夏のプログラミング･シンポジウム「ビューティフル･インターフェイス」2014.8.24

チケット駆動のサーバ/インフラ運用における問題点と手動作業の自動化

井桁正人 †1

Chef での設定管理などサーバへの操作は自動化が進んでいる．しかし，チケット駆動によるシステム障害時や作業依頼の案件管理における事務的な作業は人の手により実施されている部分が多く，自動化が必要となる．そこで特にサーバ/インフラ作業に注目し，そのシームレスな運用を実現するため，担当者の手配，サーバ状態の確認や手順書の準備，実作業までの流れにおける問題点とそれらの自動化を実現するシステムについて述べる．

Problems and Automation of manual work in ticket driven server / infrastructure operation

MASATO IGETA†1

Automation has progressed such as configuration management like the operation to the server by Chef. However, clerical work carried out by the hand of man in the ticket driven work for the project management of the request or working system failure are many, so automation is required. Therefore, I particular attention to the server / infrastructure work, I described about problems and automation tool in order to achieve its seamless operations, personnel arrangements, preparation for confirmation of the server status and procedures.

1. はじめに

組織における業務管理のため bugzilla や redmine などのチケット管理ツールが利用されている．これにより目的となるプロジェクトなどにおける個々のタスクの内容，担当者，納期，対応ログを管理することが可能となる．サーバ/インフラ運用業務の場合にはサーバ設定作業の依頼や運用システムの障害時に監視システムによるアラート検知による対応が業務として発生する．このような業務に対して，上記のようなチケット管理ツールを利用した場合にはチケットとして管理するための事務的な手続きなど手動で行われる作業が発生してしまう．そこで本件ではそのような状況で発生する問題点とそれらの自動化を実現するシステムについて述べる．

まず筆者業務環境について説明する．組織としては，

- 営業業務などを行う事業部
- サービスを提供するために PHP などのフロントエンドのアプリケーションの開発/運用担当部署（以下，APP）
- APP によるアプリケーションを動作させるためのサーバの運用を行うインフラ運用担当部署（以下，SYS）
- SYS が運用するためのサーバやネットワークや監視環境などを準備する基盤グループ

の 4 階層になっている．この中で筆者の担当部署は SYS として Infoseek や楽天オークションなどのサービスを運用するサーバ等の運用やその中での業務改善を行っている．規模感としてはサーバが開発，検証，本番環境を含めて 1000 クラスタ程度，合計約 2300 台を担当しており，ドメイン数としては社内外を含めて 400 弱の数となっている．ここで弊社楽天市場や楽天トラベルなどは別部署の SYS が存在し，適宜担当のサービスの運用を行っている．

業務の主な内容は APP からのサーバ設定などの依頼作業と監視によるアラート対応及びそれらをより効率化するためのツール開発等による仕組み化や改善となる．APP からの依頼は具体的にはサーバの増強や root 権限が必

†1 楽天株式会社
Rakuten, Inc.

要なミドルウェアの設定及び再起動などが当たる．状況によっては基盤グループからサーバ全台に影響のあるような社内標準の変更のための対応や脆弱性対応なども適宜対応することとなる．そのため業務は依頼やアラートが発生するとその調査や対応のための手順書などを準備し，作業を実施する．ここではJIRA[2]により業務がチケット管理されており，作成されたチケットが適宜 SYS 担当者にアサインされることで業務が実施される． APP のためのチケット依頼窓口のフォームも依頼カテゴリごとに存在する．なおここでは作業そのものよりもその前段階の依頼内容に対する要件確認や発生したアラートに対してどういう対応をすべきか?というような調査が必要となるが、それにかかる工数が多いものとなってしまっている．

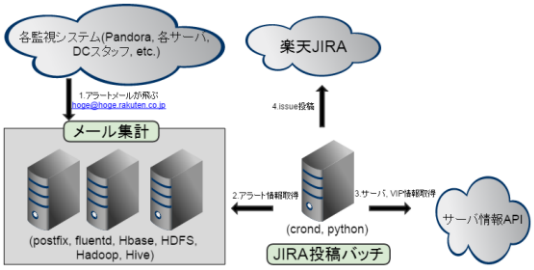

図 1　アラートのチケット管理システム
Figure 1　System of ticket driven alert controlling.

