






Aquarium: open-source laboratory software for design, execution and data management

Justin Vrana ¹, Orlando de Lange ², Yaoyu Yang ², Garrett Newman², Ayesha Saleem², Abraham Miller², Cameron Cordray², Samer Halabiya², Michelle Parks², Eriberto Lopez², Sarah Goldberg ², Benjamin Keller ², Devin Strickland ², and Eric Klavins^{2,*}

¹Department of Bioengineering, University of Washington, Seattle, WA, USA and ²Department of Electrical and Computer Engineering, University of Washington, Seattle, WA, USA

*Corresponding author: E-mail: klavins@uw.edu

Abstract

Automation has been shown to improve the replicability and scalability of biomedical and bioindustrial research. Although the work performed in many labs is repetitive and can be standardized, few academic labs can afford the time and money required to automate their workflows with robotics. We propose that human-in-the-loop automation can fill this critical gap. To this end, we present Aquarium, an open-source, web-based software application that integrates experimental design, inventory management, protocol execution and data capture. We provide a high-level view of how researchers can install Aquarium and use it in their own labs. We discuss the impacts of the Aquarium on working practices, use in biofoundries and opportunities it affords for collaboration and education in life science laboratory research and manufacture.

Key words: automation; replicability; software; LIMS

1 Introduction

As the scale of scientific research expands, systems to support replicable methods are increasingly important (1). Critical to replicability is the question of how experiments are described and how closely these descriptions are followed. Working practices for conducting biology experiments fall on a continuum between artisanal and highly standardized. On one end, idiosyncratic decisions made on the fly by a few people appear throughout the experimental design. At worst, these decisions are poorly documented, while at best they introduce extra experimental factors that make results challenging to compare and replicate. At the other end, researchers follow well-established protocols and do not deviate from them. This is due

to either top-down enforcement, as with clinical research, or to the fact that doing so is simpler and more reliable, as with kits for common procedures such as plasmid DNA isolation. Similarly, record keeping varies from hand-written laboratory notebooks and manually curated digital records to fully structured electronic notebooks.

A consequence of less structured approaches is that many experiments cannot be repeated by different researchers (2, 3). These types of issues are typically described under the terms *replicability* (reproducibility of results) (4–7). While replicability in science is a complex and multifaceted issue (8, 9), enforcement of standardization of experimental work can result in more replicable data collection (10, 11). However, the reality is that maintaining this standardization and record keeping is laborious and

Submitted: 1 October 2020; **Received (in revised form):** 12 January 2021; **Accepted:** 22 January 2021

© The Author(s) 2021. Published by Oxford University Press.

This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited. For commercial re-use, please contact journals.permissions@oup.com

researchers often fail to enforce strict standards on themselves at the bench (12).

Robotic automation has been proposed as a solution for replicable large-scale experimentation by delivering consistent performance of delicate high-throughput procedures (13). However, commercially available liquid handling robots and general-purpose manipulators have high-upfront costs and are laborious to reconfigure or reprogram (14). While robotic systems are continually improving, labs solely reliant on robots to carry out experimental work are rare. Many experiments involve tasks for which a human operator is well suited and more cost-effective than currently available robots; for example, tasks involving delicate hand-eye coordination or variable inputs and protocols. Hence, it seems likely that human researchers will be involved in benchtop experimentation for some time, and thus the potential for non-standardized execution and sub-optimal record keeping will remain as a concern.

To address this challenge, we built Aquarium—a web-based application that integrates experimental design, inventory management, protocol execution, and data collection. Aquarium supports flexible development and deployment of standardized workflows, composed of modular protocols that drive on-screen, step-by-step instructions for human technicians. During execution, experimental data and metadata are captured in forms or uploaded as files. The software automates computations involved in preparing and tracking samples through protocol execution. Aquarium also provides features to plan complex experiments involving many samples and protocols.

Aquarium integrates two key software innovations: Aquarium Workflow Language (AWL) for defining custom laboratory workflows and Krill, a protocol language for describing replicable laboratory instructions. AWL is a dataflow programming language (15) that represents a laboratory workflow as a network of modular work units linked by inputs and outputs. Borrowing concepts from visual programming languages, such as Scratch (16), protocols are represented graphically as blocks that can be wired together to create workflows. Krill is a Ruby domain-specific language that complements AWL by capturing granular instructions for a protocol as computer code. In addition to complex procedural steps such as if-then statements, loops and calculations, Krill has methods to facilitate sample flow management and render instructions for technicians working at the bench. Through AWL and Krill, Aquarium provides interactive web-based interfaces to build executable protocols, design experimental workflows based on these protocols, manage the execution of protocols in the lab and automatically record the resulting data.

Aquarium also features a Python application program interface (API), called *Trident* that provides a common interface for other applications and scripts to interact with Aquarium, for example, in planning complex workflows or extracting detailed datasets. These three programmatic interfaces, combined with inventory management and human-centered execution, make Aquarium a comprehensive, open-source software platform that facilitates low-cost scaling of laboratory research while retaining replicability and flexibility.

2 Results

2.1 Planning laboratory work with Aquarium

From the perspective of the researcher, planning laboratory work is the primary interaction with Aquarium. Researchers

design *plans* using a graphical user interface (GUI) that resembles a sketch board (Figure 1). Plans are built from workflows, essentially stereotyped series of procedures, in which materials pass from an initial state to a final state. Within plans, each input sample passes through a series of work modules termed *operations* (Figure 2b), to produce desired output samples and data. For example, a plan that ends with a sequence-verified plasmid stock (Figure 1) might include a series of operations such as PCR, DNA assembly, bacterial transformation and plasmid DNA purification. A plan may represent a fixed workflow that always executes in the same way, or it may be extended as it is executed. Thus, enabling the use of Aquarium for either manufacturing or exploratory research and development.

