Extending the Atom Editor

Milestone Report 3

CS 5150, Spring '18

Client

Kyle Harms <kyle.harms@cornell.edu>

Development Team

Saqif Badruddin <ssb229@cornell.edu>
Joseph Chuang <jcc436@cornell.edu>
Weiyou Dai <wd248@cornell.edu>
Jianhua Fan <jf773@cornell.edu>
Daniel Jordan Hirsch <djh329@cornell.edu>
Athena Danlu Huang <dh527@cornell.edu>
Tyler Yeung Ishikawa <tyi3@cornell.edu>
Jacob Ethan Rauch <jer322@cornell.edu>

Student Contact

Daniel Jordan Hirsch <djh329@cornell.edu>

Table of Contents

I. General Overview and Introduction

The goal of this project is to make an package for the Atom text editor that will enable users to configure their own or others' Atom environments at a project-based level through the use of configuration files.

Atom is a text editor that stands out on its claims of being "hackable." It supports many external plug-ins, called packages, that add unique features which improve the coding experience. Atom is written in JavaScript and CoffeeScript and is run in Electron, a framework that renders web-apps as native apps.

Atom has one or more windows, which Atom calls workspaces. Each workspace can have many panes, each with their own text editor. A workspace can also be home to one or more folders. Atom displays these folders in a file-tree view to the user, giving them quick access to all relevant files. The modifications of this project will be made on Atom v1.26.0-beta1 due to new enhancements made in this version. The extent of these changes and their ramifications will be discussed later in the paper.

II. Stakeholders

There are three main stakeholders to this project. Our client, Kyle Harms, an INFO 2300 professor; our clients intended users, the students who are taking INFO 2300; and the open source community at large. Our client has a pain point of managing different configuration settings for different workspaces corresponding to various assignments. Currently, configurations for Atom are set at a global level across all instances of the editor. The client wishes to have an easier way to set configurations on a per-project basis instead of globally. The development team has been in close contact with the client in terms of the expectation, requirements, and final deliverables of this project. During the development of our requirements analysis, the team communicated with the client via meetings and email to ensure the team and the client are on the same page and the development team is on the right track. For the student stakeholders, this package should appear to be fully behind the scenes, but our package should keep visibility in mind to allow users to understand the changes our packages is making. Open source users have wanted project level configuration for years, with GitHub Issues dating back to 2015. Our goal is to make the package configurable so users can adopt and use our package in ways that we may not necessarily anticipate.

III. Scope of Features

This is an outline of the overall features that the team has discussed with the client - details will be provided in the upcoming sections.

Atom Configuration Files

- Allows users to read in specific-typed files that describe configuration changes

Atom Per-Project Styling:

- Enable users to control styling elements including font color, tab spacing, and white-space control in different editing panes, without changing settings for each one or affecting the global configuration.

Atom Per-Project Package Management

- Enable users to control package management (enabling and disabling) for a project.

IV. Requirements

Atom Per-Project Configuration Control package

An important use-case of this package is to allow an individual to write configuration files for other Atom users, and to manage the configurations of other's Atom work environments through distributed code. While enabled, the package will search for and run the coordinated settings. Through the upcoming section, we model our use-case from the perspective of our client, with users including a professor, and his designated users, the students.

As a non-functional requirement, the package should be easy to use as defined by:

- For Student Stakeholders - Given that the only thing a student must do is enable the package, with instructions from a professor, the student should be able to have the package installed and working as designed in under 5 minutes (not including time for package or code download).
- For Professor Stakeholders - After reading the flight manual on how to create and manage configuration files, a professor familiar with the Atom editor should be able to write their first configuration file within 30 minutes of reading the guide.

The package will be able to control settings including: text color, background color, tabbing length, window styles, and blacklisted packages per window. Any setting that is configurable in a config.cson file (Atom's dedicated file for environment configuration) should be editable though our package. When multiple windows are open, each window should be able to have unique configuration. This should all be done without the user knowing; in other words, users should be able to open windows without configuration files through our package and their workspace should adopt their default Atom configurations. Additionally, there are cases, where

there may exist more than one project in a window. In this scenario, users will have control over which configuration will be used, and have the option to remember this setting.

