開発ツール連携のための ProxyChatBot フレームワーク

高木 豪[1,a)]　小林 隆志

概要：本稿では，既存の開発支援ツールを ChatOps に導入するためのモデル及びツールフレームワークを提案する．既存の開発支援ツールは出力が膨大である場合や，開発者の立場に応じて必要な出力情報が異なる場合がある．そのため，そのまま ChatOps に導入することが難しい開発支援ツールが存在する．提案モデルでは，既存の ChatOps のモデルを拡張し，ChatBot とチャットツールの間を ProxyChatBot を経由することで，開発支援ツールの出力を柔軟に加工してチャットツールに通知する仕組みを確立する．モデルに基づいて実装したツールフレームワーク ProxyChatBot を用いて，既存の静的検査ツールの除外手法を再実装する実験を通し，モデルの実現可能性及び有用性を議論する．

1. はじめに

生産性と成果物の品質は開発者が常に意識するものであり，ソフトウェア開発では高品質なソフトウェアを効率良く開発するために開発支援ツールが利用される．しかし，ソフトウェアの大規模化や分散開発が進むほど開発に用いるツールの数は増加する．その結果，開発者はツール全てを把握することに時間を割き，生産性の低下につながってしまう [1]．この問題を解決するために，複雑で大規模なツールを用いた開発プロセスの一部を自動化するツールとして Bot が利用されている．一般にルールベースの Bot は繰り返し処理や事前定義されたタスクを自動化するアプリケーションとしてみなされる．ソフトウェア開発の分野では，Bot は開発者が過去に行ってきた煩雑なタスクを自動化する．煩雑なタスクを Bot が開発者に代わって行うことで，開発者は自身が行っている作業を中断せずに集中して行うことができる．

ChatOps は，Bot を用いてチャットツール上で会話主導の開発を行う方法である．チャットツールは本来開発者間のコミュニケーションツールとして利用されている．チャットツール内で Bot はユーザーの一人としてチャットに参加し，ユーザーと対話する．Bot は開発支援ツールと接続することで，チャットツール上で開発者から送られるコマンドを受け取ってツールを実行し，結果を通知する．チャットツールは開発チームにとって擬似的な開発者間の共有コンソールを実現し，開発フローの透過性を高めることができる．

ChatOps はチーム間の情報共有手段としても有用である．ChatOps では Bot が開発支援ツールから得られた情報を集約し，一つのチャット上のタイムラインに通知する．開発者はチャットツールのタイムラインを監視するだけでツールを利用することと同等の情報を取得できる．このように，ChatOps は開発を通して行う様々な情報の出力先をチャットツールのタイムラインに一元化することで，開発者の迅速な意思決定及びチーム間の情報共有を適切に行える点で非常に有用である．

しかし，現実的には ChatOps 化することが難しいツールが存在する．それは開発支援ツールの出力が膨大である場合や，開発者の立場に応じて不必要な情報を含んでいるようなケースである．ChatOps は，開発支援ツールからの情報提示の即時性及び一元化の観点で優れる一方で，多量の情報が提示される場合，開発者の利用意欲の低下や必要な情報の見逃しの原因となりうる．そのため，これらの開発支援ツールを ChatOps に導入するには開発者にとって必要なデータを抽出しなければいけない．しかし，開発支援ツールに対する運用方法は開発者やプロジェクトに強く依存するため，出力を制御する方法は一様ではない．これらのツールを ChatOps 化するためには，ツールの出力の制御をツールをまたいで柔軟に行える仕組みが必要である．

本稿では，既存の開発支援ツールの ChatOps 化を支援するモデルツールを提案する．本モデルは既存の ChatOps のモデルを拡張し，ChatBot とチャットツールの間で ProxyChatBot を経由することで開発支援ツールの出力にデータ

[1] 東京工業大学 情報理工学院
School of Computing, Tokyo Institute of Technology
[a)] gtakagi@sa.cs.titech.ac.jp

加工をしてチャットツールに通知する仕組みを確立する．更にツールとして実際に ProxyChatBot を実装し，ProxyChatBot 上で静的検査ツールの除外手法を再実装することでモデルの実現可能性及び有用性を議論する．

