for example, has snapping point at both ends. The bottom of Figure 14, in contrast, offers only a single snapping point, as it has a bearing with only one open direction (there is no axle sticking through the bottom of the frame).

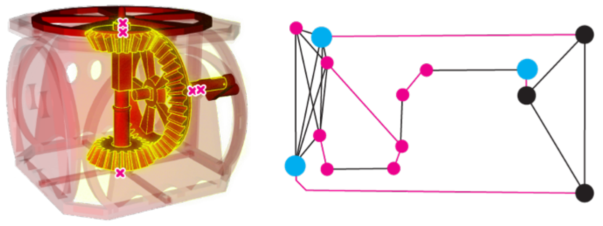

Figure 14: Snapping points of the selected bevel gears. Only the leaf nodes generate points (highlighted in blue).

Once the user has docked a mechanism to a target machine, grafter merges the mechanism's geometry into the target machine. Grafter accomplishes this in three steps: grafter (1) removes material that could prevent the mechanisms from working, (2) merges mechanism geometries, and finally (3) merges grounds.

Grafter removes material that could block mechanisms

When the user docked the test tube holder onto the siren as part of our walkthrough, grafter could have tried to mount the test tube holder directly to the end of the bevel gear mechanism. However, the resulting machine would not have been functional: the test tube mechanism requires some clear space to perform its function, as illustrated by Figure 15a.

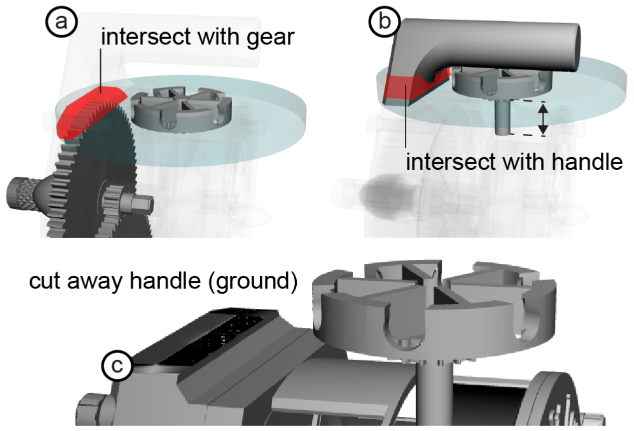

Figure 15: (a) The tube holder's swept volume collides with the gear in the back. (b) Grafter responds by translating the tube holder (and extends its axle) along the axis of rotation until the conflict is resolved. This causes the test tube holder's swept volume to instead intersect with the siren's handle, but (c) the handle is not part of the model graph and thus can be safely cut off.

Such space is commonly referred to as *swept volume.* As illustrated by Figure 16, the swept volume is defined by the space swept when the mechanism is operated. The test tube holder shown in Figure 16d is special in that it holds physical objects—the test tubes—that are not captured in the model; these also have to be considered when computing the swept volume.

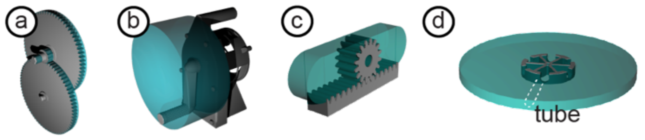

Figure 16: The swept volume of selected mechanisms.

In some cases, multiple swept volumes can intersect with one another. As in Figure 16, grafter cannot just clear out the swept volume as it would break a mechanism.

Instead, as shown in Figure 15b, grafter automatically extended the axle of the test tube holder past the gear. As shown in Figure 15c, the test tube holder's swept volume still intersected the model—however, this time only with the siren's handle. By traversing the model, grafter determined that the handle was not connected to any mechanism. Grafter therefore considered the handle non-vital and removed it in order to make space for the test tube holder and its swept volume.

Merging mechanism geometries

As shown in Figure 17, grafter fuses mechanisms by overlapping the axle of the dragged mechanism with the axle of the target machine. Then, grafter chops of all geometry past the snapping point and then merges both axles using a union operation.

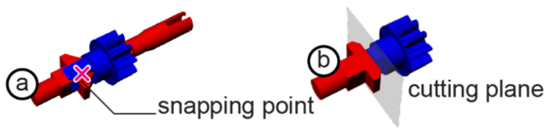

Figure 17: In order to merge the dragged mechanism into the target machine, grafter (a) overlaps the axles at the snapping point and (b) removes excess geometry.

