GSoC Application

Name: Udayan Tandon
Email Address: udayan12167@iiitd.ac.in
IRC Nick: udayan12167
Gitter: https://gitter.im/Udayan12167
Location (City,Country and/or Time Zone): Delhi/Bangalore, India, UTC +5:30

Academic Background:

1. Indraprastha Institute of Information Technology-Delhi: Currently pursuing Bachelors in Computer Science.(CGPA: 8.65/10.0 till 5th Semester)(Graduation Date: December 2016)
2. Bal Bharati Public School,Pitampura,Delhi:12th Grade CBSE Board(93.10% Aggregate)

Possible Mentor:

Lasse Schuirmann
- Gitter: https://gitter.im/sils1297
- IRC Nick: sils

Mischa Krüger
- Gitter: https://gitter.im/Makman2
- IRC Nick: Makman2

What is the ultimate goal of your proposal?

"coala is a simple COde AnaLysis Application. Its goal is to make static code analysis easy while remaining completely modular and therefore extendable and language independent. Code analysis happens in python scripts while coala manages these, tries to provide helpful libraries and provides multiple user interfaces". -coala Github page

Currently, coala only supports console output as part of its user interface. The ultimate goal, as detailed in this proposal, is to implement a GUI for coala and present an easy to use interface for the user. The GUI will have the ability to make coala configuration files and trigger runs using those configuration files. The GUI will also be responsible for showing the logs. This GUI will be implemented using PyGTK3.

What components/modules will the proposed work modify or create?

- **Mockups**: The base of every GUI application are its mockups. These mockups define how to design an effective application. While designing an application, one needs to incorporate a blend of modern design thinking, along with a concern for practicality and ease of use. All things considered while designing mockups these fundamentals are essential:
    1. **Intuitive and Consistent Design**: For a graphical interface to be functional and navigable, the components must be laid out in an intuitive and consistent fashion. Since this GUI is meant to be used by the GNOME developer community, users will be familiar with the GNOME interface. Thus, GNOME Human Interface Guidelines will be followed to the letter. Furthermore, features will be prioritised according to the preferences of the contributors.
    2. **Clarity**: The UI will strictly follow GNOME guidelines. New applications tend to require a steep initial learning curve.To mitigate initial difficulty, UI elements and lables will be short and self explanatory. Additionally, a separate standalone documentation will enable greater user support for the software.

3. **Maintainability**: After the completion of the GSoC internship, the GUI will be released with coala 0.3. It will be open to scrutiny by the developer community, with an invitation for enhancement suggestions and other criticisms. Thus, the application design must be able to accommodate constructive suggestions and enhancements. Additionally, coala being in its alpha stage will quite probably require feature enhancements. With that thought, mockups will have to be flexible making sure the application is extensible as well as scalable. The aim is to present a working model with major functionalities implemented by the end of the GSoC period, instead of a fully developed GUI. To this end, maintainability is a crucial aspect.
4. **Aesthetic Appeal**: This is not crucial as I'll be using GNOME design elements, which in my opinion, have an aesthetically pleasing design.
5. Given herewith is a link to some sample mockups which were created in balsamiq and are for demonstration purposes only:
   http://udayan12167.github.io/coala-design/

- **Prototype**: The first leg of the project will involve creating a prototype of the GUI as a proof of concept. The main purpose of the prototype is to implement a GUI with basic functionality in a simplified and swift manner. Hence, it will not be functionally exhaustive in any case.

  The prototype will require the following to be implemented:
  1. **Main Script**: Currently the main script of coala is responsible for running the SectionManager and the SectionExecutor and handle other tasks when the coala command is invoked through CLI. The prototype will have its own mainscript which will ensure its smooth operation and workflow. It will be very similar to the coala mainscript in terms of functionality and workflow but will incorporate certain parts unique to the GUI.
  2. **GUIIntercator**: This will inherit the coala Interactor class and build on that. Currently, coala has two implementations of interactors namely the ConsoleIntercator and NullIntercator. As ConsoleIntercator is responsible for interacting with the user through the Console, the GUIInteractor is a solution for the GUI. The GUIInteractor will use the GObject signals to invoke actions to the GUI and fetch data from the GUI.
  3. **GUIPrinter**: This will inherit the coala Printer class and build on that. As of now, there are several Printers that are a part of coala such as the ConsolePrinter, FilePrinter or NullPrinter. Each Printer is responsible for performing logging tasks in coala. The ConsolePrinter delivers the logging statements to the Console whereas the FilePrinter is in charge of writing the logging statements to a File. Similarly, the GUIPrinter will deliver the logging information to the GUI.

