SATToSE 2015 Hackathon

The topic of this year's SATToSE Hackathon is ***process automation***. Process automation has played a central role in software engineering and software evolution, already since the pioneering work of Osterweil [1]. These days, the increasingly distributed nature of software development has increased the demand for this technology, leading to notable innovations. Examples of such innovations are *distributed version control systems* such as Git, associated *"social coding" platforms* such as GitHub (such platforms have popularised the pull request model), *continuous integration services* (such as Travis-CI[1] and Jenkins[2]), and the DEVOPS movement [2].

Naturally, software engineering researchers have also benefited greatly from the different affordances of process automation. We are now able to empirically analyze large (open source) software ecosystems as well as integrate data from multiple types of repositories (e.g., version control systems, issue trackers, and mailing list archives), thanks to tool suites like MetricsGrimoire[3] (see also tutorial by Gregorio Robles) and others.

Your assignment is to form teams, and exercise methods and tools for software process automation. You are free to choose your own problem, as long as (i) you can frame it in this theme, and (ii) you will use a GitHub or BitBucket repository as you develop code for your solution, which will enable you to experiment with these technologies as well. For inspiration you can look up software evolution topics from the previous editions of the SATToSE Hackathon:

- 2014: http://grammarware.github.io/sattose/slides/Hackathon.pdf
- 2013: https://github.com/SATToSE/SATToSE2013/wiki
- 2012: http://grammarware.net/slides/2012/hackathon.pdf

For additional inspiration, here are some other ideas:

- Automating the evolution and deployment of your personal (researcher) website. The evolution scenario is the following. Your publications are stored in a BibTeX file, and the corresponding HTML publications page on your website is generated automatically from this file. The entire source code for your website is stored in some version control system. Whenever you publish a new paper, you only want to update this BibTeX file, and commit the change to the repository storing your website; the publications page on your website should then be updated automatically (as an extreme case of automation, you may also consider pulling your papers directly from Google Scholar or DBLP; your webpage should then automatically update when you find an update on one of these). *Hints*: Frameworks such as Jekyll[4] offer functionality for transforming Markdown into static websites and blogs, as well as free hosting and integration with GitHub pages.

[1] https://travis-ci.org
[2] http://jenkins-ci.org
[3] https://metricsgrimoire.github.io
[4] http://jekyllrb.com

Jekyll Scholar[5] is a Jekyll extension able to transform BibTeX into HTML. Continuous integration services such as Travis-CI can be set up for automatic deployment, and are integrated with GitHub.

- Automatic matching of developer profiles with job advertisements. Open source developers leave numerous traces of their skills online. For example, programming expertise can be inferred from contributions to open source repositories, or quality of answers to Stack Overflow questions. Developer profile aggregators (e.g., coderwall,[6] Masterbranch[7]) have already been collecting such information. Regarding GitHub in particular, aggregate information about one's contributions is displayed on one's profile page automatically, or it can be generated by services such as The Open Source Report Card.[8] GitHub developers can also set a "Available for hire" flag on their profiles, indicating whether or not they are interested in finding a job. Historical GitHub data is available through services such as GHTorrent[9] (both offline and in-browser) and GitHub Archive,[10] as well as through the GitHub API[11] itself. Historical Stack Overflow data is available through the official Stack Exchange data dumps[12] (offline), or the Stack Exchange data explorer[13] (in-browser). Software development job advertisements are available on many portals online, including GitHub Jobs,[14] Stack Overflow Careers,[15] Monster,[16] and Indeed.[17]
  Hints: There is some preliminary work [3, 4]

- Automatic static analysis checks for incoming pull requests. In the pull-based development model, made popular by GitHub, contributors to a project need not share access to a central repository anymore. Instead, anyone can fork the main repository (or create a branch), work independently and, when ready, submit a request to have their change merged back into the main development line, i.e., a *pull request*. Pull requests have made it possible to decouple the development effort from the decision to incorporate the results of the development into the code base, at the expense of an increased burden on project core team members (project integrators), who must review all these changes before deciding whether to accept them or not. Still, the pull-based model is becoming increasingly popular (projects such as Ruby on Rails[18] receive