次に業務として行っている具体的なチケット管理の例を幾つかあげる． まず監視システムから発生したアラートを自動でJIRAチケットを発行することで管理している．これは特に休日/夜間などに発生したが緊急度が低く，平日営業日対応で、夜間などに発生した場合には翌営業日の対応で問題ないようなアラートを対象にしている．このようなアラートはメールで連絡が来ており，大きく 2 つの問題のためのものである．まず 1 つ目は対応漏れを防ぐためである。緊急度が高いものは電話などで即時に連絡が入り対応することとなる。しかし緊急度が低いものの場合は休日明けなど多数の対応すべきアラートがメールとして送信されており，適宜そのアラートメールを確認して対応を進める必要があるが，メールは多く，それぞれメールを確認して自分の担当であるかを確認する必要があるためである．

2 つ目としてはメールでアラートが来るため具体的に誰が対応しているか?が不明瞭であり事前に取り決めはあるがその対応ログが追えない問題である．アラートレベルとしては低くとも，中長期的な対応が必要な場合にはその対応ログが長大になり，適切な対応ログの管理が必要となる．

仕組みとしては図 1 の通り

- 社内の監視システムからのアラートメールをメールサーバで受信
- 受信したメールを集計
- 毎朝定時に前日分アラート情報をバッチが確認
- サーバ情報 API から担当者を判断
- JIRA チケットを発行

という流れになる．

ここで監視システムから直接チケットを発行していないのは，社内には複数の監視システムが存在し，それぞれにチケットを直接発行するような仕組みを実装するよりは，送信されているメールを取得してパースする方が実装が容易だったためである． 監視システムとしては各サーバそのものや HTTP や ICMP 応答を監視するもの，DC スタッフからの目視点検による連絡などが存在する．発行されたチケットの運用としては図 2 のようにオペレータが毎朝発行されたチケットを確認して適宜アラートに対する対応を実施し，完了するとチケットを Fix する. そしてチーム内のリーダーがその対応が適切であったかを確認することでチケットを Close している. なお担当者を自動でアサインするためにサーバ情報 API から担当者を参照しているが，サーバによっては担当者設定が一個人に一致しないために担当者が不明な場合がある．その場合にはリーダーが適宜適切なオペレータをアサインしている．

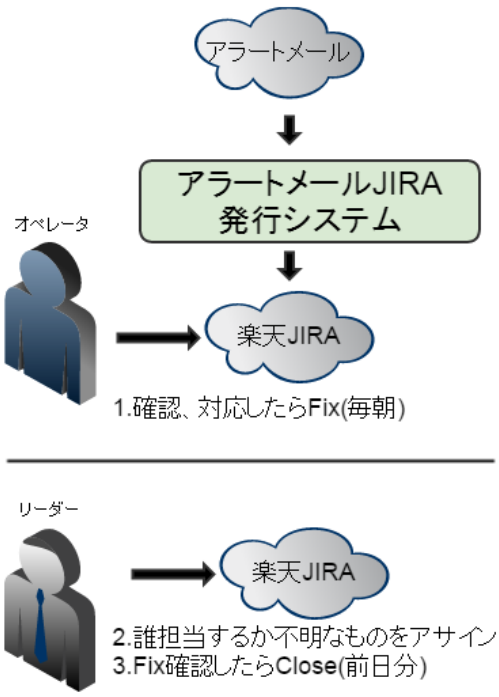

図 2　アラートチケットの運用フロー
Figure 1　Workflow of alert ticket.