2. ChatOps

2.1 DevOps と ChatOps

ソフトウェア開発を効率的に行うには，継続的かつ高速な運用改善を行ってプロダクトの成長速度を向上させることが必要である．中でも企画，運用，開発の 3 つのプロセスの垣根を取り払っていくために，ツールや組織の文化を最適化していくことを DevOps と呼ぶ．開発者は DevOps ツールを複数同時に利用し，開発を行う．しかし，ソフトウェアの大規模化や分散化が進むに連れて用いる DevOps ツールの数も増加する．その結果，開発者はツール全てを把握することに時間を割き，生産性の低下につながってしまう．

この問題を解決するために，複雑で大規模な開発プロセスの一部を自動化するツールとして Bot が利用されている．ChatOps は，Bot を用いてチャットツール上で会話主導の開発を行う GitHub が提唱した方法である．DevOps ツールで行う冗長なタスクを自動化し，また開発者間の知識量の差や必要なコミュニケーションを埋め合わせるサポートを Bot が行う．また，ここで使用される Bot は特に ChatBot と呼ばれる．

2.2 ChatBot に関する関連研究

既存の ChatOps における ChatBot の役割は，チャットを通してユーザーのコマンドを受け，開発支援ツールの出力をチャットに通知することである．

Storey らは，まだ新興の概念である ChatOps がソフトウェア開発に与える影響を明らかにするために，ChatOps がソフトウェア開発の生産性に与える影響を測るための枠組みを提案している [2]．更に，生産性を効率性と有効性の観点で区別した上で，ChatOps は煩雑なタスクの自動化や情報の集約・通知を行うことで開発者が行っている作業のフローの状態を長く維持する点で効率性を向上し，開発チームの意思決定・状況認識を促進させる点で有効性を向上させることを主張している．

Calefato らは，分散開発における開発支援ツールの多様化や情報の断片化の処理に役立つ開発者全体の状況認識を向上させるためのツールの統合モデルを提案している [3]．提案している統合モデルにおいても，ChatOps は開発全体を通して開発と運用に共有責任が生まれてチーム全体の意識を向上させることに役立つと主張されており，対話主導型の開発を行うことを推奨している．

Lin らは，チーム向けのコミュニケーションプラットフォームとしてソフトウェア開発チームが幅広く採用している Slack に対して，Slack と他のツールと Slack を連携する Bot がソフトウェア開発に及ぼす影響を理解するために，開発チームがどのように Slack を活用するのか予備的な調査を行っている [4]．開発者にとって，Slack が個人及びチーム全体で利用法がそれぞれあることを理解し，その中で Bot を用いてコードホスティング，アプリケーションのモニタリング，デプロイステータス，VCS や issue トラッキングを自動化することで，開発とデプロイのサポートに利用していることを確認した．

このように，DevOps や ChatOps に向けた ChatBot に関する研究はいくつか存在するが，著者らが知る限り，流れてくるデータを加工する役割を持った ChatBot は存在していない．

既存の ChatOps のモデルは，マイクロサービスアーキテクチャに基づいてサービスとして提供されている個々の支援ツールとのやり取りを Bot が代行して行うことで自動化等の支援を行う．サービスとして開発支援手法や支援手法の基盤となる情報を提供する研究にまつ本らが提唱する Service-Oriented MSR(SO-MSR) [5] がある．

まつ本らは，Mining Software Repository(MSR) と総称される，ソフトウェアの開発履歴等が蓄えられたリポジトリをマイニングする技術を，ネットワーク上のサービスとして実現させるためのフレームワークとして SO-MSR を提案している．更に MSR の代表的な手法の一つであるソースコードメトリクスの計算に着目し，SO-MSR のフレームワークに従った Web サービス，MetricsWebAPI の開発 [6] や MSR サービスのインタラクション改善を目的として MSR キャッシュ機構を導入している [7]．サービス指向アーキテクチャ (SOA) の考えに従い MSR をネットワーク上のサービスにすることで，既存の MSR を用いた開発支援ツールのサービス化を推進することができる．

このように，開発支援ツールだけでなく，その基盤技術に関するサービス化は提案されてきたが，現時点ではそれらを用いた ChatOps に至る研究は著者らが知る限り存在していない．

2.3 開発支援ツールの ChatOps 化の課題

ChatOps は開発支援ツールからの情報提示の即時性及び一元化の観点で優れる一方で，多量の情報が提示される場合，開発者の利用意欲の低下や必要な情報の見逃しの原因につながる．そのため，出力が膨大である場合や，開発者の立場に応じて不必要な情報が多い開発支援ツールを ChatOps にするには適していない．出力が膨大となるツールの例として TravisCI といった CI ツールがあげられる．図 1 のように，テストの可否のみを通知し，詳細は別のページにリンクする形となっている．

