

「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

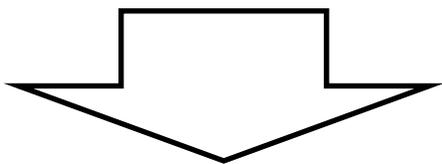
自己紹介

- 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



本セッションについて

- KYUSHU CEDEC 2015 での同名セッション
 - 概要
 - <http://kyushucedec.jp/session.html#06>
 - 発表資料
 - https://cedil.cesa.or.jp/cedil_sessions/view/1398



今回は完全版です！

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

龍が如く 極

PS3/PS4

2016年1月21日発売



複数プラットフォーム×毎年リリース

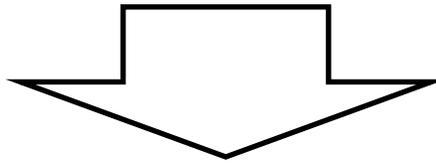
「龍が如く」を短いスパンで
リリースするための秘密は？

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

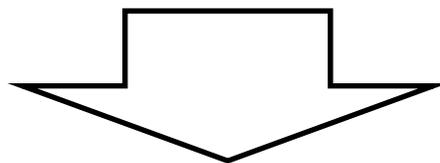


他にもやることはたくさんあるよ！
理想論じゃないの？！

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(難易度/面白さ)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

パイプラインの自動化

- ビルド
- コンバート

ビルドの自動化 (1/2)

- ビルドとは

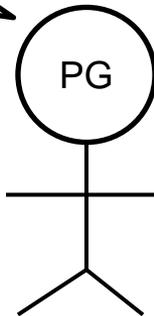
- プログラムをコンパイル・リンクして実行可能な形式に変換すること
- 自動で毎日ビルドした成果物は、Nightly Buildと呼ばれる

- **Exe更新**とは

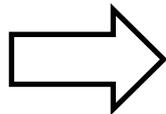
- Nightly Buildの龍が如くチーム特有の呼称
- PS3/PS4/PS Vitaに加えて、それぞれのターゲットに対応したデバッグ用のWindows版の実行ファイルを定期的に生成
- Windows版もあることから、「**Exe更新**」と呼んでいる

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

自動化

Exe更新
(Nightly Build)

コミット



最終成果物サーバ
(TiVersion)

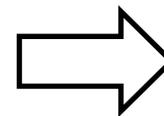
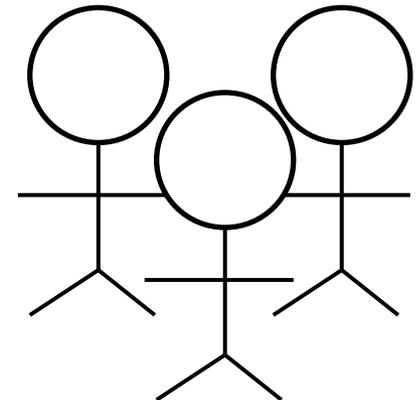
Subversionとは

- 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

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

プロジェクト全員が
最新コードの動作を確認可能



バイナリ管理を分けている理由

テキスト

バイナリ

数
KB

10MB~100MB
(1GBを超えも！)

ソース(*.cpp)

音楽
(BGM/効果音)

ムービー

絵
(3Dキャラ/背景/UI)

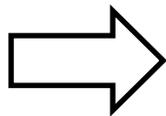
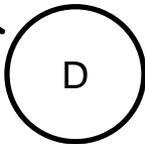
マージできなくていい！とにかく高速にやりとりしたい！

パイプラインの自動化

- ビルド
- コンバート

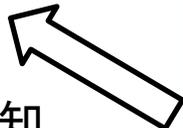
コンバートの自動化

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

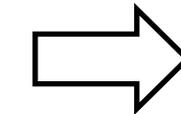
最終データサーバ(TiVersion)

Exe更新

(Nightly Build)

コミット

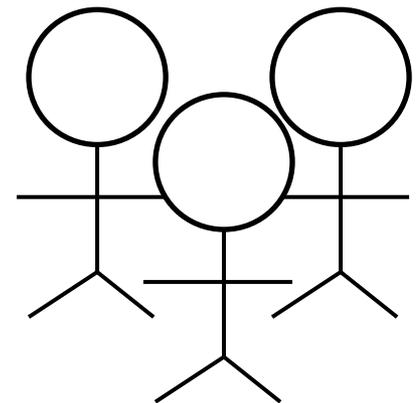
最終成果物サーバ
(TiVersion)



