

「龍が如く」の高速デバッグ術

～そびえ立つバグの山を踏破するための弾丸ワークフロー～

株式会社セガゲームス

第一CS研究開発部

アドバンスト・テクノロジー開発チーム

阪上 直樹

自己紹介

- 関わったゲームシリーズ
 - 「プロサッカークラブをつくろう！」
 - 「プロ野球チームをつくろう！」
 - 「龍が如く」
- ゲームプログラマから自動化や効率化の仕事に移行
 - QAプログラマ？
 - QAエンジニア？
 - ビルド職人？
- アドバンスト・テクノロジー開発チーム
 - 龍が如くエンジンのベース部分や各種ライブラリ・ツール等の開発



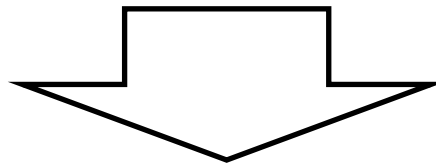
「龍が如く」を短いスパンで
リリースするための秘密は？

ないです。がんばってるだけです。

でも...

がんばる場所を限定

- ゲームの面白さ
- クオリティ
- ユーザーに感動体験を与えるための仕掛け

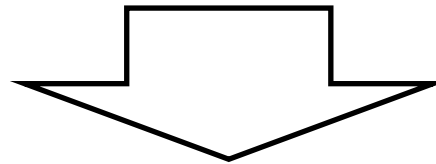


他にもやることはたくさんあるよ！
理想論じゃないの？！

がんばらないために、がんばる

- 自動化できるものは、ためらわず自動化
- 作業の効率化を徹底して追求
- ワークフローの見直し

誰でもできる作業をクリエイターにさせない



本セッションのテーマ

本セッションのゲームタイトル



龍が如く 維新！

PS3/PS4

PS Vita(無料アプリ)

2014年2月22日発売



龍が如く0 誓いの場所

PS3/PS4

PS Vita(無料アプリ)

2015年3月12日発売

本セッションの用語定義

- デバッグ
 - バグを修正する
 - テストプレイを含まない狭義の意味
- テストプレイ
 - プレイチェック
 - バグ報告
 - ゲームバランスの指摘
- **赤字**の用語
 - 龍が如くチーム用語

本セッションの概要

- 前半

- 高速デバッグ術

- 自動化

- Jenkins

- 後半

- そびえ立つバグの山を踏破するための弾丸ワークフロー

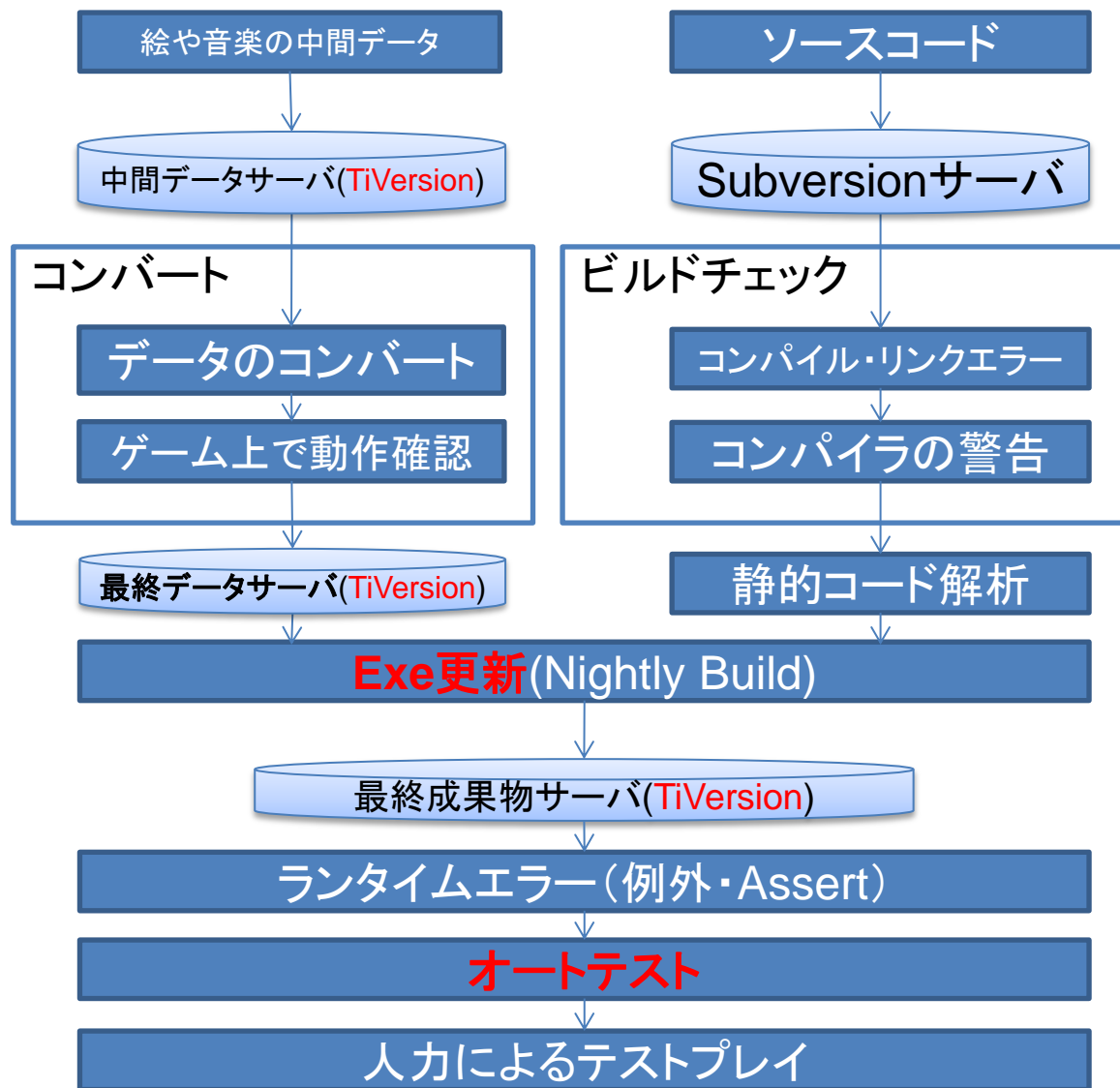
- バグ管理

- Redmine

高速デバッグ術

• 自動化

- Jenkinsを使用
- パイプライン
 - ビルド
 - コンバート
- エラー検出
 - ビルドエラー検出
 - 静的コード解析
 - 例外検出
 - テスト



Jenkinsによる自動化

- Jenkinsとは
 - CI(Continuous Integration:継続的インテグレーション)ツールの一つ
 - 日々の開発に必要なビルドやコンバートを自動化し、安定した開発環境を継続していくためのもの
- なぜ必要？
 - ゲームの動作が安定していない状態では、実装したものをすぐに確認できないので作業が滞る
 - ビルドが壊れた状態で放置すると、修正コストが増大し、他のバグを生む原因にもなる

ビルドの自動化 (1/2)

- ビルドとは

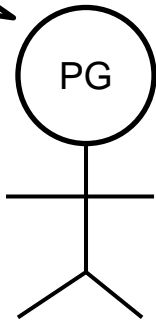
- プログラムをコンパイル・リンクして実行可能な形式に変換すること
- 自動で毎日ビルドした成果物は、Nightly Buildと呼ばれる

- **Exe更新**とは

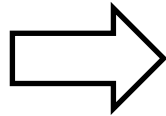
- Nightly Buildの龍が如くチーム特有の呼称
- PS3/PS4/PS Vitaに加えて、それぞれのターゲットに対応したデバッグ用のWindows版の実行ファイルを定期的に生成
- Windows版もあることから、「**Exe更新**」と呼んでいる

ビルドの自動化 (2/2)

プログラマ



プログラムを書く



ソースコード

コミット

Subversionサーバ

Subversionとは

- バージョン管理システム
- 上書きによる先祖がえりを防ぐため
- サーバにアップロードすることをコミットと呼ぶ

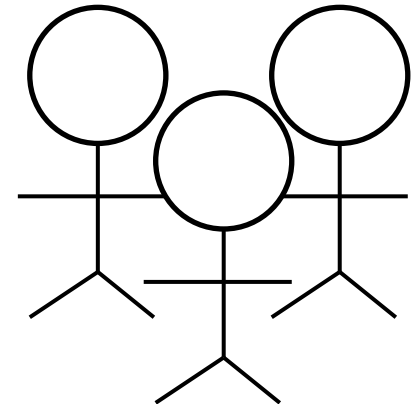
自動化

Exe更新
(Nightly Build)

コミット

最終成果物サーバ
(TiVersion)

プロジェクト全員が
最新コードの動作を確認可能

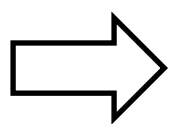
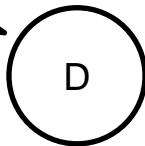


TiVersionとは

- 龍が如くチーム独自のバージョン管理システム
- ファイルサイズの大きなものを高速でやり取り可能

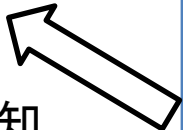
コンバートの自動化

デザイナー



背景モデルを作る

メールでエラーを通知



背景モデルの中間データ

コミット

中間データサーバ(TiVersion)