さらに，開発支援ツールに対する運用方法は開発者やプロジェクトに強く依存し，必ずしも一様なものではない．

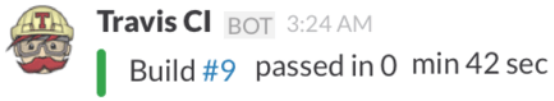

図 1　TravisCI の ChatOps 例

ChatOps では複数の開発支援ツール処理結果を，多様なステークホルダが受け取る．この際に，受け取るステークホルダによって必要な情報は異なるため，柔軟に出力の制御が必要となる．プロジェクトによっては複数のツールの出力に応じて出力の制御を変化させるケースも存在する．複数のツールの出力をまとめて，データの加工や出力の制御を行えるシステムが必要である．

既存の ChatBot 上でツールからの出力を加工する方法では，導入の際に ChatBot を理解し改変するといったコストが発生する．また，出力部を加工する方法が異なる複数の ChatBot を構成することは開発支援ツールと連携した ChatBot としての一般性を損なうことになる．このため，既存の ChatBot の構成を変えずに開発者やプロジェクトに合わせてデータの加工を行う手段を用意する必要がある．

3. 提案手法

3.1 提案モデル

開発者やプロジェクトに依存した開発支援ツールの出力を柔軟に調節するためには，以下の機能が必要である．

- 開発支援ツールの出力から必要な情報を抽出
- プロジェクト，ユーザーの状態に応じた出力の制御
- 複数の開発支援ツールの出力を受け取ってデータ加工

本研究では，これらを実現するための新たな ChatBot を定義する．

提案モデルを図 2 に示す．マイクロサービスアーキテクチャの設計に沿って既存の ChatOps モデルを立てると左の図のようになる．既存の ChatOps は ChatBot がサービスとして立っている開発支援ツールとチャットツールに対してそれぞれ双方向に通信する．ChatBot はチャットツールから送られたコマンドを受けて開発支援ツールと通信してツールを実行する．最後に，ツールから得られた出力をチャットツールに通知する．

提案モデルでは，ChatBot とチャットツールの通信を仲介する別の ChatBot を導入する．通信を仲介する ChatBot をここでは ProxyChatBot と呼ぶ．ProxyChatBot は開発支援ツールと接続した ChatBot とチャットツールの通信を仲介し，データ加工を行う．この提案モデルは，ProxyChatBot 内でデータ加工を行うことで既存の ChatBot の構成を崩さない．そのため，既存の ChatOps モデルとの切り替えを容易に行えるため，コストを低く導入できる．

開発者の部署やプロジェクトに応じて柔軟にデータの抽出を行うために，開発支援ツールと連携した ChatBot と中継する ProxyChatBot を，開発者やプロジェクトに応じて変化させる．図 3 のように，ProxyChatBot 内で受け取るユーザーに合わせたデータ加工を行う．

ProxyChatBot は既存の ChatBot から得られた出力に対してデータ加工を行い，ChatBot として代わりにチャットツールに対して通知を行う．開発者は ChatBot から結果を受け取ることに変化はないので，既存の ChatOps と差異なく ProxyChatBot を経由した ChatOps を利用することができる．更に図 2 の通り，ProxyChatBot は ProxyChatBot 同士で通信を行って，複数の ProxyChatBot を仲介してチャットツールに通知することも可能である．これによってシンプルなデータ加工を繰り返して大きなデータ加工を実現する．

開発者の状態に合わせてツールからの出力の通知を制御するために，出力結果やユーザーの状態を ProxyChatBot 内で保持することで，ProxyChatBot を中継する際に状態に応じた出力も実現させる．これによって Storey ら [2] が主張した ChatBot を用いて開発者の生産性の向上を実現させる．

3.2 Chat as Stream

開発支援ツールから得られる出力はツールの実行結果を通知するものであるため，特定の単位で区切ることができるメッセージの集まりであると考えられる．このことから，開発支援ツールの出力を加工することは，個々のメッセージに対して加工処理を行うこととかんがえられる．ProxyChatBot は，ChatBot から出力される開発支援ツールのデータを ProxyChatBot 内でストリームに変換して，ストリームを流れる個々のメッセージに対する加工処理として記述することでデータ加工を行う．個々のメッセージに対する処理としてフィルタリングや型変換，集約などの高階関数の適用として表すことで宣言的な記述が可能になる．