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



ゲーム上で動作確認

- 実際にゲームを起動して、データが正しく読み込まれ、動作しているかを確認する簡易なテスト
 1. ゲームをテストモードで起動
 2. 背景データを読み込み
 3. 背景データだけを描画
 4. エラーが出ないことを確認し、終了して成功コードを返す

パイプライン(ビルド・コンバート)の自動化



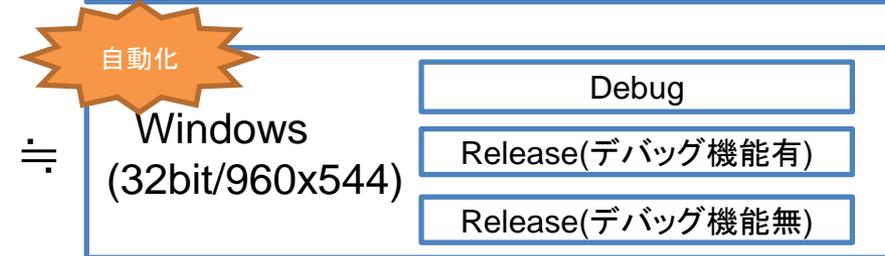
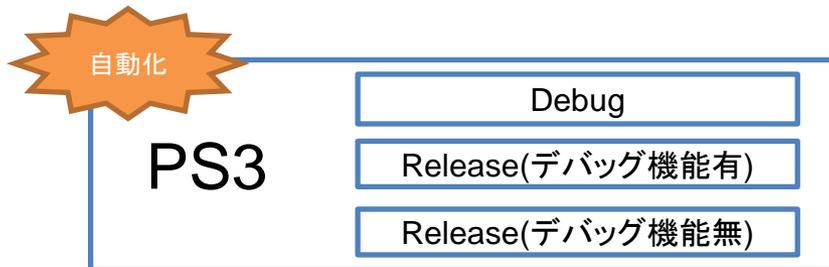
エラー検出の自動化

- 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

エラー検出の自動化

- 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

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)

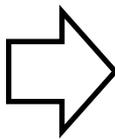
警告はエラーです

- 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

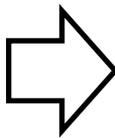
atsu_takashi

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

エラー検出の自動化

- 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

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

エラー検出の自動化

- 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

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

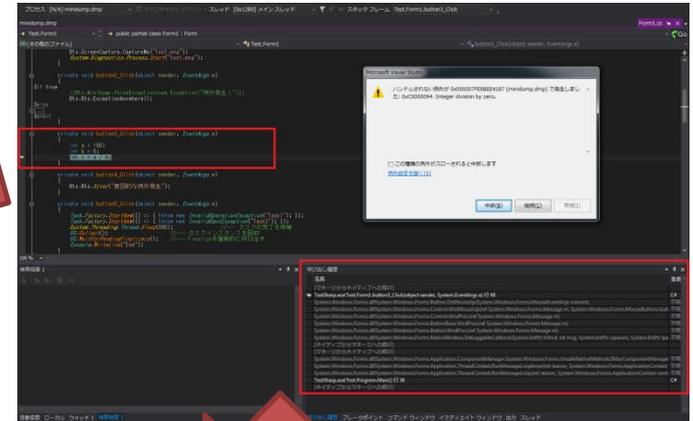
バグ管理システム
(Redmine)