背景モデルコンバート

データのコンバート

ゲーム上で動作確認

コミット

最終データサーバ(TiVersion)

Exe更新

(Nightly Build)

コミット

最終成果物サーバ
(TiVersion)

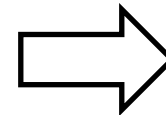
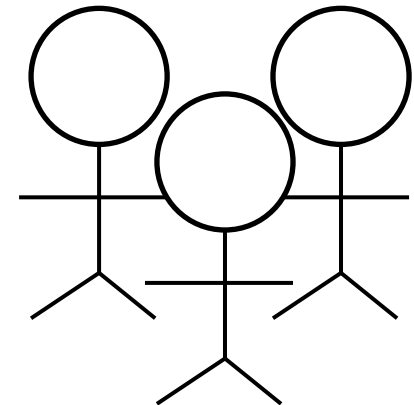
自動化

自動化

なぜ中間データを一括で
コンバートするのか？

- 各ターゲットに最適化
- 圧縮やパック
- 中間データをテキスト形式にすることで差分を取りやすく

プロジェクト全員が
最新の背景モデルを確認可能



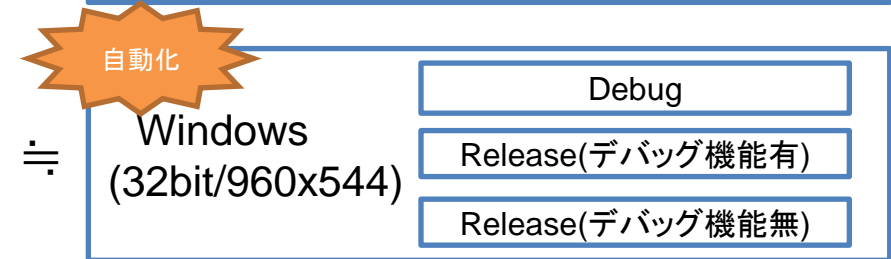
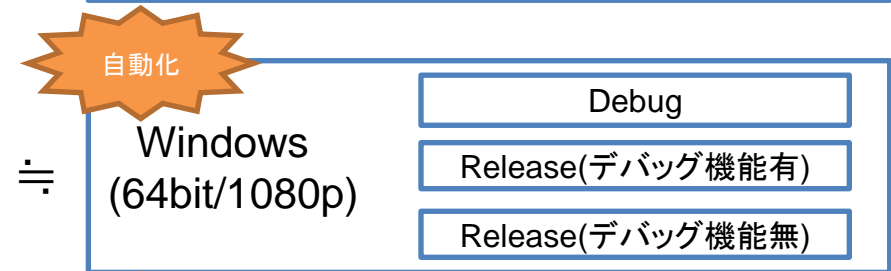
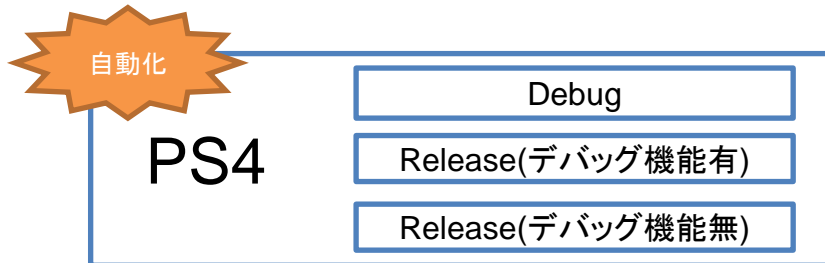
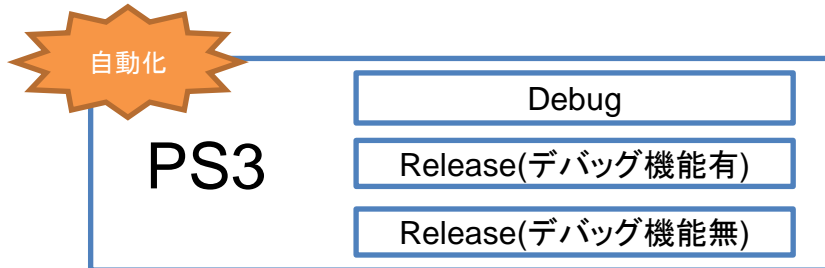
パイプライン(ビルド・コンバート)の自動化



エラー検出の自動化

- エラーの種類
 - ビルドエラー
 - 静的コード解析の指摘
 - ランタイムエラー
 - テストのエラー
- なぜ必要？
 - エラーを見つけて報告するのもコスト
 - 自動で不具合を見つけてくれるものは、どんどん使う
 - 手動だと欠落しやすいデバッグに必要なデータ(ダンプやログ、スクリーンショット等)を自動収集

ビルドエラー検出の自動化 (1/4)



手動で確認するのは大変！

ビルドエラー検出の自動化 (2/4)

- **Exe更新**とは別にSubversionサーバを監視して、コミットごとにビルドが壊れていないかチェックを行う
- ビルドが壊れていたら、Jenkinsからメーリングリストにメールを自動送信
- ビルドエラーが出たら即時直すルール
 - ビルドエラーは出さないことが望ましいが、ビルド構成が多く、各自がすべてを確認してコミットするのは無駄が多いため
- ビルドは分散ビルドを使用
- PCの台数を増やして高速化

ビルドエラー検出の自動化 (3/4)

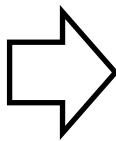
警告はエラーです

- コンパイラの警告は、エラー扱い
 - ビルドエラーになるので、即時修正が必要
 - コンパイラの警告レベルは最高に設定
 - 例外的に使いたいものは、設定で除外するか#pragmaで局所的に対応
 - まずは警告をゼロにするところから

ビルドエラー検出の自動化 (4/4)

- ビルドが壊れた状態を短縮するためには
 - 声かけが一番効果的
 - チェック結果を分かりやすく表示する
 - **ビルドチェック待ちページ**

このリストから
自分の名前が
消えたら帰宅可能



以下の方々のビルドチェックを待っています。

sakaue_naoki

r3247 (起動にこけても死なないように修正;)

r3247 (ヒートゲージのSEアサイン
IssueID #3761

jenkins_common

r3239 (ステージコンバートしました。;)

r3241 (イベントコンバートしました。;)

r3243 ([Jenkinsによる自動コミット] フィルターをソートしました。;)

r3244 (サウンドコンバートしました。;)

M /trunk/src/sound/sound_manager.cpp

atsu_takashi

r3240 (ステージ追加;)

r3242 (ステージの車を追加; ;)

r3246 (スタイル切り替え時のアニメーション; ;)

unused variable
警告はエラー扱い



リポジトリは [rev.3247](#) が最新リビジョンです。

エラーメッセージ一覧

[Zero_Win32_Release] pc-bldchk08にてビルド

/src/sound/sound_manager.cpp (950, 12) : error : unused variable 'player' [-Wunused-

[Zero_Win32_Retail] pc-bldchk16にてビルド

/src/sound/sound_manager.cpp (950, 12) : error : unused variable 'player' [-Wunused-

[Zero_PS4_Retail] pc-bldchk04にてビルド

/src/sound/sound_manager.cpp (950, 12) : error : unused variable 'player' [-Wunused-

PS3

Zero_Optimized r3244

4:24 r3247

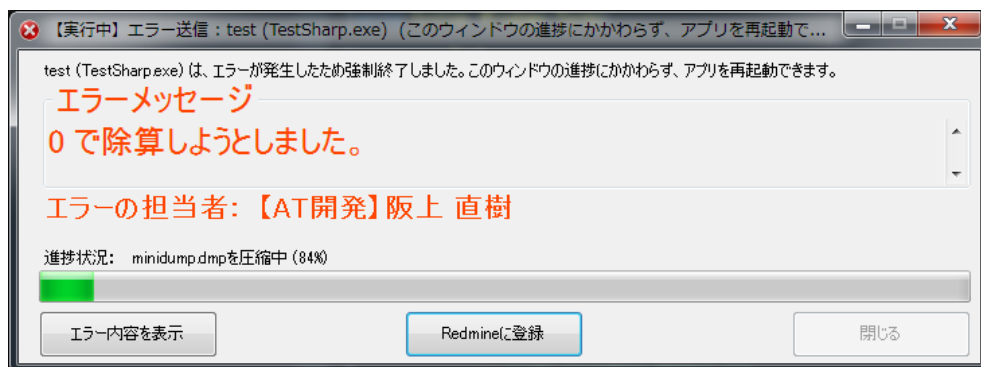
静的コード解析の自動化

- 静的コード解析とは？
 - プログラムを実行せずにソースコードを解析し、不具合やその可能性となるものを検出するツール
- なぜ必要？
 - すべてのコードを人力でレビューするのは大変
 - 不具合の芽を先に摘んでおく
- 静的コード解析ツール
 - 使用ツール
 - Coverity(精度が高い/有償)
 - Visual Studioのコード分析(Visual Studioの標準機能)
 - Cppcheck(オープンソース)
 - 使用ツールは、管理できるなら多ければ多いほどいい
 - Jenkinsを使って毎日自動的に解析して、結果をメールやチケットでフィードバック

ランタイムエラー報告の自動化

- **エラー送信** (クラッシュレポート機能の龍が如くチーム用語)

ゲームやツール
実行中に
例外発生!



