Research Proposal A GUI Bioinformatic Workflow Management tool with a basis in OpenWDL

Tra-Vaughn M.C. James
Earlham College
Richmond, Indiana
tjames19@earlham.edu

ABSTRACT

Bioinformatics is an interdisciplinary field between biology, computer science and statistics, in which software is able to augment, and analyze biological data. To convert this data into useful information the use of various tools, parameters, and dynamically changing reference data is required [16]. As a result, workflow management system such as Shakemake and OpenWDL were created to develop workflows that are scalable, repeatable and shareable. Such tools have become pivotal within large-scale labs, saving bioinformaticians time, resources and streamlining the analysis process. However, many of these workflow managers are bespoke in nature, limited to only creating workflows for specific sects of research. Moreover, newer workflow management systems, often have a steep learning curve, providing unneeded difficulty and a massive time cost. I propose a GUI workflow management tool that is easy to learn and dynamic in nature, allowing for the creation of workflows that can be applied to mainstream research in genomics and RNA-sequencing.

KEYWORDS

Bioinformatics, workflow, workflow manager, workflow management, workflow management tool, OpenWDL, WDL, Workflow Description Language

ACM Reference Format:
Tra-Vaughn M.C. James. 2022. Research Proposal A GUI Bioinformatic Workflow Management tool with a basis in OpenWDL . In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Today, Bioinformatics is an evolving field, in which computing resources have become more powerful, readily available and workflows have increased in complexity. New workflow management tools (WMT) attempt to develop software that fully harnesses this computational power, creating intuitive implementations utilizing machine learning techniques. This streamlines the design of complex workflows. However, overarching problems still remain that newer workflow management tools do not fully address: they are too specific to particular use cases, and they present a great learning curve for users unfamiliar with computing environments. Many implementations require one to spend copious amounts of time understanding the tool and adjusting already existing frameworks to ones needs, creating frustration and inefficiency. This problem is experienced by both novice and experienced bioinformaticians alike. Using OpenWDL, a Workflow Description Language, as the basis, I seek to develop an open in use workflow management tool coupled with a GUI interface. As OpenWDL is a widely known WMT, its familiarity will aid in my implementations usability. Additionally, the GUI interface will present a more welcoming environment than that of a command line interface in which many WMT's often employ. To assess the effectiveness of my implementation, I will then assess it to other WMT's, comparing its usability and openness to other bioinformatic pipeline managers such as SnakeMake and NextFlow.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 BACKGROUND

2.1 OpenWDL

Originally created by the Broad Institute for the purposes of genome analysis pipelines, OpenWDL, a Workflow Description Language (WDL), has evolved to much more than its original intent. It provides a "way to specify data processing workflows with a human-readable and -writeable syntax. It makes it straightforward to define analysis tasks, chain them together in workflows, and parallelize their execution."[5] WDL requires an execution engine to run, being compatible with engines such as Cromwell, MiniWDL and dxWDL. It further supports Python, JavaScript, and Java. Due to this and OpenWDL becoming an open source community, it has become the basis for a slew of other WMT's, workflows, as well as an inspiration for new WMT implementations. The various WMT's utilizing OpenWDL, coupled with its familiarity and widespread use within the field, makes OpenWDL a prime candidate for the basis of my own implementation.

Cromwell is a "workflow execution engine that simplifies the orchestration of computing tasks needed for genomics analysis" [6], used to execute WDL scripts. It is now an open-sourced project, but was also originally developed by the Broad Institute. Cromwell can run on a variety of systems, including local and cloud-based services e.g. AWS (Amazon Web Services). This capability is shown through the use of the Terra platform, which can run WDL scripts, Terra includes a built-in Cromwell server that "interprets your WDL workflow script, transforms it into batches of individual analysis instructions, and finally dispatches it to a Google Cloud service called Pipelines API for execution" [3]. One of the most powerful features of Cromwell with WDL is the ability to create sub-workflows. This allows the "execution of an entire workflow as a step in a larger

workflow, when a workflow calls another workflow, that second workflow is called a sub-workflow." [2] These sub-workflows can contain within themselves other sub-workflows, thus, allowing the workflow nesting, which aids in workflow reuse. [7]