Merging grounds

After connecting the axles, parts of the newly implanted mechanism may have become "ungrounded", i.e., they dangle loosely as they are not connected to ground anymore. Ungrounded geometry typically prevents the resulting machine from functioning. In Figure 18 the red bearing holding one of the bevel gears has become ungrounded.

Grafter identifies ungrounded mechanisms by comparing the merged model graph with the original one and identifying where ground nodes have not been replaced.

Grafter then reattaches ungrounded mechanisms by bringing in additional ground geometry from the mechanism's original model. Grafter therefore computes the intersection between the ground of the target model (here the blue siren) and the ground of the source model (the red bevel gears). If there is an intersection, it splits the mesh after the intersection and removes the excess. It fuses the grounds together with a Boolean union operation.

In the rare case that there is no overlapping geometry between the grounds, grafter currently fails. An approach as shown in AutoConnect [13] in which rods between models are generated could be used as an extension to solve this problem.

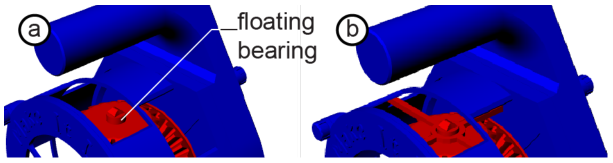

Figure 18: (a) The red bearing is *ungrounded.* (b) In order to resolve this, grafter brings in ground from the red source model (record player).

ANNOTATING MACHINES

Now that we have learned about model graphs and interfaces, we are well equipped to explain grafter's model annotation tool. The objective of the annotation process is to allow grafter to create the model graph. In particular, this requires users (1) to mark all polygons/triangles that belong to the same interface and (2) to combine interfaces into mechanisms.

In the following, we demonstrate the annotation workflow at the example of a vise—a workshop tool capable of clamping a workpiece (Figure 19). When the user turns the knob on the left, the middle gear rotates. This causes the two gears left and right to rotate, which pushes the threaded rods and the attached jaw towards the workpiece. The device achieves this functionality with the help of 9 mechanisms, which together consist of 15 interfaces. This is what we will annotate.

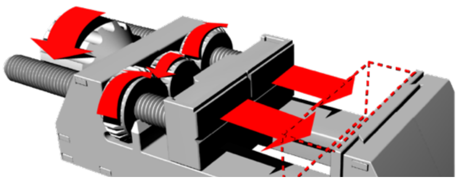
Figure 19: The vise model to annotate.

The process is straightforward except for the step of selecting all triangles that together form an interface. Doing so naively can be tedious, as an interface may contain thousands of triangles. Grafter's annotation tool therefore offers a specialized cylinder-shaped selection tool that is able to select all relevant triangles at once. Grafter supports the cylinder selection tool by automatically locating the axes around which cylinder selections can be formed.

Figure 20a: when loading a model, grafter invokes its *axes of rotation* script (a script similar to the one mentioned in the user study section earlier). The script locates cylindrical shapes using the RANSAC algorithm [5]. This identifies axes in models, which the editor shows as red lines.

(b) The user starts by annotating the gear in the front. The user selects the axis (which causes it to be highlighted in yellow) and using grafter's cylinder selection tool the user constructs a cylindrical selection that roughly matches the interface (by clicking the start of the cylinder, the approximate radius and the end of the cylinder) (c) grafter finds meshes that are close to the constructed cylinder and identifies these as an interface, here the teeth of the gear.

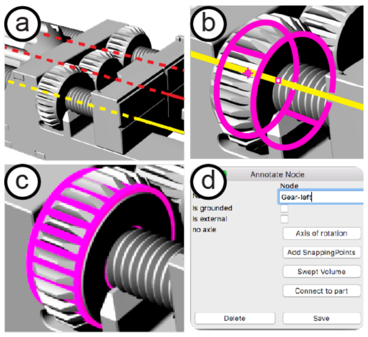

Figure 20: (a) Upon loading a model, grafter automatically locates axes of rotation in the model. (b, c) Based on the selected axis, grafter's cylinder selection tool creates the shown pink selection in three clicks (start, radius, end). (d) users can indicate whether an interface is external or grounded with the annotation dialogue.

As shown in Figure 20d, the user now labels the interface as grounded geometry or an external interface. Other options are set by grafter, but can be adjusted by the user.