開発者やプロジェクトに依存した開発支援ツールの出力を柔軟に調節するためには，複数のツールの出力をまとめてデータの加工や出力の制御を行えるシステムが必要である．ProxyChatBot では図 2 にある通り複数の開発支援ツールと接続した ChatBot とのやり取りをそれぞれストリームに変換し，ストリームの分岐・合成を行うことができる．これによって複数の開発支援ツールから送られてくるメッセージを統合した出力や，開発者・プロジェクトごとにストリームを分岐させて各々必要なデータ加工を行って通知することが可能となる．

4. ProxyChatBot フレームワークの実装

提案モデルの実現を支援するために，本研究では ProxyChatBot フレームワークを実装する．開発者が ChatBot に対してコマンドを送るためのユーザーインターフェー

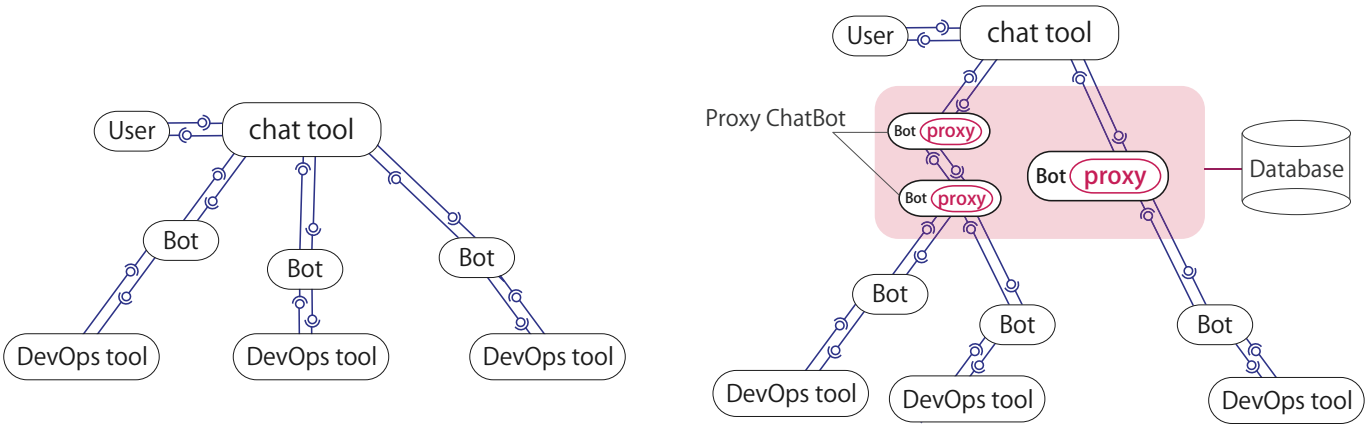

図 2　従来の ChatOps モデルと ProxyChatBot を介した ChatOps モデル

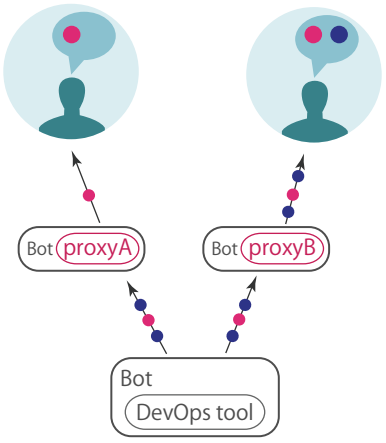

図 3　ユーザーに応じた出力の実現

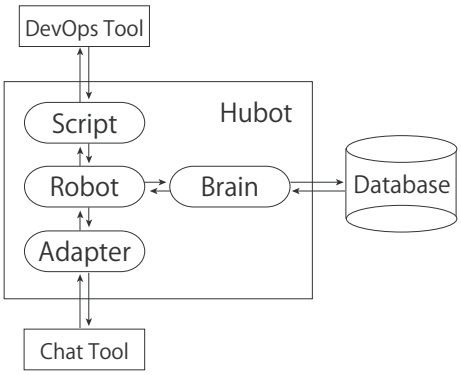

図 4　通常の Hubot を用いた ChatOps の実装

スにあたるチャットツールに今回は Slack を使用する．ChatBot には Github が提供している Hubot を用いる．実装では ChatBot は Hubot，チャットツールは Slack に限定している．しかしこれらに限らず，既存の開発支援ツールと連携してチャットツールに通知する ChatBot に対して本モデルは適用可能である．

