ソフトウェア開発者にフィードバックを与えるチャットボットの実現に向けて

平尾 俊貴[1,a)]　伊原 彰紀[1,b)]　上田 裕己[1,c)]　桂川 大輝[1,d)]　Dong Wang[1,e)]　松本 健一[1,f)]

概要：近年，ソフトウェア開発プロセスの効率化を目的として，Bamboo や Jenkins などの継続的インテグレーション (Contenious Integration) システムが，ソフトウェア開発現場に導入されている．CI システムは，ソフトウェア開発者が設定したビルド・テストを実行して，フィードバックする．しかし CI システムは，ソフトウェア開発者が事前に設定した内容以外の問い合わせに対して，フィードバックすることは難しい．本研究では，予め設定された内容だけではなく，開発者が求める様々な問い合わせに対して，フィードバックするチャットボットの実現を目指している．本論文では，開発中の４つのチャットボットを紹介する．ワークショップでは，チャットボットの活用方法，及び，従来のソフトウェア開発における支援技術と比べた，チャットボットの期待される効果について議論したい．

1. はじめに

近年，ソフトウェア開発の効率化を目的として，Bamboo [*1] や Jenkins [*2] などの継続的インテグレーション (Contenious Integration) システムが，多くのソフトウェア開発に導入されている [1][2]．CI システムは，ソースコードのビルドやテストを継続的に実行する手法であり，ソフトウェアの潜在的問題を早期に検出が可能となる．CI システムにより，ソフトウェア品質の検証時間の削減，ソフトウェア開発の生産性の向上が期待される．

CI システムでは，ビルドやテストの内容を開発者が設定する．設定された実行内容に従って，CI システムが自動的にビルドやテストを実行して，そのフィードバックをソフトウェア開発者に与える．しかし，CI システムは，開発者が設定した内容以外のフィードバックを返すことができない．例えば，バグ修正のためにソースコードを変更した場合，開発者が事前に設定したテスト内容を実行するが，設定した内容以外のフィードバック（バグ修正方法の妥当性，他の修正方法の提案など）を返すことができない．

本研究で開発するチャットボットでは，ソフトウェア開発者間の議論に参加して，ソフトウェア開発者の様々な問い合わせに対して，リアルタイムにフードバックを返すことを目指す．具体的には，我々の研究チームで開発中の 4 つのチャットボット，「レビューコストの自動見積りボット」，「修正指摘内容の自動推薦ボット」「ソフトウェアライブラリの自動推薦ボット」，「重複レビューの自動検出ボット」を紹介する．

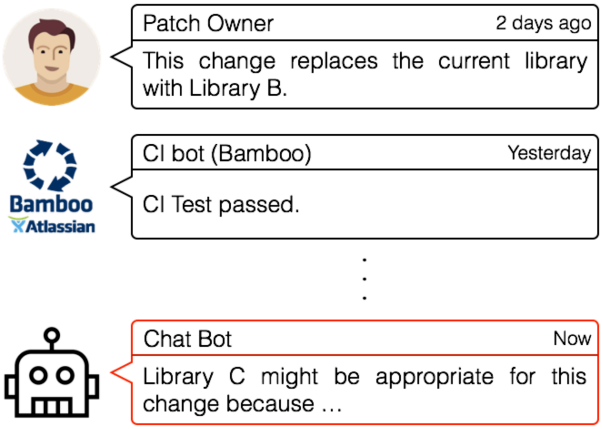

図 1　チャットボットと CI システムの実行例

本論文では，２章でチャットボットのシナリオを説明して，３章で開発予定及び開発中の４つのチャットボットを説明する．最後に，４章で本研究内容のまとめを述べる．

2. チャットボットのシナリオ

近年のソフトウェア開発プロジェクトでは，主に分散管理システム GitHub を利用して，ソフトウェア開発履歴を

[1] 奈良先端科学技術大学院大学
8916-5, Takayamacho, Ikoma, Nara, 630–0192, Japan
[a)] hirao.toshiki.ho7@is.naist.jp
[b)] akinori-i@is.naist.jp
[c)] ueda.yuki.un7@is.naist.jp
[d)] katsuragawa.daiki.ka7@is.naist.jp
[e)] wang.dong.vt8@is.naist.jp
[f)] matumoto@is.naist.jp
[*1] https://ja.atlassian.com/software/bamboo
[*2] https://jenkins.io/

© 2012 Information Processing Society of Japan

管理している [3]．GihHub 上で，ソフトウェア開発者は，ソフトウェア品質の改良を目的としてソースコードを改変する．そして，その改変されたソースコード群（パッチ）をプルリクエトとして GitHub 上に投稿する．我々が開発するチャットボットは，GitHub における開発者のアクション（プルリクエスト，イシュー報告）に反応するものである．

