# An Inside Look into Addgene's Dev Team

By Sophia K. Cheng

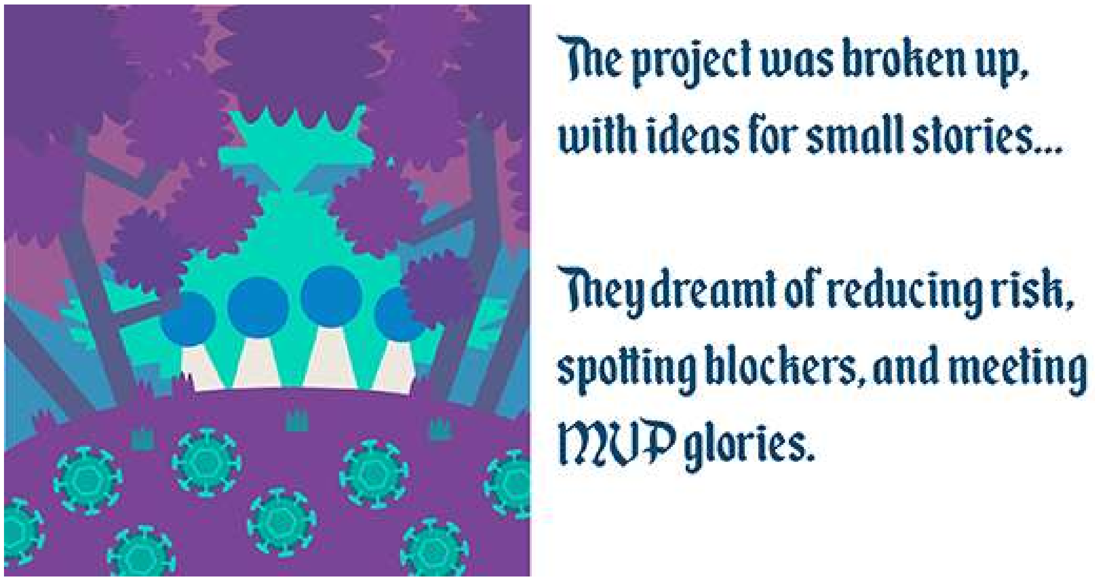
When you think of Addgene, chances are software development is not one of the things that comes to mind. The truth is, there is a lot of software development happening at Addgene under the hood. A significant portion of our development may not be visible to users like yourselves but you have definitely reaped the benefits. We support everyone that works with Addgene in some way. For users, we just released a major redesign of how search works on our site. For Addgenies, development includes everything from tracking plasmids from the moment a deposit kit is sent to a depositor, to streamlining the order fulfillment process to reduce viral order fulfilment from an average of 40 minutes to 10 minutes. While users can’t tangibly see these improvements, you are benefited with reassurance your plasmids will not get lost in a freezer and your viral orders are being fulfilled more quickly. As Addgene grows and evolves, we will be here to increase all of our teams’ efficiency and productivity. Get to know our software development team and the development process at Addgene.

## Diversity supports our innovation

Our software development team is a microcosm of Addgene’s work culture. Our developers are a diverse group of ethnicities, cultures, genders, educational, and professional backgrounds. Sometimes this can lead to wildly different ideas of how to approach a problem, but it is these very differences that make our software so great. If we all came from similar backgrounds, we would have similar cognitive biases and this would become apparent in the software we develop. We would all have the same blind spots, and there wouldn’t be anyone to question these blind spots. Blind spots often result in less than optimal user experience and the overall soundness of the software design. Each of us brings slightly different cognitive biases as a result of our diversity which in turn minimizes the blind spots in our work.

# Agile at Addgene

We follow an agile methodology of software development called Scrum. The Scrum framework is based on an iterative and collaborative approach to development. Scrum teams plan and break up work into small, well defined chunks, called *stories*, and work on stories during short periods of time called *sprints*. The team is mindful of the number of stories we commit to for each sprint, to ensure we will have the capacity to finish all of the work. Since we work in iterative chunks, we have the flexibility to adapt and easily respond to changing needs of the teams at Addgene. Another advantage of Scrum is that it provides transparency to everyone involved about the status of each project. This creates accountability on the part of each developer and creates trust amongst the team.


The project was broken up,
with ideas for small stories...
They dreamt of reducing risk,
spotting blockers, and meeting
MVP glories.

Whether we are a highly collaborative team because we are using the Scrum framework or we are using the Scrum framework because we are highly collaborative, we love teamwork. Working collaboratively allows our team to best serve Addgene and ourselves. We are also able to consider different technical approaches that may come up during discussion.

One example of agility at work is our process for working on a new feature. We start with a story that describes the feature and what the criteria is for accepting the feature as done correctly. These initial stories are usually a result of collaborations between Addgene's Product Team and the stakeholders (other Addgenies). Once we begin work on a story, we continue to collaborate closely with each other and with other teams.

