GitHub

開発ツール社内導入攻略ガイド

－ツールを社内に浸透させるためのベスト・プラクティス－

はじめに

IoT、AI、コネクテッド・カー、フィンテックなどのキーワードに代表されるよう、昨今、「ソフトウェア化」はすべての産業において必須事項となりました。そのような背景から、ソフトウェア開発環境の整備・改善は企業にとって重要な課題であるにもかかわらず、多くの企業ではまだ十分に進んでいません。

その要因の１つとして、開発環境が開発者たちに及ぼす影響が軽視され、結果として組織が開発環境を整備するシステム管理者たちに十分にコストを割いていないことが挙げられます。開発環境の整備･改善は、単純なサーバ管理ではありません。技術が進化するように、組織が変化するように、最適な開発環境も日々変化します。そのような中で、開発環境の整備・改善を目指し、最適な開発ツールの選定と導入にチャレンジすることは容易ではありません。技術に対する感度、またその技術に対応するための多種多様な技術スキルを要します。それに加えて、組織にとって最適なツールを見極めるための冷静な判断力と迅速かつ戦略的な決断力が必要です。また改善を推し進めるための推進力も必要でしょう。

本稿は、そうした開発環境の整備・改善を行う管理者を支援するために、開発ツール管理者であった私の実体験を基に「開発ツール社内導入攻略ガイド」としてまとめました。本ガイドの前半では、開発ツールを導入する際にそのツールの選定と技術検証で確認すべきポイントについて、後半では、導入した開発ツールを浸透させることの重要性と浸透させるために管理者が心がけることについて紹介します。

開発環境の整備により開発フローを効率化することで、開発者から無用な足かせを外し、素晴らしいプロダクトが世の中に沢山生まれることを願います。

GitHub
ビジネスサポートエンジニア
鈴木 順子

GitHub

目次

GitHub

第 1 章

組織へのツールの浸透

どんなに技術選定と検証がしっかりでき、どんなに組織の要件に合った素晴らしい開発ツールを導入しても、ユーザに利用されなければ全く意味がありません。本章では、私個人の体験談も交えて、開発ツールを組織に浸透させるために大切なポイントや、また浸透させることの重要性について紹介します。

開発環境が整備できていないことの弊害

私は以前、あるWeb系大手IT企業において、主にJavaを使ったソフトウェア開発に従事していました。その中で、基盤システムの新規開発に携わった際に、ソースコードやライブラリ、APIなど必要な情報を効率的に検索、参照できることの重要性に気づきました。

当時私は、100を超える社内のAPIを、社外から利用できるように拡張するためのAPI基盤の新規開発に携わっていました。そのため、社内APIの仕様を把握する必要がありました。しかし、多くの社内APIのソースコードやライブラリは所在が不明でした。APIの仕様を記したドキュメントもほとんどありませんでした。当時はなんとか人脈を駆使して、ソースコードやライブラリを発見してAPIの仕様を把握して開発を進めることができましたが、その際に、開発環境が整備されていないことによる弊害を強く感じました。

当時の社内の公式と言われる開発ツール全般は、インフラ担当者が“サーバ管理”の一環として管理していました。加えて、把握できないほどの沢山の“野良”開発ツールが沢山存在していました。お世辞にも、効率的に開発できる環境とは言えない状態でした。野良開発ツールとは、導入されたものの組織全体に浸透することなく、一部プロジェクトでのみ利用され、その管理品質も担保されていないものを指しています。

タスク管理ツールとしては数え切れない程の“野良”Redmineと、一部BacklogやTrackなどを利用しているプロジェクトがありました。CIツールはJenkinsが主流でしたが、把握できないほど存在し、チャットツールはSkype、IRC、ChatWorkとプロジェクトによってコミュニケーション方法が異なっていました。