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;
}
```



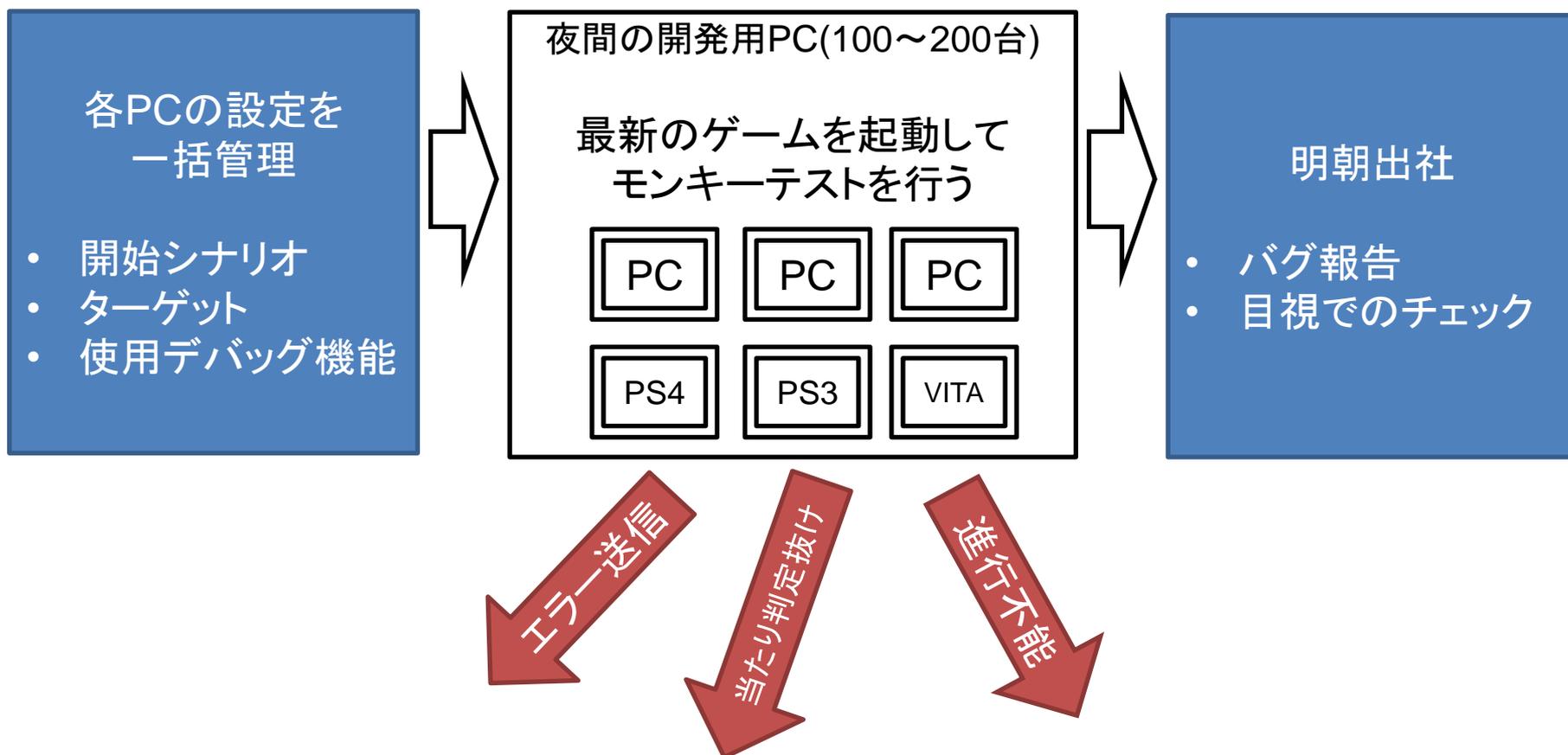
呼び出し履歴
名前
[マネージからネイティブへの移行]
TestSharp.exe!Test.Form1.button3_Click(object sender, System.EventArgs e) 行 49
System.Windows.Forms.dll!System.Windows.Forms.Button.OnMouseUp(System.Windows.Forms.MouseEventArgs mevent)
System.Windows.Forms.dll!System.Windows.Forms.Control.WmMouseUp(ref System.Windows.Forms.Message 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]

エラー検出の自動化

- 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

テストの自動化 (1/4)

- 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)



テストの自動化 (2/4)

- Excelファイルに設定を記述

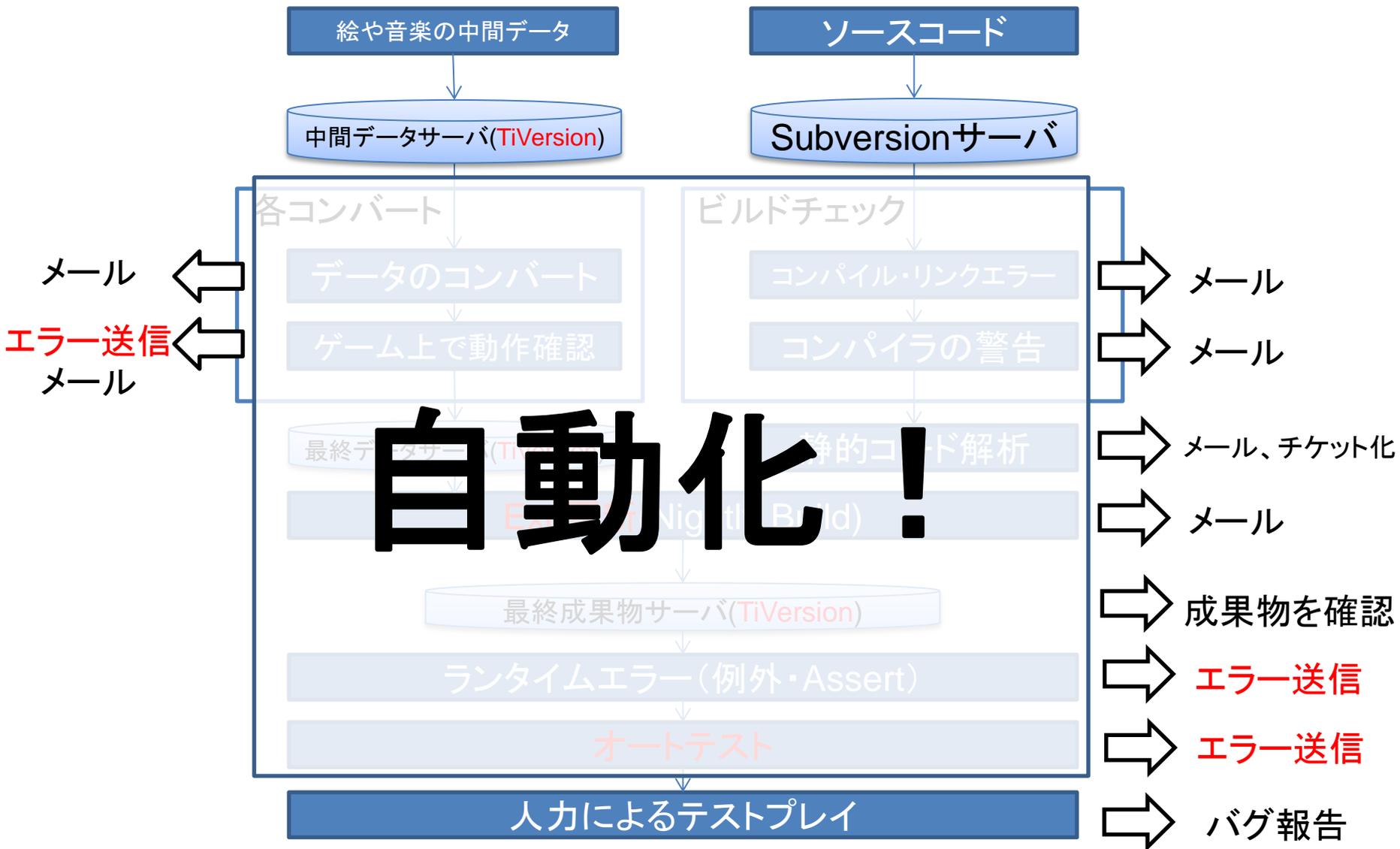
条件

種類	シナリオチャプター	シナリオタイトル	AUTO	dip SYSTEM	その他										
DIPカテゴリ															
PC名	開始シナリオ														
名前	シナリオチャプター	シナリオタイトル	自動スタート	W92	W84	PSS	PS4	VITA	ムービーキャプチャ	ランダムイベント	非アクティブ時も実行する	パッドを振動させない	主人公無敵	敵無敵	最後にdps_logを送る
名無し用	PADV	プレミアムアドベンチャー	<input type="checkbox"/>												
name01	PADV	プレミアムアドベンチャー	<input type="checkbox"/>												
name02	PADV	プレミアムアドベンチャー	<input type="checkbox"/>												
name03	PADV	プレミアムアドベンチャー	<input type="checkbox"/>												
