Software Citation Tools
Project Plan

Latest Revision: 2016-10-20

Team ForCite:
Eric Lee
Rob Lowe
Sam Mosher
Colin O'Neill

Overview

Software Citation Tools will assist users who need to cite software. The primary goal is to provide the scientific community with a tool that streamlines the process of citing software while following the standards set by the Force11 paper *Software Citation Principles*. The streamlined process should lower the barrier to citation and help to ensure that the people who write the software get the credit that they deserve.

The Software Citation Tools will be built on a core npm package, which will be written in JavaScript. This package will take a repository URL and return a formatted citation. Our team will then develop a suite of applications that can be used by researchers to more easily take advantage of the package. Potential examples include a web application that will act as a more user-friendly interface to the package, and a browser plugin that will generate a citation when the user is on a repository site. The exact applications to be developed will be determined by user feedback, and described in the next revision of this document.

The target users are researchers and academics who use or create software for their jobs and studies. The aim is to make it easier for these users to cite software so that the original creator gets credit for the software that they developed.

One of our secondary goals for this project is fostering community involvement. This tool has the potential to be highly beneficial to the scientific community. To provide the highest possible benefit, we will leverage the community for ideas, requirements, and feedback. In addition, the project will need to be maintained after our Senior Project has been completed. By building and engaging a community of invested contributors who want to see the project succeed, we can insure the longevity of the project beyond the duration of our Senior Project.

Project Scope

The scope of Software Citation Tools will include a core package that will allow the user to enter the address of a GitHub repository and return a citation string to the user. From there, that package will be integrated into different web applications or browser plugins so that users can either navigate to the website and enter the address of the repository to get the citation, or just navigate to the repository and get the citation from the plugin window in the browser. Currently we only plan to support git/GitHub repositories. Another goal is building the community around this application so that the application does not die once our senior project ends.

There are, however, some things that are out of the current scope for the project. The main features that are considered out of scope will be accessing repositories other than those hosted on GitHub, as well as additional format options, and creating additional webapps in addition to the ones mentioned above.

Deliverables

Due to the open nature of development of Software Citation Tools, the project will not have traditional "releases". The heavy community focus means that even after the core team finishes with an area of development, community contributors can continue to tweak and develop in those areas. It also means that the product will always be available in its most up to date form in a rolling release instead of discrete releases. Instead the “Releases” will be points when the core development team will shift its focus. These releases will be described by a set of features that the core team feels is enough make a particular facet of the product sufficiently useful to the end user.

Planned Releases

The planned releases of Software Citation Tools will be broken up into three components, the Core Package, Application 1 and Application 2.

Core Package

The Core Package of Software Citation Tools is an npm library library that provides core citation functionality in a modular fashion to allow for other applications and tools to be built on top of it.

Core Package Stage 1

The first stage of Core Package development is focusing on getting enough features developed in the Core Package that it can be tested with target end users to facilitate feedback to direct development of the second stage of development. In particular this release will focus on processing a git repository and generating a simple citation object or string following the Force11 Software Citation Principles as an output.

Core Package Stage 2

The second stage of Core Package development focuses on eliciting and reacting to end user feedback gathered from the Stage 1 of the Core Package. The development team will focus on bringing only the most important features to the core package and will agree on a set of features that constitutes Stage 2 to address potential scope creep.

Application 1

Application 1 is a tool developed on top of the Core Package which provides a more user friendly interface to the tools for a specific application. This will be an application based on the core package that addresses the most critical functionality of the core package. It will also serve as an example application using the core package for potential contributors.

Application 2

Application 2 is a second tool developed on top of the Core Package which provides new functionality to the tool suite. This will address the most demanded functionality of the tool suite. During development the team will also focus on getting the project ready for hand off to the community of contributors after the conclusion of the core team's academic obligations.

Risks and Mitigations

See Appendix A-Risks for detailed descriptions and mitigation plans.

ID	Name	Description	Risk Exposure
Risk 01	Severe Scope Creep	Addition of extra features and changes causing considerable delays in completing development stages.	11.2 Work Days
Risk 02	Compounded Delays	Delays to project due to late start compounding over time and causing project to fall even further behind	8.4 Work Days
Risk 03	Inadequate Community Involvement	The functionality of the tool suite relies on eliciting feedback from the community. The tool suite will be an evolving resource and failing to cultivate contributors will damage its ability to continue to be a worthwhile resource.	11.2 Work Days

Technical Process

The project will use an iterative and incremental development methodology with evolutionary prototyping with the goal of producing a stable deliverable at the end of each iteration. Each iteration will consist of four phases: requirements gathering, analysis and design, development, and deployment.