デバッグに必要な情報を自動収集

- ダンプ
- ログ
- コールスタック
- スクリーンショット ... etc

アップロード

ネットワークドライブ

メール送信

- ダンプ表示batのURL
- ログ表示batのURL
- コールスタック
- リビジョン
- ターゲット

バグ報告

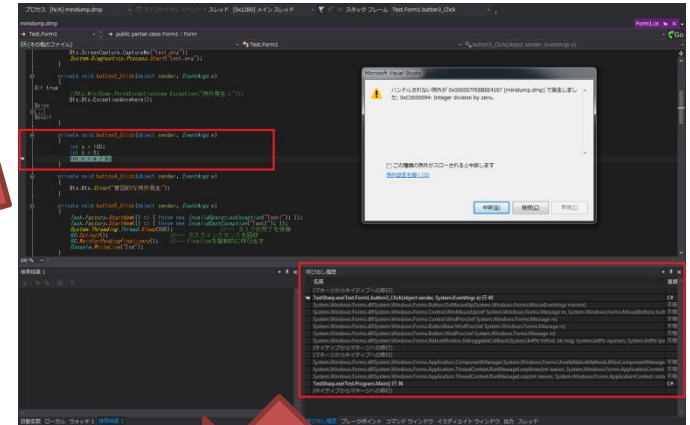
バグ管理システム
(Redmine)

エラー送信の実装例 (Windowsの場合)

- C++
 - 未処理例外のキャッチ
 - SetUnhandledExceptionFilter
 - コールスタック
 - StackWalk
- C#
 - 未処理例外のキャッチ
 - System.Windows.Forms.Application.ThreadException
 - AppDomain.CurrentDomain.UnhandledException
 - System.Threading.Tasks.TaskScheduler.UnobservedTaskException
 - コールスタック
 - Environment.StackTrace
- ダンプ出力
 - MiniDumpWriteDump
 - C#ではアンマネージド・コード呼び出し
 - ダンプを開くときは、ダンプファイル(.dmp)以外に、exeファイルと対応するpdbファイルが必要

ダンプを開いた例(C#)

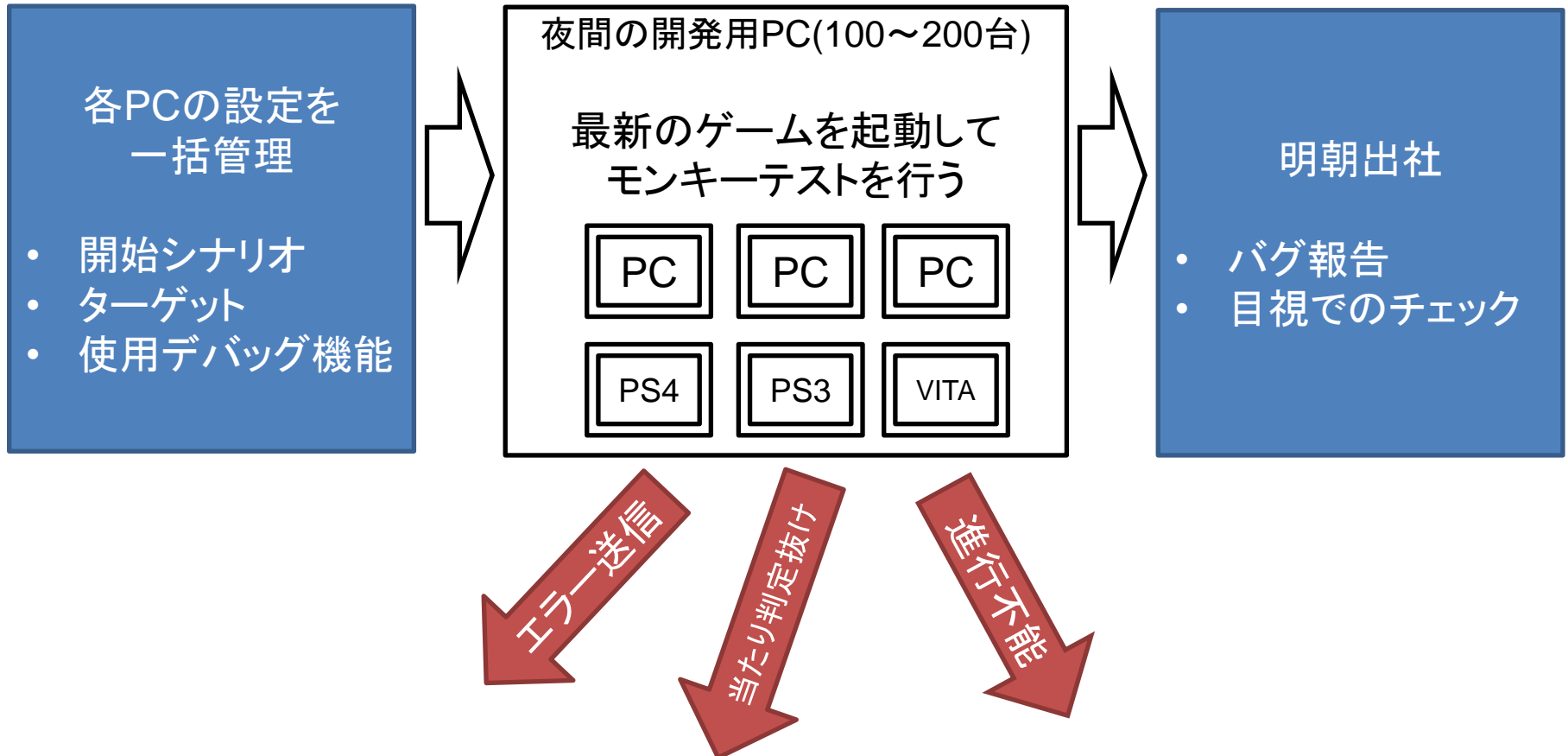
```
private void button3_Click(object sender, EventArgs e)
{
    int a = 100;
    int b = 0;
    int c = a / b;
}
```



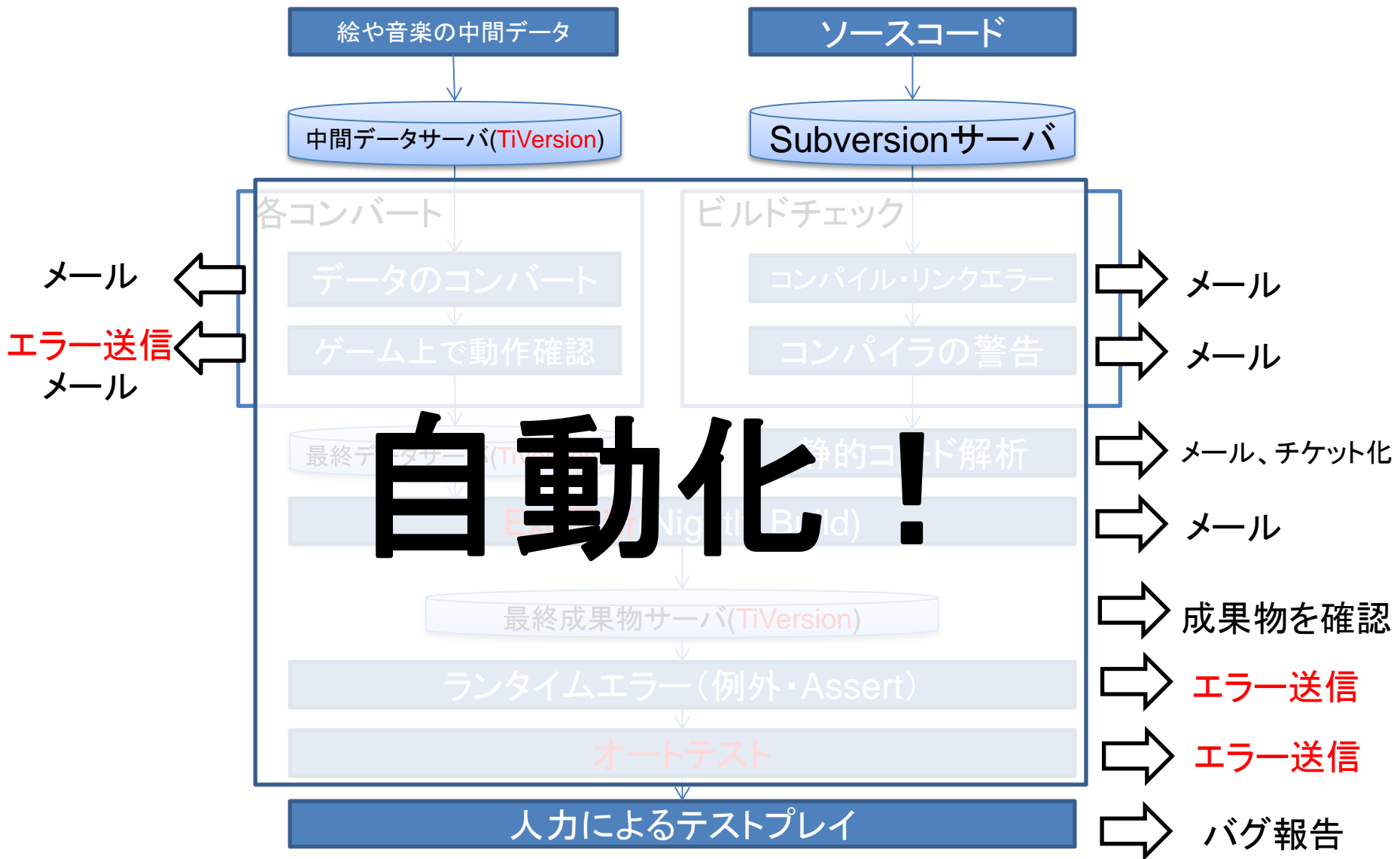
呼び出し履歴	
名前	
[マネージからネイティブへの移行]	
TestSharp.exe!Test.Form1.button3_Click(object sender, System.EventArgs e) 行 49	
System.Windows.Forms.dll!System.Windows.Forms.Button.OnMouseUp(System.Windows.Forms.MouseEventArgs mevent)	
System.Windows.Forms.dll!System.Windows.Forms.Control.WmMouseUp(ref System.Windows.Forms.Message m, System.Windows.Forms.MouseButtons butt	
System.Windows.Forms.dll!System.Windows.Forms.Control.WndProc(ref System.Windows.Forms.Message m)	
System.Windows.Forms.dll!System.Windows.Forms.ButtonBase.WndProc(ref System.Windows.Forms.Message m)	
System.Windows.Forms.dll!System.Windows.Forms.Button.WndProc(ref System.Windows.Forms.Message m)	
System.Windows.Forms.dll!System.Windows.Forms.NativeWindow.DebuggableCallback(System.IntPtr hWnd, int msg, System.IntPtr wParam, System.IntPtr lpa	
[ネイティブからマネージへの移行]	
[マネージからネイティブへの移行]	
System.Windows.Forms.dll!System.Windows.Forms.Application.ComponentManager.System.Windows.Forms.UnsafeNativeMethods.IMsoComponentManager	
System.Windows.Forms.dll!System.Windows.Forms.Application.ThreadContext.RunMessageLoopInner(int reason, System.Windows.Forms.ApplicationContext	
System.Windows.Forms.dll!System.Windows.Forms.Application.ThreadContext.RunMessageLoop(int reason, System.Windows.Forms.ApplicationContext conte	
TestSharp.exe!Test.Program.Main() 行 36	
[ネイティブからマネージへの移行]	