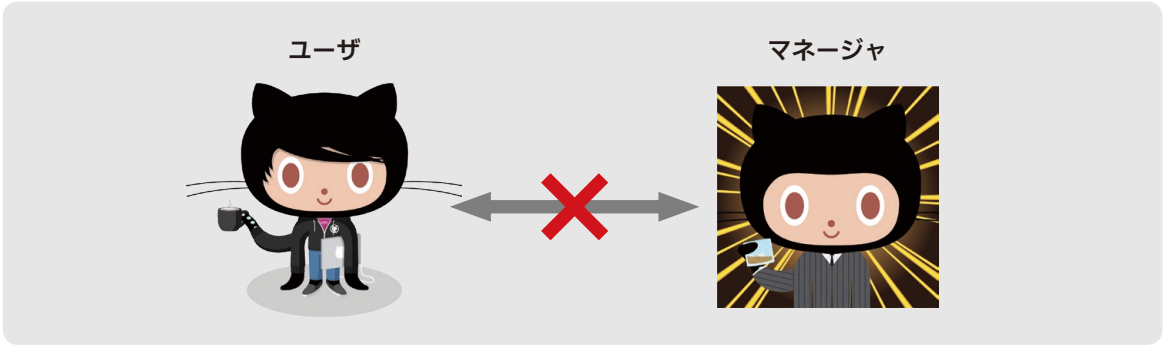
バージョン管理ツールは公式にはSubversionが推奨されていましたが、CVS上にも数百ものリポジトリが存在していました。GitLabやSubversionなどの野良サーバが多くあり、アカウント管理者だけでなくサーバ自体が行方不明なものが沢山ありました。GitHub.comのプライベートリポジトリもありましたが、アカウントの管理者が不明なことも多く、社外の開発者のアカウントが特に管理されず登録されており、追跡も非常に困難でした。

野良開発ツールの乱立は多くの問題を引き起こしていました。まずは、ユーザの学習コストです。ユーザは、プロジェクト毎に様々な開発ツールを使用しており、プロジェクトを異動するたびに、新しいツールの使い方を覚え直す必要がありました。次にツールの管理コストです。システムやアカウントの管理は、開発者が、開発の傍らに行っていました。その結果、システム管理やアカウント管理が繁雑になり、データロストの危険性や、セキュリティ上の問題を抱えていました。さらに、管理者は、プロジェクトの異動に伴って簡単に管理者が不在になり、開発ツールのサーバが行方不明になりました。

開発ツールの管理者がユーザとマネージャの架け橋になる

当時の私は、このような状況に陥ってしまったのは、マネージャ（決裁者）とユーザ（開発者）の、開発環境に対する意識のギャップが原因だと考えました。マネージャは開発環境の整備よりは目先のリリースを重視し、一方で開発者はプロジェクトを効率的に進めるために、本業である開発の合間に野良の開発ツールを導入する。結果として、組織全体としては開発環境を軽視した形になり、開発者たちは、十分に管理されていない開発ツールを乱立することになります。

開発者はマネージャ（導入を承認する人、または決裁権を持つ人）に対して、「マネージャは、どうせ、自分たちの開発環境にコストを割かないだろう」と考え自分たちで解決しようとしていました。マネージャは、「開発者は、どうせ、新しいツールをいれても、いずれちゃんと管理しなくなるだろう」といったように、お互いの信頼関係が失われてしまった結果です。



導入した開発ツールを組織に浸透させるには、開発ツールのシステム管理者が、ユーザとマネージャの間に立ち、「ユーザからの信頼」と「マネージャ（決裁者）からの信頼」を得る必要があります。この信頼を得られなければ、開発ツールの導入は中途半端に終わってしまいます。

ここで、「ユーザからの信頼」とは、ユーザが開発環境について困った時に、開発者自身で解決せずに「あの人達に相談しよう」と思い浮かべてもらえることを指します。この信頼を失うと、ユーザが導入した開発ツールを利用せずに、類似の野良ツールが乱立する要因となります。「マネージャ（決裁者）からの信頼」は、ユーザからの信頼に応えるために、既存ツールの管理や新しい改善のためのコストをかけるために必要です。

それでは、当時私が具体的にユーザとマネージャからの信頼を得るために心がけたことをいくつか紹介します。これらがうまく機能するかは組織の性質に寄りますが、1つでも参考になれば幸いです。