ChatBot は Web サーバーと異なり，能動的に発話するため対話対象を指定しなければいけない．図 4 の通り，通常 Hubot の場合は Adapter クラスを継承した各チャットツールの Adapter を実装して各種チャットツールと接続する．

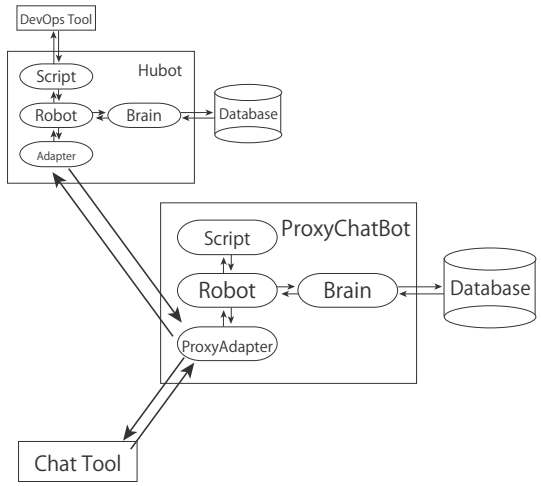

図 5　Hubot Adapter を拡張した ProxyChatBot フレームワーク

これは Adapter というインターフェースを通してチャットツールに寄らない汎用的な応答処理を Hubot 上で記述することを可能にしている．提案モデルの実装にあたって，Hubot の Adapter を拡張した hubot-proxy Adapter を実装することで，ProxyChatBot を実現するためのフレームワークを実装した[*1]．この Adapter は Hubot 2.0 以上で動作確認している．

hubot-proxy Adapter を用いたものが図 5 である．ProxyChatBot は発話対象をチャットツール，ChatBot と少なくとも 2 つ以上持つ．そのため ProxyChatBot の Adapter はチャットツールからの出力に対して任意のデータ加工を行い ChatBot への通知，及び ChatBot の出力に対して任意のデータ加工を行いチャットツールへの通知を行う必要がある．実装では，ChatBot からチャットツールへの出力のデータ加工に限定し，ProxyChatBot 内の hubot-script で，開発支援ツールからの出力を受けとり，一定の規則で区切られたメッセージが流れるストリームに変換してデータ加工を行う．

前述したとおり，ChatBot も能動的に発話するため対話

[*1] hubot-proxy Adapter (`https://github.com/shimastripe/hubot-proxy`)

対象を指定しなければいけない．実装では，ProxyChatBot に対して発話するための Adapter も用意する．

3.2 で議論したように，ProxyChatBot 内での柔軟なデータ加工，複数の ChatBot との接続を行うために，ProxyChatBot 内をストリームに変換して，ストリーム中の個々のメッセージに対する加工処理として記述することが必要である．実装では ReactiveX を使用してストリームに変換した．ReacriveX を用いることで，開発支援ツールから流れてくるデータに対して関数型インターフェースを受け取るメソッドを使って加工することができる．また，ReactiveX は複数のストリームに対する処理も柔軟に記述できるため，複数の ChatBot と接続してデータ加工を行うことも可能である．

5. 評価

提案モデル及びモデルに基づいたフレームワークの評価を行うために，既存の静的検査ツールの出力を制御する渥美らのツール MAFP [8] を ProxyChatBot 上で再実装した[*2]．

5.1 MAFP とその実現方法

MAFP は桑原らによるソースコードの検査における警告の版間追跡手法 [9] を実現したものである．MAFP は静的検査ツールが報告するツールを版間追跡し，過去の検査において開発者が確認した警告と同じ警告を将来の検査において報告しないツールである．版間履歴として Git の blame 機能を用いて，報告された警告内容を確認済みとして記録する際に，報告された警告内容と，該当する行が変更された直近のコミットと，その時のファイル名と行番号も合わせて記録しておく．あるコミットに対して静的検査ツールが報告する警告について，警告内容及び報告された行が変更された直近のコミット，ファイル名，行番号の全てが等しい警告が確認済みと記録されていれば除外する．