Operations correspond to units of work that can be performed on one sample, by one person, within a single work session. Each operation will generally output a sample that can be stored or used in a variety of other operations. Operations are wired together such that the output of one operation is automatically routed to and triggers the execution of one or more subsequent operations (Figure 2b). Each operation is defined by valid inputs and outputs as well as a detailed laboratory protocol, written in Krill, that renders on-screen instructions to guide technicians. An Aquarium plan can encode arbitrarily large and complex programs of work that progress automatically. As the plan is executed, the state of inventory is automatically updated and data are captured, stored and made available through the GUI as well as the Python API.

2.2 Executing laboratory work

Aquarium contains its own laboratory information management system (LIMS) that tracks lab inventory. Through the LIMS, changes in inventory are recorded automatically as a part of protocol and workflow execution, rather than requiring manual updates. Hence, the workflow planner and Krill can reliably use the LIMS as an up-to-date representation of the laboratory.

In Aquarium, a physical object is referred to as an *item*. Each item has a recorded location and may have associated data (Figure 3). An item is an instance of a *sample*, which is a class of physical objects in the laboratory defined by a set of descriptors determined by a *sample type*. The information fields used to define each sample type are chosen by the user and then apply to every sample of that sample type. For example, a user may define a 'plasmid' sample type, including fields for information on sequence, length and selectable markers, which would have to be defined for each plasmid sample in the database. Using sample types ensures that inventory descriptions are standardized while allowing the flexibility of custom definitions based on user needs.

Each item belongs to a user-defined *object type*. One key parameter for the definition of an object type is the default *location wizard* for items of this object type. Location wizards correspond to storage locations such as fridges and freezers and are represented as matrices with unique numbered positions for items within boxes, organized into rows and shelves (Figure 3b). Other than the location wizard, the name of each object type is its most important defined parameter and will indicate a physical state of the sample, reflecting how it manifests or is used in the laboratory. For example, a plasmid sample might have items associated with it belonging to a number of object types such as 'Plasmid miniprep', 'Gibson assembly reaction product' or 'E. coli overnight culture'. As items are generated in the course of laboratory work, they are automatically assigned ID numbers as

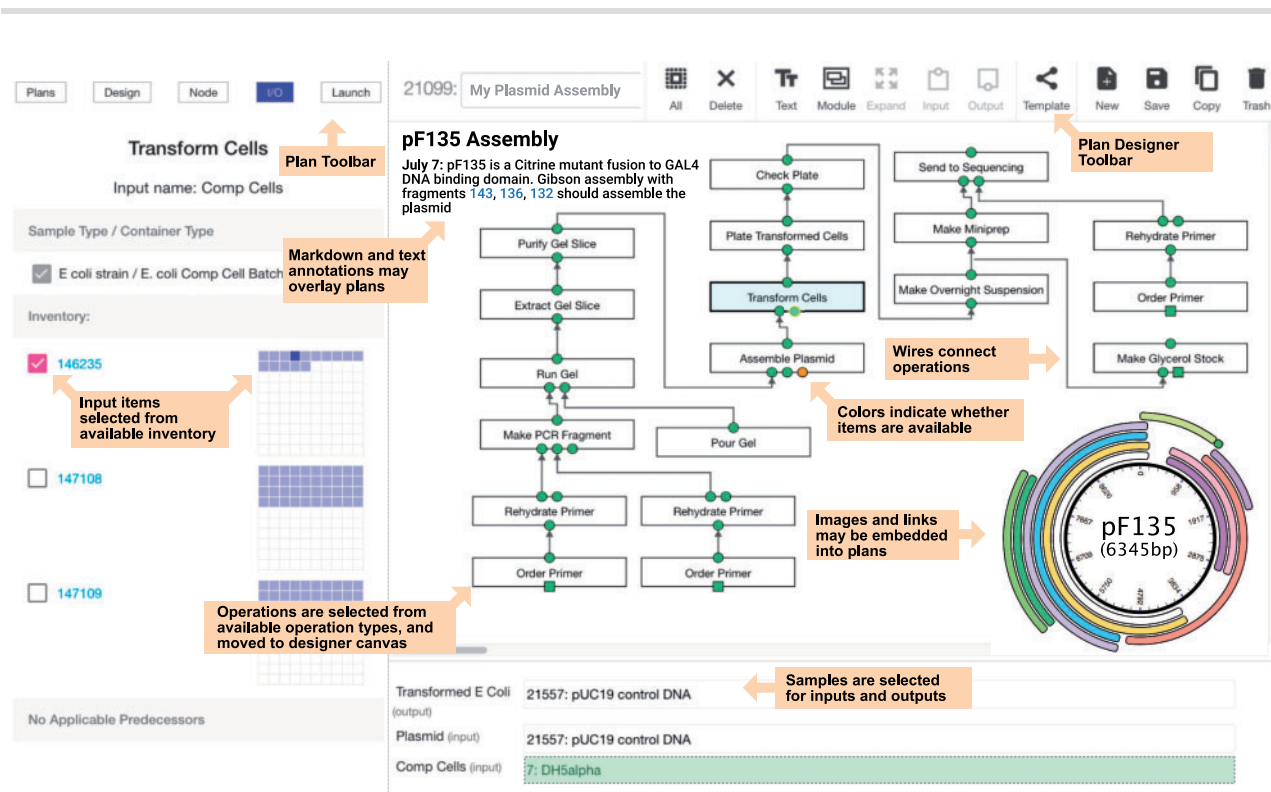


Figure 1. Workflow designer interface. Experimental plans can be created by dragging operation types (not shown) onto the designer canvas (right side) to create operations. Selecting a given operation input or output node users are prompted to select from a list of compatible up or downstream operations to create a custom workflow. Available inventory for each operation input is selected in the input view (bottom) and the I/O view (left). Designer tools are available for creating templates and modifying/copying plans. Additionally there are several plan tools (top left) available for investigating input/output specifications, managing existing plans, and launching plans. The designer also features annotation capabilities, allowing embedded text (such as Markdown; <https://daringfireball.net/projects/markdown/>), images or links.

well as locations according to the location wizard defined by the item's object type.