テストの自動化

- モンキーテストとは
 - ランダムに擬似パッド入力を行うテスト手法
- **オートテスト** (モンキーテストの龍が如くチーム用語)



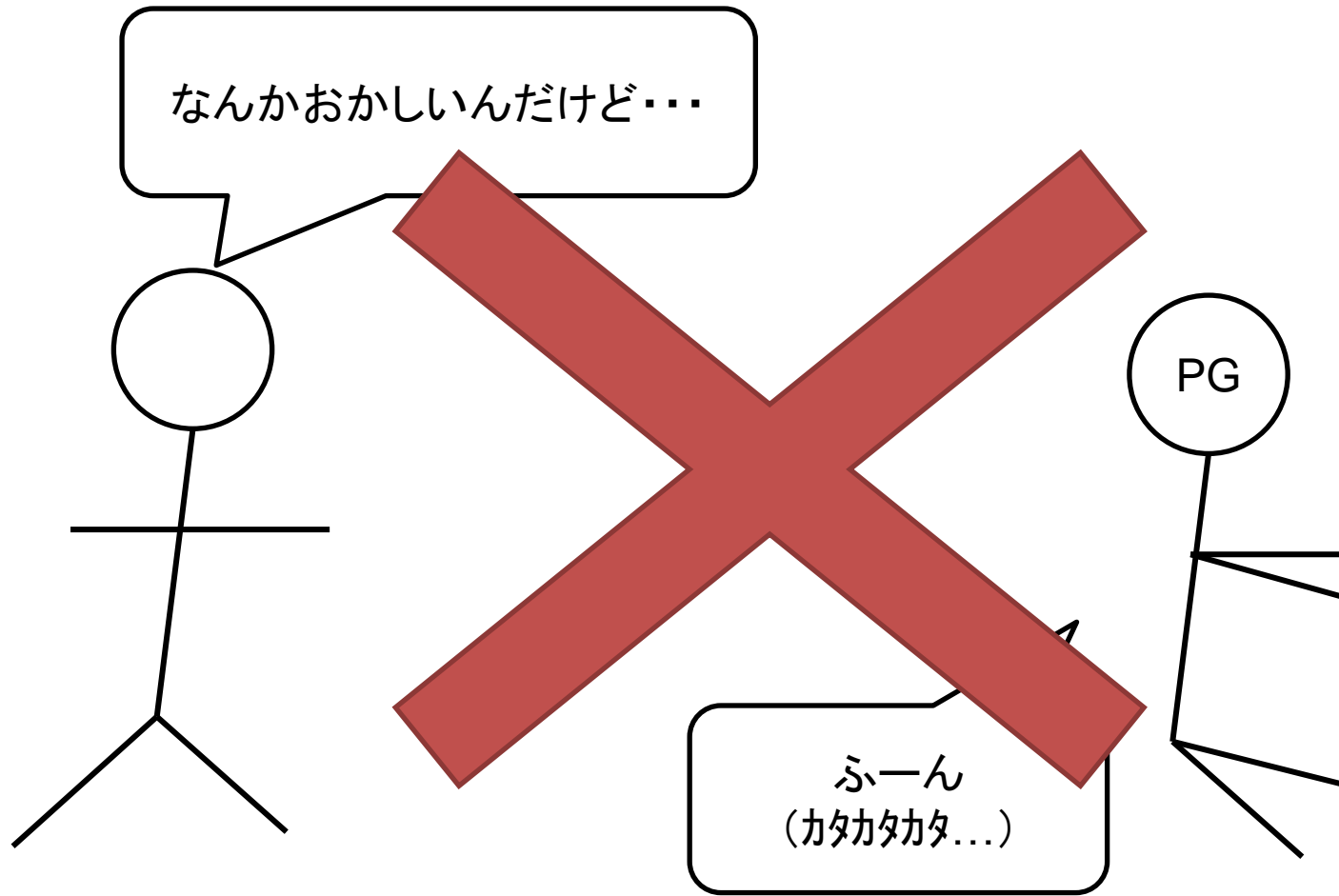
高速デバッグ術(自動化)のまとめ



そびえ立つバグの山を 踏破するための弾丸ワークフロー

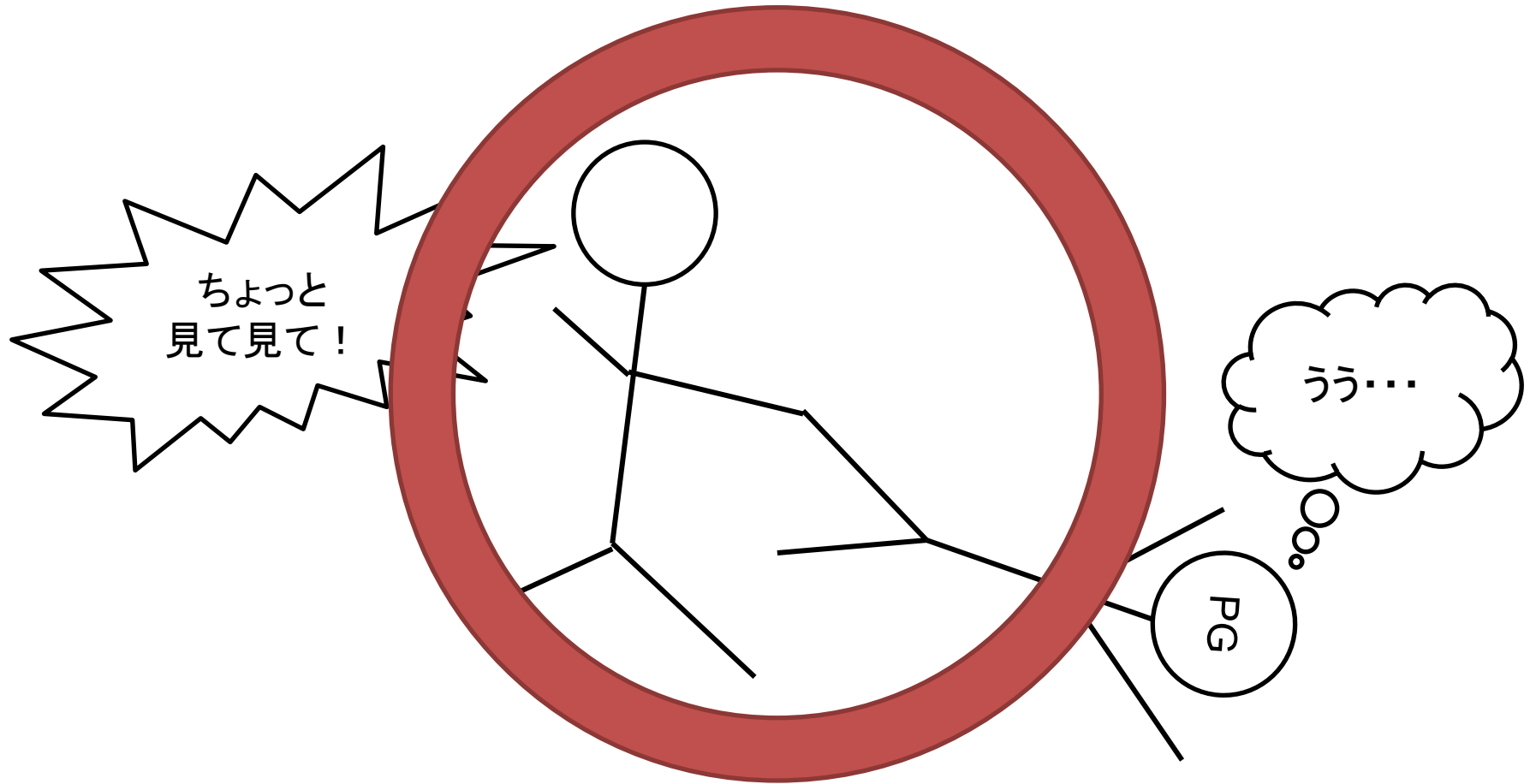
- バグ管理
 - Redmineを使用
 - バグ報告の掟
 - バグ管理システムの導入の歴史
 - 紙で管理
 - Redmineで管理
 - バグ管理システムの問題と解決