name04	PADV	闘技場	<input type="checkbox"/>												
name05	PADV	闘技場	<input type="checkbox"/>												
name06	PADV	闘技場	<input type="checkbox"/>												
name07	PADV	闘技場	<input type="checkbox"/>												
name08	第一章	第一章 タイトル表示	<input type="checkbox"/>												
name09	第一章	第一章 タイトル表示	<input type="checkbox"/>												
name10	第一章	第一章 タイトル表示	<input type="checkbox"/>												
name11	第二章	第二章 タイトル表示	<input type="checkbox"/>												
name12	第二章	第二章 タイトル表示	<input type="checkbox"/>												
name13	第二章	第二章 タイトル表示	<input type="checkbox"/>												
name14	第三章	第三章 タイトル表示	<input type="checkbox"/>												
name15	第三章	第三章 タイトル表示	<input type="checkbox"/>												
name16	第三章	第三章 タイトル表示	<input type="checkbox"/>												

テストの自動化 (4/4)

- テストで検出されたバグを見逃さない
 - エラー送信
 - ログ出力
 - 動画撮影
- 人力で出せないバグの検出
 - 1/60秒単位の特定のパッド入力で発生
 - 長時間同じ動作を繰り返すことで発生
- 人力で出せないバグを見つける必要があるのか？
 - 数十人では出ないバグも、数十万人がプレイすると出てしまう可能性がある
 - 人力のチェックはリソースに限りがあるため、バグを自動で検出する仕組みは、大規模ゲーム開発には欠かせない

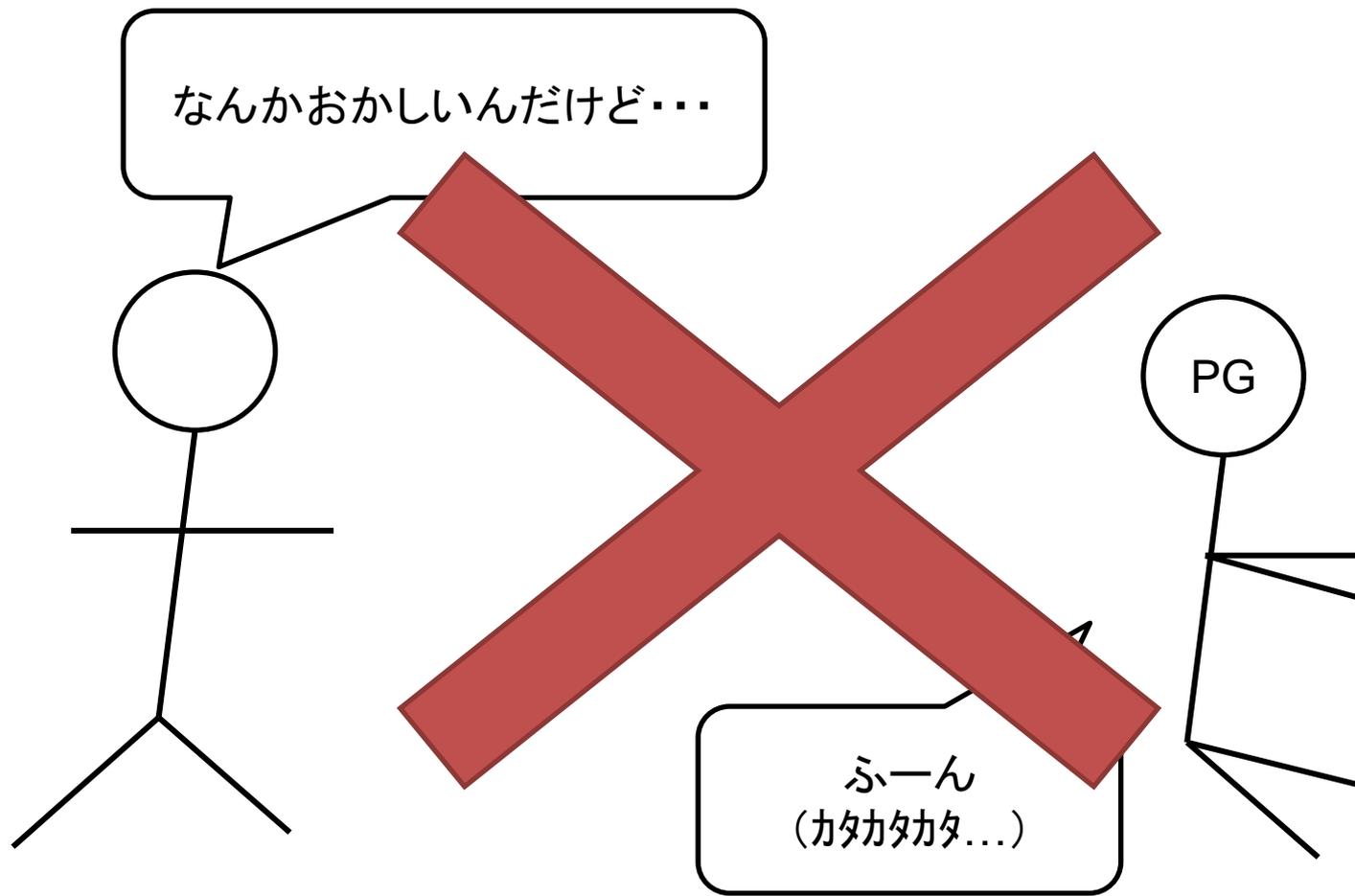
高速デバッグ術(自動化)のまとめ



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