Once the items required to initiate an operation in a plan are available in the lab, the inputs to the operation are satisfied and the operation can be executed. In the manager subsystem, an executable operation is batched into a *job* with operations of the same type to be executed together (Figure 2). A job may include operations from many different plans and researchers but can only be created from operations of the same type. A strict execution policy governs how and when operations can be batched into jobs and executed in the lab (Supplementary Figure S1). Running a job launches a graphical user interface that displays step-by-step instructions to guide a technician through the steps of the operation protocol (Figure 4).

2.3 Rendering instructions using the krill protocol language

The Krill protocol language produces detailed, context-specific instructions for how materials and data should be handled for each operation (Figure 4). To accomplish this, Krill provides methods that allow arbitrary computations to be rendered dynamically, so that specific instructions presented to the technician can be made to reflect not only the number of items being processed, their locations and ID numbers, but also the results of calculations such as pipetting volumes based on the molarity of a solution. Thus, a Krill protocol describes a procedure in enough detail that it can be replicated by another person or lab by following specific instructions on a tablet or computer screen. Krill methods include those to execute complex calculations, retrieve

and generate data, add and remove inventory, display videos and photos, retrieve user input, create interactive timers, output audio alarms and send emails, among other functions. Krill extends Ruby (17), a popular, dynamic, object-oriented language used in web development. To facilitate development, Krill provides ways of creating libraries of reusable code. A version control system allows a lab to record changes to their protocols over time, or revert their protocols to past versions.

The Krill protocol is supplemented by two functions that facilitate the proper execution of the protocol (Figure 4). The *cost model* calculates how much an operation may cost at the time of execution. Cost models can use any information from Aquarium to perform cost calculations, but typically calculations use properties of samples or items used in an operation. For example, an operation that orders a synthesized piece of DNA may use the length and sequence of the DNA and check with a vendor website to establish an accurate monetary cost. Some protocols may be labor-intensive, and so operation cost models can include an estimation of the 'labor rate' and the average length of time required to complete the protocol. Additional features in Aquarium allow the generation of monthly spending reports and budget tracking for each user or user group. The *precondition* defines conditions required for a protocol to be run; the default precondition is always true. Preconditions are a critical part of an operation's execution policy (Supplementary Figure S1) and can be used to institute more complex experimental workflows. For instance, enforcing a 12 h delay to wait for an *E. coli* plate to grow. Like the cost model, precondition code may use any of the other subsystems to establish running conditions. For example, an operation that runs a