次にチーム内などのレビューや共有でもJIRA を利用している．こちらはメールやメッセンジャーによる連絡が通常社内ではよく使われる連絡手段であるが，それらでは確認したかどうか?のステータス管理が不明瞭であり，それを管理するためである． 当然メールやメッセンジャーでも同様のことは可能ではあるが，レビューやタスク単位での担当者や対応フローの管理がチケットの場合にはより柔軟になる．

上記のようなインフラ運用の中で現在ではChef[1]という構成管理ツールを代表に，サーバの構成情報などを DSL によりコード化して管理/設定を反映するという流れが存在する．これは作業の自動化およびサーバ構成の暗黙知をなくすために活用されている．このようにインフラ運用業務における具体的な作業の自動化や管理をコードレベルで行うことで職人技のような属人化した業務からより普遍的な運用を行える仕組みが技術として発展してきている．

2. 問題点

2.1 業務完全自動化は困難

組織が大きい場合に直面する問題点として，組織の細分化によるコミュニケーションコストの増大があげられる．より多くの人数で物事を行う場合には適切な役割分担が必要となるが，割り振られた役割がそれぞれの部署単位で異なるために，その間に発生する調整コストが膨大なものとなるためである．このような場合，依頼を行う部署と依頼を受ける部署，そして関係部署がそれぞれ別の部署として業務を行う必要があり，社内全体で共通化するための標準化およびそれに準拠する必要や他部署の承認が必要な場合が発生する．これはサーバの設定一つでも，基盤グループの中にあるサーバ構築担当の部署がまず構築し，運用段階において APP からの依頼により SYS は設定作業を行うが，APP の依頼と社内標準を SYS は考慮の上で最適なサーバ設定を実施する必要がある，ということである．これは現状人間が調整を行い対応しているためそのような部分の自動化は容易ではなく，コストが高くなってしまう，という問題が生じている． このような問題に対しては組織全体としてのワークフローの改善が必要であり，適宜そのような施策を実施してはいるが完全な状態にはなっていないのが現状である．

また，あくまでサーバの設定作業など具体的な作業に注目しても自動化は困難な点が存在する．Chef のような構成をコード管理することで自動化を推進するものが存在するが，そのように現状の設定を完全にコード化することは困難である．これは既存のサーバはすでに数年以上の昔に設定されていたり，その設定作業を手動で作業されているため現状の設定から構成情報を管理するためのコードを再構成することが非常に大きなコストとなってしまうためである．特に筆者の業務担当ではクラスタ数に対するサーバ数が少なく，また他社から会社ごとサービスを購入し，そのまま利用しているものも存在するためにサーバ構成は多種多様で複雑なものとなってしまっている．

このような問題に対しては既存の運用を開始しているサーバの設定情報のコード化は困難であるため，新規にサーバを構築する場合

に対して構成情報をコード化することで管理することは可能であり，実施している．これにより社内標準でサーバが構築されることが保証されているため，その後の設定作業などの一元管理が容易なものとなる．また別のアプローチとしてあくまでサーバの状態を保持するのではなく，定型作業の自動化として例えばアカウント作成作業などを自動化するような作業のツール化を実施している．このような依頼の多い作業については適宜作業そのもの，ひいては依頼そのものを自動化することで全体のコストを削減することが可能であるが，まだ現状では可能な箇所がすべてツール化されているわけではない．

2.2 作業前業務が多く存在

例えば web サーバに新しいサブドメインを追加するような依頼があった場合，それに関連した DNS 登録，監視登録，apache 設定作業などが必要になる．しかしここでドメインの承認，DNS 作業の承認，監視設定ツールの主管などは他部署との調整が必要になる場合が存在する．このように直接サーバに設定作業するコストよりもその妥当性や準備のための作業前の業務コストが大きなものとなってしまっている． これはチケット駆動での問題点であるチケットの粒度と完了条件が不明瞭な場合にワークフローが効率的に回らないという問題点も影響している．そのため，

- APP からの依頼チケットの実際の要望はどういうものか?
- それに対してサーバの状態は現状どうなっているか?
- 他部署に承認の必要な承認は別途必要か?
- 依頼に対する作業は社内標準に準拠しているか?