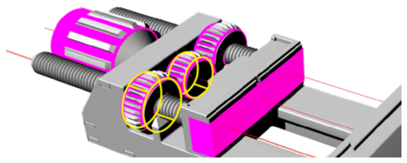
Figure 21: After all interfaces are selected, the user continues to select pairs of them and link them as mechanisms. In the shown figure, these two gears form a mechanism together.

Repeating this step 15 times labels all interfaces in this model as shown in Figure 21. The user then combines pairs of interfaces into mechanisms by selecting two interfaces and linking them (using the link tool) for each of 9 mechanisms. The resulting model graph is shown in Figure 22

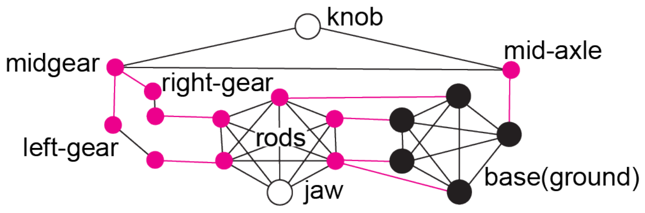

Figure 22: The resulting model graph.

Figure 23 shows the parent models remixed in Figure 2. Annotating them by a trained user required 23min on average (see Figure 23 for individual annotation times). When models are remixed repeatedly, as in Figure 2, each parent model requires annotation only once.

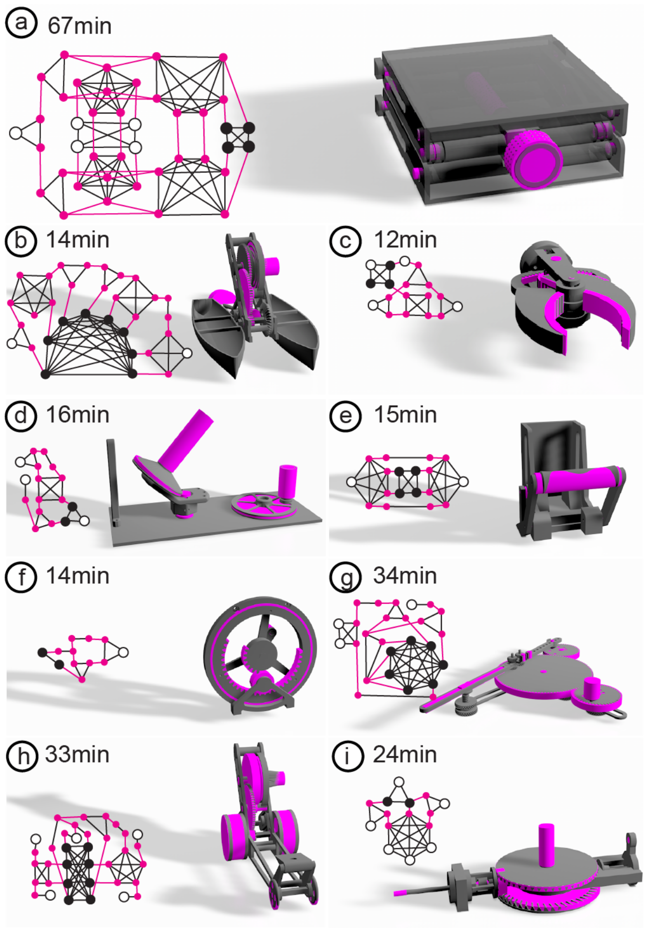

Figure 23: The parent models remixed in Figure 2 annotated using grafter's annotation tool and how long it took a trained user to annotate these models.

IMPLEMENTATION

Grafter and its model annotation tool are implemented as a *Rhinocommon* plug-in. This is a C# SDK for Rhino [19].

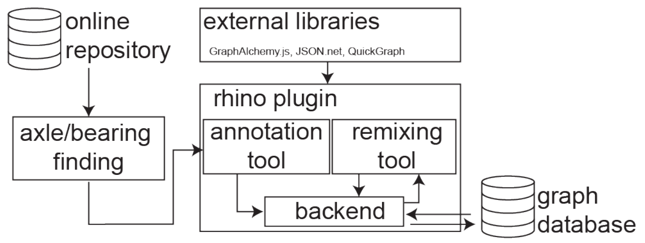

Figure 24: The architecture of grafter.

