Extended Abstract: GenApp: Extensible Tool for Rapid Generation of Web and Native GUI Applications

Alexey Savelyev and Emre Brookes,*
Department of Biochemistry and Structural Biology,
The University of Texas Health Science Center
at San Antonio, 7703 Floyd Curl Drive, San Antonio, TX 78229-3900
**Corresponding author:* email: emre@biochem.uthscsa.edu

Abstract: *GenApp is a universal and extensible tool for rapid deployment of applications. GenApp builds fully functioning science gateways and standalone GUI applications from collections of definition files and libraries of code fragments. Among the main features are the minimal technical expertise requirement for the end user and an open-end design ensuring sustainability of generated applications. Because of the conceptual simplicity of use, GenApp is ideally suited to scientists who are not professional developers, to disseminate their theoretical and experimental expertise as embodied in their code to their communities by rapidly deploying advanced applications.*

1. Introduction

GenApp (Generalized Application Framework) [1,2,3,4] is designed to simplify creation and deployment of local GUI and web based applications over a collection of modules. GenApp was originally conceived during NSF and UK's EPSRC funded CCP-SAS project [5] to provide a web-based GUI for advanced analyses of structural data in chemical biology and soft condensed matter [6,7,8]. Over time, GenApp's capabilities were significantly enhanced and broadened driven by other multiple science cases, transforming it into an established framework. Currently, GenApp supports a growing number of web portals primarily, but not exclusively, in the small-angle scattering (SAS) community.

Unlike conventional approaches to web development which often require a large team of computer scientists working on a project, GenApp greatly simplifies the work of users and system administrators by utilizing libraries of fragments and user defined modules as directed by definition files (see below). GenApp's simplicity, as seen from the user perspective, is manifested by the fact that these definition files are the only input (apart from the scientific code intended to be deployed) necessary for building final application. As elaborated below, GenApp effectively divorces the user from having to develop software environment specific user interfaces and provides an extensible collection of application environment targets.

2. Significance

Quite often scientific codes developed in a typical research laboratory become unsustainable ("dark" codes) beyond the lifetime of funding or shortly after staff rotation. Hiring expensive CS expertise diverts scarce resources from the lab's primary goals and often translates the problem without resolving it. The diversity and continually changing nature of software environments compounds the issue. GenApp can readily address the issue to insure preservation[1] of scientific codes and to enhance the level of their usability and recognition by producing fully functional science gateways and native GUI applications in multiple target languages. The scientific community would greatly benefit and be better served if the tools and infrastructure provided by GenApp were available to enable and maintain the work of talented developers worldwide.

Another important aspect of GenApp broadly impacting the scientific community is the potential of its use as a versatile and powerful educational tool. In particular, not only GenApp can carry a "research" component (scientific code), but also a rich educational and demonstration content accompanying main application, including supporting manuals, images, videos, use examples, and the like. An XSEDE/ECSS project utilizing GenApp is currently using these capabilities [9]. Thus, GenApp directly addresses points mandated

[1]Forcing the module developer to uniformly define inputs and outputs and modify their code to accept inputs and outputs as defined, by itself, helps insure preservation since any competent programmer can utilize these definitions minimal effort. Additionally, within the GenApp framework itself, future target languages can be developed which will automatically wrap the code into an application.

Presented at Gateways 2016, San Diego Supercomputer Center, La Jolla, CA, November 2-3, 2016.
https://gateways2016.figshare.com/

by funding agencies such as improved education, increased public scientific literacy and public engagement with science and technology and creating enhanced infrastructure for research and education.

3. GenApp Structure, Technical Details

3.1 Roles

GenApp defines four primary roles. These are, in order of descending C.S. expertise, the framework developer, the target language developer, the application developer and the module developer. These roles parallel the organization of GenApp as shown in Fig. 1. The framework developer develops and maintains the generator tool. The target language developer is responsible the contents of one or more target languages and implements and maintains them by building up code fragments and defining their assembly. The application developer organizes modules in a menu definition file and runs the GenApp generator to create working applications. Finally, the module developer wraps executable modules by writing a module definition file and ensuring the wrapped executable accepts defined input and produces defined output (see Fig. 2).

This structure segregates the application and module developers, which require minimal CS expertise from the framework and target language developers, which require advanced CS expertise. This enables a researcher, acting as an application