を考慮する必要があり，事前のタスク洗い出しも大きなコストとなっている．

このような状況においてナレッジマネジメントが完全ではない点も問題となる．運用ルールや手順書等は簡単には纏まっているが，多用な要望を完全に網羅したものではなく，依頼を引き受けた SYS オペレータは具体的に何をすればいいか判断が困難であり，過去の依頼を参考にしたり，自身の経験から必要な情報を集める必要が発生してしまう．

また運用ルールなど完全なドキュメントがあればよい，というわけでもない．そのドキュメントが常に最新に保たれているか?メモ書きのような不明瞭な形式でないか?など，ドキュメントそのものの運用についても考慮が必要である．

3. 目的

本件では前述の問題点に対して特に作業前業務をより効率的に実施するためのナレッジマネジメントを支援するシステムの開発，運用を目的とする．作業前の準備について具体的なワークフローを仕組み化することで実現する．筆者の業務環境の場合には，実際の作業は 1 サーバなら数分で対象は数台程度であり，比較的大きなコストとなってはいない．そのためサーバ設定作業の自動化そのものよりもそのための準備の支援を行う．

また前述したドキュメントを作成してもその運用を行う必要がある問題に対して，その依頼案件に対するチーム内のノウハウとなる情報をプル型で自分から探しに行くのではなく，プッシュ型で自動でオペレータに対して伝える仕組みが必要になる点も考慮したい．これは不完全にまとめられたドキュメントを散在させずに一元管理し，チームとして得た経験値をチームとして享受するような仕組みが必要なためである．

4. 提案と実装

4.1 提案システム

本件では Chef での DSL により設定情報のコード化による管理および設定情報の反映を，チケットと実際のオペレータに当てはめて実施する．作業前業務を事前に DSL 化して作業に対する暗黙知をなくし，TODO，手順書，確認コマンド実行結果，特殊仕様を主な内容として DSL を構成する．内容の水準としては他部署の SYS エンジニアであれば分かる水準を担保することで，チームのメンバーであれば混乱なく業務を行える粒度でナレッジを明文化する．

さらにまとめたノウハウである DSL を依頼の 1 チケットにサブチケットとして付与することで依頼担当者はチケットがアサインされた状態でチケットを確認すれば作業前業務及

び，サーバ作業をより効率的に実施できると考えられる．

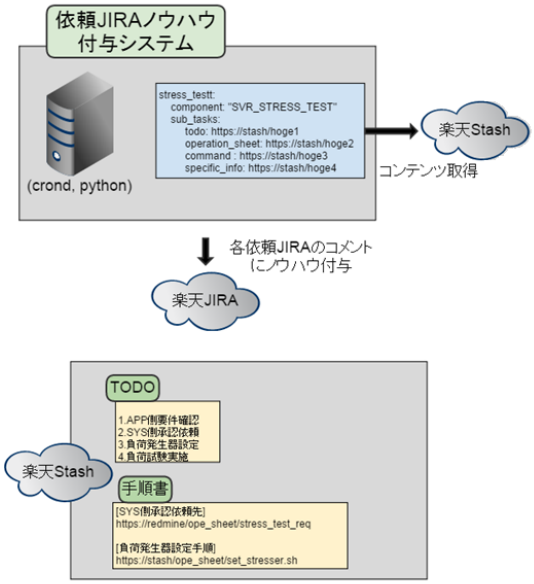

図 3 提案手法のシステム図
Figure 3 Suggested system.

4.2 実装

実装の全体像としては図 3 のようになる．Python[3]製のバッチが JIRA の API を経由して定期的に新規の依頼チケットを監視し，作業カテゴリごとに適切な情報を DSL からサブチケットに添付する． ここで作業カテゴリは JIRA のコンポーネントという単位で依頼チケット作成時にフォームに紐付いて割り当てられている．

ここで DSL としては 2 段階に分かれている．1 段階目は上記 Python によるバッチがまず読み込む yaml が存在する．これはコードが読み込むための設定情報のため，依頼種別とコンポーネント名，そして付与すべきサブチケット名とその中身を社内 github である stash[4]のリンクとして保持している．そして 2 段階目として stash のリンク先に自然言語で作業を行う上で必要なノウハウが記載されている．