For scenarios and use cases, please see the appendix.

V. Revised Milestones

In our feasibility study, we proposed an iterative refinement process. We followed the iterative process up to finishing milestone 1, before transitioning from Iterative to an Agile Development model with four short sprints. This new Agile model would greatly enhance the robustness and deployability of our Atom extension project.

At the date of publishing (April 13th, 2018), we are currently on track with this schedule and plan to proceed as described below.

Sprint 3 (April 10 - April 30, 2018)

Atom Configuration Control package

User testing, Acceptance Testing

The project will be user-tested and debugged at this milestone. The team would preferably do user testing on realizing the scenario described previously in Section IV. Comments would be collected and if any issues arise, they should be resolved at this point.
(Details about how the team designs the User Testing procedure can be found in Section X. Testing)

Key Dates

April 16 - April 18: Meet with client to finalize User Testing procedure
April 18 - May 1: Work through iterative Acceptance Testing
April 18 - April 30: Run through User Testing

Sprint 4 (May 1 - May 9, 2018)

During this sprint, the team will commit all our focus to completing the following requirements and guarantee satisfying delivery:

Presentation 4

Entire package should be ready for delivery. Documentation and user instructions, preferably as a flight manual page featuring the package, should be completed. At this point, the code should be ready for production use.

Final delivery

All deliverables are prepared and a presentation is presented to the Client. The project source code should be handed over to the Client for the final milestone.

Key Dates
May 1 - May 7: Complete Acceptance Testing
May 7 - May 9: Presentation and Package Delivery

VI. System Design

The solutions we offer in this project are all based on Atom operating on a single computer/node. Our code will remain self-contained within an Atom package, which exists as a small Node.js styled package conforming to Atom's plugin structure. In the newest version of Atom, v1.26.0 (currently in Beta), the developers at Atom have "reintroduced the idea of per-package configuration." In other words, there is now a built-in way to specify project-specific settings that are local to a single window. This addition to the API is an essential prerequisite to our project. This is done by configuration objects now having a field called `Project Specification`. Upon loadup, this configuration is layered on top of the global configuration at files that are in the directory of the `path` variable. For further details about the functionality that this provides, please see GitHub Pull Request #16845 found at the link below.

https://github.com/atom/atom/pull/16845

VII. Program Design

In sprint 2, the team worked on developing the Atom package using the "core rework" method, i.e. using the functionality implemented in the previously mentioned pull request.

