チームで行うコーディング作業を容易にする方法
原題：Making it easier to collaborate on code

著者：Adam Conner-Simons, 訳者：平木場 風太

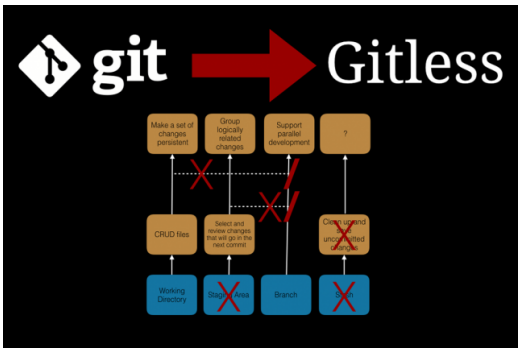
原文公開日：October 26, 2016, 翻訳日：February 7, 2017

はじめに

この文書は，Pyhs.org という英語のニュース・コラムサイトにある、“Making it easier to collaborate on code”という記事を和訳したものだ。git について調べていたところ Gitless という git を簡略化する手法を知り、日本語で説明しているサイトが無かったため、本記事を和訳することにした。また、各ページの下部にある脚注は訳者が付け加えたものである。

1　概要

図 1　Gitless のイメージ



「Gitless」は、git の核となる機能を根本的に変えずに、「staging」や「stashing」といった複雑なコンセプトを取り除く。
– Gitless 開発者の 1 人である Santiago Perez De Rosso 氏の言葉

git はオープンソースのシステムであり、プログラマの間では偏極的な評判がある。開発者がコードの変更を追跡できるようにする強力なツールだが、多くの人が使用するのが非常に難しいと考えている。そこで、より使いやすいように MIT のコンピュータ科学・人工知能研究所（CSAIL[1]）のチームは、システムの核となる部分を変更することなく根本的な問題の多くを修正するインターフェイス「Gitless」を開発した。

「我々が開発した Gitless というツールは、習得と使用が簡単にできる。それでも git の人気の核となる要素は残している」と Santiago Perez De Rosso 氏は述べている。彼は MIT の Daniel Jackson 教授と関連論文を共著した大学院生だ。「このツールを特に奨励する理由は、Dropbox や Google Inbox などの他のシステムがユーザビリティを向上させるために同じアプローチを使用する可能性があるからだ」

Gitless の一部分は、有名なプログラミングサイト StackOverflow[2]の約 2,400 件の git 関連の質問を調べることによって、開発された。チームは、「staging」と「stashing」という概念を含む、git の最大の問題のいくつかをピックアップし、それらの問題を最小限に抑えることを目指した変更を提案した。

Gitless は git 上に実装されているから、Gitless の利用者はコードを一方から他方へ移行することな

[1] MIT Computer Science and Artificial Intelligence Laboratory
[2] 有名なプログラミングに関する Q&A サイト。日本語サイトもある。

く、Gitless と git を簡単に切り替えることができる。さらに、利用者が git の大ファンではないことは利用者の Collaborator[3]達にとって問題にはならない[4]。

Perez De Rosso 氏は、来月[5]にアムステルダムで開かれる「システム、プログラミング、言語とアプリケーション：人類のためのソフトウェア」に関する ACM SIGPLAN 会議でこの論文を発表する予定だ。

2 Gitless の特徴

git は「分散型バージョン管理システム」と呼ばれるものである。 これは、複数のプログラマーが、コードの変更を追跡することを可能にする。ユーザは変更を行い、誰もが何をしたのかを誰もが知るように保存（またはコミット）する。あなたと同僚がファイルのバージョン 10 を使用していて、新しいものを試したい場合、あなたの友人が ”master”で働いている間に、別の ”branch”を作成することができる。理にかなっているだろう。しかし、物事はすぐに混乱する。

2.1 staging を取り除く

Gitless の 1 つの機能は、ファイルの特定の部分だけを保存できる「staging」を排除することだ。たとえば、変更が完了したファイルと変更が完了していないファイルがあり、変更が完了したファイルをコミットしたい時、「staging」を利用すると、作業中の他のファイルを維持しながら変更が完了したファイルのみをコミットできる。ただし、staging されたバージョンと作業中のバージョンの両方を持つファイルを作成すると、厄介な状況が発生する。ファイルを staging してから変更を加え、さらにコミットした場合、コミットしているバージョンは、あなたが今作業しているものではなく、前に staging したものである。

Gitless は本質的に staging 領域を隠すことで、プロセスをはるかに明瞭にしユーザーにとっての複雑さを軽減する。その代わり、はるかに柔軟な ”commit”コマンドが存在し、コミットするコードのセグメントの選択などを行うことができる。

2.2 stashing を取り除く

Gitless が取り除くもう一つのコンセプトは、「stashing」だ。プロジェクトの途中で、別のブランチに切り替える必要があるが、まだ半分の作業をコミットしたくないとする。 stashing は、行った変更を、後で復元できる”未完了の変更のスタック”に保存することだ。 （stashing と staging の主な違いは、stashing で変更が作業ディレクトリから消えることだ）