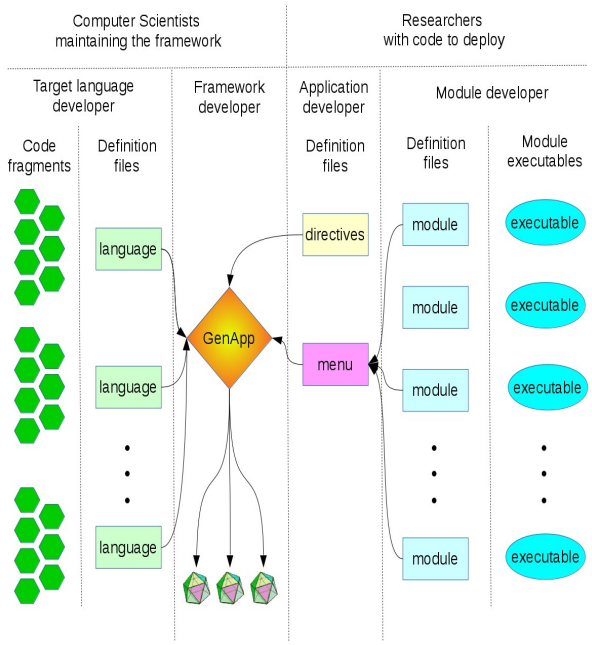

Fig. 1 An overview of the GenApp framework. Four roles are defined at the top as areas of responsibility (see the text). Separation of C.S. and researcher expertise allows researchers to rapidly deploy applications in a variety of target languages and take advantage of new developments within the framework without affecting their underlying executable.

and module developer with an executable that they wish to expose, to rapidly deploy advanced user interfaces. If a new target language, variant or feature is developed by a target language developer, it becomes available to all application

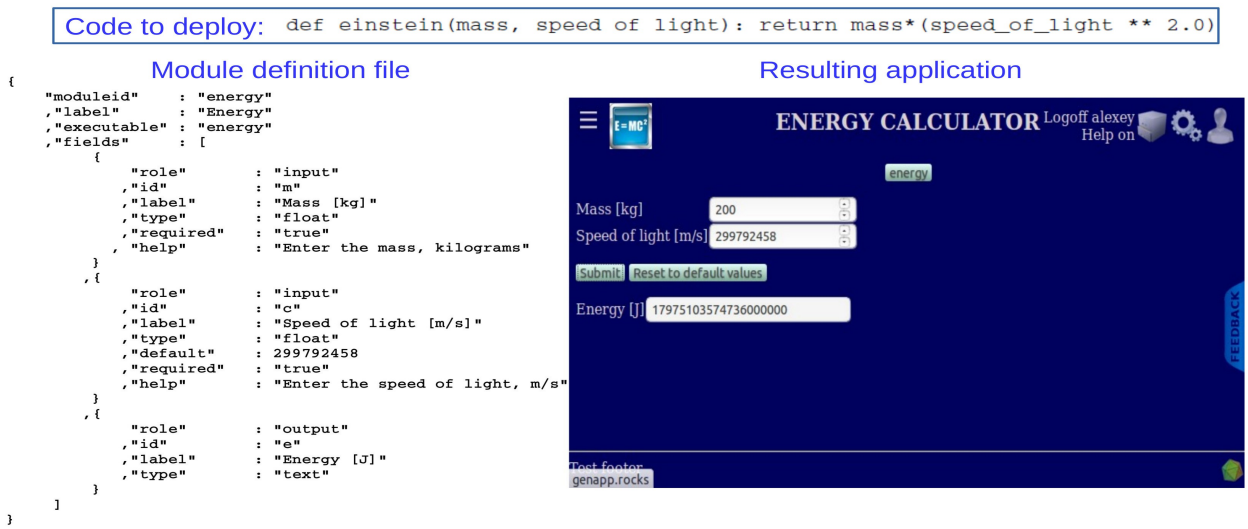

Fig. 2 An example python code intended to be deployed, GenApp module definition file, and the corresponding application output. Definition file is a JSON text file describing input and output.

and module developers and they can deploy the new or modified target.

3.2 Modules

A module is some defined executable within GenApp. The module definition file (Figs. 1 and 2) contains all information about the module. This, of course, includes all input and output fields. Each field is uniquely defined with an *id*. In addition, a primary attribute for each field is the *type*, e.g. “integer”, “text”, “plot”, “atomicstructure”, etc.

3.3 Repeaters

Early on, module developer requirements presented the necessity to define input fields which appear to the user based upon the values of other input fields. The GenApp answer to this was the *repeater* structure. For example, a “checkbox” field could have the attribute of *repeater*, subsequently, other fields could reference a *repeat* on the *repeater*'s *id*. This enables a field to be displayed dynamically to the user based upon the setting of the “checkbox”. Current *repeater types* include “checkbox”, “integer” and “listbox”. The “checkbox” *repeater* has been described and a reverse logic “checkbox” *repeater* is also available. The “integer” *repeater* will create some number of instances of the dependent *repeat* fields. The “listbox” *repeater* displays dependent fields based upon the “listbox” choice. Repeaters can be arbitrarily nested.

3.4 Summary of Main Features

Currently developed GenApp's *target languages* include web [HTML5+Javascript+Php] and native GUI [Java, Qt4, Qt5] fragment collections. GenApp generated science gateways include the following features: user management and statistics; job managements with multiple simultaneous reattachment to job running or run; a “cloud” file system; caching of results; integrated feedback mechanism which links full failed job information for easy debugging. Additional features are added as needed by use cases. Websites generated are typically hosted on a VM, be it on a developer's laptop, dedicated host, or cloud resources such as NSF/Jetstream or AWS. The extensible variety of current execution models include running on local or managed compute resources such as those available from NSF/XSEDE. Recently, GenApp integration with Apache Airavata for the application execution has been prototyped [2]. Apache Airavata [10] is a software framework that enables one to compose, manage, execute, and monitor large scale applications and workflows on distributed computing resources such as local clusters, supercomputers, computational grids and clouds.