1 組織（チームや部署）を作り、その知名度を上げる

組織を設立した当時はメンバーは私一人で非公式でしたが、個人名を使わずに組織のように振る舞うことにこだわりました。これは、開発環境の整備・改善が属人的になることを避け、「開発者を支える組織がいつもある」ことを目指していたためです。「プロジェクト名」を作り、自分たちは開発者を公式にサポートする組織なのだという存在感を積極的に示しました。開発環境に関わる業務連絡には、タイトルや内容の中に必ずプロジェクト名を入れました。また、ロゴを作成して、開発ツールのヘッダに表示しました。さらに、ポスターやフライヤーを作成して社内のいたる所に貼ったり、週報メールを送信し続けるなど知名度を高めるようにしました。最終的には、組織として公式に認められた際も、知名度を下げないためにプロジェクト名をそのまま組織名に使用しました。

2 相談するきっかけを増やす

開発者が困ったときにすぐに相談してもらえるように、相談するきっかけをなるべく多く作りました。また、開発者の中にもレイヤーが存在していたため、現場の開発者、マネージャ、ボードメンバーとそれぞれから雑談や意見を吸い上げる工夫をしました。これは、相談をしてもらうだけでなく、野良ツール乱立の抑止力にもなりました。相談窓口がわかりやすいように、週報メールの中に記載したり、当時の公式ドキュメント管理ツールのトップ画面に問い合わせ先を明記しました。また、チャットツール上で、問い合わせ用のチャンネルを作成して、なるべく気軽に問い合わせができるように工夫しました。野良のツールを見つけ出しては、積極的に管理を引き取り、公式ツールに移行を進めました。さらに、各部署の開発マネージャと週一で定例を行い、開発環境で困っていることを吸い上げるようにしました。この定例を開始した頃にはすでにタスク管理ツールを統一した後であったため、すべてのリクエストに対してチケットを作成して進捗を明確化するようにしました。さらに、月一で、ボードクラスの開発メンバーと組織全体の開発環境について議論し、その議事録や参加者をドキュメント管理ツール上で公開しました。これは、現在進行中の事項を誰でも確認でき、重要な決定事項をブラックボックスにしないことが狙いでした。

3 相談に対して必ずアクションを起こす

相談を受けたときは、必ずアクションをして結果を伝えるように気をつけました。開発者からのリクエストに対して、要望に応えられないケースもありましたが、その場合も、経緯や理由を明確に伝えるように気をつけました。開発者からのリクエストを実現するために、取れる手段はすべて取るようにしました。私自身が開発者だったので、どうしても技術的にこだわりそうになったこともありましたが、とれる手段が技術的なことではなくても、開発環境を改善するために必要なことは全てやりました。スピード感も大事です。時間がかかるものでも、必ず進行していることを相談者に伝えるように気をつけました。

4 ドキュメントや勉強会の開催

窓口をどんなに宣伝しても、敷居が高いと感じる開発者は沢山いました。そんな開発者たちの悩みを少しでも解決できるように、困っていることをなるべく先読みしてFAQなどのドキュメントを用意しました。ドキュメントは、該当する画面のスクリーンショットを入れたり、実行コマンドを記載して、直感的に理解できるように工夫しました。さらに大規模な勉強会やセミナーを開催したり、エンジニアに限らず、デザイナーやマーケティング担当者たち向けの勉強会も個別に実施しました。

5 ルールをなるべく作らない

1000人近い組織であったため、ある程度のルールを作成しましたが、可能な限りルールを作らないように気をつけました。開発ツールを導入していくと、どうしてもルールを作りがちですが、それによって開発者が窮屈に感じてしまわないように必要最低限を心がけました。開発者から新しいツールを利用したいとリクエストがあった際は、公式に検証できる仕組みを作りました。検証に必要な契約手続きは引き受け、代わりに検証とフィードバックは提案者に任せました。その後導入が決まったものについては、その管理を引き取りました。

6 固執しない

一度決めた手段に固執しないことにも気をつけました。どんなに検証に苦労しても、新しい良い方法を見つけたのであれば、それを選択する。本当に自信を持って、これは開発環境を一番改善できる方法だと思うものを選択するように心がけました。開発ツールは導入して浸透したら終わりではありません。最適な開発環境は日々変化します。特定の方法にこだわらずに、改善を続けていく必要があります。

7 開発者の味方である

日々の運用に追われて忘れないように。

GitHub

第 2 章

GitHub Enterprise

GitHub Enterprise は世界中で広く使われている GitHub.com と同等の機能を利用できるオンプレミス版の開発プラットフォームです。この章では、GitHub Enterprise が、検証項目をどれくらい満たしているかについて表で紹介します。