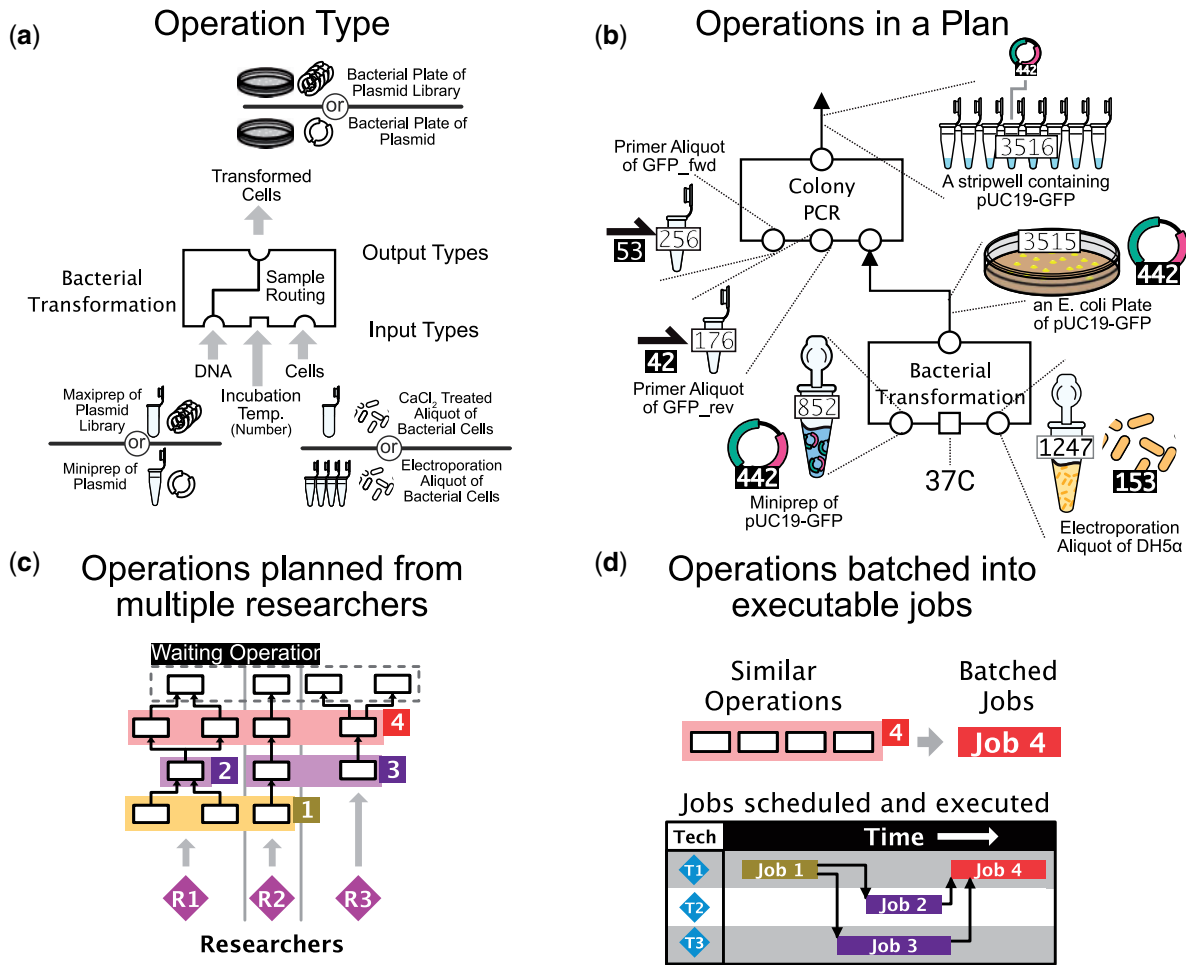


Figure 2. AWL Depictions of operation, operation type, plan, and job models that comprise an AWL. (a) An example of a ‘Bacterial Transformation’ operation type is displayed. Operation types define specific ways in which input samples and items can be processed to produce outputs. Each operation type contains specifications for its input and output types. For example, the ‘DNA’ input of a hypothetical bacterial transformation operation type may be satisfied by a ‘maxiprep of plasmid library’ or a ‘miniprep of plasmid’. Input and output types are entirely customizable and may include any number of sample type and object type specifications. Sample routing, if provided, ensures the input and output samples are mapped correctly upon operation execution; here the input ‘DNA’ sample will be mapped to the ‘Transformed Cells’ output sample. Non-inventory inputs (i.e. parameters) can also be defined as inputs to operation types. (b) An example of a bacterial transformation uses specific items in the LIMS and a parameter (37°C) to produce a bacterial plate. After executing the transformation, the output plate is wired to the colony PCR operation. The colony PCR outputs the amplicon to an empty well in a stripwell. Notice that the operation type sample routing ensures sample information (here pUC19-GFP, sample 442) is maintained throughout the series of tasks. (c) Operations from several different researchers and different plans can be batched together into jobs if they have the same operation types. For instance, all ‘Colony PCR’ operations from all users can be run as a single job. (d) Once operations have been batched into jobs, jobs can be run divorced from Aquarium plans because all necessary information for execution is included in the job. These jobs can then be performed concurrently by separate technicians.

colony PCR on a bacterial plate may halt operation if there are data associated with the plate indicating there was contamination, or if the plate is less than 12 h old and thus not ready to be run. Finally, the *documentation* contains human-readable markdown text that describes how the protocols are used and executed.

2.4 Recording and accessing experimental data

Aquarium records data in several ways. By default, Aquarium automatically logs protocol metadata during job execution. These metadata include operations batched within the job as well as the identity of submitting users and IDs of source plans for each operation. Job logs also capture technician identity, job start and end time, timestamps for each step in a protocol, job error records, and inventory handled. In addition to the job metadata, Aquarium has generic data associations that record

data as attachments to inventory items, operations or plans. Data associations may include numerical (e.g. DNA concentration), text (e.g. experiment notes) or file uploads (e.g. sequencing results). Data associations can be created manually through the GUI or automatically by a protocol during a job. During execution, protocols may include specific steps instructing the technician to record or upload data (Figure 4-vii). Post-execution, data can be accessed by researchers through the GUI, or by scripting via Aquarium’s Python API, known as Trident. Trident allows custom Python applications that power visualizations, interfaces, reports or machine-learning workflows to communicate easily with Aquarium.

2.5 Interacting with Aquarium

There are five major interfaces in Aquarium: designer, plans, manager, samples and developer. The designer tab provides

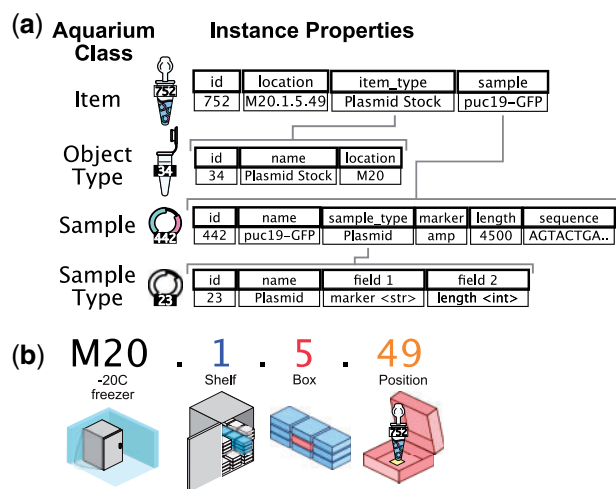


Figure 3. Aquarium inventory types. The procedures of a given research lab will require handling of multiple different sample types, representing things like DNA plasmid, various cell lines or chemical reagents. (a) For instance samples of type 'Plasmid' may be defined by a marker, length or sequence. Properties for sample types are defined by the user allowing for custom definition of inventory. In Aquarium, there are no hardcoded concepts of 'Plasmids' or any other form of laboratory inventory. Once a sample type has been defined samples can then be added to the database. Items are physical manifestations of samples and always have a location, as well as the ability to carry data associations. Each item is of a given type, known as an object type defining relevant physical properties relevant to laboratory handling. (b) Each item produced automatically gets assigned a location. How this location is assigned is reconfigurable in Aquarium. Here, a -20C freezer is designated 'M20' and has three dimensions (shelf, box and position). The location designation and capacity along the dimensions are customizable. Which types of items go into which locations are defined in the item's object type. Proper management of item locations in Aquarium is critical as this feeds into protocol execution, which may use (or alter) item location during execution.

access to the AWL interface, in which users can draft and launch plans, selecting from available operation types. The plans interface offers a summary of launched plans including up to date sample and status data. The manager tab is used to batch operations into jobs and run them. Within the manager interface, all operations are accessible, grouped by category, operation type and status. Launching a job from the manager interface starts with the on-screen instructions used by technicians (Figure 4). The samples interface provides searchable access to inventory. The developer tab provides access to an Interactive Development Environment (IDE) where Krill code for protocols can be written and tested directly in the web browser (Figure 5).

3 Discussion

3.1 Specialization of roles

Aquarium facilitates, but does not require, a division of personnel roles roughly corresponding to the different front-end interfaces, thereby facilitating the standardization of laboratory workflows as a low-cost and flexible alternative to robotic automation systems. The module composition approach implemented by AWL is intended to allow for flexible workflow design, reflecting the reality of discovery-phase research, while gaining the benefits of standardization, including replicability and efficiency gains from batched jobs. Laboratory roles can further be divided into lab managers, scheduling and assigning jobs, and technicians, executing jobs. The following role

descriptions are based on our experience using the system while recognizing that individual members of laboratory personnel have often adopted multiple or blended roles.

