Simplifying Complex Software Assembly

The Component Retrieval Language and Implementation

Eric L. Seidel[1], Gabrielle Allen[2,3], Steven Brandt[2], Frank Löffler[2], and Erik Schnetter[2,4]

[1]Department of Computer Science, City College of New York
[2]Center for Computation & Technology, Louisiana State University
[3]Department of Computer Science, Louisiana State University
[4]Department of Physics & Astronomy, Louisiana State University

Abstract

Assembling simulation software along with the associated tools and utilities is a challenging endeavor, particularly when the components are distributed across multiple source code versioning systems. It is problematic for researchers compiling and running the software across many different supercomputers, as well as for novices in a field who are often presented with a bewildering list of software to collect and install.

In this work, we introduce a language (CRL) designed to address these issues, as well as our implementation called `GetComponents`, which is now used by the Cactus Framework.

Introduction and Motivation

Distributed Software Frameworks are becoming increasingly difficult to assemble and maintain:

- They can be spread across multiple repositories and various version control systems.
- They can require a large number of individual commands to retrieve completely.
- Many users want to run their applications on a variety of compute resources.
- Users may also want to include other tools to assist in building and running the application.

The Cactus Framework [1] used a tool called `GetCactus` to facilitate this process. `GetCactus` was created in 2000 and designed only for use with a few specific *Version Control Systems* (VCS), and after 10 years was no longer sufficient. We designed and developed a new application, `GetComponents`, to address these issues and to create a general-purpose tool.

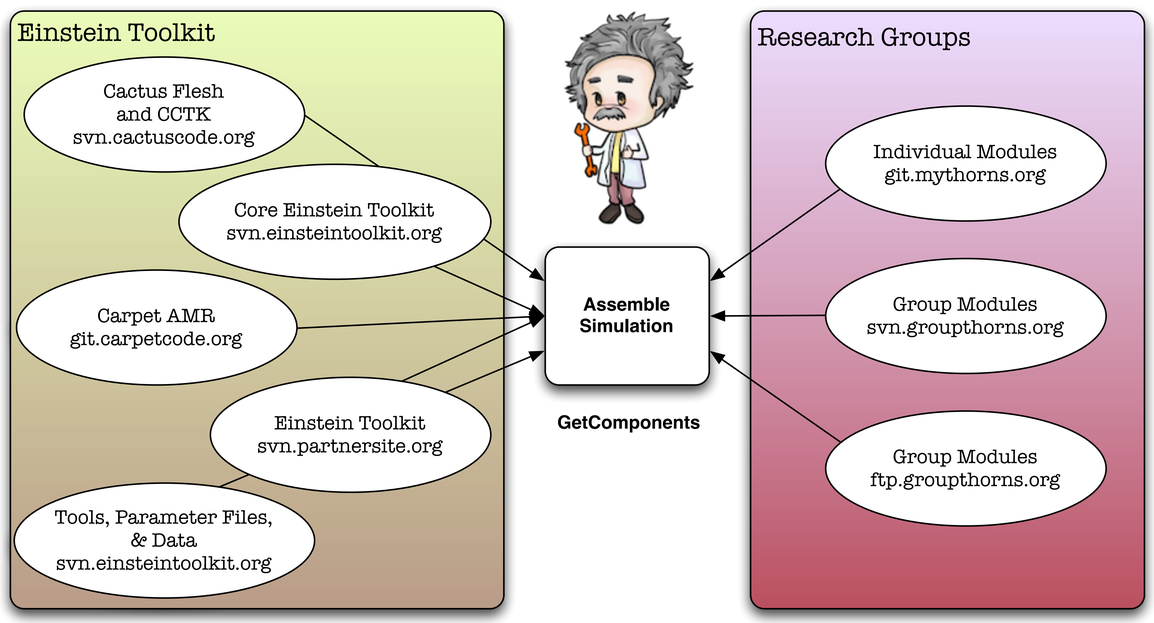

Figure: Applications such as the Einstein Toolkit [2], built from component frameworks such as Cactus, can involve assembling hundreds of modules from distributed, heterogeneous source code repositories. The Einstein Toolkit is comprised of 135 components, most of which reside in their own repository. `GetComponents` was released as the primary means of distribution for the Cactus Framework and the Einstein Toolkit.

