Improving Build and Deployment Process of the SBML Solver

Taeyonn Reynolds

Abstract

This paper presents Systems Biology Markup Language (SBML) solver which is an extensible, high-performance, cross-platform, open-source software library. SBML is a shared library that loads, compiles and executes models written in the SBML language. Building the SBML solver involves tracking down a lot of dependencies and they are important for users who want to compile it to have clear instructions on how to do so. The SBML Solver will also be connected to integrate a PhysiCell software package. It fits into a suite of physics (fluids, soft-matter, visco-elastic) and computational biology called 'Mechanica'. This module is designed to calculate the time evolution of a chemical network that's attached to some simulated physical object (such as cell, membrane, volume), so forth.

Introduction

Software deployment includes all the process required for preparing a software application to run and operate in a specific environment[8]. It involves installation, configuration, testing and making changes to optimize the performance of the software[8]. The software deployment phase provides many advantages for the users. It reduces the time needed to develop the software and focuses on the work needed to be completed. Software deployment makes it easy to monitor user actions effectively and gain insight into user activities around the software[8]. Through software deployment accurate updates, software, maintenance tasks and uninstall can be targeted automatically[8].

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later[1]. Throughout history developers faced an issue as working on projects with other developers remotely. This is important as it speeds up the development process greatly by allowing collaboration simultaneously.
There were two different version controls created which are distributed and centralized.

Centralized version control systems was the first version control created to fix this major issue by having a single server that contains all the versioned files, and a number of clients that

check out files from that central server[1]. A disadvantage of this version control system is the possibility of the server being down or corrupted. If the server is down the users will not be able to work or commit changes for the duration. Also, all work will be lost if the central database is corrupted and the proper backups were not updated regularly.

Distributed version control systems are different due to them allowing every user to clone a copy of the repository and allowing the project to be moved from multiple systems with ease. Once the copy is created the user is allowed to push and pull the project to collaborate. The pulling aspect is when the user downloads the work to the server and the push is when the changes are sent back to the server for collaboration. It is effective as it allows the user to make changes without affecting others work and keeps the changes private until the user pushes them. An example of a distributed version control system is Git.

Methodology

The project was initiated to both document and improve the software build process for the SBML solver. It was important to document the software development process with user and developer roles to make the instructions easy to complete. This consisted of repeating multiple steps and forking, committing changes and issuing pull request via GitHub.

GitHub is a code hosting platform for version control and collaboration founded by Tom Preston-Werner, Chirs Wanstrath, Scott Chacon and P. L. Hyett[2]. GItHub allows multiple users to utilize the master code to make their own copy which is called forking. By forking, the user will pull the code to their platform to make and check updates that will allow them to complete the task. Once the code has been successfully changed it can be updated to GitHub by providing a pull request that allows others to see your changes. If the changes are accepted the master code will be changed allowing everyone to see the correct code remotely.

Due to the versatility of the SBML solver on cross platforms it was necessary to ensure the builds and fix issues as they are encountered. Linux and Windows were the two operating systems used to create the SBML solver. Also, building the SBML solver included a better documentation process and determining the missing features that were not in the build instructions. While documenting the build system I noticed there were a few missing third-party libraries that were needed to run the SBML solver. For example, I tracked and updated the build instructions to add and download libxml2, zlib12 and libncurses. While building the SBML solver on Windows the dependency library needed to be updated to enable a 64-bit build.

Libxml2 is a free software that is a version of XML which is a metalanguage to design markup languages, i.e. text language where semantic and structure are added to the content using extra "markup" information enclosed between angle brackets[7]. The library is written in C but there are a variety of language bindings to make it available in multiple environments[7]. Libxml2 is effective due to is flexibility to run on most operating systems such as Linux, Windows and MacOS X.

Zlib12 was created by Jean-loup Gailly and Mark Adler to be a free open source lossless data-compression library to be used on virtually any computer hardware and operating system[4]. It's compression method, an LZ77 variant called deflation, emits compressed data as a sequence of blocks[5]. Various block types are allowed, one of which is stored blocks—these are simply composed of the raw input data plus a few header bytes[5].

Libncurses is a freely distributable library, fully compatible with older versions of curses[3]. It forms a wrapper over working with raw terminal codes, and provides highly flexible and efficient API (Application Programming Interface)[3]. It provides functions to move the cursor, create windows, produce colors, play with mouse etc[3]. The application programs need not worry about the underlying terminal capabilities [3].

Figure 1 : The following shows the necessary steps for the addition of a documented third-party library needed to run the SBML solver.

Results

The software distribution and uptake of the SBML solver resulted in a streamlined document build process. SBML Solver is a form of Ordinary Differential Equations(ODEs) that is used for communicating and storing computational models of biological processes[6]. Its capacity to deal with complex models and quick model execution speed have prompted its fast reception in an assortment of uses. SBML solver is quick enough to help huge scale issues, for example, tissue models, examines that require enormous quantities of rehashed runs and intelligent reenactments.

Conclusion

SBML solver is an independent library, ready to run both as a part inside different instruments by means of its C++ and C ties, and intuitively through its Python interface. The documentation of the software build is a key component to making the solver user friendly as the SBML solver had to successfully run on Linux, Windows and Mac OS X. The SBML solver speed and efficiency will enable analysts to illuminate enormous models, incorporate models implanted in multi-scale frameworks and run huge outfits of smaller models.

Acknowledgements

I would like to thank the Research Experience for Undergraduates (REU) at Indiana University for providing me the opportunity to make this research possible. I would also like to thank Andy Somogyi and Gregor von Laszewski for their assistance.

References

[1] “1.1 Getting Started - About Version Control.” *Git*, git-scm.com/book/en/v2/Getting-Started-About-Version-Control.

[2] “GitHub Guides.” *GitHub* , 7 Apr. 2016, guides.github.com/activities/hello-world/.

[3] Padala, Pradeep. “NCURSES Programming.” *The Linux Documentation Project* , 2001, www.tldp.org/HOWTO/NCURSES-Programming-HOWTO/intro.html.

[4] Roelofs , Greg, and Mark Adler. “Zlib Home Site.” *Zlib Home Site*, 15 Jan. 2017, zlib.net/.

[5] Roelofs, Greg, and Mark Adler. “Zlib Technical Details.” *Zlib Technical Details*, 1 May 2006, zlib.net/zlib_tech.html.

[6] Swat, Maciej, and Julio Belmonte. “SBML Solver.” *Python Scripting Manual*, 2017, pythonscriptingmanual.readthedocs.io/en/latest/sbml_solver.html.

[7] Veillard, Daniel. “The XML C Parser and Toolkit of Gnome.” *The XML C Parser and Toolkit of Gnome*, www.xmlsoft.org/

[8] Williams, Ellisse. “What Is Deployment in Software.” *PDFelement*, 30 June 2019, pdf.wondershare.com/business/what-is-software-deployment.html.