  Reused coala elements:
  1. **SectionManager**: This class is responsible for first, setting up the logging objects(interactor and printer) and then, reading and setting up the sections to be executed.
  2. **SectionExecutor**: This class is responsible for executing the sections and invoking actions in the GUI as and when required. It executes the sections using multiprocessing.

- **The GUI Application**: This is the main focus of the project and my primary concern during my tenure as a GSoC intern. Firstly, the GUI application will require intricate software architecture planning. This will be done under the guidance and in close collaboration with my mentor, armed with the knowledge and insights gained from developing the prototype. Secondly, the details of all the features to be implemented will only be finalized post architecture planning. However, below is a list of the features currently planned:
  1. **Running Sections** : This is the main focus of the project. Using the GUI application, the user should be able to run sections and display the results in an organised format.

2. **Building a final GUI** : This will be based on the mockups that will be finalised with the GNOME team and will serve as the blueprint for a fully functional GUI.
3. **Showing Logs** : The user will be able to view the logs within the GUI itself rather than viewing them in a separate text file. My first concern will be to only display the logs to the user. Once this is achieved, the logs view can be stylised and the logging statements can be segregated on the basis of their tags.
4. **Ability to open editors to apply patches** : Post-processing, coala gives users the option to open an editor of their choice, in case they wish to apply patches.
5. **Editing .coafiles(Optional)**: .coafiles are the files that are used to configure running of coala in the CLI. An optional functionality could be opening a .coafile in the GUI and allowing the user to migrate his CLI based coala project to the GUI. This can be done if there is time to spare after the higher priority features have been implemented successfully.
6. **Showing progress(Unit based or time based) (Optional):** As the codebase and number of tests that coala runs for code analysis grows, the time taken to process the code also grows. Therefore, a progress bar will be useful feedback for the user. This is an optional feature. However, a time prediction algorithm or a defined unit can be used for this purpose.
7. **Writing Tests** : I hope to write tests on the fly parallel to my coding implementation and do a test driven development. Dogtail is a python framework for automated GUI testing. It currently supports testing of GTK+ applications. This framework will be used to perform behavior driven tests with the GUI.
8. **Getting code integrated with coala**: This will be done in the last phase of the project which will involve checking in and merging the code with the coala repository and then, testing if the code works on another user's machine.

- Some other modifications:
    1. **Make coala non-linear** : As of now, coala has a linear pattern of execution wherein the mainscript calls every other component. The GUI requires a non-linear flow because a linear execution would require the user to re-start the GUI upon each run. Thus, incorporating nonlinearity in coala is a major segment of the architecture design phase of the GUI which, precedes everything except the prototype. For more information in this regard, refer to the following link: https://github.com/coala-analyzer/coala/issues/252
    2. **Ctrl+C to quit coala gracefully (Optional)** : Currently, when Ctrl+C is pressed coala waits for all the bears to finish their execution and then quits. a more time-efficient approach seems to have the bears exit gracefully midway and then have coala quit immediately. A possible solution is to have callback methods when the SIGINT signal is registered i.e. Ctrl+C is pressed. This also provides an additional GUI feature wherein the user can cancel a run in the middle of the execution.

What benefits does your proposed work have for GNOME and its community?

coala is a code analysis application, whose main goal is to make static code analysis easy and mainly modular. coala is not language specific in its code analysis. The best part about coala is that people can write bears and extend its functionality as per their requirements. These bears can be as simple as checking for indentation as opposed to far more complex ones like Code Duplication Detection. So far, coala offers an easy to use CLI interface wherein users are prompted for input and actions, as and when required. This accounts for a streamlined and hassle-free experience. My project proposes to build a GUI for coala, which I believe will have the following benefits:

1. A GUI offers a faster and more efficient way to use coala. It will have many features which will help coala users perform their static code analysis with greater speed and have a more stylised way to look at the results and perform actions on them.

2. While I am quite comfortable and well-versed with the command line interface now, I found my initial time with it to be quite tiresome. A GUI provides a more visual means of performing actions and is therefore, easier to use. Hence, a coala GUI would help new developers joining the GNOME community to focus on analysing their code without the added complication of getting acquainted with the command line interface.