バグ報告の掟



バグ報告の掟(破り)

なんとかしてプログラマにバグの現場を見せる！



バグ管理システムの必要性

- 口頭でのやり取りは、数人程度が限界
- バグ管理システムが必要な場合
 - 数百人規模のプロジェクト
 - それぞれが別の場所で作業している

紙でバグ管理 (1/3)

- 入社してすぐ、メダルゲームのテストプレイに参加
- ディスプレイがついていたので通常のビデオゲームと同等のチェックが必要
- 手がメダルで真っ黒に
 - メダルは裏で店員さんが洗浄している
- 体力仕事
 - ゲーム開発ってガテン系？
- 湯水のようにメダルを投入し続けた結果、メダルの価値観が崩壊

メダルゲームができない体になってしまった...

紙でバグ管理 (2/3)

バグ1つに紙1枚(チケット)
違うバグを1枚にまとめない!

報告者	阪上直樹
発生日時	2015/10/17 13:30
ラスボスのやられ演出中の モーションブラーでフリーズし ました。	

バグ報告

バグ箱

担当者に
振分

PG

修正

修正NG

修正確認済

済

紙でバグ管理 (3/3)

- メリット

- バグの見える化
 - 各自の記憶に頼らない
 - 誰が何個バグを抱えているのか
- アナログならでは
 - バグの報告者・担当者間のコミュニケーションがスムーズ
 - 少人数では一番高速

- デメリット

- かかわる人数が増えると、チケットの管理が煩雑
- チケット全体を管理できない
 - 総数が分からない
 - バグの傾向が分からない
 - 各バグのステータス(対応状況)が分からない
 - どのバグから手をつけていいのかわからない

Redmineでバグ管理 (1/4)

- Redmineとは
 - バグを含めた課題管理システム
 - Webブラウザでアクセス
 - オープンソース

バグ #1181 作成者を変更 編集 時間を記録 ★ ウォッチ コピー 削除

【(PADV)プレミアムアドベンチャー】 どこでもバグ報告のテスト < 前 | 1/3 | 次 >

【AT開発】 阪上 直樹 が 2015/10/06 18:54 に追加。

[リマインダー送信]

ステータス:	新規	開始日:	2015/10/06
優先度:	通常	期日:	2015/10/06
担当者:	-	進捗率:	0%
カテゴリ:	-	作業時間の記録:	-
対象バージョン:	-		
ターゲット:	W32_Debug	ステージ:	神室町
重要度:	B	プレイヤー座標:	(-57.091, 0.000, 104.806)
言語:	日本語	時間帯:	夜
発生リビジョン:	54107	天候:	晴
プレイヤーキャラ:	桐生		

説明 引用

テストです。

RedmineAnywhere Ver 3.1.0.232

[dbg_log_w32.txt](#) (5.49 MB) 【AT開発】 阪上 直樹, 2015/10/06 18:54

[dip_switch.ini](#) (69 KB) 【AT開発】 阪上 直樹, 2015/10/06 18:54

[20151006_185205.jpg](#) (114 KB) 【AT開発】 阪上 直樹, 2015/10/06 18:54



子チケット 追加

関連するチケット 追加

Redmineでバグ管理 (2/4)

- メリット

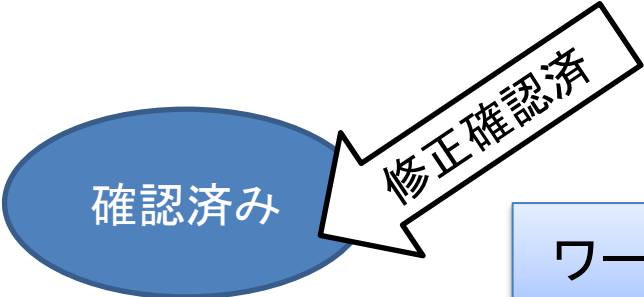
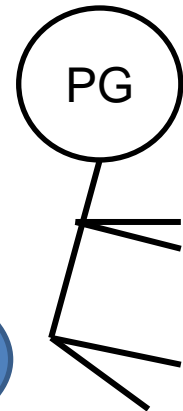
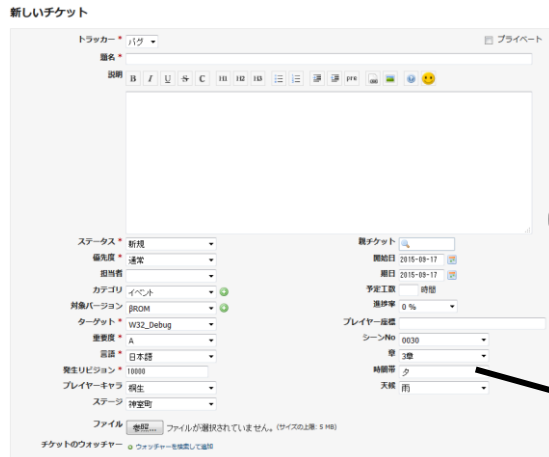
- バグ情報のデジタル化
- サーバによる一元管理
- スクリーンショットやログファイルを添付できる
- コメント入力やステータス、優先度を設定できる
- APIやプラグイン等の拡張機能が豊富で、開発も活発
- タスクも管理できる

- デメリット

- 自由度の高いツールなので、運用ルール作りが必要
- プラグインを多用すると、保守・メンテナンスコストが高くなる

Redmineでバグ管理 (3/4)

バグのステータスの一例



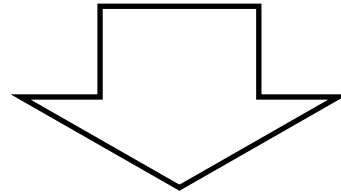
ワークフローは紙と同じ！

Redmineでバグ管理 (4/4)

- なぜ「確認待ち」なのか？
 - 「修正完了」ではない？
 - バグ修正を実装した時点で、チケットは終わっていない！
 - 修正を確認して、はじめてチケットは完了する。
 - ステータス名は、見ただけで次に何をしなければならないかがすぐ分かるのがよい

ツールを使えば、すべて解決？

もちろん、そんなことは無いです。



- バグ管理システムの問題点
 - バグ報告の問題点
 - バグのワークフローの問題点
 - 各ツールにプロジェクトの情報が分散

バグ報告の問題点

- 必要な情報を入力するのは結構大変
 - テストプレイがメインの仕事でない時に、バグ報告を怠りがちになる
- ログの添付を忘れやすい
 - 手動だと、必要な情報を入力・添付し忘れる
- 入力ミスも発生する
 - リビジョン番号が違う
 - ターゲットの選択を間違える

バグ報告に必要な情報

基本情報

- 原因
- 結果
- 再現率
- 重要度
- カテゴリ
- 報告者
- ターゲット
(Win/PS3/PS4/VITA)
- 言語
(日本語/中国語)
- リビジョン

添付ファイル

- ダンプファイル
- デバッグログ
- 使ったデバッグ機能
(設定ファイル等)
- スクリーンショット
- セーブデータ

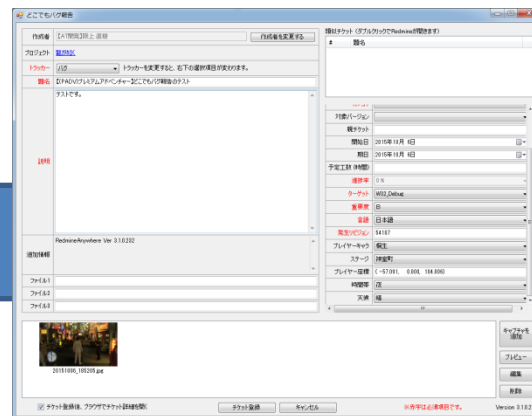
龍が如く独自

- プレイヤーキャラ
(桐生/真島/秋山...)
- ステージ
(神室町/蒼天堀...)
- プレイヤーの座標
- シナリオ進行度
- 時間帯(昼/夜)
- 天候(晴/雨/雪)

自動入力できる！

バグ報告の自動化

- **どこでもバグ報告**とは



- Redmine REST APIを使用して、専用アプリ経由でゲームアプリから直接Redmineにチケット登録
- 全ターゲット(Win/PS3/PS4/PS Vitaに対応)
- **エラー送信**からも直接バグ報告が可能
- デバッグ機能がないと使えない

どこでもバグ報告 (1/3)



デバッグメニューから**どこでもバグ報告**をクリック！

どこでもバグ報告 (2/3)