「問題は、ブランチを切り替えるときに、どこに隠し場所があるのかを覚えにくいことだ」と、Perez De Rosso 氏は言う。「ファイルの競合を含むマージのようなアクションの途中にいる場合、stashing は役に立たない」

Gitless は、ブランチを完全に独立させることで、この問題を解決できる。これにより、タスクを常に切り替える必要のある開発者にとって、これ[6]はずっと簡単になり、混乱も少なくなる。

3 Gitless の評価

Gitless は確かに git を改善するための最初の努力[7]ではない。しかし、このプロジェクト[8]に参加していないサンディエゴにあるカリフォルニア大学の認知科学の助教授である Philip Guo 氏によれば、これは git のインターフェースを超え、実際にはコアの概念上の問題を扱う最初のものだという。「この作業には、世界で最も広く使用されているソフト

[3] 協力者という意味。実際、共同で開発を行う人は Collaborator と呼ばれている。

[4] なぜ問題にならないかというと、Gitless は git の操作を簡略化するツールであり、ローカルリポジトリ内の.git にしか影響を及ぼさないため。他の collaborator に不利益は生じない。

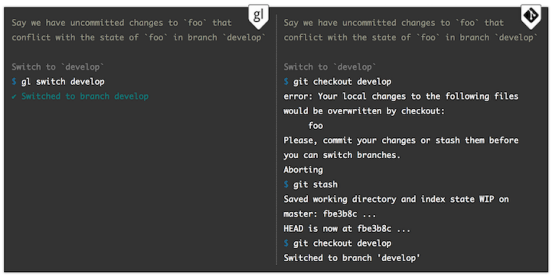
[5] 2016 年 10 月 30 日から 11 月 4 日にかけて開催された

[6] プロジェクトの途中で、別のブランチに切り替える必要があるが、まだ半分の作業をコミットしたくない場合

[7] git-flow や git クライアントなどの git を拡張したり使いやすくしたりするツールのことだと思われる。

[8] Gitless の開発のこと

図 2　Gitless 対 git



> Gitless（左）は、git に存在するより複雑な概念の多くを単純化します。
> – Santiago Perez De Rosso 氏の言葉

ウェアの欠点を明らかにするために、厳密なソフトウェア設計の研究手法を適用している」と Guo 氏は言う。「過去には、多くの実務家が git に対して賛否両論の逸話的な議論を行ってきたが、先の研究はそれらの議論を解く科学的アプローチをとっていない」

また、git に対する Gitless の効果をテストするユーザー調査も行った。Gitless ユーザーは git ユーザーよりもタスクを完了するのに成功していることがわかった。少なくとも 1 つのタスクでは、これを大幅に迅速に実行した。(Perez De Rosso 氏は、この研究の参加者はすべて git に精通していると指摘し、git の経験のない人で構成されているチームに Gitless をテストした場合、その結果がさらに顕著になるかもしれないことを示唆している)

タスク後の調査では、参加者は Gitless のブランチ間の移行能力に特に感銘を受け、「非常にスムーズ」と「より直感的な方法」と述べた。Guo 氏は初心者のプログラマーが git を使い始めるのを助けるために、Gitless を ”トレーニングホイール[9]”の貴重な形として説明している。より高いレベルでは、チームのフレームワークは他のシステムを見るための重要なツールになると彼は言う。

Perez De Rosso 氏は、Dropbox の「共有フォルダ」の概念だけでなく、Google Inbox のまとまった「会話[10]」の概念を分析できる可能性に特に興味があると述べている。

「おそらく、この研究のもっとも良い貢献は、git 自体の分析ではなく、作成者が人気のあるソフトウェアの分析、解読、再設計に使用した方法論である」と Guo 氏は言う。「設計上の欠陥を分析するための作成者のアプローチの力は、多くの種類の一般的なソフトウェアに適用できる」

4　おわりに（訳者）

Gitless とは何か気になって、この記事を和訳したが、訳してみると思ったよりも簡単な説明しかなく、著名人の台詞が多いと感じた。よく考えてみると、原文がニュースサイトに掲載されていた文章なので当たり前のことかもしれない。次は、開発者の Perez De Rosso 氏の論文 [2] を和訳したい。

参考文献

[1] Researchers tackle issues surrounding security tools for software developers,
https://phys.org/news/2015-08-tackle-issues-tools-software.html

[2] Santiago Perez De Rosso Daniel Jackson Purposes, concepts, misfits, and a redesign of git. November 2016
DOI: 10.1145/2983990.2984018
http://people.csail.mit.edu/sperezde/pre-print-oopsla16.pdf

[3] Towards a theory of conceptual design for software. October 2015
DOI: 10.1145/2814228.2814248
http://dl.acm.org/citation.cfm?id=2814248

[9] 補助輪という意味。

[10] 恐らくメールのやりとりを連ねたスレッドのこと。