[5] https://github.com/inukshuk/jekyll-scholar
[6] https://coderwall.com
[7] https://masterbranch.com
[8] Currently unavailable. More information here: http://blog.kris-lab.com/open-source-report-card/
[9] http://ghtorrent.org
[10] https://www.githubarchive.org
[11] https://developer.github.com/v3/
[12] https://archive.org/details/stackexchange
[13] http://data.stackexchange.com
[14] https://jobs.github.com
[15] http://careers.stackoverflow.com/
[16] http://www.monster.com/
[17] http://www.indeed.co.uk/
[18] https://github.com/rails/rails/pulls

hundreds of pull requests each month), to the extent that in some projects core team members (who would otherwise be able to push their changes directly) also submit their code as pull requests, so that *all* code gets reviewed (and tested) before being merged. Continuous integration (CI) is one of the technologies that facilitates this process. Whenever a pull request is received by a project using CI, it is first automatically merged into a separate testing branch, where the code is automatically built and the existing test suite is automatically run (see, e.g., Travis-CI https://travis-ci.org). The GitHub pull request user interface is then updated automatically with the test results, and the pull request submitter and project integrators can be notified by email of the outcome. Typically, pull requests undergo manual code review by project team members *only* after all the tests have passed. Yet, despite the potential benefits of CI, and despite the availability of numerous static analysis tools (e.g., Checkstyle[19] - finds violations of coding conventions, Findbugs[20] - looks for bugs in compiled Java code, PMD[21] - looks for potential problems in Java source code, to name a few) few projects go beyond unit testing [5] in their CI process.
Hints: There have been some attempts to integrate cppcheck and the clang static analyzer into a Travis-CI continuous integration process for C++ projects https://legalizeadulthood.wordpress.com/2014/12/07/adding-static-analysis-to-your-c-github-repository/; Coverity Scan and Travis-CI for Java, C, C++, and C# projects https://scan.coverity.com/travis_ci; R CMD check and Travis-CI for R packages on GitHub http://r-pkgs.had.co.nz/check.html#travis; similarly for Ruby projects using Jenkins http://codingfearlessly.com/2014/11/06/continuous-static-analysis/

- A tool for supporting the evolution of cloned code. The blocks of code cloned from a project to another one create implicit dependencies between projects that are typically not managed. However, changes applied on the original code should be taken into account by the developers of the cloning project, since they may reveal a bugfix or any other improvement that should be replicated in the other copies. A tool that detects clones shared by a client project and some external projects, that tracks changes that occurred in clones of the external projects, and that offers an automatic, fast, and easy approach (e.g., by submitting a pull request to the project's source code repository) for applying similar changes to the client project would help the developers to manage their cloned code. This idea can be adapted to the support of any implicit and asymmetric relation between projects.

You can also look at the winners of the GitHub Data Challenge:

- 2014: https://github.com/blog/1892-third-annual-data-challenge-winners
- 2013: https://github.com/blog/1544-data-challenge-ii-results
- 2012: https://github.com/blog/1162-github-data-challenge-winners

[19] http://checkstyle.sourceforge.net
[20] http://findbugs.sourceforge.net
[21] http://pmd.sourceforge.net

References

[1] L. Osterweil. Software processes are software too. In ICSE, 1987
[2] B. Phifer. Next-generation process integration: CMMI and ITIL do devops. Cutter IT Journal, 24(8):28, 2011
[3] A. Capiluppi, A. Serebrenik, and L. Singer. Assessing technical candidates on the social web. Software, IEEE, vol. 30, no. 1, pp. 45–51, Jan 2013
[4] C. Hauff and G. Gousios. Matching GitHub developer profiles to job advertisements. MSR 2015 http://gousios.gr/pub/dev-profiles.pdf
[5] G. Gousios, A. Zaidman, M.-A. Storey, and A. van Deursen. Work Practices and Challenges in Pull-Based Development: The Integrator's Perspective. In ICSE, 2015 http://www.gousios.gr/pub/pullreqs-integrators.pdf