図 1 は，プルリクエストを投稿したパッチ作成者（Patch Owner）に対する，CI システムとチャットボットの実行例を示す．はじめに，Patch Owner が GitHub 上にプルリクエストを投稿した．当該パッチは，現在使用しているライブラリを変更していることを述べている．その変更内容に対して，CI ボットが事前に設定された内容に従って，自動テストを実行した．その後，チャットボットがライブラリの変更に反応し，他のライブラリを推薦している．図 1 のように，我々が目指すチャットボットは，ソフトウェア開発者が CI システムに設定されている動作検証に限らず，他プロジェクトの比較内容（例えば，最適なライブラリの推薦）などをソフトウェア開発者にフィードバックする．

3. チャット Bot 開発内容

我々が現在開発に取り組んでいる４つのチャットボットの開発内容を紹介する．

3.1 コードレビュー時間の自動見積ボット

コードレビューは，ソフトウェア開発に要する時間の約 50%を占めており，膨大な時間的コストを要することがわかっている [4]．しかし，従来の CI システムでは，レビューに要する時間的コストについてリアルタイムにフィードバックを与える機能は備わっていない．我々が提案するチャットボットでは，投稿されたプルリクエストに対して，時間的コストの見積もり結果を通知することで，膨大な時間的コストを要する変更に対するレビューを防ぐことが期待される．

3.2 修正指摘内容の自動推薦ボット

CI システムを利用することで，ソースコードの妥当性を検証することができる．しかし，大規模なソフトウェア開発プロジェクトでは，検証する項目が多いことから，膨大な時間的コストを要する．本開発では，パッチに含まれるソースコードを分析し，未完成のソースコードに対して修正すべき内容を推薦するチャットボットを開発する．本チャットボットは，開発者の開発能力に合わせたフィードバックを出力することで，コード理解の補助が期待される．

3.3 ライブラリの自動推薦ボット

ソフトウェア開発の生産性の向上を目的として，開発者はソフトウェアライブラリを活用する．既存の機能を再利用するため，開発時間を削減できる一方で，最適なライブラリを選択することが難しい．当該ボットでは，他の開発プロジェクトのライブラリ利用履歴に基づき，最適なライブラリを自動推薦する．最適なライブラリを自動的に推薦することで，開発者の負担を軽減することができる．

3.4 重複したプルリクエストの自動検出ボット

日々大量のプルリクエストが GitHub に投稿されるため，同じ変更内容を持つ重複したプルリクエストが投稿される可能性がある．当該ボットでは，プルリクエストのタイトル・説明文・検証者からのコメントから，特徴的なキーワードを抽出して，重複したプルリクエストを検出する．重複したプルリクエストが投稿されたときに，当該チャットボットがパッチ投稿者に警告することで，不要なレビューを防ぐことができる．

4. おわりに

本論文では，現在，我々の研究チームで開発予定及び開発中の４つのチャットボット「レビューコストの自動見積りボット」,「修正指摘内容の自動推薦ボット」「ソフトウェアライブラリの自動推薦ボット」,「重複レビューの自動検出ボット」を紹介した．我々は，本研究を進めることで，今後のソフトウェア工学分野に貢献できると考えている．

近年のソフトウェア工学分野では，ソフトウェア開発プロセスの一部を自動化する研究が数多く発表されている．その一方で，その研究成果を実際の開発現場で活用するプラットフォームが用意されていない．本研究で提案するチャットボットに，研究成果を追加機能として実装できるプラットフォームを提供することで，研究成果をソフトウェア開発現場に素早く適用できると期待する．

今後は，これまでの CI システムによる限られた内容のテストにチャットボットを加えていき，プロジェクトや開発者個人に沿った最適なフィードバックを提供できるシステムの実現を目指す．

参考文献

[1] Daniel Ståhl, Jan Bosch, “Industry Application of Continuous Integration Modelling: A Multiple-Case Study, ” Proc. of the 38th IEEE International Conference on Software Engineering Companion (ICSE), pp.270-279, 2016.

[2] Tony Savor, Mitchell Douglas, Michael Gentili, Laurie Williams, Kent Beck, Michael Stumm, “Continuous Deployment at Facebook and OANDA, ” Proc. of the 38th IEEE International Conference on Software Engineering Companion (ICSE), pp.21-30, 2016.

[3] Georgios Gousios, Martin Pinzger, Arie van Deursen, “An exploratory study of the pull-based software development model,” Proc. of the 36th International Conference on Software Engineering, pp.345-355, 2014.

[4] David S. Alberts, “The Economics of Software Quality Assurance” , Proc. of the National Computer Conference and Exposition,” pp.433-442, 1976,

© 2012 Information Processing Society of Japan