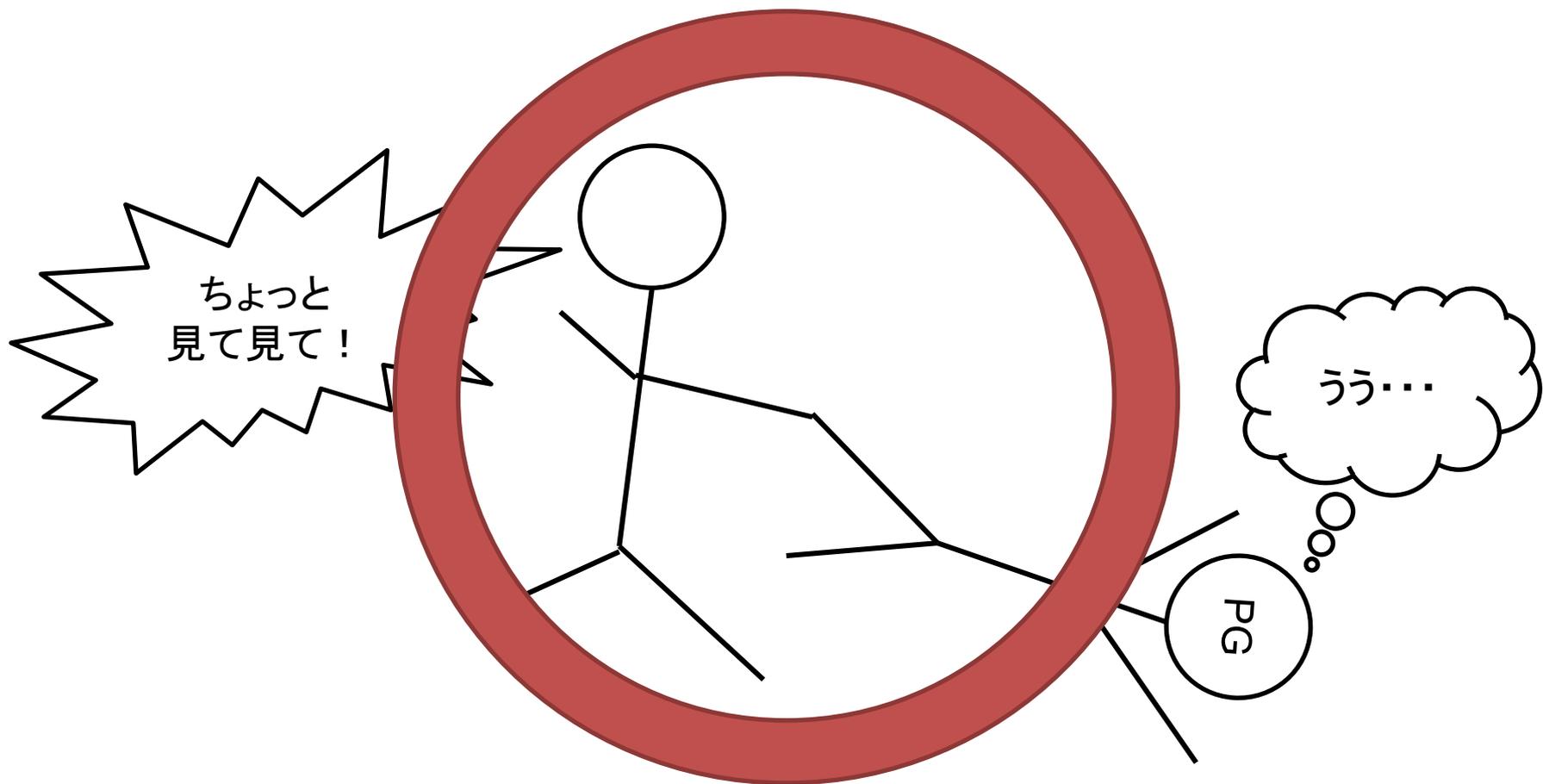
- バグ管理
 - Redmineを使用
 - バグ報告の掟
 - バグ管理システムの導入の歴史
 - バグ管理システムの問題と解決

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

バグ管理システムの導入の歴史

- ~~口頭だけ~~
- 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(現在)

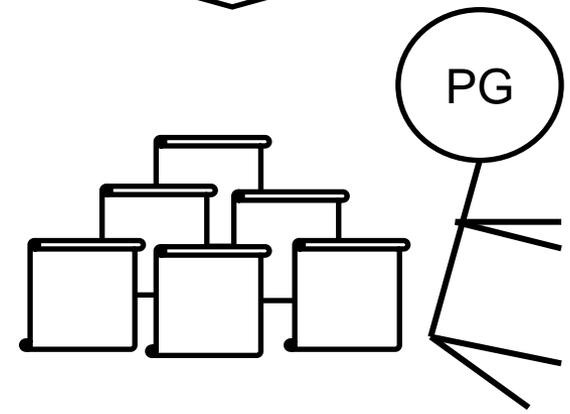
バグ管理システムの導入の歴史

- ~~口頭だけ~~
- **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(現在)~~

紙でバグ管理 (1/2)

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	2016/3/8 14:00
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。	



紙でバグ管理 (2/2)

- 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が分からない

バグ管理システムの導入の歴史

- ~~口頭だけ~~
- 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(現在)

Excelでバグ管理

※Office 365がなかったころの話

- メリット
 - 情報のデジタル化
 - 共有設定にして、1ファイルですべてのバグを管理
- デメリット
 - 誰が編集したのか分からない
 - 共有設定にすると、ファイルがたびたび壊れる
 - 部外とのやり取りが煩雑
 - マージ作業が発生してしまう
 - 情報が漏洩しやすい
 - バグのステータスが分からない

バグ管理システムの導入の歴史

- ~~口頭だけ~~
- 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(現在)

オリジナルツールでバグ管理

- メリット
 - サーバで一元管理
 - ツールの開発者が社内にいるので、要望を追加しやすい
- デメリット
 - ツールの属人化
 - 開発者がいなくなったらメンテナンスできない
 - ツールの学習コスト
 - 社内システムなので、ツールに慣れてもらう時間が必要

バグ管理システムの導入の歴史