The backend component processes the model graph into the 3dm file format of Rhino. To do so, it encodes the graph into JSON using the JSON.net library. For the undirected graph data structure as well as graph algorithms we utilize the QuickGraph library. For visualizing the model graph we use GraphAlchemy.js

USER STUDY

Our main objective with grafter is to allow for fast remixing by eliminating the tweaking process traditionally associated with combining parts from different parent models. To verify this assumption, we conducted a study in which participants used grafter to remix machines. Given that participants performed this task using mechanism-based remixing, our hypothesis was that participants' models would 3D print successfully on the first attempt.

Task and Procedure

Half of the participants were given the task to create a hand-cranked wool winder by remixing elements from two parent machines; the other half were given the task to create a hand-cranked wire twister. Figure 25a and c show intended solutions; participants, were given verbal description of what to make.

They were then given access to grafter and 15 parent models shown throughout this paper, in annotated form.

Participants received 10 minutes of training, during which they created the centrifuge model shown in Figure 1. They then performed their remixing task. Finally, participants were asked about their experience. All participants completed the study in 30 min or less.

Participants

We recruited 12 students (1 female, average age 25.5 years old). All participants had used a 3D modeling system at least once. Three participants had more than 50h of experience with other 3D modeling programs, namely Fusion360, Blender, and SketchUp. Two users had used Rhino before, albeit for less than 2 hours overall.

Results

Figure 25 shows 3D prints of the models made (note that many participants achieved similar results). The 3D prints of all devices worked on first attempt and without any need for tweaking or test printing. This confirms our hypothesis.

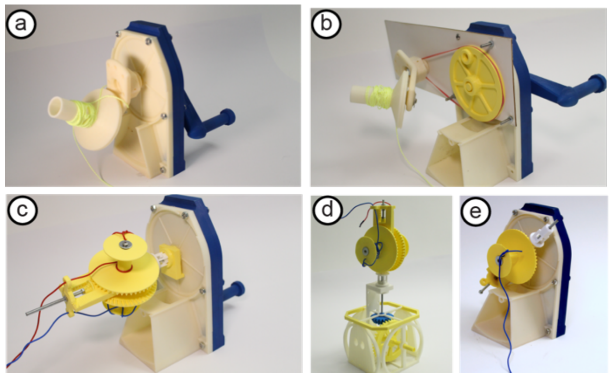

Figure 25: (a and b) The wool winders and (c-e) the wire twisters made by participants.

Figure 26 shows participants' task times. All participants completed their tasks quickly: 4:50 minutes on average for the wool winder and 4:54 for the wire twister).

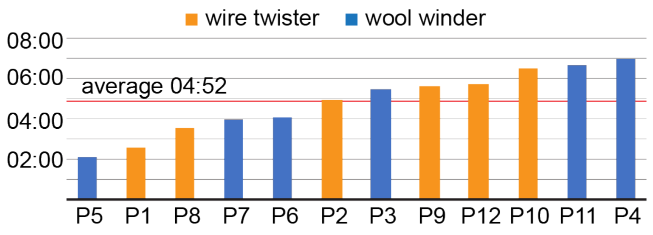

Figure 26: Task time ordered by performance.

In the first go, 10/12 participants managed to make a working version of the remix we asked them to make. 2 participants seemed not to understand how the machine is supposed to work (our participants had no mechanical engineering background) and remixed the models in a way that would not work out (the solution does print and work, but not twist wires). After the experimenter explained how the machines do work, they managed to make the intended remix.

P4 commented on the interaction: "at first it was a little hard to do, but once I figured out how the interaction worked it was actually quite easy". P7 and 3 others in other words expressed "I really enjoyed the experimental style of modelling, trying combinations out is a useful way to model".

All participants suggested additional features, 5 participants suggested animating mechanisms to show what the machine does. 4 participants had issues with the search, 2 of them suggested to use the mechanisms that were searched before to fine tune the results.

CONCLUSION AND FUTURE WORK

In this paper, we presented grafter, a tool that allows users to remix 3D printed mechanical machines. In addition to the system itself, our main contribution is that grafter does *not* follow common maker practice of extracting and re-assembling *parts*. Instead, grafter extracts self-contained mechanical units, aka *mechanisms* that remain functional throughout remixing. The main benefit for users is that grafter eliminates the need for tweaking and test printing.