Researchers, including graduate students and postdocs, use Aquarium's LIMS to define new samples and the AWL (Figure 1) to design and launch plans. Once a researcher is ready to launch a plan, costs are computed with the operation cost models, and the researcher assigns the total to a budget that Aquarium uses to automatically track spending and generate reports. After launching, plans become visible within the plans interface, where the researcher can see the status of each operation in the plan, as well as access collected data. Some power users in the researcher role entirely bypass the Aquarium browser front-end and instead add sample definitions, submit plans and retrieve data through the Trident API. We have found that these tools allow researchers to spend minimal time at the lab bench, and more time reading literature, planning experiments and analyzing data.

Developers use Aquarium's IDE to specify operation types and associate code (Figure 4). In our experience Aquarium protocol drafting typically begins with a pre-existing paper-based or digital protocol as references, with the developer often working with an experimentalist for guidance. Once a protocol has been tested in the IDE, it can be deployed, making it available to add to plans and run in jobs. Developers also work with researchers and managers to develop the cost model, documentation and preconditions to create cohesive workflows (Figure 4d).

Managers batch operations and schedule and launch jobs, in the process deciding how many operations to include in each job, when each job should be executed and which technician should run the job.

Technicians execute jobs at the lab bench in accordance with the on-screen instructions provided by the protocol code (Figure 4). Instructions typically include item retrieval and storage, sample preparation and handling, operation of laboratory instruments and data uploading. Technicians may be guided to directly upload data files from cameras or other equipment, or asked to create data based on prompts (e.g. answering whether or not a band of a given length is present on a gel).

As well as formalizing personnel roles, Aquarium facilitates a conceptual shift to thinking about all laboratory work, including both manufacturing and experimentation, as composed of modular units, with the steps of each modular unit standardized. Standardization of laboratory methods can be beneficial for replicability (10, 11) and Aquarium provides a means to ensure that standardized procedures are both established and followed. This arrangement can also reduce experimental bias and shield sensitive sample information.

3.2 Aquarium use cases

Aquarium has been used for a number of applications beyond the work of academic research groups. These include biofoundries, service laboratories and laboratory skills training.

Biofoundries are facilities providing laboratory services to the synthetic biology research community, generally including plasmid assembly and strain construction (1). Aquarium's built-in abstraction barrier between design and execution, and system for efficient task management are well suited to support biofoundries. Aquarium has supported a biofoundry at the University of Washington, the UW BIOFAB that was first developed for internal use in 2014 and then made publically accessible in 2016. Between 2014 and 2020, the UW BIOFAB has run over 30 000 jobs, serving 319 different users. BIOFAB technicians