どこでもバグ報告

作成者: 【AT開発】阪上 直樹 作成者を変更する

プロジェクト: [龍が如く](#)

トラッカー: バグ トラッカーを変更すると、右下の選択項目が変わります。

題名: 【PADV】プレミアムアドベンチャー】どこでもバグ報告のテスト

説明: テストです。

追加情報: RedmineAnywhere Ver 3.1.0.232

ファイル1
ファイル2
ファイル3

類似チケット (ダブルクリックでRedmineが開きます)

#	題名
---	----

対象バージョン
親チケット
開始日: 2015年10月6日
期日: 2015年10月6日
予定工数 (時間)
進捗率: 0%

ターゲット: W32_Debug
重要度: B
言語: 日本語
発生リビジョン: 54107
プレイヤーキャラ: 桐生
ステージ: 神室町
プレイヤー座標: (-57.091, 0.000, 104.806)
時間帯: 夜
天候: 晴

キャプチャを追加
プレビュー
編集
削除

チケット登録後、ブラウザでチケット詳細を開く チケット登録 キャンセル ※赤字は必須項目です。 Version 3.1.0.232

自動入力

自動撮影 & 自動添付

どこでもバグ報告 (3/3)

バグ #1181

作成者を変更 編集 時間を記録 ウォッチ コピー 削除

【(PADV)プレミアムアドベンチャー】 どこでもバグ報告のテスト

<< 前 | 1/3 | 次 >>

【AT開発】 阪上 直樹 が 2015/10/06 18:54 に追加.

[リマインダー送信]

ステータス:	新規	開始日:	2015/10/06
優先度:	通常	期日:	2015/10/06
担当者:	-	進捗率:	0%
カテゴリ:	-	作業時間の記録:	-

ターゲット:	W32_Debug	ステージ:	神室町
重要度:	B	プレイヤー座標:	(-57.091, 0.000, 104.806)
言語:	日本語	時間帯:	夜
発生リビジョン:	54107	天候:	晴
プレイヤーキャラ:	桐生		

← 自動入力

説明

引用

テストです。

RedmineAnywhere Ver 3.1.0.232

dbg_log_w32.txt (5.49 MB) [AT開発] 阪上 直樹, 2015/10/06 18:54

dip_switch.ini (69 KB) [AT開発] 阪上 直樹, 2015/10/06 18:54

20151006_185205.jpg (114 KB) [AT開発] 阪上 直樹, 2015/10/06 18:54



← 自動添付

子チケット

追加

関連するチケット

追加

バグ報告自動化による効果

	どこでもバグ報告 使用件数	バグの 総数
龍が如く 維新！	9,338件	17,448件
龍が如くO 誓いの場所	6,307件	12,955件

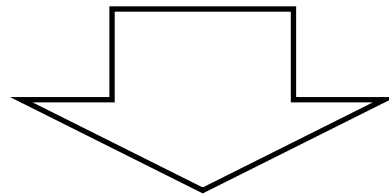
2タイトル(9,338件+6,307件) × 10分(短縮時間/件)
= 約**2,600時間**を削減

