Project Phase III

by

Andy Li
Kai Song
Stephen Harrichand
Steven Diao
Yang Wang

November 13, 2014
CSCC01H3 F
Anya Tafliovich
University of Toronto at Scarborough

Table of Contents

State of Current Project

After Phase II, we have improved core features for the front end of our app. We built on and polished the user interface in order to make it more user friendly and more clear as to what the purposes of our app are. It was very confusing from the previous iteration as the bulk of our features were not fully implemented. Furthermore, we previously mentioned in our interview regarding working on a prototype of a web crawler implementation. We are anticipating to have this implemented to a certain degree, but it may extend into the next iteration for polishing. Also, an important spike we discovered in this iteration was the WARC format. It is very difficult to incorporate into our current application because it's very complex and requires a significant amount of work. As a result, we are currently moving forward to implement our web crawler.

For our iteration plan, we found it hard to keep up with the plan. As our burndown chart shows, we are behind schedule. We initially anticipated our project velocity to be much higher than the previous iteration. However, merging in Git becomes very complex and time consuming when multiple teammates work on semi-dependant features because of large scale conflict resolutions. In addition, all of us had midterms (we did not anticipate conflicting time slots) at some point within this iteration and it was very difficult to produce a productive hour (our unit for cost) without compromising our success in other courses. As a result, the amount of time invested is significantly lower than the estimated project velocity, but we completed work of higher quality after last week.

Also, we had to push various features into the next iteration as we had to recalculate our project velocity in this iteration. The good news is that we now have a satisfiable application that meets the expectations from our previous iteration and the back-end core is underway in development. We expect that in our next iteration our project velocity is significantly higher and that we can focus more on the remaining core features.

Iteration Plan

In Asana within the project "*Sprint 3 Iteration Plan"*, under the project description (to the left on a maximized screen, our iteration plan, project velocity and cost calculations will be stated in there since it reflects what our task board looks at the beginning of the iteration.

Task Board

Our task board is in Asana under the project "*TaskBoard for Phase III"*. There are several categories listed: *TO DO*, *IN PROGRESS*, *TO BE TESTED*, *UNDER TEST* and *COMPLETED*.