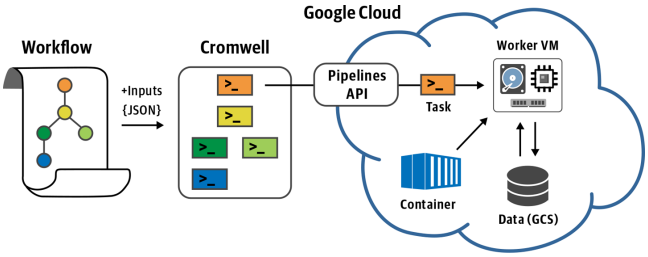

Figure 1: An overview of Cromwell with Terra. (https://3qim1p3jo50i2s787c216eqb-wpengine.netdna-ssl.com/wp-content/uploads/2020/11/cromwell-gcp-overview.png).

MiniWDL, developed by Mike Lin, is a newer WDL execution engine.It adds "developer productivity tools such as a local runner and source code linter, a python library for programmatic access to its WDL parser, static analysis framework, and run time, system."[1]. It reuses Docker's "built-in Swarm mode for many aspects of container scheduling, rather than re-implementing parallel resource allocation and queuing logic. MiniWDL is a sound basis for building "WDL tools and platform-specific runners".[1]

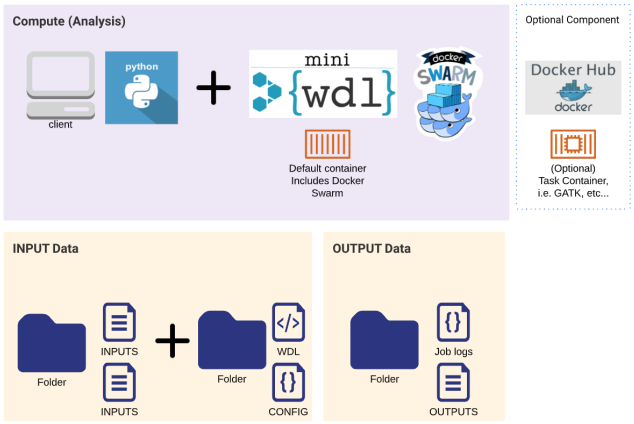

Figure 2: An overview of MiniWDL. (https://raw.githubusercontent.com/openwdl/learn-wdl/master/images/miniwdl-dev.png).

As aforementioned OpenWDL requires an execution engine in order to run, the most well known options are Cromwell and MiniWDL. Each engine, while providing the same service, have different methods of implementation and different features in which holds certain advantages. By juxtaposing both engines I can ascribe one or the other for a user, in their own particular circumstance. For instance, if a user seeks to run their workflow within the field, Cromwell would be a more suitable engine as it allows them to leverage more computational power in the cloud. Or, if a user is running a smaller workflow that does not necessarily need that much computational power, the lightweight MiniWDL maybe a better option. By leveraging the features of each execution engine I can indirectly increase the usability of my WMT implementation.

3 RELATED WORK

3.1 BioShake MIGNON

Many bioinformatic workflow managers use and build upon existing workflow systems to address specific issues with the tool or to create a new workflow management system for a specific application. These workflow managers allow for the advancement of new workflow technologies while receiving the robust and time tested implementation of a preexisting workflow system. Bioshake, is an Embedded Domain Specific Language (EDSL) built in Haskell. It is built on top of Shake, another build tool implemented as an EDSL in Haskell [8]. It inherits Shake's, "reporting features, robust dependency tracking, and resumption capabilities" [8]. However, unlike Shake, BioShake supports forward specification of workflows. One of the most important aspects of Bioshake is its ability to prevent errors before execution, which are caught by their type system. Moreover, its interchangeability to use a different back end, such as Toil or Cromwell, allows for "the leverage of the cloud and containerisation facilities" of them both. Bioshake provides a good example implementation I can analyze in order to understand methods in which WDL can be changed, but also, methods of how to utilize existing technologies to advance the software. While more specific to RNA-Seq experiments, MIGNON, uses WDL as underlying framework [10]. The steps of the workflow are "wrapped into WDL tasks that must executed on an independent unit of containerized software through the use of docker containers. Similar to Bioshake it can also be utilized with cloud based services like cromwell, but also with personal and HPC computers. MIGNON was tested by 6 different human data-sets (total of 42 samples) in which Cromwell and docker coupling produced a fast and easy to deploy workflow. MIGNON, serves as a reference for ways to leverage Cromwell or other execution engines for WDL, that I can use also use to enhance my own implementation.