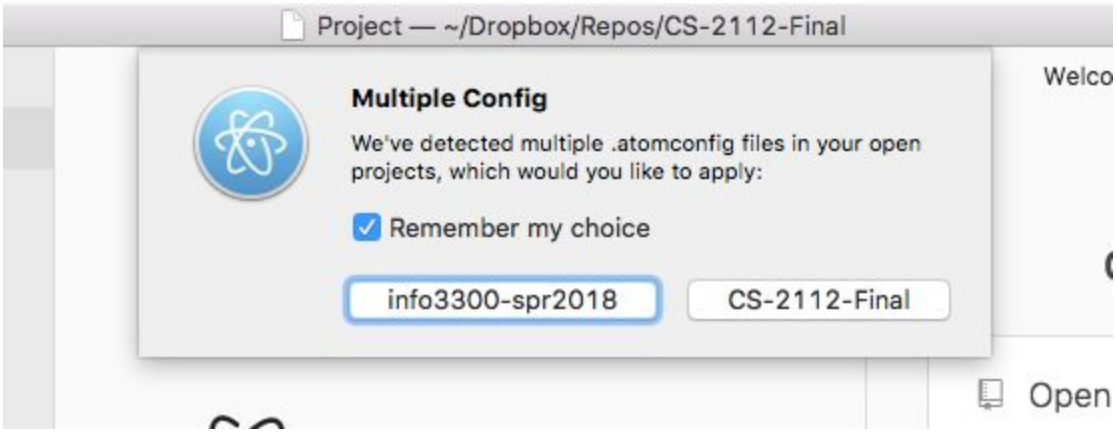
Functionality

Our package implementation works by replacing the command line functionality included in the pull request with an Atom package that automatically handles loading in package configurations. When a project is opened in Atom, our package checks for any files in the directory that end with "`.atomconfig.json`" or "`.atomconfig.cson`". If one is present, it is parsed into a JavaScript/CoffeeScript object and set as the configuration of the current window. The path variable is set to the project the user is opening.

Opening Multiple Projects in a Single Window

When multiple projects are opened in a single window (using the "Add Project Folder" command in Atom), it is possible that each of the projects will have its own .atomconfig file. When this happens, our package will prompt the user to ask which configuration file to use. The

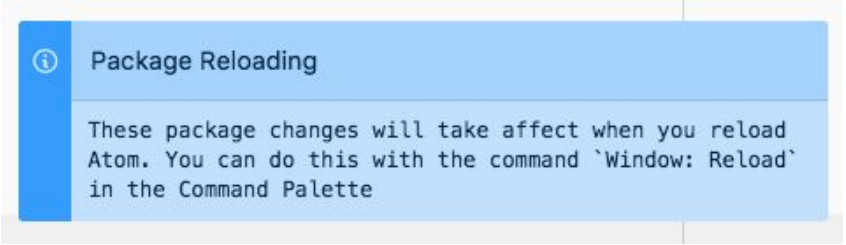
dialogue also contains a checkbox that, if selected, will remember the choice for the next time the user opens the same projects.



At the current time, the behavior of the package forces a user to choose between two different configurations. In future versions, users may have individual control over how each setting conflict is resolved.

Editing the .atomconfig File

When the .atomconfig file is edited and saved, the new project configuration is applied to the current window immediately. The only settings that are not applied instantaneously are those that enable/disable packages. For these settings, a window reload is needed. Instead of forcing a window reload, our package alerts the user that a window reload is necessary and instructs the user on how to do so. Future versions will give users the options to reload the window immediately. Moreover, future versions will give the user the option to turn off this automatic refresh of configuration on package saving.



Toggling the Package

Using the menu option, a user can toggle whether our package is actively updating styling, without needing to fully reload the window. When the package is toggled off, global configuration is reinstalled. When the package is toggled on, project level configuration is enforced.

Implementation Risks

Producing code for a version of Atom that is not yet released presents the risk that this version may not be released as is. This risk has been minimized by keeping track of this pull request, observing its merge progress, as well as following written responses to the developer of this version by the staff at Atom.

There is also the risk that this version of Atom will not be released by our delivery date. However, given that the most recent release of 1.25.x was April 5, 2018, there is a strong chance that this feature, present in 1.26.x, will be included in Atom's May release.

Lastly, there is the risk that users may not have the correct Atom versioning. Our package mitigates this risk by enforcing a dependency on the required version of Atom. This risk is further minimized by Atom's auto-updating feature.

VIII. Progress To Date

As of the date of publishing, April 13, 2018, the team has:

- Developed a system that meets all of the technical requirements
- Conducted internal code reviews to review all technical aspects of the system with attention to robustness and error behavior.
- Written User Tests with User Guides to conduct during Phase 3
- Written a README/Flight Manual explaining the use, installation, and features of our package
- Written a guide on creating .atomconfig files

IX. Testing

Unit Testing

The team used Jasmine as a test framework for the project, a popular choice for Atom package test development. Jasmine is a behavior-driven development framework for testing JavaScript code. The team started off with Behavior Driven Development and created numerous test cases. Then, the team fed the test suites with the developed package and figured out the output from the test suites. The team made sure the output is the same as expected during mocking.

During development, the team wrote several unit tests to represent the configuration that users normally want to change, including the background color, tab size, text color, highlighting color. In addition, tests are covered for enabling/disabling packages. All the test cases have passed.