The project will be maintained via GitHub to make use of GitHub's tools for tracking work, in particular GitHub issues. GitHub issues will be created during requirements gathering, formalized during analysis and design, and utilized to track work status during development.

For more detailed information on the project's technical process, see the Process and Methodology Document.

Community Engagement Plan

A core component of the Software Citation Tools project will be its development according to open source best practices. In order to work openly with feedback and contributions from the research community, and to ensure a sustainable group of contributors, we plan to solicit and engage a community of researchers, open science advocates, and code contributors. This will be done by keeping development open on GitHub, directly engaging the community via community chat platforms, and keeping the community informed of the project team's status via the project website.

Development of the Software Citation Tools will be kept open to the public on GitHub. All source code, documentation, and project history will be available. Tasks and their statuses will be publically available via GitHub issues. A CONTRIBUTING document will be available to describe how interested parties can get involved with the project's development.

The community will be able to communicate at any time via a community chat. This chat will be hosted on Gitter, and will be mirrored to an IRC chat room for maximum accessibility. The project team will be available via this chat.

Finally, the status of the development team will be made publically available via the team website. Important updates regarding team and project status will be made to a News page on the website. Documentation, such as this project plan, weekly updated "four-up" status reports, and reported metrics will also be available to any interested party.

Project Schedule

Metrics

Our first metric will be to observe the team’s story points per week or velocity. Story points indicate the complexity of a task. Each task shall be assigned a story point before it can be worked on. After a few weeks story points can be used to determine the number of tasks a team can complete within a certain time period.

Our second metric will be tracking the number of issues closed per week. We’d like to examine the difference between story points completed per week vs number of issues closed per week. We hypothesise that there will be a strong correlation between the two to the degree where analysing the number of tasks will provide enough granularity to create estimations.

In order to track community engagement, two community engagement measures will be tracked: chat activity and GitHub activity. The chat activity will reflect the number of messages posted to the Gitter and IRC chats, and will serve as an estimate for the relative amount of conversation regarding the project each week. The GitHub activity will be the total number of stars, forks, issues, comments, etc. created on the project's GitHub each week, and will serve as an estimate of the project's community activity.

Appendix A - Risk Descriptions

Risk 01 - Severe Scope Creep

Description

Addition of extra features and changes causing considerable delays in completing development stages.

Probability: 0.8

Impact: 2 Work Weeks

Risk Exposure: 14 Days * 0.8 = 11.2 Work Days

Mitigation Tactics

- Scope will be defined by the team before a development stage can begin.
    - Changes to this scope will be evaluated for schedule impact and require sign off from the entire development team before they can be accepted

Risk 02 - Compounded Delays

Description

Delays to our project due to our late start compounding over time. Being off schedule means we don’t know what needs to be done which will allow us to fall even further behind

Probability: 0.6

Impact: 2 Work Weeks

Risk Exposure: 14 Days * 0.6 = 8.4 Work Days

Mitigation Tactics

- Attempting to complete more than what is needed each week(catching back up)
- Recreating Sr Project Schedule so we know when things are “late” again

Risk 03 - Inadequate Community Involvement

Description

Community involvement is critical to the success of the project. The functionality of the tool suite entirely relies on eliciting feedback from the community that we are targeting the tool suite for. Similarly, the tool suite will be an evolving resource and failing to cultivate contributors from the project’s inception will damage its ability to continue to be a worthwhile resource for academics.

Probability: 0.4

Impact: 4 Work Weeks

Risk Exposure: 0.4 * 28 Work Days = 11.2 Work Days

Mitigation Tactics

- Integrating community involvement in our process and plan to increase a feeling of involvement
- Encouraging one time contributors to continue collaboration
    - E.g. Encouraging a person who answered our questions to join a mailing list to hear about the status of our project/help in other ways
- Maintaining an open development methodology so potential contributors can learn about our project and join in easily
    - Maintaining a public chatroom about our project so potential contributors can ask one of the core team about it.