3.2 aCLImatise a Tool to Aid in WDL Workflow Creation

While there exists a multitude of workflow management technologies there are many others used to advance and simplify existing ones. These tools provide necessary advancements to one seeking to better their workflow software without the need to move on to a completely new product. One such tool is aCLImatise, a tool "designed to streamline the creation of new portable workflows by providing automatically generated tool definitions for any tool with a conventional command-line interface" [14]. aCLImatise initially executes the command "of interest by trying a variety of help flags, storing the standard output from each". The results are then extracted using Parsing Expression Grammar. Finally, the data model is outputted into a WDL workflow format. A tool definition basically describes a piece of software and thus has no effect on the creation of a workflow.

4 DESIGN AND IMPLEMENTATION

The development approach of my software implementation is comprised of two parts/implementations. The first implementation, the GUI interface, deals with the design and creation of my GUI interface as well as a small back end to test it. The Second, the workflow streamlining implementation, deals with applying a method, which can be used to augment and streamline the workflow creation process. As a means to further optimize my software, I will compare the performance and features of both Cromwell and MiniWDL execution engines with my software ensuring compatibility with both. Thus, giving users who are familiar with one engine, but not the other, the ability to use either.

4.1 GUI Implementation

Within the GUI interface I plan on utilizing a Java library such as Swing or JavaFX for creation of the GUI interface. The planned design of the GUI will support the following capabilities:

- The user, upon opening the application, will be greeted with a welcome screen that describes the basic functionality of the GUI.
- There will be an add button that will allow the user to choose what Bioinformatic software they seek to use e.g. BLAST and Quime2.
- A search bar will be in the middle of the screen allowing the user to type the beginning of a command and then options for that command will be listed out on the screen, allowing the user to specify values and certain criteria.
- A place in workflow button, that will allow the user to add commands to there workflow.
- Drop Box for dropping in important files.
- A save button that will allow the user to save the workflow they have created.
- A run button to run the workflow.
- Optional Addition: Output section that lists out the output files. Allows the use to view the contents of each one.
- Optional Addition: A Type Command box, the user can type the cli (command line interface) version of the command if so desired.

When the user runs their workflow, selections made within the GUI will transfer to a WDL file, which will then be run by the workflow execution engine of there choosing.

4.2 Workflow Streamlining Implementation

Within the Workflow Streamlining Implementation I seek to utilize and configure a parser. When a user types a specific command from a particular piece of software, the program will list out and display the options and parameters of that command in a GUI format, using buttons, scrollers, checkboxes, etc, as a means to adjust each one. Thus giving the user discretion to specify and declare all of the supported parameters for that command to their needs. Secondly, if time allows, I seek to implement a software method similar to that implemented in BioShake, that will allow the user to prevent errors before execution or stop execution at particular point in the workflow, giving the user the ability to correct or adjust any mistakes they may have made.

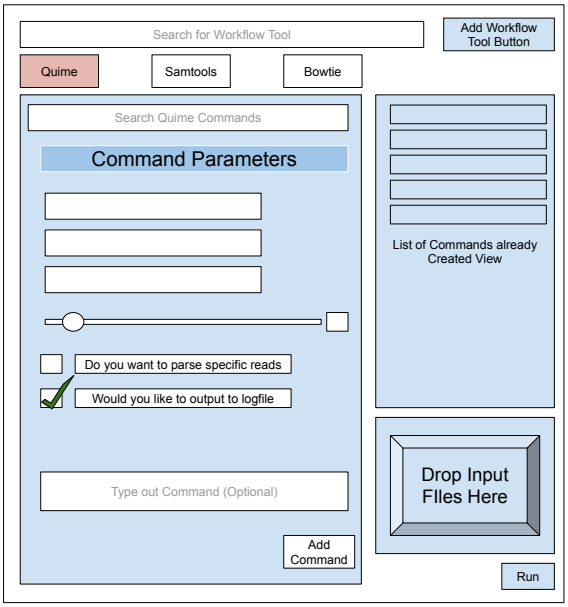

Figure 3: An Example Template of what my GUI may look like.