- ~~口頭だけ~~
- 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(現在)

Mantisでバグ管理(1/2)

- Mantisとは
 - バグ管理システム(BTS)
 - Webブラウザでアクセス
 - オープンソース



利用者名: administrator (admin - 管理者) 2015-09-25 17:09 JST プロジェクト: MantisConnectTest [変更]

[メイン](#) | [マイビュウ](#) | [検索](#) | [登録](#) | [変更履歴](#) | [ロニドマップ](#) | [要約](#) | [ドキュメント](#) | [システム管理](#) | [アカウント設定](#) | [ログアウト](#)

チケット # ID指定

▲自分用のお知らせ情報などを閉じる

担当者の場合はステータスを"対応待ち"にして開始日、終了日を入力してください。担当者違いの場合は、適切な担当者に回して下さい【^】(0 - 0 / 1)

#	カテゴリ	重要度	担当者	要約	開始日	終了日	最終変更日
0001341	テストカテゴリ1	B	admin	MantisConnectテスト5	2013-03-14	2013-03-14	2014-02-10 17:47

参照履歴: 0001344

報告者:	監視者:	担当者:	カテゴリ:	重要度:	解決状況:	プロフィール:
どれでも	どれでも	どれでも	どれでも	どれでも	どれでも	どれでも
ステータス:	ステータス非表示:					優先度:
どれでも	クローズド (とそれ以上)					どれでも
表示数:	公開:	チケット (Sticky) の表示:	重要 (時間):	日付フィルターの適用:	簡潔:	
50	どれでも	はい	6	いいえ	どれでも	
プラットフォーム:	OS:	バージョン:	タグ:			
どれでも	どれでも	どれでも	どれでも			
シーンNo	終了日	開始日				
どれでも	どれでも	どれでも				
コメント登録者:	どれでも	ソート順:	更新日 降順			

 [\[詳細フィルター \]](#) [\[固定リンクの作成 \]](#)

チケット一覧 (1 - 50 / 53) [一括印刷] [CSV エクスポート] [Excelエクスポート] [グラフ] [先頭] [12 ページ]

	P	ID	#	カテゴリ	重要度	担当者	ステータス	更新日	要約	開始日	終了日