The original pull request that enabled per-project settings had test cases that specify the behavior of the Atom editor and the new APIs exposed with the update. This likely means that we will be extended said tests, and this will be part of our development in the next cycle.

We will add the unit test cases in the coming Sprints (Sprint 3) and try to brainstorm innovative ideas to break our package and apply test patch onto our package to make our package robust and maintainable from internal usage and external usage. Here are some but not all of the examples that we will look into in the coming Sprints.

1. Users try to enable a package that is not installed.
2. Users put typos in the .atomconfig JSON or CSON files
3. Users want to change the configuration of another package in Atom.
4. Users delete the .atomconfig file by accident/on purpose
5. Users open up large number of configuration files(number at least greater than 3)

Moreover, issues that arise during user testing will be handled with concrete unit testing cases. The advantage of the Jasmine framework is that we can test our code against behaviors that potential users believe to be correct if they run into a currently undefined behavior.

User Testing

In our third sprint, we will begin testing with users. As per our requirements, we will look for:

- A professor with knowledge of CSS given a guide on how to make configuration files can complete their first configuration file in under 30 minutes
- Students with no knowledge of our package can have their files configured within 5 minutes

We will pay close attention to:

- Difficulties professors have with setting any fields
- Fields that professors believe they should have control over (and whether that field is not configurable or our documentation was not clear whether that the field was configurable)
- Difficulty students have loading our package
- Complaints students have about their styling environment or package environment.
- How users respond to errors

In order to accurately convey our intentions and goals of user testing, we will begin by reading the following script that will also be printed on a document presented to the testee. The

documents can be found under the "Testing Introduction Letter" found in the appendix. The important notes of this letter are the time this test will take to conducts, the guarantee of anonymity to testees, and the methods we will be using to collect data.

The development team will be making qualitative notes and records of performance both live and based off the screen record of the user on the following categories of usability:

- Efficiency:
    - [Students and Professors] How long (in minutes) did it take for the user to achieve the required task
    - [Students] How long did it take for a student to open a file with the correct configuration
    - [Student] If we configure a setting that we ask the user to change, can they figure out how to change it given that project settings may overwrite their initial method
    - [Students and Professors] How do users respond to purposefully placed errors in the system
    - [Students and Professors] Do users seem to click on the same area multiple times. Do they get stuck on one part. Is this part covered in the guide presented to them?
- Effectiveness
    - [Students] Do students get confused about opening a project vs. opening a file
    - [Professors] When making configuration files, do professors use the full range of specified functionality
    - [Professors] How often does the professor use our guide in creating their configuration files
    - [Professors] Do professors try testing their configuration files on sample files.
    - [Students and Professors] What errors occurred?

- Satisfaction
    - [Students and Professors] On a scale of 1-5, one being not intuitive at all, 5 being highly intuitive, how would you rank our system
    - [Students and Professors] Following on the last question, please explain your response. Please highlight any areas where you were confused or where you thought the system was effective
    - [Students and Professors] Who do you see as potential users of this program
    - [Professors] What functionality do you believe should be configurable that is currently not
    - [Students] On a scale of 1-5, 1 being uncomfortable and 5 being very comfortable, how comfortable are you with others configuring your atom workspace. Would it depend if the person configuring your workspace was a teacher, employer or other superior?

- [Students and Professors] If you were designing this system, what would you have done differently.

At this point, the user test will be concluded, and the testee will be reminded of the person they can contact with any further inquiries. If they have no further question, they will be thanked for their participation.

Between each task in the test, the administrator of the test will go and quit out of Atom, setting the testee back to the original test. After completing all tasks or after 20 minutes, whichever occurs first, the development team administrator will follow up with the following questions to gauge the users satisfaction with our product. Following the recommendation of our client, we have chosen to use neutral questions when getting opinions about our products as to elicit candid responses from our users.