4.3 Evaluation Plan

To provide a good juxtaposition of my implementation, Nextflow, and Shakemake, I will follow the evaluation procedures of Mainzer and colleagues. Using a variety of scripts they compared CromWell/WDL, Nextflow, and Swift/T using criteria such as [**?**]:

- User interface: the means by which the user interacts with the software.
- Containerization support: methods to virtualize an OS to run on a host without separate virtual machines.
- Check-pointing: ability to save workflow state periodically, allowing for rerun from it upon failure.
- Caching: ability to store frequently used data in memory to reduce data retrieval time.
- Portability: usability of software in a variety of different operating environments.
- Distributed execution engine: makes the computer cluster look like a single machine.Circumvents the use of task scheduler and resource manager.
- Modularity: program implemented as a library of modules, allowing for design flexibility and maintainability.
- Error handling strategy: functionalities to address and resolve errors that arise during program execution
- Parallelization: methods to distribute data among multiple compute nodes, allowing many instances of the same function to run at the same time.
- SPARK support

I will base my evaluation on similar criteria, excluding SPARK support, but adding other criteria such as ease of use. As all of these

criteria are highly considered when new or novice Bioinformaticians chose a WMT, this evaluation method will ensure I compare my implementation to others in a manner that is relevant to what users actually seek. Moreover, the comprehensive nature of this evaluation ensures that I unbiasly and equally judge my own WMT as compared to NextFlow and Shakemake.[**?**].

5 MAJOR RISKS

As I am using new to using WDL, a potential risk is the learning curve required to understand the program and its workings as well as the time it takes to do so, rendering it a hindrance to my overall plan. Additionally, as OpenWDL is created in Java, there is a slight Java learning curve. Although I am familiar with Java, my work within it is limited. I may have to take a brief amount of time refreshing my knowledge, and comprehending its other language features. Finally, there is the challenge of creating the GUI interface. Even though various Java frameworks like JavaFX and Swing make development easier, to make it simple to understand and easy to use, as well as the overall planning of it will take some time to effectively develop.

6 ACKNOWLEDGEMENTS

My appreciation is extended to David Barbella for his assistance in helping me in articulating my proposal and research basis, as well as Charlie Peck's assistance in acquiring sources.