We start with writing demo details, step-by-step instructions on how to demonstrate the feature. For example, we recently reduced the number of steps involved during checkout. Demo details for this would be something along the lines of showing that the checkout takes the user directly to the shipping step, and no longer takes the user to a terms step. These demo details help us see the problem from the user's perspective which in turn helps us decide how to best approach it in our code.

Sometimes while thinking through these steps, we discover we need additional information. It is quite possible that the clarifications lead to the story being scrapped or broken up into smaller stories. Since this occurs before a single line of code is written, we are able to use our developers' time efficiently. While writing the code, we may need additional clarification and again we collaborate with the Product Team who in turn collaborates with stakeholders to resolve the questions.

After the code is written, it is reviewed by at least one other developer. This is another point of collaboration for us. I enjoy doing code reviews because it's like a little peek into the mind of the developer and I get to see an approach that is likely different from what I would have done. And finally the stakeholder accepts the new feature as done and we're onto the next story.

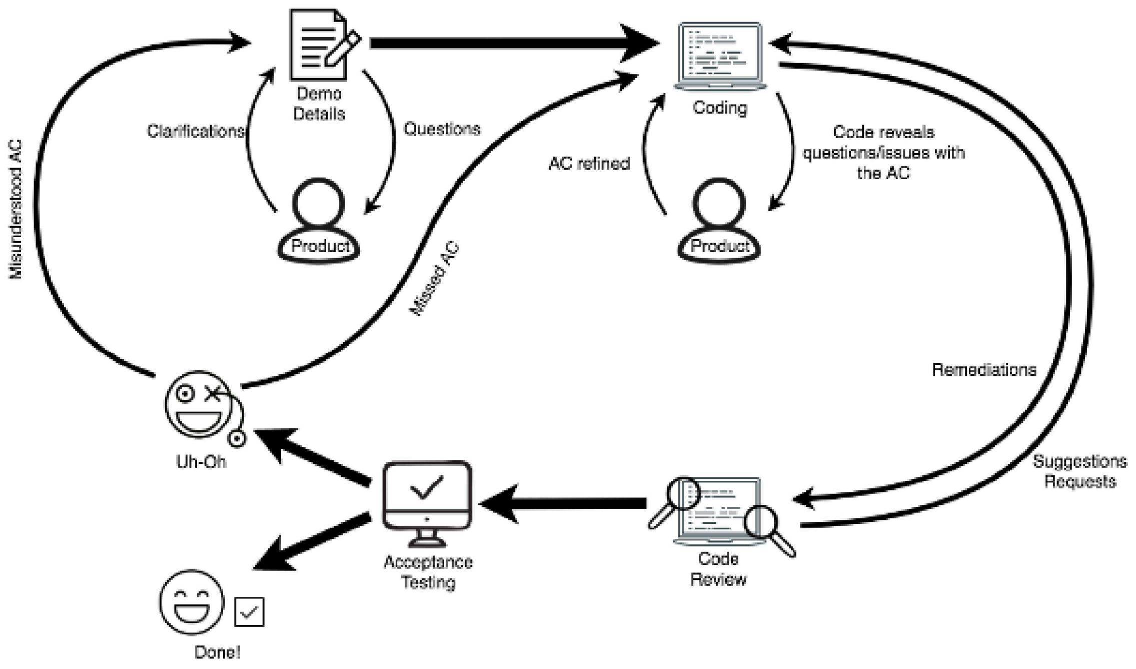


***Figure 1: Addgene's Agile Approach***

## Sharing is caring

There are many moving parts to our software, which has grown with the company. It would be easy to assign each developer to a single area/part of the code and have each developer work only on their assigned area. It's natural that some of us have more in-depth knowledge of one area, and when something urgent comes up, that person is the first in line to resolve it for expediency. But otherwise, this isn't how we work. We all take turns working on different parts of the code. I think the best way to learn about how any code works is to poke it and see what wiggles. In addition to diversifying our own knowledge of the software, sharing knowledge allows for life to happen without impacting the overall work.

Each of our sprints is themed and the stories in that sprint relate to the theme. For example, we could have a shipping sprint and all the stories will touch different parts of the shipping code. The following sprint we might have an EMTA theme and we will all be working in the EMTA area of the code. By having everyone work on every part of the software, we are able to get different perspectives on how things are implemented and from that, develop better software.

## Maintenance

We’re not just about developing new features and fixing bugs: we take time during each sprint to do some maintenance work on our code. One way to look at this is tidying up your home. You could choose to only do the dishes once a month, resulting in a dangerous mountain of dirty dishes or you could choose to do the dishes after every meal and be reassured that your dirty dishes won’t tumble out of the sink and on to a poor unsuspecting pet.