今回は Java 言語を対象として，静的検査ツールには Checkstyle を利用する．Checkstyle は Hubot の子プロセス上で実行して出力を受け取った．Checkstyle の設定にはパッケージに標準で含まれている google_checks.xml を利用している．開発者が確認した警告の記録には mongoDB を利用する．Git のリポジトリを取得するために，Node.js 向けの API である NodeGit を用いる．NodeGit を用いて Github からリポジトリデータを取得する．

5.2 CheckStyle と連携した ChatBot

CheckStyle を ChatOps 化するにあたって，現状 CheckStyle はサービスとして存在しない．そのため，本実装では NodeGit を用いてリポジトリを clone し，その上で clone したリポジトリに対して，hubot-script 上で Node.js の子プロセスとして CheckStyle を実行し，得られた検査結果を発話する．この hubot-script は coffeescript で記述し，72 行で記述することができた．

[*2] mafp-proxy(`https://github.com/shimastripe/mafp-proxy`)

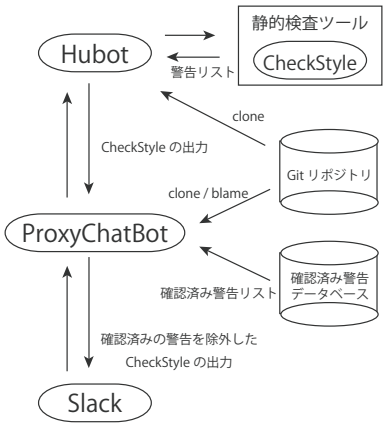

図 6 実装構成

5.3 MAFP を再実装した ProxyChatBot

ProxyChatBot では ChatBot から送られてきた CheckStyle の検査結果を hubot-script 上で ReactiveX を用いてストリームに変換し，データ加工を行う．Hubot にはデータを永続化する機能として Brain というモジュールが存在するが，今回は Brain を用いずに hubot-script 上で直接 mongoDB と接続している．ProxyChatBot 側でも対象のリポジトリを clone し，CheckStyle が警告している行に対して git blame を実行して版間履歴を取得する．これらを用いて，対象の警告が確認済み警告リストに登録済みでない警告のみを集約し，チャットツールへと応答する．この hubot-script は coffeescript で記述し，145 行で記述することができた．

図 6 が実際の実装の構成である．ProxyChatBot 上では，事前にデータベースに確認済みの警告を登録した上で，ユーザーへ Checkstyle の結果を送信するときに該当する警告は除外して通知するように実装した．

実装の結果，図 7 のように確かに確認済みの警告は出現されないことを確認し，提案モデルの実現可能性およびモデルに基づいて実装したフレームワークの有用性を確認した．

6. おわりに

本稿では，開発者やプロジェクトに依存した開発支援ツールの出力を柔軟に調節を実現するために支援するモデル及びモデルに基づいたフレームワークを提案した．既存の開発支援ツールの出力は膨大である場合や，開発者の立場に応じて不必要な情報を含んでいるため，現状 ChatOps に導入することができない開発支援ツールが存在する．本モデルは既存の ChatOps のモデルを拡張し，ProxyChatBot を