バグ報告の質が均一化して、精度も向上

テストプレイヤーは、バグ探しに専念！
クリエイターは、バグの修正・バランス調整・クオリティの向上に専念！

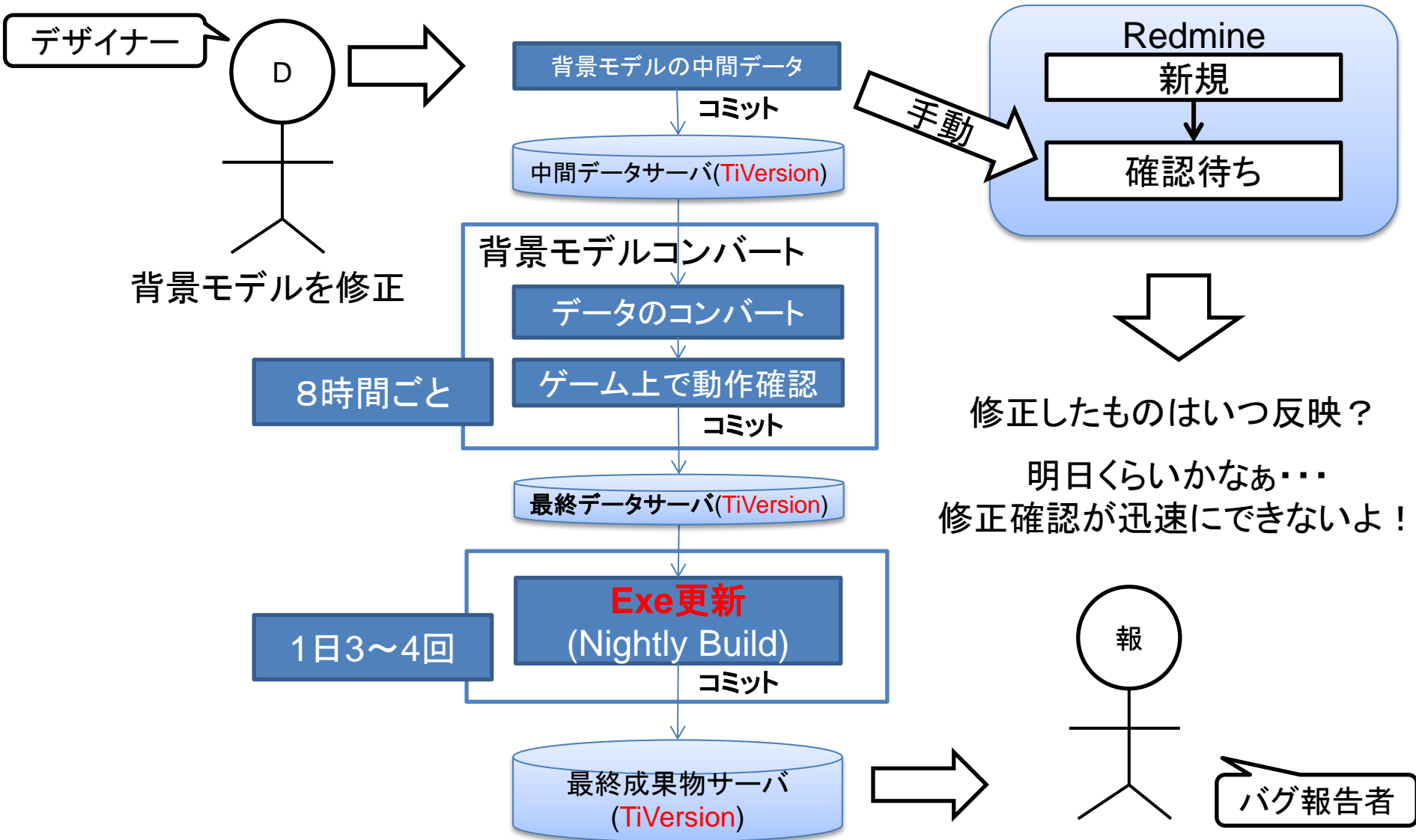
バグのワークフローの問題点

- 修正したものがいつ反映されたか分からない
- バグチケットに対して、どういう修正が入ったのか分かりにくい

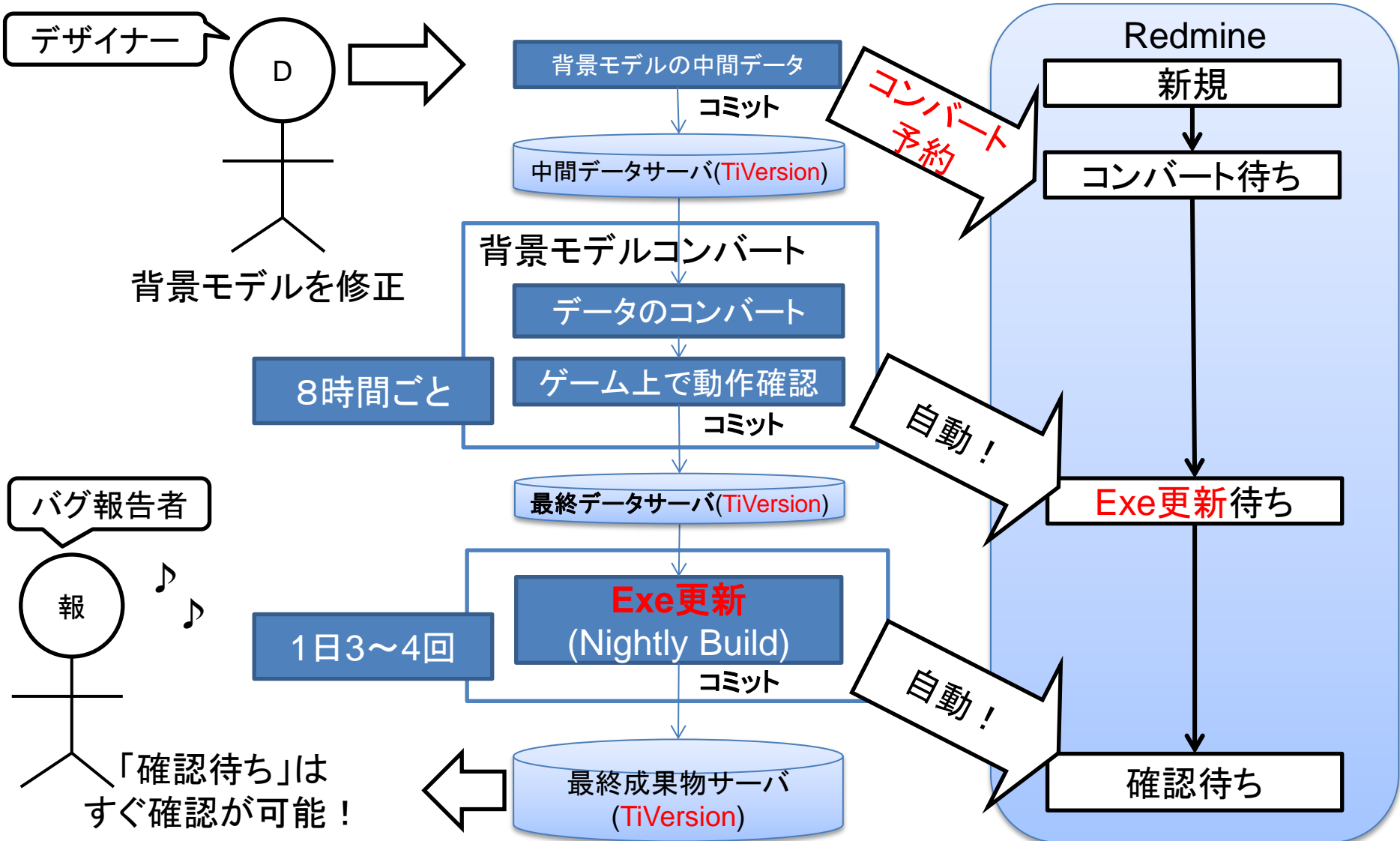


Redmineと各システム・ツールを連携して
ワークフローを自動化

Redmineとパイプラインの連携 (1/4)



Redmineとパイプラインの連携 (3/4)



Redmineとパイプラインの連携 (4/4)

コンバート予約とは

コンバート予約

• バグ #111: コンバート予約テスト

コンバート名: サウンド

注記

B I U S C H1 H2 B

効果音を修正しました。

コンバート完了後に差し戻す

作成 プレビュー

コンバートの種類を選択して
予約を作成

「確認待ち」になったら
確実に修正確認が可能

バグ #111

作成者を変更 編集 時間を記録 ☆ ウォッチをやめる コピー 削除

コンバート予約テスト

< 前 | 1/69 | 次 >

阪上 直樹 が 2015/09/25 16:49 に追加, 2015/09/25 16:52 に更新.

[コンバート予約] [リマインダー送信]

ステータス:	確認待ち	開始日:	2015/10/17
優先度:	通常	期日:	2015/10/17
担当者:	阪上 直樹	進捗率:	0%
カテゴリ:	プログラム	作業時間の記録:	-
対象バージョン:	-		
重要度:	B		

コンバート予約 (ヘルプ)

[コンバート予約]

子チケット

追加

関連するチケット

追加

履歴

阪上 直樹 が 2015/09/25 16:50 に更新

#1

• ステータスを [新規] から [コンバート待ち] に変更

• コンバート予約: サウンド

効果音を修正しました。

【自動】 Redmine API が 2015/09/25 16:52 に更新

#2

• ステータスを [コンバート待ち] から [exe更新待ち] に変更

• コンバート完了: サウンド

• コンバート予約: exe更新

【自動】 Redmine API が 2015/09/25 16:52 に更新

#3

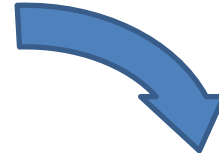
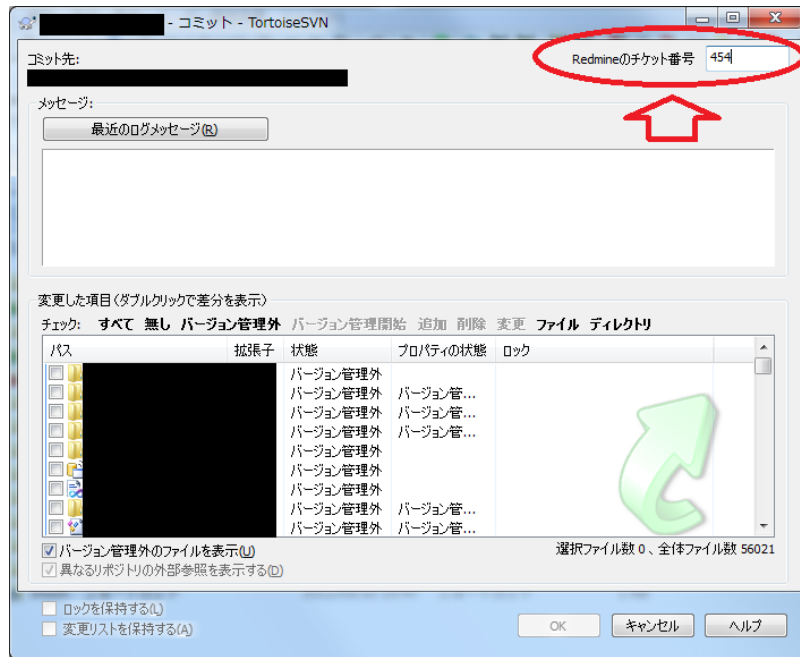
• ステータスを [exe更新待ち] から [確認待ち] に変更

• コンバート完了: exe更新

RedmineとSubversionの連携 (1/2)