Prior to testing, the development team will be refining the above user testing procedures with the kelp of Professor Harms. As an expert in Human Computer Interaction, we believe Professor Harms will be an invaluable resource.

The development team plans on conducting user testing between Wednesday, April 18th and Monday, April 30th. Participants will come from three main sources. The first source is students from Kyle Harms' class. We have already spoken to our client about using his students for testing, and we have pitched the idea to his classes. We also posted our request on the course Piazza page. Students have already begun responding to our request, with users tests already on the schedule. We are reserving private rooms in Uris, Olin, and Mann on the requested dates to conduct our tests. To maximize the likelihood of his student's attendance, he has recommended we sell this as a good opportunity to learn new technologies that will be used in this class, which will be an important skill for those looking to TA this class in the future. The second source of user testing is from Cornell Professors; the team has already compiled a list of professors that we are going to ask to do our user tests. The location and time of these tests will be decided on a case by case basis. The third source for user testing is users from the open source community. The development team has already reached out to users on the Github issue tracker following the issue for per project configuration, asking for users to help test. These sessions will be conducted through appear.in, allowing the development team to capture the screens of distant users as well as speak to them face to face for conducting the test and follow up questions.

The development team's goal is to conduct at minimum 12 tests, 6 each for professor and student use cases. While we will have a sample that is mostly made up of student users, we believe that given that many of these students may seek TA positions in some capacity, they will be appropriate subjects for professor use-case testing, which we will evaluate on a case by case basis.

X. Appendix

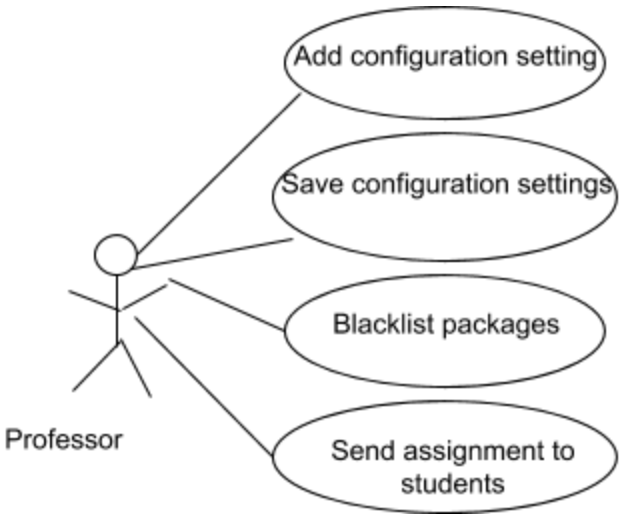
I. Scenarios & Use cases

Atom Per-Project Configuration Control package

Scenario - Professor Configuring Atom Settings

- Purpose: Describe a typical representative professor's usage of the Atomic Management package.
- Individual: A Computer & Information Science Professor who teaches an introductory programming course with use of Atom, similar to our client.
- Equipment: A computer with internet access. While an installation of Atom is NOT required for this, it is highly recommended for testing purpose.
- Assumptions: The professor has read through the tutorials of the extending the atom package and is aware of the functionalities the package provides.
- Steps:
    1. Professor creates a new configuration file in the top level directory of the assignment.
    2. Professor adds all desirable configuration settings into the configuration objects files in the text editor.
    3. Professor lists all packages he wants blacklisted to the students in the configuration file.
    4. Professor saves configuration file
    5. Professor publishes assignment with desired config files to students. This can be distributed through a course website, a management system such as CMS, a version control system such as github, or other code sharing method.
    6. Professor repeats steps 2 through 6 throughout the academic semester for homework and demo files of different levels of difficulty.