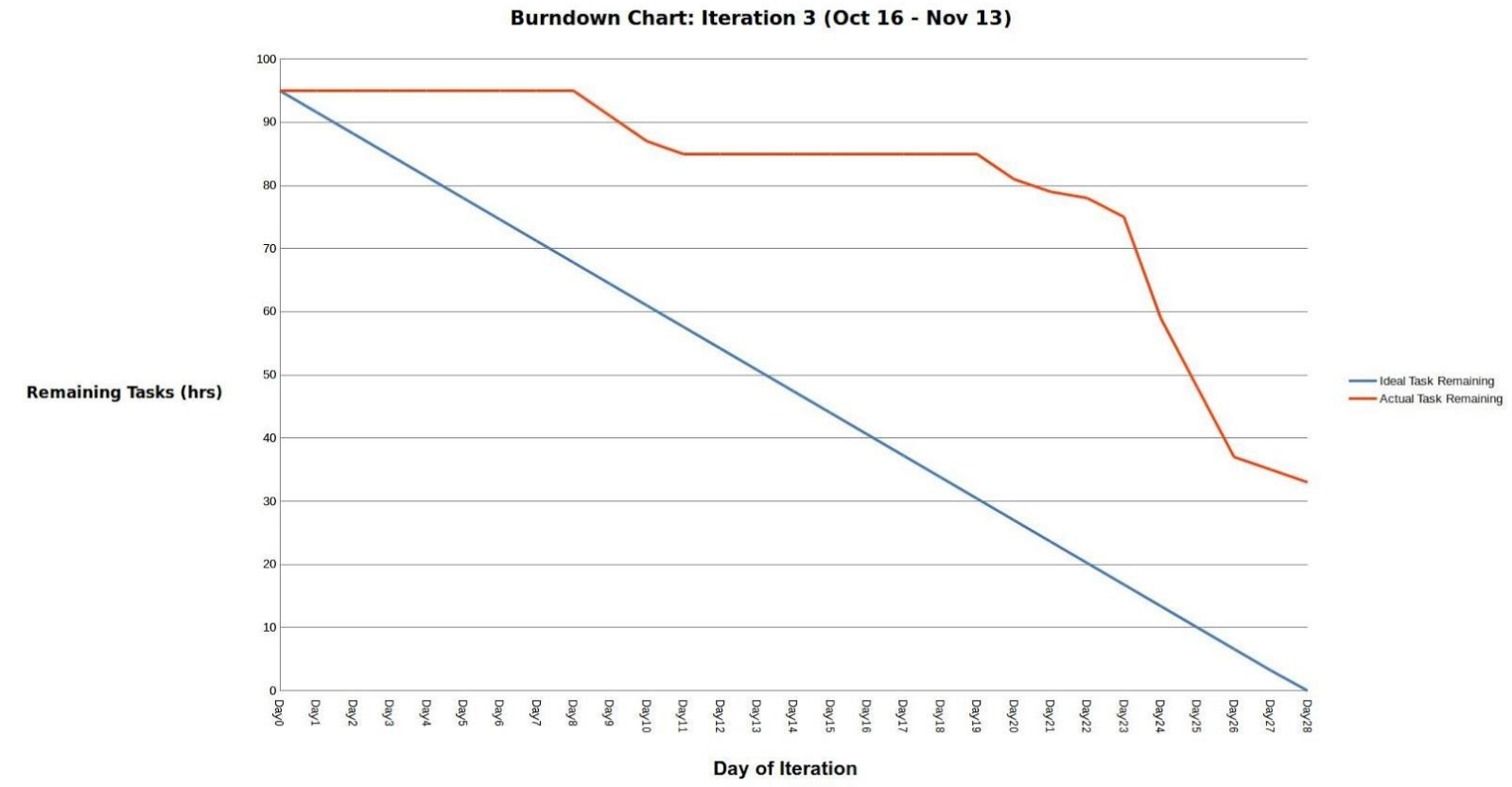
- Tasks are considered complete only when they have passed the acceptance tests.
- Completed tasks are checked off as completed, which means they will not appear in the default view that only shows remaining tasks. Hence to view all tasks, in the project screen, select view and click on *View all complete and incomplete tasks.*

Release Plan Changes

In Asana, there is a project called Release Plan and we pushed user stories with heavier costs into the next iteration because of the recalculated project velocity as well as new spikes in this current iteration. This means costs have been re-evaluated and estimations have been updated to match our current and projected progress.

Burndown Chart

The burndown chart for our project is as follows:



Code Review - ANDY

Reviewer: Andy Li
File under review: crawler/views.py
As of commit： 7268592639161ff7497171c79b6cdc403f4bd138
Date: Nov 13, 2014

Correctness
It is correct because for every independent session, its respective data does not leak into other sessions. In addition, it also links to the proper page as requested on the interface.

There seems to be features that are currently not working, but that will be pushed to the next iteration.

Coding Style
The code is very clean and easy to follow. Excellent use of indentation and great naming conventions. This is a great model to follow regarding coding style.

Coding Guidelines
It's satisfied due to the above reasons.

Quality of Documentation
Documentation is very formal - very precise and concise to the point. However, when commenting out non-functioning or glitchy code, be sure to state why it’s not working.

Otherwise the commented code shouldn’t exist within the body of the code.

Quality of Testing
Automated testing is not available, however to test this we would need to do this manually. At the moment, it lacks validation, but it does what it's supposed to.

In addition, manual testing has been done for it. Very simple cases has been used, however, validation still remains.

Code Review - KAI

Reviewer: Kai Song
File under review: WebSpider/crawler/views.py
As of commit： 7268592639161ff7497171c79b6cdc403f4bd138
Date: Nov 13, 2014

Coding Style Issues:
[line 21] what does the global variable l = [] do?

[line 27, 37, 46, 62, 84, 231, 234, 245, 248, 258,]
Coding lines are too long to read in one monitor

[line 91 - 137] [line 264 - 297] useless comments

[line 239 - 240] bad variable names - What know that does the p contain?