3. coala helps in improving both patch and code quality. As a new contributor to the open source community, I often found myself fixing spacing issues along with other trivial mistakes related to formatting. This is frustrating and counter-productive for the user as well as the reviewer. coala and its GUI will greatly help in solving this problem. It helps new developers to contribute in a more efficient and effective manner.

4. Provision of non-real time static code analysis is another pro of coala. IDE's have several tools to provide support while writing code but nothing to check code after it has already been written. coala to the rescue!

5. GNOME is an organisation known for its GUI applications. Thus, creating a GUI for coala seems like the natural order of things at GNOME.

Why are you the right person to work on this project?

I, Udayan Tandon, am an undergraduate student studying Computer Science at Indraprastha Institute of Information Technology. I'm extremely passionate about software development and contributing to open-source feels like the next step towards fulfilling my passion. I have whet my skills in multilingual programming. I'm a proficient coder in Python, C, Java, Ruby, PHP, HTML and CSS, having implemented at least a couple of projects well over 1000 Lines of Code in each.

Technical Ability:

1. Strong Python Knowledge: I have been working with Python for the past 2.5 years. I am very familiar with the language and have a fair amount of dexterity with it. I have done various projects in Python ranging from modifying libraries to writing automated testing suites. As this project is Python based, I feel that I can utilize my skills to their fullest.

2. Familiarity with coala codebase: Although, I have been involved with coala only for a brief period of time, I have become hugely familiar with the total 6000+ lines of coala code. This familiarity with the code will save on time that would otherwise have been spent getting acquainted with the coala libraries. My in-depth understanding of the codebase equips me to do the tasks at hand with considerable speed and accuracy. In evidence to my expertise with coala code, I have 3-4 merged pull requests which involve around 17 commits.

Discussions with the mentor: During my brief period of working with coala, I have had detailed discussions with my mentor (Lasse Schuirmann) on the best approaches towards implementing the project. I'm also quite familiar with the contributors of the coala community.

During my constant communication with Lasse, we discussed the best possible architecture for the GUI implementation. Moreever, I have been an active participant in conversations on the architecture of some of the current coala features.

Collaboration Experience: Contributing to open source is all about effective collaboration. While this is my first adventure into the open-source world, I have been involved in many group projects ever since my first year of college. These projects have involved working with 4-5 team members

with hard deadlines that have all been diligently met with full cooperation and coordination amongst all the team members. A major challenge with GUI applications is integrating the front-end with the back-end and coordination and understanding amongst the teams taking care of these two departments. I have some experience in this area and know how to integrate the front-end with the back-end, both in theory and practice.

How do you plan to achieve completion of your project?

Pre Coding Period (27th April - 25th May)

Due Date	SMART Goal	Measure
10th May	Familiarity with PyGTK+3 Library.	Go through tutorials of PyGTK+3 and figure out components to be used in the project.
15th May	Finalise the mockups for the GUI application.	Complete the final design mockups for the GUI by discussing it with the GNOME-desgin community.
25th May	Architecture planning for Prototype	Have discussions with mentor on the proposed architecture and coding structure of the prototype.

First Trimester (25th May - 21st June)

Due Date	SMART Goal	Measure
7th June	Implement Prototype with all its dependencies	Prototype will be implemented along with its mainscript, GUIInteractor and GUIPrinter using PyGTK and python
14th June	Test prototype and plan architecture for final GUI.	Run the prototype multiple times and assess the problems and deficiencies in it. Accordingly plan the architecture of the final GUI and required changes to coala libraries.
21st June	Work on making coala non-linear and in parallel get some skeletal code ready for the GUI.	Non Linearity for coala is an essential part to the GUI and the architecture for it would have been discussed in the previous week.

Second Trimester (22nd June - 19th July)

Due Date	SMART Goal	Measure
28th June	Make coala non-linear.	coala will now be non-linear and the wont be controlled by the mainscript but rather by a control element which responds to signals from external sources as well.
12th July	GUI will be able to run sections	The GUI will be capable of configuring,enabling,disabling and running sections of coala.

19th July	Displaying results of the run	The application will be able to display results of the run of coala.

Third Trimester (20th July - 21st August)

Due Date	SMART Goal	Measure
26th July	Make displayed results actionable	Enable actions on results like opening editors to apply patches or auto apply patches.
2nd August	Display log files	Log files are displayed as part of the GUI.
16th August	Review and test all the code and features. Clear backlogs of any implementation work and focus on optional objectives.	There could be backlogs or some deadlines missed this time period is kept to work on them. In parallel thoroughly test the application and run all tests repeatedly. Optional goals might get time. (All coding ends here
21st August	Get all the code checked in and intergrated with coala.	