<input type="checkbox"/>		0001341		テストカテゴリ1	B	admin	担当者決定	2014-02-10 17:47	MantisConnectテスト5	2013-03-14	2013-03-14
<input type="checkbox"/>		0001344	1	テストカテゴリ1	B	admin	確認済	2014-02-07 14:15	MantisConnectテスト8	2013-03-14	2013-03-14
<input type="checkbox"/>		0001342	1	テストカテゴリ1	B	admin	確認済	2014-02-07 14:15	MantisConnectテスト6	2013-03-14	2013-03-14
<input type="checkbox"/>		0001340		テストカテゴリ1	B		未着手	2013-03-14 18:04	MantisConnectテスト4	2013-03-14	2013-03-14
<input type="checkbox"/>		0001337		テストカテゴリ1	B		未着手	2013-03-14 18:04	MantisConnectテスト1	2013-03-14	2013-03-14
<input type="checkbox"/>		0001338		テストカテゴリ1	B		未着手	2013-03-14 18:04	MantisConnectテスト2	2013-03-14	2013-03-14
<input type="checkbox"/>		0001339		テストカテゴリ1	B		未着手	2013-03-14 18:04	MantisConnectテスト3	2013-03-14	2013-03-14
<input type="checkbox"/>		0001336		テストカテゴリ1	B		未着手	2013-03-14 18:04	MantisConnectテスト0	2013-03-14	2013-03-14
<input type="checkbox"/>		0001335		テストカテゴリ1	B		未着手	2013-03-14 17:56	MantisConnectテスト9	2013-03-14	2013-03-14
<input type="checkbox"/>		0001333		テストカテゴリ1	B		未着手	2013-03-14 17:56	MantisConnectテスト7	2013-03-14	2013-03-14
<input type="checkbox"/>		0001334		テストカテゴリ1	B		未着手	2013-03-14 17:56	MantisConnectテスト8	2013-03-14	2013-03-14