		管理者目線の検証項目
ネットワークなどのアクセス制御ができるか	●	社内ネットワークに GitHub Enterprise 用のインスタンスを配置できるため、アクセス制御が容易です。
アカウント管理のしやすさ	●	**認証プラットフォーム** Built-in、CAS、LDAP、SAML（AD FS、Azure AD、Okta、OneLogin、PingOne、Shibboleth）といった多くの認証方式をサポートしており、組織のポリシーに合った認証方法を選択できます。
運用を自動化できる仕組み	●	**API、コマンドなど** ユーザ向けの API だけでなく、GitHub Enterprise を管理するための API やコマンドを多くサポートしています。
システムの管理のしやすさ	●	**容易なアップグレード** パッケージダウンロード、コマンド実行などの簡単アップグレードに対応するほか、バグ修正などのパッチリリースは基本的にダウンタイムゼロで適用可能です。
	●	**システム監視の仕組み** SNMP サポート、システム状況のグラフ、ログ転送、監査ログに対応しています。
	●	**復旧方法** バックアップデータから簡単に復旧する仕組みや、ネットワークやハードウェア障害が発生した際の高可用性の仕組みを提供しています。
サポート体制	●	**窓口の有無** 英語でのテクニカルサポートを、平日 24 時間受け付けています。また平日午前 9 時から午後 5 時までの間は、日本語でもサポートを行っています。さらに、システムダウンなど緊急の問題が発生した場合は、英語限定ですが、24 時間 365 日サポートします。
	▲	**管理者用ドキュメントの有無** 公式ドキュメントは、現在日本語化を進めていますが、英語で記載されたものがほとんどです。

GitHub

ユーザ目線の検証項目		
ユーザ用公式ドキュメントが充実しているか？	▲	公式ドキュメントは、英語で記載されたものがほとんどです。また、公式ではありませんが、インターネット上で数多くの日本語の記事が公開されています。
ツールを学ぶためのワークショップなどのイベントが開催されているか？	●	ギットハブ・ジャパン合同会社主催のセミナーを継続的に開催しています。
新しいツールへの移行が簡単か？	●	**連携ツールのサポートが充実しているか？** 様々なサードパーティツールと連携可能です。
	●	**データ移行のための機能が提供されているか？** Subversionやその他類似ツールから、リポジトリやユーザ情報といったデータを移行するための手段を多くサポートしています。例えば、移行APIや専用の移行ツールを提供しています。
WindowsとMacの両方にクライアントは対応しているか（ツールの性質に寄る）？	●	WindowsとMacに対応したGitHub Desktopなどのクライアントを用いて、リポジトリのクローンやブランチの作成、履歴の参照、コードのコミットなどを快適に行えます。

GitHub

まとめ

ソフトウェア化するビジネス環境において、開発環境の整備・改善は競争力のある製品やサービスを開発するだけでなく、それを開発する優秀なエンジニアの採用活動を優位に進めるためにも重要です。開発環境の整備・改善において最も大切なことは、新しい開発ツールを導入することではなく、組織の要望にあったツールを選定して、それを組織に浸透させることです。開発ツールの管理者はただのシステム管理者ではありません。より良いサービスを世の中に生み出すための縁の下の力持ちであり、常に開発者を支える存在なのです。

本稿で述べたチェックリストを参考にすることで、ユーザ・マネージャ・管理者の3者すべてがメリットを実感する提案ができるよう願っています。

ギットハブ・ジャパンについて

米国カリフォルニア州サンフランシスコ市に本拠を置くGitHubは、2008年に設立され、Andreessen HorowitzやSequoia Capitalなどの主要投資家から総額3億5000万ドルを調達している、最も利用されているソフトウェア開発プラットフォームのひとつです。GitHubが運営するGitHub.comは、世界最大規模のオープンソースコミュニティであり、大きな影響力を持つテクノロジーが数多く開発・管理されています。ギットハブ・ジャパンは、2015年6月にGitHub, Inc.の日本支社として設立されました。日本におけるGitHubの普及と情報提供に努めるとともに、ソフトウェア開発にGitHubを導入する企業に向けて強力なサポートを展開しています。

GitHub

ギットハブ・ジャパン合同会社

〒105-0012 東京都港区芝大門 1-10-18 PMO 芝大門 7F

https://github.co.jp/