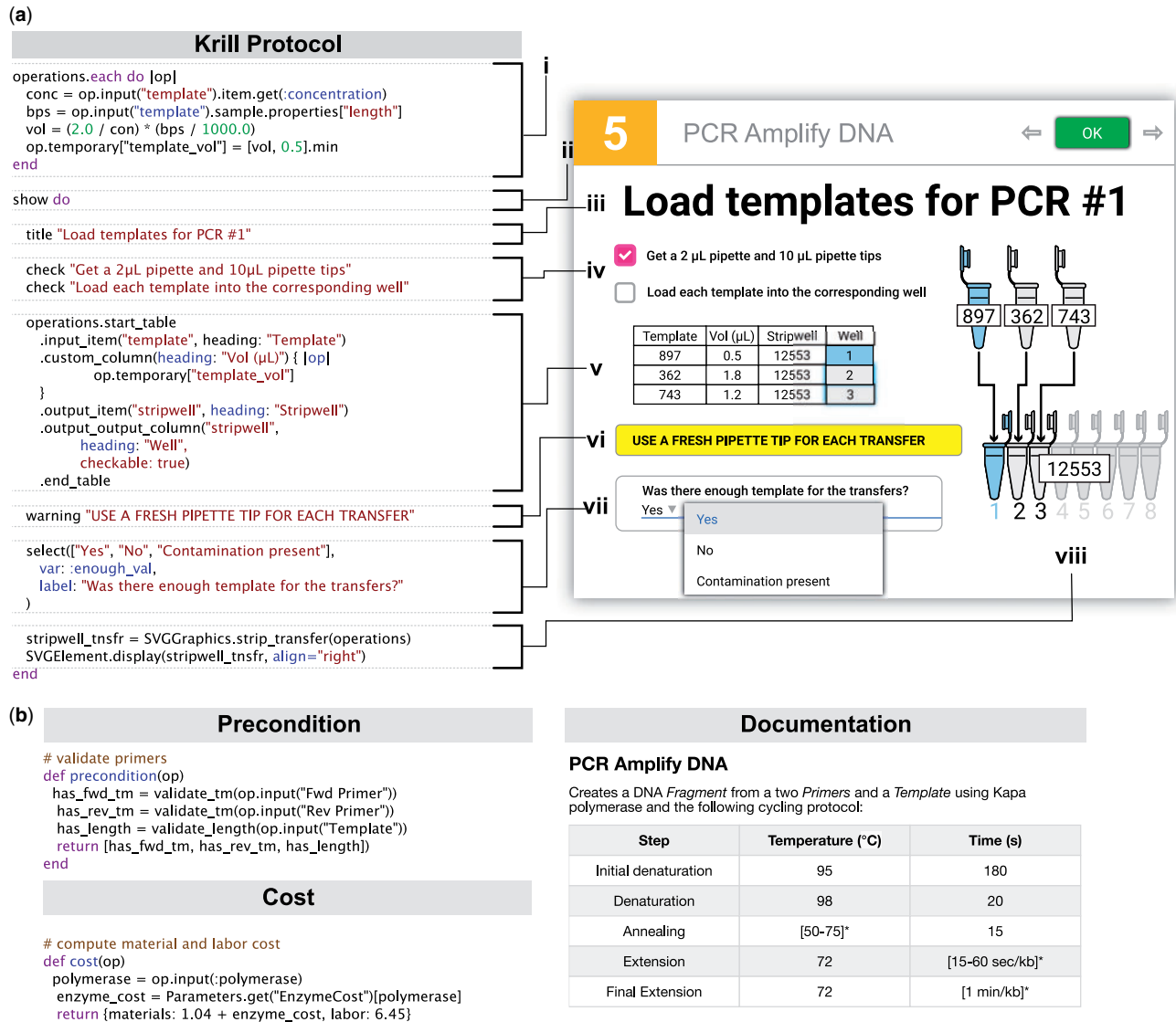


Figure 4. Krill protocol. An example of Krill protocol code and its corresponding rendered protocol instructions. Operation types have four sets of code that govern operation behavior and scheduling: Krill protocol, precondition, documentation and cost. (a) Shown here is a snippet of Krill protocol code for the load template step in a PCR amplification protocol with lettering highlighting aspects of the code. Operations are batched into a single job; when the job is executed, Krill code can access the input and output information of all batched operations. (i) In this simple example, the template volume is calculated for each operation before generating instructions for loading template DNA into wells. (ii) A new protocol step is rendered with a `show` block. (i–vii) Within the `show` block, various elements are rendered for the technician. (iii, iv) A title is displayed and has two checkboxes. Checkboxes must be checked before proceeding to the next step, forcing the user to be attentive. Operations are iterated to display a table that uses the computed template volume, inputs and outputs of the operations. (v) Tables can be interactive and may include text inputs or checkable boxes. (vi) A visible `warning` is displayed. (vii) A `selection` input instructs the user to select from a list of options; numerical, textual and file upload inputs are also possible through Krill. (viii) Finally, an SVG graphics element can be rendered on the fly using operation information. (b) Optional precondition code governs when operations can be scheduled into jobs. Cost model computes monetary costs prior to plan launch and documentation provides readable instruction about the underlying protocol.

have assembled 23 million base pairs of DNA using 8.8 million base pairs of fragment DNA amplified in-house, and have built over 5700 different yeast strains. This work has supported synthetic biology research efforts of the Klavins lab and collaborators (18–22), as well as other users with no shared research interests. The UW BIOFAB first implemented cloning and yeast construction services but has since moved on to offer plant cultivation and transformation, mammalian cell culturing, protein engineering, and next-generation sequencing and other workflows.

Operated by private companies or public institutions, service laboratories support clinical diagnostics, agricultural soil and

crop analytics, and forensics. The impacts of the global pandemics (such as COVID-19) highlighted the importance of low-cost, flexible tools that can support the rapid scaling of laboratory services both in terms of throughput and geographical reach. Aquarium was recently used to support an HIV-resistance screening workflow for use in the developing world (23), taking advantage of Aquarium’s graphical technician interface, data collection management and options for rapid deployment into new devices and locations.

Given the instructional efficacy of the technician interface, we have also found utility for using Aquarium as an education tool, teaching university laboratory courses with the software.

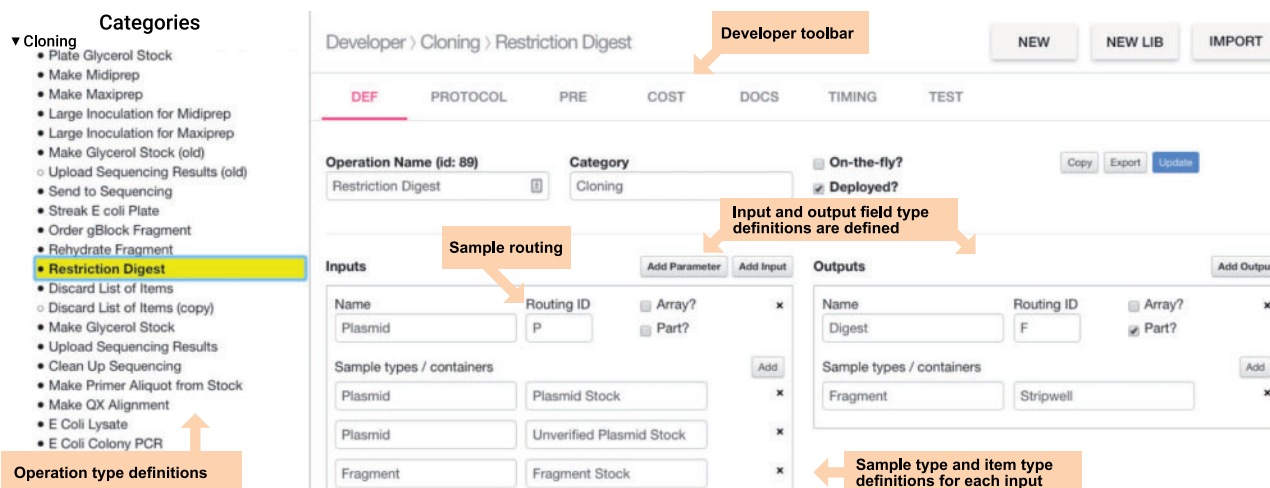


Figure 5: Operation type integrated development environment (IDE). New operation types are developed through the operation type IDE. Input/output specifications are defined for each operation, along with sample type and object type specifications for each input or output. Several tools are available (top) for editing Krill protocol, precondition, cost and documentation code. A built-in protocol testing environment is available (top right) to speed workflow development.

Aquarium's technician interface (Figure 4) delivers step-by-step instructions at the lab bench, reducing the need to front load learning of methods, similar to Just-in-Time Teaching (24). It has also been used to support undergraduate laboratory training courses at the University of Washington and elsewhere.