Correctness:
[line 18 - 19] useless import : import urllib import re

Guidelines:
index.html seems to do most works, it might be a good idea to separate

Single responsibility - Try breaking into smaller htmls

It is better if we put all ".html" filenames at the top of codes, so when we want to change to the different html, we don't go through all code to change them

Documentation Quality:
Code style looks neat, and easy to follow

Testing Quality:
There needs to be more simple base cases
no setUp, should initial [line 9, 14, 19, 24] to setUp part
no tearDown part

Code Review - STEPHEN

Reviewer: Stephen Harrichand
File under review: example.py in mybot/mybot
As of commit： 7268592639161ff7497171c79b6cdc403f4bd138
Date: Nov 13, 2014

Correctness

- Right now it is doing what we want it to do
- Still need to work through it and adjust the algorithm to scrape the ONLY the items that are valuable to the customer

Coding Style

- Good use of variable naming
- Many, many nested if/else statements in parse()
- Perhaps could write small helper functions to fix this

Coding Guidelines

- Seems to be following guidelines

Documentation

- Some documentation which helps reader understand what is going on
- More would be helpful
- Documentation is minimal
- Could be a lot for descriptive

Testing

- No testing other than making sure it works and pulls some data
- Could possibly write dummy-sites and test on those
- Some kind of test suites need to be developed

Code Review - STEVEN

Reviewer: Steven (Shuo) Diao
File under review: WebSpider/views
Date: Thu Nov 13 12:03:57 2014

Correctness:
[233] What does invalid user do? It is never actually called.

[239] Should use something other than subprocess and wait() as this limits it to one functionality at a time and it would be hard to run on multiple threads

[70] Add monitor and add sessions seems very similar maybe there is a way to make that into a function and both call that instead

Coding Style Issues:
[10] Messages have already been imported no need to import again

[21] What does the global variable l do?

[91] Home request should be deleted if not used rather than completed commented out
There should be more comments in the code to explain what is going on

No docstrings what-so ever so

Coding Guidelines:
Some code can be broken down and better documented

Documentation:
code it decently styled variable names, functions and class names are all good, however lacking in comments and docstrings

Testing Quality:
Tests exists but not much of it works and some of them are empty.

Code Review - YANG

Reviewer: Yang Wang
File under review: mybot/spiders/example.py
As of commit： 7268592639161ff7497171c79b6cdc403f4bd138
Date: Thu Nov 13 00:03:57 2014

Coding Style Issues:
[6] Name of Class = Example Spider?
[7] Why is name=example? - Not clear what Example Spider is or does.
[24, 38] There are better ways to check if a list is empty or not, rather than != [].
Follow Python style guide.
[38-62] Too many conditionals, not clear what each conditional does.
Try refactoring into helper functions

Correctness:
[8] This code grabs all sites instead of sites belonging to a particular session. Don't we want sites belong to a single session

[9-12] Should go into def __init__ since it's part of initialization?

[15] Why do we check for data inside //ul/li? Ie. Why do we check for data inside the <ul> and <li> tags only? What's the rationale behind ignoring all other html content?

[27] Why do we only take the 1st link name? Is it safe to assume that the list has only one element?

[36] Description takes the entire content of text inside "site"? Extract() returns a list, so we are storing the entire list into item['desc']. Is that intended?

[45] str_urls is just a string. Why are we iterating on a string. The for loop seems to be incorrect.

[57-62] That's the wrong way to update Django database objects. It iterates over all objects to update a single object. O(n) complexity
Should query for the specific origin and link, and modify that object. The database is optimized for this operation. (This may explain why our scraping is so slow).

Guidelines:
def parse seems to do too much: Single responsibility - Try breaking into smaller functions

Documentation Quality:
Code is poorly styled in terms of variable/method names, and isn't self-explanatory. Could use more documenting, especially on what parse is supposed to return.

Testing Quality:
No tests found… Should write some unit tests to verify that def parse is working as intended.

Code Review Video Instructions

Our video is located in our GitHub repository within the directory named *Media* and the file is named ***code_review.mp4*.**

Testing Instructions

Currently, we only have automated tests for the Django code. We are still figuring out how to run tests for Scrapy without running into dependency problems.

The unit tests for Django are in a file called testview.py which is located in WebSpider/crawler directory.

To run the tests, you must be in the directory of where the makefile is located in and then type the command *make test* in the terminal.