The software equivalent to this would be staying on top of any third party library updates. Third party libraries are reusable software components that can be incorporated into software. You can think of these libraries as ingredients to a recipe. It’s unrealistic to plant, grow, and harvest grains every time you want to bake a cake. Instead, you go to a store and purchase flour. However, buying a bag of flour is not the only thing you need to make a cake. The flavor, shape, and texture of a cake depends on how you combine it with other ingredients.

For example, we use a third party library to handle label printing in our software. What this means for us is that we can focus on the meat of the software and not waste time writing code for common tasks, it would be like re-inventing the wheel over and over again for every single software today that prints labels. These libraries in turn need to stay up to date with *their* third party libraries and any security updates. Most of the time, updating these libraries is straight forward. However, there are times where a library is no longer being maintained (i.e. hasn’t been updated in several years) or they make significant changes that make it incompatible with our software. When this happens, we need to research and hopefully find a comparable library if it’s gone stale or we will need to update our code to be compatible with the updated library.

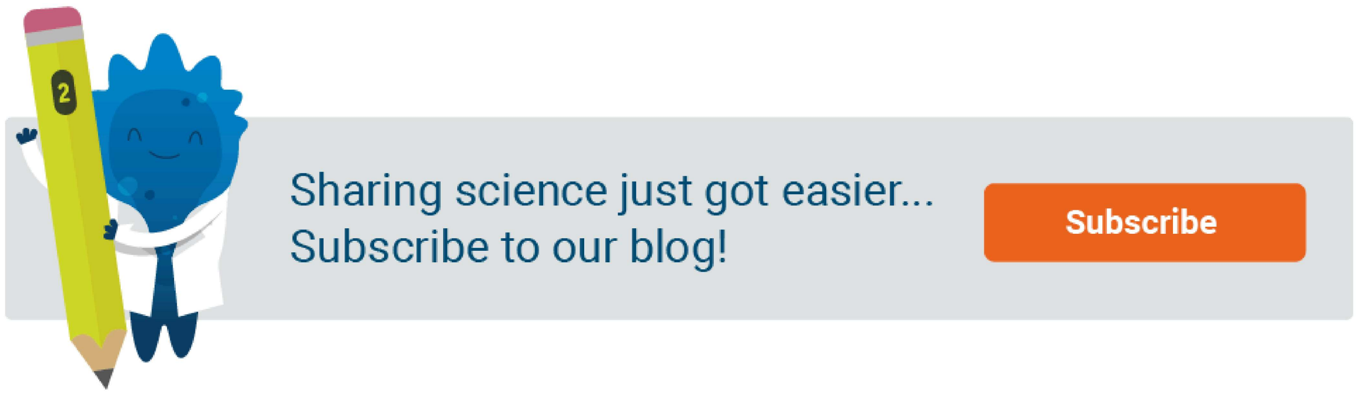
A recent example is updating our software from Python 2.7 to Python 3.5. Python 2.7 is retiring this year, which means there will be no further updates, security or otherwise. This might sound like Greek to you, which brings me to my next analogy. Human languages evolve over time, for example from Ancient Greek to Modern Greek. Books written in Ancient Greek need to be translated into Modern Greek to be easily accessible today. Roughly speaking, we translated our software from Python 2.7 to Python 3.5. There are significant changes to the language and it took us a total of two years, from planning to completion, to do this upgrade while continuing to add new features.

Another area of maintenance is agreeing to the names of the things in our code. In some parts of our code, we might be calling something a sample and in other places, we might be calling it an aliquot. This can lead to confusion as to what the code is supposed to be doing. When this happens, we discuss and agree on a common name. While we do our best to write clean, reusable code when working on a feature, sometimes we have to take short cuts or add a band-aid to get a feature to work. We will then go back to this area later and tidy it up.

Careful consideration was given to all the code that was penned...
Maintainable, stable, and reusable was the mantra repeated on end.

## Conclusion

While none of us are life science majors, we do our best to understand the science. By understanding the hows and whys of our users, we are able to make informed software design choices, resulting in software that helps our users focus on the fun stuff, accelerating research and discovery. We work with transparency, gladly share our knowledge, and love to collaborate, which, incidentally, are the same principles for open science. I hope you enjoyed learning about Addgene's software development team and how we work.



---

**Additional resources on the Addgene blog**

- Learn about what went into our search engine update
- Find all Addgene news

**Resources on Addgene.org**

- Check out our job postings
- Learn more about Addgene's mission

**Topics:** Addgene News

## Leave a Comment

**Deepika Joshi** 1/27/2020, 7:09:44 AM

great post thank you for sharing this information with me
broadband plans

Reply to *Deepika Joshi*

Add Comment