3.3 Comparison with other software

Aquarium is part of a growing ecosystem of laboratory research software that we believe will become central to the working practices of researchers over the next decade. Similar to Aquarium, existing software platforms like those provided by Benchling, Riffyn, Teselagen and Transcriptic have fully featured LIMS capabilities that connect to protocols and workflows in meaningful ways. However, as far as we are aware, Aquarium is unique in its support for human-centric workflow execution which allows labs to leverage existing equipment and personnel. This is in contrast to robotic automation approaches, like those provided by Transcriptic, that integrate workflow execution primarily via laboratory robotics, which allows high-throughput experimental automation. Other platforms specialize in other aspects of the laboratory research process, such as Riffyn, which provides sophisticated tools for connecting and integrating workflow processing to data analytics; Benchling and Teselagen integrate aspects of biodesign to LIMS and workflow processes. Unlike these tools, Aquarium has limited capabilities for data processing and biodesign. Instead, Aquarium focuses on flexible workflow planning and execution, leaving design and data analytics to other software better suited to that task, such as those mentioned above. Aquarium's open-source Python API and flexible LIMS invite future integrations of Aquarium with other software systems.

3.4 Future development of Aquarium

We support a growing Aquarium user community and are aware of at least eight groups that have set up and operated independent Aquarium servers for applications ranging from plant transgenics, to microbial strain construction to biomedical diagnostics. Aquarium is distributed under the MIT license to promote adoption by, and contributions from, users in any setting, whether academic, commercial or educational.

An online hub (<https://www.aquarium.bio/>) for sharing and peer-curation of Aquarium workflows supports the growing user community. Aquarium workflows currently can be exported and published as Github repositories. Current development plans include simplification of the Krill protocol language to lower barriers and allow for wider use of Aquarium in life science research labs.

While Aquarium provides a way to formalize scientific workflows and their execution so as to allow researchers without high-level knowledge of the protocol to perform experiments reliably, it does not provide guidance on experimental design choices. However, there have been many recent advances on computer-aided design (CAD) tools for science (25–29). Using Aquarium and its Python API provides a way to execute experimental plans developed by CAD software and return results in a machine-readable format. In the future, one can imagine combinations of such systems that mediate automatic design and submission of experiments, execution through Aquarium, automated extraction and analysis of results, and rapid redesign.

4 Materials and methods

4.1 Glossary of Aquarium terminology

The following is provided for disambiguation and covers Aquarium terminology used in this article for which an alternative common-usage definition exists. For a more complete description of relevant terminology, please refer to the documentation found at www.aquarium.bio.

Sample: A biologically unique entity, with properties defined by the needs of the user. A description of a specific plasmid is a sample.

Sample type: A category of samples, such as 'Plasmid' or 'Mammalian Cell Line'.

Item: A physical manifestation of a sample that exists in the laboratory. A miniprep stock is an item of a given plasmid sample.

Object type: A category of items that includes a name and a default location, and belongs to a particular sample type. Examples could be 'Plasmid Stock' or '400 mL Bottle of Media'.

Operation: The basic unit of laboratory work planned in Aquarium, in which inputs are converted to outputs according to a protocol defined using the Krill protocol language.

Operation type: A protocol definition, which governs how human-readable instructions are rendered for a batch of operations, and how operations change the inventory and other data.

Plan: A set of operations that are linked by connecting inputs and outputs.

Job: A batch of operations of the same type that are run concurrently by a technician following instructions generated from the operation type protocol written in Krill.

Krill: The domain-specific language used to define protocols, a core element of an operation type. Krill extends Ruby by including methods specific for managing Aquarium objects and generating on-screen instructions for technicians.

Data association: Key-value pair that is associated with plans, operations or items. Data associations are added automatically during the execution of a job or manually by a user.

4.2 Aquarium license and software availability

Aquarium is distributed under the open-source MIT license. Aquarium, documentation and installation instructions are freely available (<https://www.aquarium.bio/>) along with links to Dockerized versions of the software. Code is maintained on Github (<https://github.com/aquariumbio/aquarium>). Aquarium's Python API (Trident) is also under the open-source MIT license and is hosted on the open-source python repository at PyPI (<https://pypi.org/project/pydent/>) and its documentation and installation instructions are also freely available (<https://aquariumbio.github.io/trident/>).

4.3 Aquarium software implementation

Aquarium is implemented as a browser-based Ruby-on-Rails application (<https://rubyonrails.org/>), with an AngularJS (<https://angularjs.org/>) and HTML5 front-end. The current implementation of Krill leverages Ruby (<https://www.ruby-lang.org/en/>), which is a popular, dynamic, object-oriented language used in web development. Data models are implemented using a MySQL relational database (<https://dev.mysql.com/>).

Aquarium is distributed as a Docker (<https://www.docker.com>) image (<https://hub.docker.com/repository/docker/aquariumbio/aquarium>), along with Docker Compose (<https://docs.docker.com/compose/>) scripts (<https://github.com/aquariumbio/aquarium-deployment>) that can be used to orchestrate backend, relational database and front-end services. The Trident API is implemented in Python using open-source libraries and is available as a Python package via pypi.org (<https://pypi.org/project/pydent/>).

SUPPLEMENTARY DATA

Supplementary Data are available at SYN BIO Online.

Funding

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001117C0095. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency

(DARPA), the Department of Defense or the United States Government.

Acknowledgements

Justin Vrana, Orlando de Lange, Devin Strickland, and Ben Keller wrote and revised the manuscript. Eric Klavins and Yaoyu Yang conceptualized the software and wrote most of the application code. Ben Keller, Abe Miller, Garrett Newman, Phuong Le, Justin Vrana contributed to the application code. Abe Miller and Ben Keller prepared documentation and installation scripts. Tileli Amimeur, Nick Bolten, Leandra Brettner, Cameron Cordray, Miles Gander, Sarah Goldberg, Samer Halabiya, Seunghee Jang, Yokesh Jayakumar, Eriberto Lopez, Jon Luntzel, Cannon Mallory, Abraham Miller, Garrett Newman, Michelle Parks, Sundipta Rao, Ayesha Saleem, Devin Strickland, Chris Takahashi, Justin Vrana, Yaoyu Yang and David Younger developed Aquarium workflows. Cami Corday, Samer Halabyla, Aza Allen and Michelle Parks executed and tested workflows and contributed to project ideas while managing the experimental laboratory.