What will be showable two months into the project ?

I plan to have the following things ready two months into the project:

1. Prototype : A functioning prototype of the GUI which can be presented to anyone and everyone in the community. This prototype will showcase the ability of coala along with its features, to some extent. It will comprise of the following features:
    1. Making a .coafile : These files are responsible for all the configurations of a single coala run. It constitutes of sections which contain specific parameters.
    2. Enabling/Disabling certain sections.
    3. Running Sections: A user will be able to use the prototype and initiate a coala run.
    4. Display Results: Display the results for each execution.
    5. Interact with the user: The prototype will prompt the user for certain required parameters.
    6. The Prototype GUI will need to be restarted for each run of the section due to the linear nature of coala. Non-linearity will be implemented with the main application.

2. coala becomes non-linear: There will be a control object and I will be able to show it on the CLI that coala is executing in a non-linear manner.

3. Main Application: The main GUI will be partially ready at this stage. The final GUI will be viewable once the application is complete in the last leg of this summer project.

What are your past experiences with the open source world as a user and as a contributor?

I'm relatively new as a contributor to the open source community and began contributing about 1.5 months back to the coala project. I have become a fairly proficient contributor and a valuable part of the coala project. Besides fixing multiple issues for coala, I have also been involved in discussions on how to take the project forward. I have been in constant communication with the creators of the project, discussing coding architectures, providing bug fixes and reviewing

commits made by them. I have come to learn a lot during my short association with the coala project and hope to further this learning throughout the summer. I feel confident about contributing to other open-source projects and expanding my knowledge and skills as a developer. Given below are some of the links to my work:

- Coala repository: github.com/coala-analyzer/coala

- Issues Fixed:
  - Bug #386 : https://github.com/coala-analyzer/coala/issues/386
  - Bug #332 : https://github.com/coala-analyzer/coala/issues/332
  - Bug #303 : https://github.com/coala-analyzer/coala/issues/303
  - Bug #256 : https://github.com/coala-analyzer/coala/issues/256

- Currently assigned:
  - Bug #282 : https://github.com/coala-analyzer/coala/issues/282

- Summary of involvement can be found here: https://github.com/coala-analyzer/coala/commits?author=Udayan12167

As a user, I have delved in several open-source softwares. The life of a developer practically revolves around open-source softwares i.e. from IDEs to text editors to mockup designers. My experience with GNOME specific software has been very pleasant. I have generally found GNOME software to be well thought out. I have had little to no grievances everytime that I have used GNOME software. Below is a small list of softwares designed and implemented by GNOME that I have used fairly extensively:

- Gedit : A simple and lightweight text editor with infinitesimally many plugins.

- GNOME Desktop : I have used it with Ubuntu and found it to be simply delightful.

What other relevant projects have you worked on previously and what knowledge did you gain from working on them?

1. **Mobile Computing Group Internship**: This internship required me to develop on top of a python library called sMap. sMap is responsible for providing archival capabilities. This internship required me to modify sMap to improve its functionality when interfacing with electronic meters. Skills and knowledge acquired:
   - Getting an idea of how to understand a large unknown codebase.
   - Useful knowledge in developing python based projects.

2. **Robotics and Programming Project**: Design a java application with a GUI to guide a robot through a random world. Skills and knowledge acquired:
   - Working on designing and implementing a GUI although the application was very basic.

3. **Internship at EMC Corporation**: I'm currently an intern at the Isilon division in EMC^2 . Isilon is a company which makes clustered NAS storage. I have been heavily involved in writing automated tests in python. My current project consists of writing APIs in Python for VMWare. Skills and knowledge acquired:
   - A deeper insight into object oriented programming in python.
   - Planning and writing automated test suites.
   - Working in a team and coordinating with team members.
   - Discussing architectural problems and implementing the solutions.

- Collaborating with people in a remote location.

What other time commitments, such as school work, another job, planned vacation, etc., will you have between May 25 and August 21?

I have no commitments other than GSoC between May 25 and August 10. I plan to contribute in an excess of 40 hours per week without fail. The next semester of my college begins from August 10, however, as is the general trend, the workload is expected to be light till August 21. I plan to devote an excess of 40 hours per week even after my college semester commences, utilizing weekends to the fullest, as required.

I have no planned vacations or school work during this period and plan to fully concentrate on my GSoC project.