これは頻繁に更新されるであろうノウハウ情報はシステムから切り分け github で管理することにより過去のバージョンや更新する場合での pull request による承認フローの機能を利用するためである．実際の運用の流れとしては図 4 のようになる．また各オペレータが確認する付与されたサブチケットは図 5 のようになる．

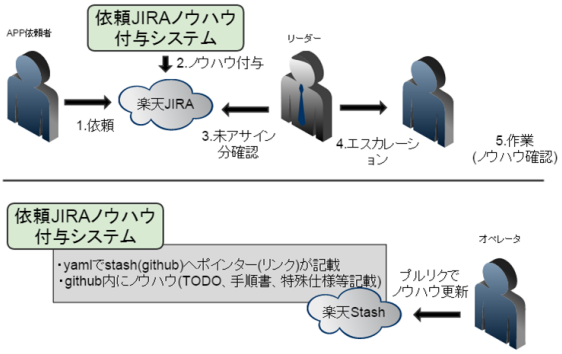

図 4 提案手法の運用フロー
Figure 4 Suggested system's workflow.

図 5 付与されたサブチケット
Figure 5 Added sub-ticket.

5. 期待される効果

現在本件の提案システムは実装は完了しているが運用のため調整中であるため評価による考察ではなく，ここでは期待される効果について述べる．

まず 1 つ目としては情報をプッシュ型で提示することによる利点があげられる．これによりオペレータがそれぞれ必要なドキュメントを毎回探す必要はなくなる．また社内に散在するドキュメントの中でどれが正しいものなのかもチケットにノウハウが付与されることでそれは承認済であり信頼に足るドキュメントであることがわかる．

次にドキュメントの可読性，保守性が向上することが期待される．DSL により情報の形式を事前に固定しているため，ドキュメントとしてどのような情報を記載すべきかが明白になる．ドキュメントを作成する際にどのようにそのドキュメントを参照されるか?という状況も作成者は事前に理解することができ

るため，その目的や文脈にそったドキュメントを作成することになるためである．さらにチケットに付与された情報が不十分な場合にはそのドキュメントを更新し最適なものにするべきかどうかが自らが作業を進める上で明確になるので，ドキュメントの保守性も向上すると考えられる．ここでドキュメントの更新は github にて誰が更新したかを確認することができるため，それを人事的な評価に利用するなどインセンティブへも役立てることが可能であると考えられる．

6. 今後の課題

6.1 評価の実施

前述の通り本件の提案システムはまだ評価を行っていないため，実際の業務コストを削減できたかモニタリングする必要がある．これは具体的な工数によるものや，それによって作成/保守されたドキュメントの数，及び各オペレータの主観評価を確認することとなる．

6.2 DSL の最適化

DSL で記載すべき内容は現状ではあくまで必要と思われる項目や情報の列挙にすぎない．そのためその文法や内容をチームに合わせて最適化している必要がある．これはチーム内での既存資料へのアクセスログ解析や作業担当者へのヒアリングによって適宜実施すべきであると考えられる．またそのための DSL の保守ルールを明確に定義することも必要になると考えられる．

6.3 作業自動化が可能になった場合の作業用 DSL の生成

サーバ設定作業が完全にコード管理され，作業そのものの自動化が確立された場合には，本提案の DSL からそのような構成自動管理ツールのための DSL の自動生成も必要であると考えられる．その場合にはワークフローの上で必要となるヒューマンループを適切に洗い出し効率と安全性を確保した自動化を進める必要がある．

6.4 チケット依頼時のフォーム UI の改善

チケット作成から作業までの SYS 側での業務コストの自動化について言及してきたが，実際の業務としては APP 側でのチケット作成コストや，事前の要望に対してどのように SYS へ依頼すべきか?が問題点で述べた組織の細分化により問題となっている．そのためチケットを作成するフォームにおいて効率的に SYS への依頼のチケットを作成できるよう支援する必要がある．これは具体的には例えばサーバ名の自動補完や既存設定の自動取得などが検討される．また依頼項目としてのカテゴリが適切であるかも適宜見直す必要がある．これは社内基盤などの移り変わりによって適宜依頼カテゴリごとの件数の推移が発生するため，そのチケット発行状況を適宜モニタリングすることで適切な項目の作成/削除を実施すべきであると考えられる．