Conflict of interest statement. None declared.

References

- Jessop-Fabre, M.M. and Sonnenschein, N. (2019) Improving reproducibility in synthetic biology. *Front. Bioeng. Biotechnol.*, 7, 18.
- Begley, C.G. and Ellis, L.M. (2012) Drug development: raise standards for preclinical cancer research. *Nature*, 483, 531–533.
- Fang, F.C. and Casadevall, A. (2012) Reforming science: structural reforms. *Infect. Immun.*, 80, 897–901.
- National Academies of Sciences, Engineering, and Medicine. (2019) *Reproducibility and Replicability in Science*. National Academies Press.
- Goodman, S.N., Fanelli, D. and Ioannidis, J.P.A. (2016) What does research reproducibility mean? *Sci. Transl. Med.*, 8, 341ps12.
- Prinz, F., Schlange, T. and Asadullah, K. (2011) Believe it or not: how much can we rely on published data on potential drug targets? *Nat. Rev. Drug Discov.*, 10, 712–712.
- Ioannidis, J.P.A. (2005) Why most published research findings are false. *PLoS Med.*, 2, e124.
- Baker, M. (2016) 1,500 scientists lift the lid on reproducibility. *Nature*, 533, 452–454.
- Fanelli, D. (2018) Opinion: is science really facing a reproducibility crisis, and do we need it to? *Proc. Natl. Acad. Sci. USA*, 115, 2628–2631.
- Fonio, E., Golani, I. and Benjamini, Y. (2012) Measuring behavior of animal models: faults and remedies. *Nat. Methods*, 9, 1167–1170.
- Arroyo-Araujo, M., Graf, R., Maco, M., van Dam, E., Schenker, E., Drinkenburg, W., Koopmans, B., de Boer, S.F., Cullum-Doyle, M., Noldus, L.P.J.J. et al. (2019) Reproducibility via coordinated standardization: a multi-center study in a Shank2 genetic rat model for autism spectrum disorders. *Sci. Rep.*, 9, 1–10.
- Nussbeck, S.Y., Weil, P., Menzel, J., Marzec, B., Lorberg, K. and Schwappach, B. (2014) The laboratory notebook in the 21st century: the electronic laboratory notebook would enhance

- good scientific practice and increase research productivity. *EMBO Rep.*, 15, 631–634.
13. Miles, B. and Lee, P.L. (2018) Achieving reproducibility and closed-loop automation in biological experimentation with an IoT-enabled lab of the future. *SLAS Technol.*, 23, 432–439.
 14. Naugler, C. and Church, D.L. (2019) Automation and artificial intelligence in the clinical laboratory. *Crit. Rev. Clin. Lab. Sci.*, 56, 98–110.
 15. Dennis, J.B. (1975) *First Version of a Data Flow Procedure Language*. Programming Symposium, Springer Berlin Heidelberg, Berlin, Heidelberg. 362–376.
 16. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. et al. (2009) Scratch: programming for all. *Commun. ACM*, 52, 60–67.
 17. Ruby Programming Language. (2020) <https://www.ruby-lang.org/en/> (20 October 2020, date last accessed).
 18. Younger, D., Berger, S., Baker, D. and Klavins, E. (2017) High-throughput characterization of protein-protein interactions by reprogramming yeast mating. *Proc. Natl. Acad. Sci. USA*, 114, 12166–12171.
 19. Khakhar, A., Leydon, A.R., Lemmex, A.C., Klavins, E. and Nemhauser, J.L. (2018) Synthetic hormone-responsive transcription factors can monitor and re-program plant development. *Elife*, 7, e34702.
 20. Groves, B., Khakhar, A., Nadel, C.M., Gardner, R.G. and Seelig, G. (2016) Rewiring MAP kinases in *Saccharomyces cerevisiae* to regulate novel targets through ubiquitination. *Elife*, 5, e15200.
 21. Khakhar, A., Bolten, N.J., Nemhauser, J. and Klavins, E. (2016) Cell-cell communication in yeast using auxin biosynthesis and auxin responsive CRISPR transcription factors. *ACS Synth. Biol.*, 5, 279–286.
 22. Gander, M.W., Vrana, J.D., Voje, W.E., Carothers, J.M. and Klavins, E. (2017) Digital logic circuits in yeast with CRISPR-dCas9 NOR gates. *Nat. Commun.*, 8, 15459.
 23. Panpradist, N., Beck, I.A., Vrana, J., Higa, N., McIntyre, D., Ruth, P.S., So, I., Kline, E.C., Kanthula, R., Wong-On-Wing, A. et al. (2019) OLA-Simple: a software-guided HIV-1 drug resistance test for low-resource laboratories. *EBioMedicine*, 50, 34–44.
 24. Marrs, K.A. and Novak, G. (2004) Just-in-time teaching in biology: creating an active learner classroom using the Internet. *Cell Biol. Educ.*, 3, 49–61.
 25. Rohl, C.A., Strauss, C.E.M., Misura, K.M.S. and Baker, D. (2004) Protein structure prediction using rosetta. *Methods Enzymol.*, 383, 66–93.
 26. Hillson, N.J., Rosengarten, R.D. and Keasling, J.D. (2012) j5 DNA assembly design automation software. *ACS Synth. Biol.*, 1, 14–21.
 27. Untergasser, A., Cutcutache, I., Koressaar, T., Ye, J., Faircloth, B.C., Remm, M., Rozen, S.G. et al. (2012) Primer3—new capabilities and interfaces. *Nucleic Acids Res.*, 40, e115–e115.
 28. Appleton, E., Tao, J., Haddock, T. and Densmore, D. (2014) Interactive assembly algorithms for molecular cloning. *Nat. Methods*, 11, 657–662.
 29. Nielsen, A.A.K., Der, B.S., Shin, J., Vaidyanathan, P., Paralanov, V., Strychalski, E.A., Ross, D., Densmore, D. and Voigt, C.A. (2016) Genetic circuit design automation. *Science*, 352, aac7341.