4. Applications

The prototype of GenApp was first applied to create SASSIE-web [6] by the CCP-SAS consortium [5]. This has had a significant positive impact on SAS user community with over 300 registered users and 3900 jobs submitted in the first six months of the current year. Over 40 known publications have been produced.

During past several months, we have extensively used the GenApp framework to build a number of web portals for other SAS related applications, including *SoMo* [7], *Willitfit* [11], *Quafit* [12], *Genfit* [13], *MULCh* [14], *Denfert* [15] and *BayesApp* [16]. Additionally, we have recently prototyped a web portal, *ParamMD* [17], for parameterizing and simulating physical systems by means of Molecular Dynamics (MD) using different computational models (e.g. CHARMM, GROMOS atomistic force fields) and highly scalable simulation engines (NAMD, LAMMPS). A GenApp generated gateway, *NAMDrunner* [18], which allows execution of multi-scale NAMD based MD simulations, is currently in alpha testing.

The above mentioned and newly developed portals will be continually updated with new features, including new models, analysis tools, advanced methods for parametrization and setup of models of different resolution (all-atom vs. coarse-grained) and others. These requirements demand corresponding advancements to GenApp, which can be exemplified by generating more dynamic UI content, introducing complex functional dependencies among various input fields, enabling sophisticated workflows and execution staging.

5. Acknowledgments

This work is supported by the NSF grant CHE-1265817 to E. Brookes. We are grateful to application developers and their users for their valuable feedback and suggestions.

6. References

[1] Brookes, E.H. 2014. *An Open Extensible Multi-Target Application Generation Tool for Simple Rapid Deployment of Multi-Scale Scientific Codes.* XSEDE '14. ACM, doi: 10.1145/2616498.2616560

[2] Brookes, E.H., Anjum, N., Curtis, J.E., Marru, S., Singh, R., and Pierce, M. (2015), *The GenApp framework integrated with Airavata for managed compute resource submissions.* Concurrency Computat.: Pract. Exper., 27(16):4292-4303, doi: 10.1002/cpe.3519.

[3] GenApp. http://genapp.rocks

[4] Brookes, E.H., Kapoor, A., Patra, P., Marru, S., Singh, R., Pierce, M. (2015) *GSoC 2015 student contributions to GenApp and Airavata,* Concurrency Computat.: Pract. Exper., 28(7):1960-1970, doi:10.1002/cpe.3689

[5] Perkins, S., Butler, P., *CCP-SAS – Collaborative Computational Project for advanced analyzes of structural data in chemical biology and soft condensed matter.* http://ccpsas.org

[6] Curtis, J. E, Raghunandan, S., Nanda, H., and S. Krueger. (2012) *SASSIE: A program to study intrinsically disordered biological molecules and macromolecular ensembles using experimental restraints.* Comp. Phys. Comm.183:382-389. http://www.smallangles.net/sassie

[7] Brookes, E., Demeler., B, and Rocco, M. (2010). *The implementation of SOMO (SOlution MOdeller) in the UltraScan analytical ultracentrifugation data analysis suite: enhanced capabilities allow the reliable hydrodynamic modeling of virtually any kind of biomacromolecule*. Eur. Biophys. J, 2010 doi:10.1007/s00249-009-0418-0, http://somo.uthscsa.edu

[8] Wright, D. and Perkins, S., SCT software, http://dww100.github.io/sct/

[9] http://gw165.iu.xsede.org/vortexshedding

[10] Marru S., Gunathilake L., et al. 2011. Apache airavata: a framework for distributed applications and computational workflows. Proc. Workshop Gateway computing environments. ACM

[11] Pedersen M. C., Arleth L. and Mortensen K., "WillItFit: a framework for fitting of constrained models to small-angle scattering data" *J. Appl. Cryst.* 46, 1894–1898, 2013

[12] Spinozzi F. and Beltramini M., "QUAFIT: A Novel Method for the Quaternary Structure Determination from Small-Angle Scattering Data", *Biophys. J.* 103, 511-521, 2012

[13] Spinozzi F. et al, "*GENFIT*: software for the analysis of small-angle X-ray and neutron scattering data of macromolecules in solution", *J. Appl. Cryst.* 47, 1132–1139, 2014

[14] Whitten A. E., Cai S., and Trewhella J. "MULCh: ModULes for the Analysis of Small-angle Neutron Contrast Variation Data from Biomolecular Complexes." *J. Appl. Cryst.* 41, 222-226, 2008

[15] Koutsioubas, A. & Pérez, J. (2013). "Incorporation of a hydration layer in the `dummy atom' ab initio structural modelling of biological macromolecules" *J. Appl. Cryst.* 46, 1884-1888

[16] Hansen, S. (2012). "BayesApp: a web site for indirect transformation of small-angle scattering data". *J. Appl. Cryst.* 45, 566–567.

[17] https://somo.chem.utk.edu/parammd/ http://js-170-47.jetstream-cloud.org/namdrunner/