As future work, we plan on using the model graphs to make machine designs work across different printers and materials.

REFERENCES

1. Celena Alcock, Nathaniel Hudson, and Parmit K. Chilana. 2016. Barriers to Using, Customizing, and Printing 3D Designs on Thingiverse. In *Proceedings of the 19th International Conference on Supporting Group Work* (GROUP '16). ACM, New York, NY, USA, 195-199. DOI: https://doi.org/10.1145/2957276.2957301

2. Chris Anderson 2012. Makers: The New Industrial Revolution. *Crown Business*, New York, 2012.

3. Patrick Baudisch and Stefanie Mueller. 2017, "Personal Fabrication", *Foundations and Trends in Human–Computer Interaction*: Vol. 10: No. 3–4, pp 165-293. DOI: http://dx.doi.org/10.1561/1100000055

4. Giorgos Cheliotis and Jude Yew. 2009. An analysis of the social structure of remix culture. In *Proceedings of the fourth international conference on Communities and technologies* (C&T '09). ACM, New York, NY, USA, 165-174.
DOI: http://dx.doi.org/10.1145/1556460.1556485

5. Martin A. Fischler and Robert C. Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6 (June 1981), 381-395. DOI: http://dx.doi.org/10.1145/358669.358692

6. Christoph M. Flath, Sascha Friesike, Marco Wirth, & Frederic Thiesse, Copy, transform, combine: exploring the remix as a form of innovation. *Journal of Information Technology*, 1-20.
DOI: https://doi.org/10.1057/s41265-017-0043-9

7. Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. 2004. Modeling by example. *ACM Trans. Graph.* 23, 3 (August 2004), 652-663.
DOI: http://dx.doi.org/10.1145/1015706.1015775

8. David Garlan, Robert Allen & John Ockerbloom 1995 Architectural Mismatch: Why Reuse Is So Hard. *IEEE Software 12(6), 17-26*
DOI: http://dx.doi.org/10.1109/52.469757

9. Björn Hartmann, Scott Doorley, and Scott R. Klemmer. Hacking, mashing, gluing: Understanding opportunistic design. *IEEE Pervasive Computing* 7.3 (2008).
DOI: https://doi.org/10.1109/MPRV.2008.54

10. Scarlett R. Herring, Chia-Chen Chang, Jesse Krantzler, and Brian P. Bailey. 2009. Getting inspired!: understanding how and why examples are used in creative design practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09). ACM, New York, NY, USA, 87-96.
DOI: http://dx.doi.org/10.1145/1518701.1518717

11. Benjamin Mako Hill and Andrés Monroy-Hernández. 2013. The cost of collaboration for code and art: evidence from a remixing community. In *Proceedings of the 2013 conference on Computer supported cooperative work* (CSCW '13). ACM, New York, NY, USA, 1035-1046.
DOI: http://dx.doi.org/10.1145/2441776.2441893

12. Malcolm Douglas McIlroy, 1969. Mass Produced Software Components. In *Software Engineering*, ed. P. Naur and B. Randell, NATO Science Committee, Jan 1969, pp. 138-150.

13. Yuki Koyama, Shinjiro Sueda, Emma Steinhardt, Takeo Igarashi, Ariel Shamir, and Wojciech Matusik. 2015. AutoConnect: computational design of 3D-printable connectors. *ACM Trans. Graph.*34, 6, Article 231 (October 2015), 11 pages.
DOI: https://doi.org/10.1145/2816795.2818060

14. Ranjitha Kumar, Jerry O. Talton, Salman Ahmad, and Scott R. Klemmer. 2011. Bricolage: example-based retargeting for web design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '11). ACM, New York, NY, USA, 2197-2206.
DOI: http://dx.doi.org/10.1145/1978942.1979262

15. Brian Lee, Savil Srivastava, Ranjitha Kumar, Ronen Brafman, and Scott R. Klemmer. 2010. Designing with interactive example galleries. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '10). ACM, New York, NY, USA, 2257-2266.
DOI: http://dx.doi.org/10.1145/1753326.1753667

16. Lin, M., Shao, T., Zheng, Y., Mitra, N., & Zhou, K. (2017). Recovering Functional Mechanical Assemblies from Raw Scans. *IEEE Transactions on Visualization and Computer Graphics*.
DOI: https://doi.org/10.1109/TVCG.2017.2662238