REFERENCES

[1] 2019. *miniwdl, a runtime and developer toolkit for the bioinformatics Workflow Description Language.* https://medium.com/czi-technology/miniwdl-17ecdaf40944
[2] 2020. *Cromwell Subworkflows.* https://cromwell.readthedocs.io/en/stable/SubWorkflows/
[3] 2020. *The freedom of portable workflows.* https://terra.bio/the-freedom-of-portable-workflows/
[4] 2020. *MiniWDL.* https://miniwdl.readthedocs.io/en/latest/
[5] 2020. *OpenWDL.* https://openwdl.org/#three
[6] 2021. *Cromwell on AWS.* https://aws.amazon.com/government-education/cromwell-on-aws/#:~:text=Cromwell%20is%20a%20workflow%20management,%2C%20based%20in%20Cambridge%2C%20MA.&text=Cromwell%20is%20a%20workflow%20execution,tasks%20needed%20for%20genomics%20analysis.
[7] 2021. *Cromwell Workflow Manager and WDL Workflows.* https://sciwiki.fredhutch.org/compdemos/Cromwell/
[8] Justin Bedő. 2019. BioShake: a Haskell EDSL for bioinformatics workflows. *PeerJ* 7 (2019), e7223.
[9] Xiaoling Chen and Jeffrey T Chang. 2017. Planning bioinformatics workflows using an expert system. *Bioinformatics* 33, 8 (2017), 1210–1215.
[10] Martín Garrido-Rodriguez, Daniel Lopez-Lopez, Francisco M Ortuno, María Peña-Chilet, Eduardo Muñoz, Marco A Calzado, and Joaquin Dopazo. 2021. A versatile workflow to integrate RNA-seq genomic and transcriptomic data into mechanistic models of signaling pathways. *PLoS computational biology* 17, 2 (2021), e1008748.
[11] Michael Jackson, Edward Wallace, and Kostas Kavoussanakis. 2020. Using rapid prototyping to choose a bioinformatics workflow management system. *bioRxiv* (2020).
[12] Samuel Lampa, Martin Dahlö, Jonathan Alvarsson, and Ola Spjuth. 2019. SciPipe: A workflow library for agile development of complex and dynamic bioinformatics pipelines. *GigaScience* 8, 5 (2019), giz044.
[13]]ComparitiveMainzer LS Mainzer. [n. d.]. *Comparative Analysis of Genomic Sequencing Workflow Management Systems.* https://swift-t-variant-calling.readthedocs.io/en/latest/_downloads/poster_ISCB_2018.pdf
[14] Michael Milton and Natalie Thorne. 2020. aCLImatise: automated generation of tool definitions for bioinformatics workflows. *Bioinformatics* 36, 22-23 (2020), 5556–5557.
[15] Francesco Strozzi, Roel Janssen, Ricardo Wurmus, Michael R Crusoe, George Githinji, Paolo Di Tommaso, Dominique Belhachemi, Steffen Möller, Geert Smant, Joep de Ligt, et al. 2019. Scalable workflows and reproducible data analysis for genomics. In *Evolutionary Genomics*. Springer, 723–745.
[16] Laura Wratten, Andreas Wilm, and Jonathan Göke. 2021. Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers. *Nature methods* 18, 10 (2021), 1161–1168.

A ONLINE RESOURCES

Git repositories of OpenWDL, Cromwell, MiniWDL and various open source workflow management tool implementations. Introductory series of getting started with OpenWDL.

[12] [10] [9] [11] [4] [15] [5]

Table 1: Timeline

Weeks	Tasks
Week 1 -> January 31 - February 6th	Learn the basics of OpenWDL and its architecture. Examine other WMTs that utilize OpenWDL as a reference.
Week 2 -> February 7th - 13th	Begin working on first software implementation. Assess and choose GUI libraries for Java: SwingX, JavaFX, etc. Develop template of what the GUI should look like.
Week 3 -> February 14th - 20th	Template of GUI should be finalized. Start work on both front and back end features. Develop and submit first draft of research paper.
Week 4 -> February 21st - 27th	Produce and submit first workable software implementation. Basic structure should be established in which the GUI is working with a simple back-end despite bugs being present.
Week 5 -> February 28th - March 6th	Correct major bugs that break the program. Add additional features as needed for the front or back-end.
Week 6 -> March 7th - March 13th	Limited errors in the Software should be present. Perform tests on entire program to ensure it is working correctly.
Week 7 -> March 14th - March 20th	First part of Software implementation (GUI interface) should be complete with minimal to no errors. Begin work on advancing workflow creation capabilities of software (i.e second part of software implementation). Use sources for reference.
Week 8 -> March 21st - March 27th	Evaluate and choose methods and techniques for streamlining Workflow creation Implement in Code Working on second part of software implementation all parts of first implementation must be complete.
Week 9 -> March 28th - April 3rd	Continue working on second implementation. Write second Draft of paper.
Week 10 -> April 4th - April 10th	Basic infrastructure of second part of software implementation should be established or near completion.
Week 11 -> April 11th - April 17th	At this point software should be relatively fully functional although bugs are present.Submit second version of software.
Week 12 -> April 18th - April 24th	Second part of software should be completed with a few bugs remaining. Begin comparison testing between Shakemake and Nextflow
Week 13 -> April 25th - May 1st	Second implementation should have minimal to no errors. Testing with Mini-WDL and Cromwell should be complete
Week 14 -> May 2nd - May 8th	Finalize all loose edges and perform overall cleanup of both the first and second software implementations. By the end of the week all software work should be completed. First draft of poster should be submitted and final draft of paper should be in the works.
Week 15 -> May 9th - May 15th	Poster, final draft, and software should be completed and submitted. Practice for poster presentation and present.