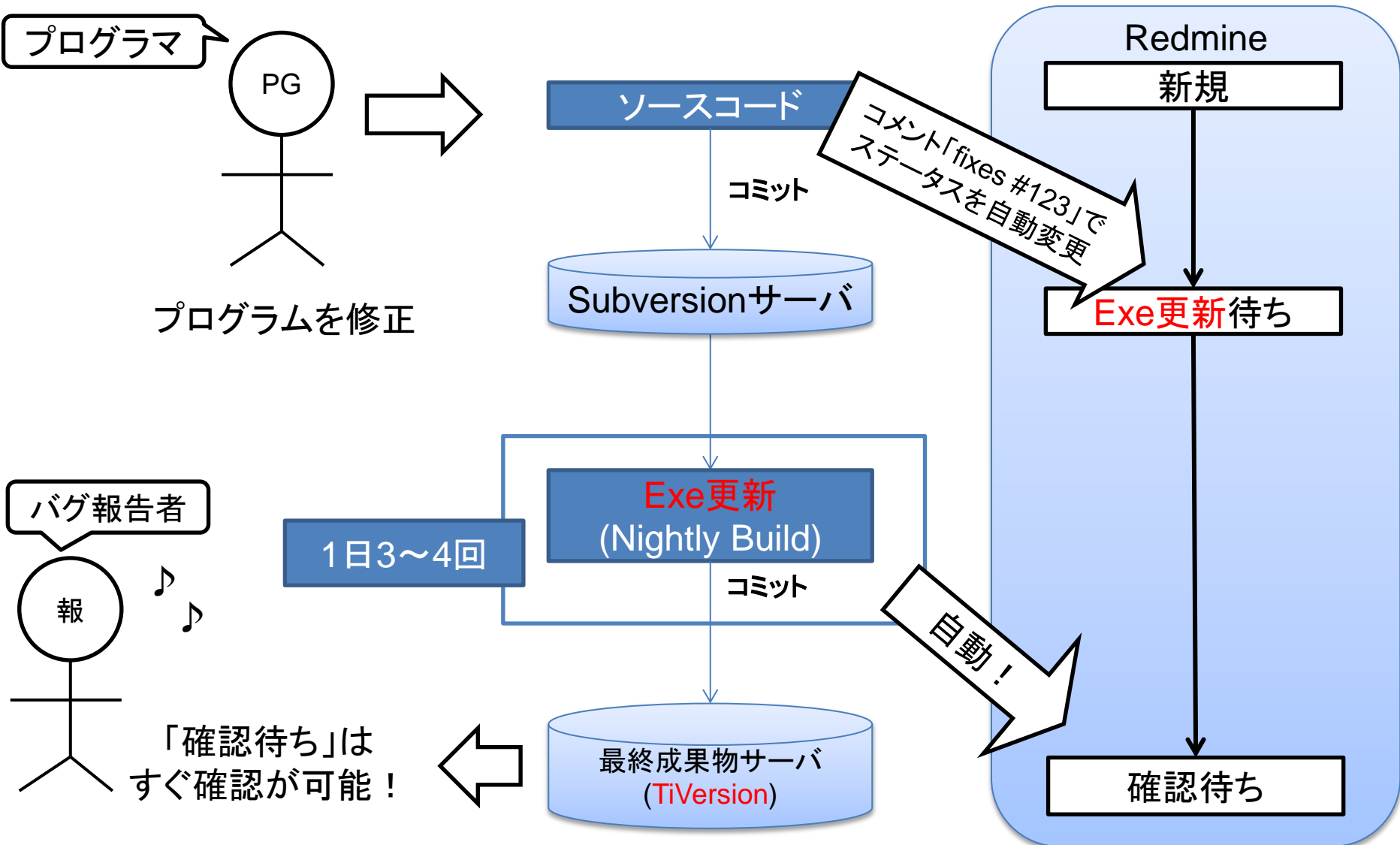
- バグチケットに対して、どういう修正が入ったのか分かりにくい

→Subversionのコミットログにチケット番号を入力



バグチケットに対して、どういう変更が入ったのか
一目瞭然！

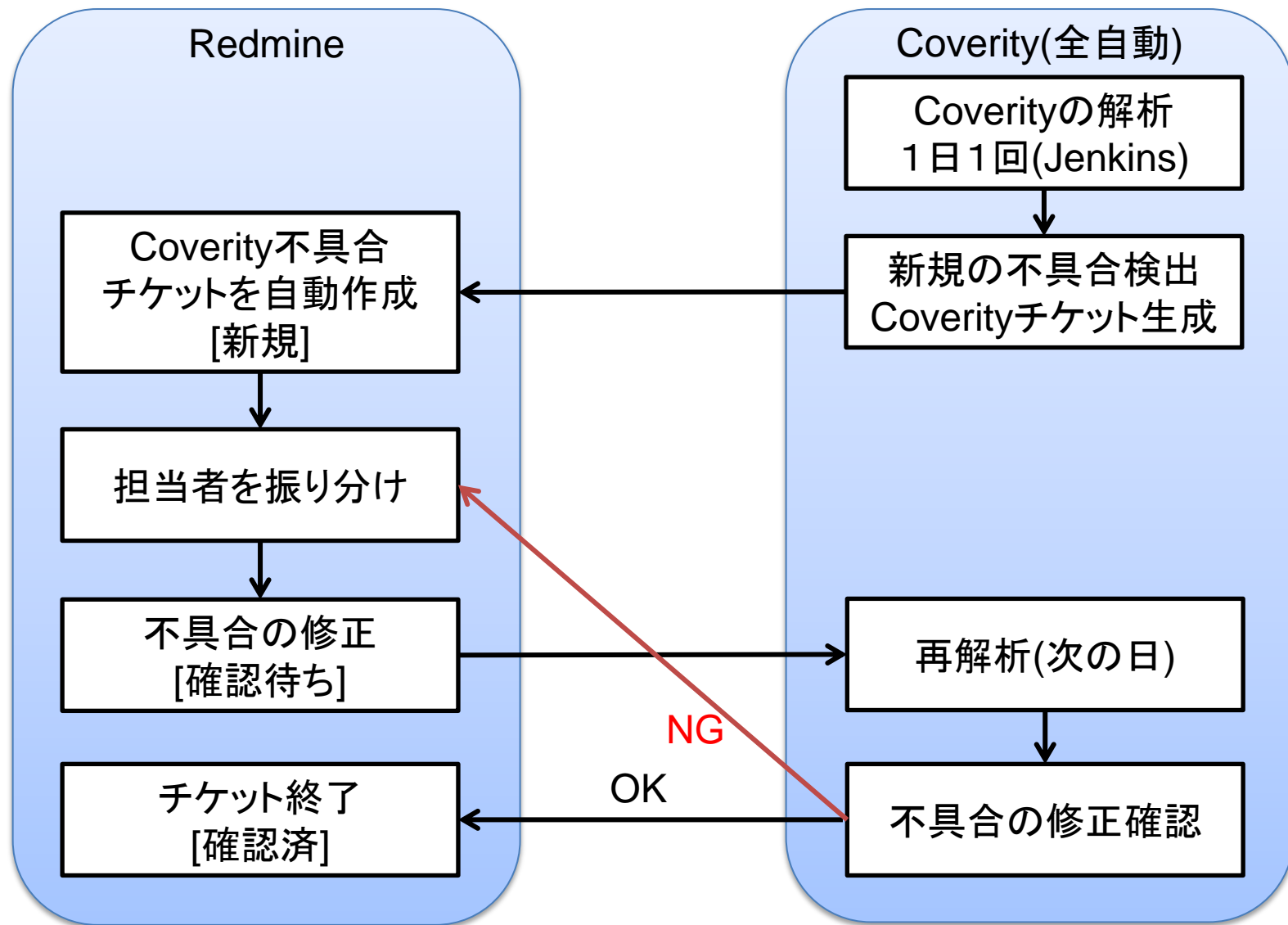
RedmineとSubversionの連携 (2/2)



RedmineとCoverityの連携 (1/2)

- 各ツールにプロジェクトの情報が分散
→ Redmineに情報を集約
- メリット
 - バグやタスクと静的コード解析の不具合を一括管理
 - Redmineのワークフローに統一して作業を効率的に
- 連携方法
 - RedmineとCoverityのそれぞれのAPIを使った同期ツール
 - セガゲームス開発技術部の粉川氏制作

RedmineとCoverityの連携 (2/2)



そびえ立つバグの山を踏破するための 弾丸ワークフロー(バグ管理)のまとめ

- Redmineを使ってバグを一元管理
- バグ報告に必要な情報は自動入力
 - **どこでもバグ報告**
- バグ管理システムと他のシステムの連携
 - ワークフローを自動化
 - **コンバート予約**
 - Subversionと連携
 - Redmineにプロジェクトの情報を集約

本セッションのまとめ

- 大規模 & 高速リリースの秘密は無い
 - 地道な自動化・効率化
- 自動化
 - パイプラインの自動化
 - エラー検出の自動化
 - バグ報告の自動化
 - ワークフローの自動化
 - ためらわず、小さなことから、こつこつと
- 効率化
 - 開発チームの規模や環境・風土に適したツールを導入
 - 情報を1つの場所(Redmine)に集約

最後に

私(QAプログラマ)がいないと
龍が如くは毎年出せない！

参考資料

- 「10ヵ月でHDゲームを開発する方法 ～龍が如くを支えたテクノロジー～」(CEDEC 2010)
 - 前半の高速デバッグ術の関連情報
 - TiVersion
 - エラー送信
 - オートテスト etc..
 - 概要
 - http://cedec.cesa.or.jp/2010/program/PG/C10_P0064.html
 - CEDiL資料
 - https://cedil.cesa.or.jp/cedil_sessions/view/324

【おまけ】バグ管理システム導入の全歴史

- 口頭で
- 紙で管理
- Excelで管理
- オリジナルツールで管理
- Mantisで管理
- Redmineで管理



龍が如くはここ

【おまけ】システム移行の苦悩

- タスクも管理するためにMantisからRedmineに移行
 - あれがない、これがない、使いにくいと言われる
 - 代替手順を提示して、なんとかツールに慣れてもらう
 - どうしても必要と思われる機能は、プラグインとして実装
 - リマインダー送信

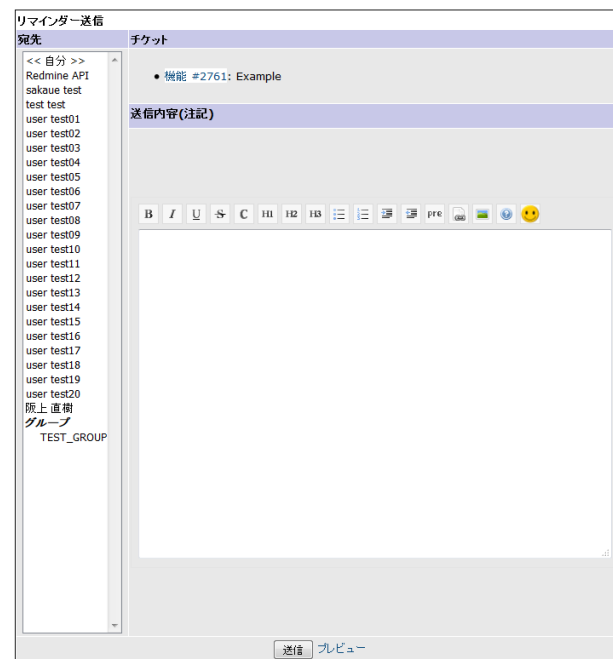
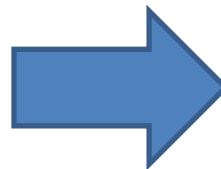
Mantisのリマインダー送信



The screenshot shows the Mantis reminder form. It has a header 'リマインダー送信' and two main sections: '宛先' (Recipient) on the left and '送信内容' (Message Content) on the right. The '宛先' section contains a list of users, with 'user test01' through 'user test20' visible. Below the list is a '送信' (Send) button.

このチケットのフィードバックを要求する受取人に送信します。受取人は、このチケットの監視ユーザーに追加されます。監視を止める場合は、「監視終了」ボタンを押してください。送信内容は、このチケットのコメントに追加されます。

見た目もそのまま
Redmineに移植



The screenshot shows the Redmine reminder form. It has a header 'リマインダー送信' and two main sections: '宛先' (Recipient) on the left and 'チケット' (Ticket) on the right. The '宛先' section contains a list of users, with 'Redmine API', 'sakau test', 'test test', and 'user test01' through 'user test20' visible. Below the list is a '送信' (Send) button. The 'チケット' section contains a '機能 #2761: Example' and a '送信内容(注記)' (Message Content) field with a rich text editor toolbar.

チケットへのリンク付きのメールを指定した宛先に送信し、宛先に指定したユーザーをウォッチャーに追加します。また、送信内容はチケットの注記にも追加されます。

【おまけ】RedmineとExcelの連携

- それでもExcelが必要なときがある
- 数百個のタスクを一括で登録したい
 - redmine_importプラグインを導入
 - インポート用のテンプレートを自動生成
- Excelで見たい人にチケット一覧をxlsファイルに変換
 - redmine_xls_exportプラグインを導入