17. Linjie Luo, Ilya Baran, Szymon Rusinkiewicz, and Wojciech Matusik. 2012. Chopper: partitioning models into 3D-printable parts. *ACM Trans. Graph.* 31, 6, Article 129 (November 2012), 9 pages.
DOI: http://dx.doi.org/10.1145/2366145.2366148

18. MakerBot Customizer. Retrieved February 16, 2017 from http://customizer.makerbot.com

19. McNeel, R. (2017). Rhinoceros. *NURBS modeling for Mac: http://www.rhino3d.com/mac*

20. Niloy J. Mitra, Mark Pauly, Michael Wand, & Duygu Ceylan (2013, September). Symmetry in 3d geometry: Extraction and applications. In *Computer Graphics Forum* (Vol. 32, No. 6, pp. 1-23).
DOI: http://doi.org/10.1111/cgf.12010

21. Niloy J. Mitra, Yong-Liang Yang, Dong-Ming Yan, Wilmot Li, and Maneesh Agrawala. 2013. Illustrating how mechanical assemblies work. *Commun. ACM* 56, 1 (January 2013), 106-114.
DOI: http://dx.doi.org/10.1145/2398356.2398379

22. Norton, R. L. (1999). *Design of machinery: an introduction to the synthesis and analysis of mechanisms and machines*. McGraw-Hill Inc., US.

23. Lora Oehlberg, Wesley Willett, and Wendy E. Mackay. 2015. Patterns of Physical Design Remixing in Online Maker Communities. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, New York, NY, USA, 639-648. DOI: https://doi.org/10.1145/2702123.2702175

24. Pan, W., Gao, S., & Chen, X. (2016). An approach to automatic adaptation of assembly models. *Computers in Industry*, *75*, 67-79 .
DOI: http://dx.doi.org/10.1016/j.compind.2015.06.005

25. Spiros Papadimitriou and Evangelos E. Papalexakis. 2014. Towards laws of the 3d-printable design web. In *Proceedings of the 2014 ACM conference on Web science* (WebSci '14). ACM, New York, NY, USA, 255-256.
DOI: http://dx.doi.org/10.1145/2615569.2615660

26. Spiros Papadimitriou, Evangelos Papalexakis, Bin Liu, and Hui Xiong. 2015. Remix in 3D Printing: What your Sources say About You. In *Proceedings of the 24th International Conference on World Wide Web* (WWW '15 Companion). ACM, New York, NY, USA, 367-368. DOI: http://dx.doi.org/10.1145/2740908.2745943

27. Thingiverse Remix Challenge November-December 2016 http://thingiverse.com/challenges/RemixChallenge

28. Pratt, M. J. (2001). Introduction to ISO 10303—the STEP standard for product data exchange. *Journal of Computing and Information Science in Engineering*, *1*(1), 102-103.
DOI: http://dx.doi.org/10.1115/1.1354995

29. Adriana Schulz, Ariel Shamir, David I. W. Levin, Pitchaya Sitthi-amorn, and Wojciech Matusik. 2014. Design and fabrication by example. *ACM Trans. Graph.* 33, 4, Article 62 (July 2014), 11 pages.
DOI: http://dx.doi.org/10.1145/2601097.2601127

30. Maria Shugrina, Ariel Shamir, and Wojciech Matusik. 2015. Fab forms: customizable objects for fabrication with validity and geometry caching. *ACM Trans. Graph.* 34, 4, Article 100 (July 2015), 12 pages.
DOI: https://doi.org/10.1145/2766994

31. Cesar Torres and Eric Paulos. 2015. MetaMorphe: Designing Expressive 3D Models for Digital Fabrication. In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition* (C&C '15). ACM, New York, NY, USA, 73-82.
DOI: http://dx.doi.org/10.1145/2757226.2757235

32. Yaz, I. O., & Loriot, S. (2014). Triangulated surface mesh segmentation. *CGAL User and Reference Manual*, *4*(1).

33. Ran Zhang, Thomas Auzinger, Duygu Ceylan, Wilmot Li, and Bernd Bickel. 2017. Functionality-aware retargeting of mechanisms to 3D shapes. *ACM Trans. Graph.* 36, 4, Article 81 (July 2017), 13 pages.
DOI: https://doi.org/10.1145/3072959.3073710

34. 3D contentcentral. Retrieved February 17, 2017 from http://3dcontentcentral.com