Use Cases:

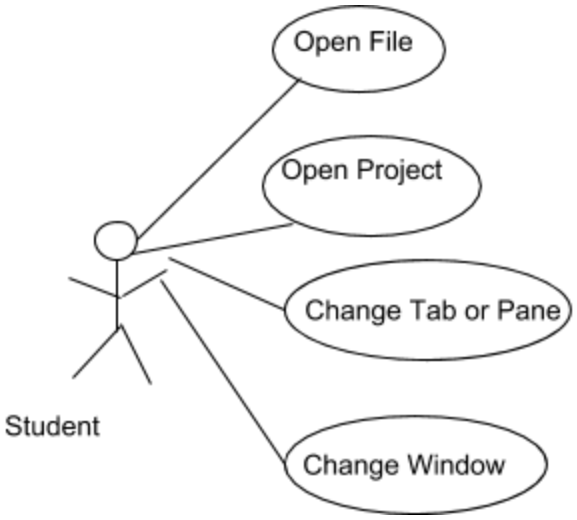


Scenario - Student Using Pre Configured Atom Settings

- Purpose: Describe a student's interaction with the configuration control package
- Individual: A student who is taking an introductory computer science class such as INFO 2300 that use a specific text editor such as Atom with little to no previous exposure to coding or related course material.
- Equipment: A computer with access to the Atom text editor with the package pre installed
- Assumptions: The student has the knowledge to turn on Atom packages and has turned on our package
- Steps
    1. Student opens the distributed project, Project A, in Atom, opening from the root of the project itself
    2. Student opens a file in the project directory
    3. File opens with correct configuration
    4. Student changes tabs or panes with file from same project
    5. File opens with the same correct configuration as before
    6. Student changes tabs or panes with a file from a different project, Project B
    7. Styling is the correct form for that project
    8. Message pops up to confirm which packages to run. The student is prompted to choose between A's configuration and B's, with the option to remember these settings.
    9. Student opens new window and opens up a file from project A, opening just the file, not the whole project

10. The file opens with no configuration unless otherwise specified in global, themes, or profiles

Use Cases:



II. Previous Sprints

Sprint 1 (*February 17 - March 15, 2018)*

Feasibility Studies

Finished compiling, presenting and reviewing feedbacks with the client.

Requirements Analysis

The team prepared a formal document that details the client's requirements for both components of the project, categorizing the requirements into primary, desired, and optional features. This was completed after two formal requirements gathering meeting with the client, and with revisions of the requirements document between meetings.

System Design document

Since the team is building on top of a sophisticated, open-source project, software architecture and compatibility was designed and confirmed at an early stage. A design document of the software architecture was done and presented to the client to gather feedback.

Prototype for Configuration Control package and Tests

A simple prototype for Configuration Control along with tests was prepared. It will support per-project configuration (as opposed to the editor-wide, global configuration by default), and a demo was presented to the client.

Sprint 2 (March 16 - April 9, 2018)

Atom Configuration Control package

Supports per-window settings and environment

Design to be completed and finalized. Implementation for the next-stage feature should be done: user should be able to control all settings for all packages including window styles and enabled/disabled packages per window. When multiple windows are open, each window should be able to have unique configuration. A demo will be presented to the client.

III. Testing Introduction Letter

"Thank you for participating in user testing for our new additions to the Atom Text Editor. Today, you will be presented with a variety of tasks using our new package and the Atom Text Editor. The focus of this test will be on the program itself, not on you as a user.

This test will take approximately 30 minutes, and you may opt out of this testing at any time. All personal data will be kept confidential, reviewed only by the development team and the CS 5150 course staff. The results of your test will be written up in a report with all personally identifiable data removed to preserve anonymity.

Over the course of the test, the development team will be recording your own screen interaction using screen capture. Moreover, the development team will be personally observing and noting your interaction with our program. Again, the focus of our notes will be on testing the program, not you as the user.

You will be presented with a guide on how to work with our package, but this will be the only guidance the development team will give you regarding the user testing. If you are truly stuck on a particular aspect, you may skip it; however, the development team will not be able to answer any questions you have while the test is being conducted. At the end of the test, the conductors of the survey will ask you some questions regarding your experience with out package.

If you have any questions regarding the use of the results of the user testing, feel free to contact the development team liaison, Daniel Hirsch, at djh329@cornell.edu. We thank you for your participation."