Design Issues

The `Component Retrieval Language` (CRL) [3] was designed to alleviate issues that had arisen with `GetCactus`. CRL and `GetComponents` focus on the following design issues:

- Easy distribution of component lists.
- Support for both anonymous and authenticated retrieval of components.
- Support for different repository and distribution types.
- Support for updating components.
- Support for multiple component lists specified together.

Grammar

```
# NAME is an alphanumeric or '.' character

DOCUMENT : DIRECTIVES ;

DIRECTIVE : DEFINE NAME '=' PATH EOL
          | CHECKOUT '=' COMPONENTLIST EOL
          | CHECKOUT '=' EOL COMPONENTLIST EOL
          | REPO_LOC '=' LOC EOL
          | AUTH_LOC '=' LOC EOL
          | PATH_DIRECTIVE '=' PATH EOL
             # !REPO_PATH, !CHECKOUT, !TARGET,
             # !ANON_PASS, !NAME
          | NAME_DIRECTIVE '=' NAME EOL
             # !CRL_VERSION, !AUTH_USER,
             # !ANON_USER, !TYPE
          ;

DIRECTIVES : DIRECTIVE
           | DIRECTIVES DIRECTIVE
           ;

LOC : PSERVER PATH            # CVS repository
    | NAME ':' '/' '/' PATH   # Git/SVN repository
    | NAME '@' NAME ':' PATH  # Git repository
    ;

PATH : NAME
     | '/' NAME
     | PATH '/' NAME
     ;

COMPONENTLIST : PATH
              | COMPONENTLIST EOL PATH ;
```

Figure: `CRL` has a full grammar written in Bison, a variant of Backus-Naur Form (BNF).

Authentication

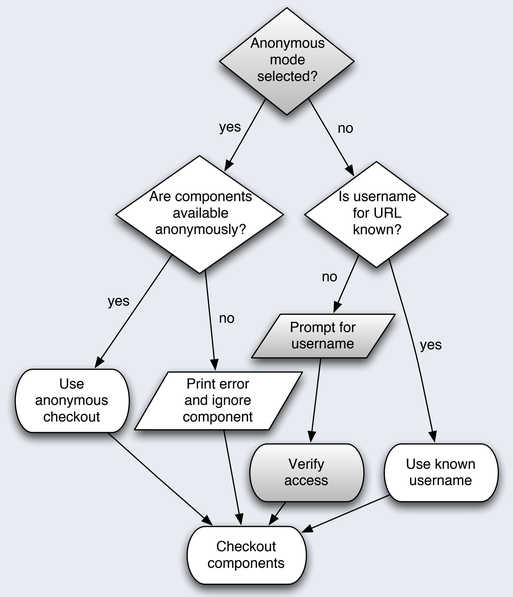

Figure: `GetComponents` features a robust authentication process that emphasizes security. It uses the built-in authorization methods for each VCS, never seeing the user's password itself.

References

[1] Cactus Computational Toolkit, URL `http://www.cactuscode.org`.

[2] The Einstein Toolkit, URL `http://www.einsteintoolkit.org`.

[3] E. L. Seidel, G. Allen, S. Brandt, F. Löffler, and E. Schnetter, *Simplifying complex software assembly: The component retrieval language and implementation*, TeraGrid '10 (New York, NY, USA), ACM, to be published.

Acknowledgements

This work was supported by NSF #0904015, NSF #0725070, NSF #0721915, NSF #1005165, and the Blue Waters Undergraduate Petascale Internship. We used TeraGrid resources under allocation TG-MCA02N014.