6.5 リアルタイムアラート対応

背景で述べたとおり緊急度の低いアラート対応のチケット管理はすでに実現しているが，リアルタイムの緊急度の高いアラート対応に対しても展開していく必要がある．これは従来の電話やメール連絡だけでなく，それに付属したチケットの発行や Asterisk による電話自動応答および連絡，Hipchat[5]などのメッセンジャーによる連絡など，緊急時の関係者の連絡をより適切かつ迅速に行うための仕組みが必要となる．これに本件での対応ナレッジの付与なども検討される．

なお新たな問題としてアラートとチケットの粒度のすり合わせが必要になる．既存の緊急度の低いアラート対応のチケット管理ではアラートが複数発生した場合にどの単位でチケットを 1 つとして管理するか?という点に対して 1 日という期間の中で同様のアラートが発生した場合には 1 つのアラートとして対応していた．しかしリアルタイムでアラートメールがきたらチケットを発行してしまうとメールの数だけチケットが発行されてしまい，管理手間が増大してしまう．そのため同一であると判断できるようなチケットについては既存のアラートのチケットにサブチケットとして紐付けるなどの考慮が必要になると考えられる．

7. 結論

本件ではサーバ/インフラ業務におけるチケット駆動の運用について，サーバ作業前業務が多いことに注目しそのためのシステムを提案した．提案手法によって，作業前業務のノウハウを DSL で明文化し, JIRA チケットにそ

れをサブチケットとして付与することでオペレータが作業を進めやすくなることが期待される．

今後としては効果測定と継続性を高めるための施策が必要であり，業務全体の効率化のためには，本件の対象外としていた依頼前や作業実施段階での自動化を推進する必要がある．

謝辞 本研究の設計開発にご協力いただいた皆様に，謹んで感謝の意を表する．

参考文献

1) Chef
https://www.getChef.com/Chef/
2) Jira
https://www.atlassian.com/ja/software/jira
3) Python
https://www.python.org/
4) Stash
https://www.atlassian.com/ja/software/stash
5) Hipchat
https://www.atlassian.com/ja/software/hipchat

質疑・応答

斉藤(ハートビーツ) 作業の正当性の評価はどのように確認してますでしょうか?
井桁 インフラオペレータによる目視確認が主です.
斉藤(ハートビーツ) オペレーションの自動化について依存関係はどこまでカバーできますでしょうか?
井桁 現状はあくまで一番最初の構築に必要になるところまでです.
不明 そもそもアラートが発生しないような対応は進めているのでしょうか?
井桁 アラート種別によっては自動で設定変更されるような仕組みもいくつか実施しております.
不明 APP から来た依頼内容はどう確認しているのでしょうか?
井桁 依頼フォームの各内容やサーバ状態を目視やツールで確認している.
中山(夏プロ幹事) なぜ JIRA や Confluence を選んだのでしょうか?
井桁 社内標準として決まったため．導入は別部署のため選択の主権はありませんでした．API が提供されていたのは幸いでした.
中山(夏プロ幹事) 参考で kompira というものもあるので参考にするといいかもしれません.
中山(夏プロ幹事) 今後定量評価を実施したあとの考察などを今後発表頂けるか?
井桁 内容や間に合うかなど考慮の上で判断します.
中山(夏プロ幹事) 夜間に発生するアラートなどの対応は自動化できるのか?
井桁 アラート内容によっては可能です．ただし基盤グループが関与するものもあるので開発全体として対応が必要です.
丸山(夏プロ幹事) 依頼元の人とやりとりすることが多いのでしょうか?
井桁 はい．事業部まで遡ることもあります.
丸山(夏プロ幹事) 提案の DSL を準備していくことで新人教育に利用できるでしょうか?
井桁 補助としては可能ではありますが，社内業務知識持っている前提で記述されているためそれを教える必要があります.