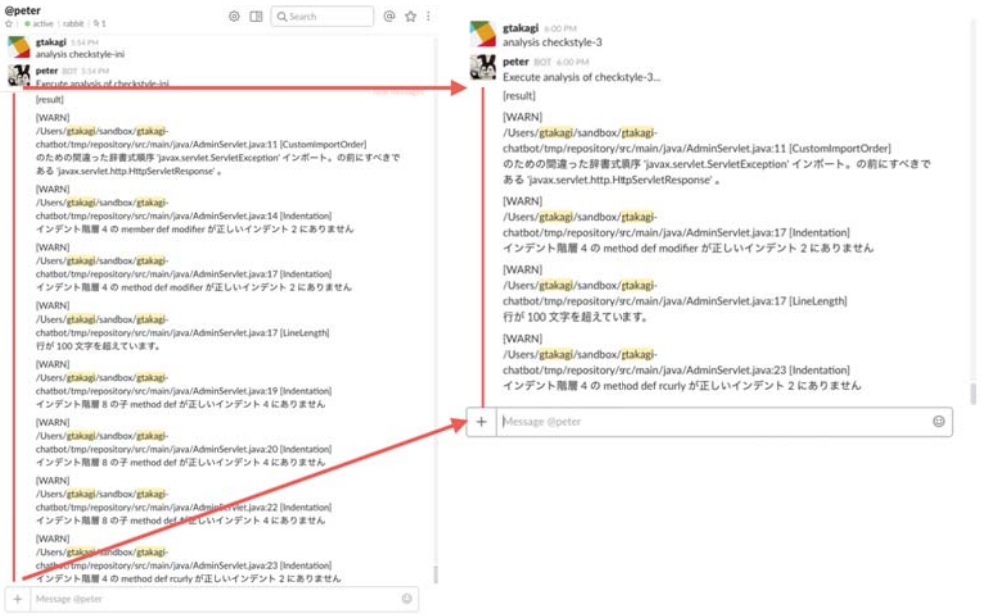

図 7 ProxyChatBot を用いた MAFP の再実装

経由することで既存の開発支援ツールと連携した ChatBot が送る出力にデータ加工をしてチャットツールに通知する仕組みを確立した．更にツールとして Hubot とチャットツール Slack の間で仲介する ProxyChatBot，及び Hubot の接続 Adapter を実装し，実際に ProxyChatBot 上で渥美らのツール MAFP を再実装することでモデルの実現可能性及び有用性を議論した．

本モデルツールではユーザーの状態に合わせて通知のタイミングを調整するような ChatBot の実装は行っていない．提案モデルに対する有用性を評価するためにこれらも評価する必要が残っている．

実装したモデルツールは Slack のみに対応しているため，様々なチャットツールに対して適用できるように Hubot の Adapter をそれぞれ実装しなければいけない．複数の ProxyChatBot を経由するための ProxyChatBot 同士を接続する Hubot の Adapter も実装しなければいけない．

今後の課題として，評価実験に実装した ProxyChatBot を用いて MAFP の再実装を行ったが，本モデルツールは静的検査ツールに限らず様々な開発支援ツールに用いることができる．それらに対しても本モデルツールを適用して評価を行う必要がある．また ChatOps ツールに導入できない開発支援ツールに限らず，既存の ChatOps モデルに対しても ProxyChatBot を経由させてデータの加工を行うことができる．これらに対しても同様に提案モデルを適用し，評価することで提案モデルが既存の ChatOps モデルの拡張として，より複雑で高機能な ChatOps モデルとして確立することを期待できる．

謝辞 本研究の一部は JSPS 科研費 JP15H02683 の助成を受けた．

参考文献

[1] Storey, M. A., Zagalsky, A., Filho, F., Singer, L. and German, D.: How Social and Communication Channels Shape and Challenge a Participatory Culture in Software Development, *IEEE TSE*, Vol. 43, No. 2, pp. 185–204 (2016).
[2] Storey, M.-A. and Zagalsky, A.: Disrupting developer productivity one bot at a time, *Proc. FSE2016*, pp. 928–931 (2016).
[3] Calefato, F. and Lanubile, F.: A Hub-and-Spoke Model for Tool Integration in Distributed Development, *Proc. ICGSE2016*, pp. 129–133 (2016).
[4] Lin, B., Zagalsky, A., Storey, M. and Serebrenik, A.: Why Developers Are Slacking Off: Understanding How Software Teams Use Slack, *Proc. CSCW2016 Companion*, pp. 333–336 (2016).
[5] まつ本真佑，中村匡秀：リポジトリマイニング技術共有のためのサービス指向フレームワーク，ソフトウェア工学の基礎ワークショップ FOSE2011，pp. 231–236 (2011).
[6] Matsumoto, S. and Nakamura, M.: Service oriented framework for mining software repository, *Software Measurement, 2011 Joint Conference of the 21st Int'l Workshop on and 6th Int'l Conference on Software Process and Product Measurement (IWSM-MENSURA)*, IEEE, pp. 13–19 (2011).
[7] 坂元康好，まつ本真佑，中村匡秀ほか：サービス指向リポジトリマイニングを効率化するキャッシュ機構の実装，研究報告ソフトウェア工学 (SE)， Vol. 2013, No. 12, pp. 1–6 (2013).
[8] 渥美紀寿，桑原寛明：MAFP:ソースコードに対する静的検査における警告の管理ツール，コンピュータソフトウェア，Vol. 33, No. 4, pp. 50–66 (2016).
[9] 桑原寛明，渥美紀寿：ソースコードの静的検査における警告の版間追跡ツール，ソフトウェアエンジニアリングシンポジウム 2015 論文集，pp. 38–47 (2015).