Mantisでバグ管理(2/2)

- メリット
 - 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
 - 有名なツールなので、学習コストを削減できる
- デメリット
 - バグ以外の管理には不向き(タスクなど)

バグ管理システムの導入の歴史

- ~~口頭だけ~~
- 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(現在)

Redmineでバグ管理 (1/5)

- 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

ID	トラッカー	ステータス	優先度	題名	担当者	更新日
1177	機能	コンバート待ち	通常	testttt	[TST] 阪上 直樹	2015/07/15 13:54
1176	機能	新規	通常	インポートテスト2		2015/07/15 13:36
1175	機能	新規	通常	インポートテスト1		2015/07/09 18:31
1174	機能	新規	通常	インポートテスト子		2015/07/09 18:31
1173	バグ	新規	通常	インポートテスト親		2015/07/09 18:31
1172	バグ	新規	通常	sdvds	sokaue test	2015/07/09 18:31
1171	バグ	対応中	通常	safsf	sokaue test	2015/07/09 18:31
1170	バグ	新規	通常	TESTTT	sokaue test	2015/07/09 18:31
1169	機能	新規	通常	TESTTT	[TST] 阪上 直樹	2015/07/09 18:31
1168	バグ	新規	通常	99999	[TST] 阪上 直樹	2015/07/09 18:31
1167	バグ	新規	通常	9999999	[TST] 阪上 直樹	2015/07/09 10:47
1166	バグ	exe更新待ち	通常	TESTTTTTT	[TST] 阪上 直樹	2015/07/06 15:14
1094	バグ	差し戻し	通常	TEST		2015/06/22 10:20
1093	機能	新規	通常	Example		2015/06/18 15:18
1092	機能	新規	通常	Example		2015/06/18 15:09
50	バグ	新規	通常	Example		2015/06/18 14:29
45	機能	新規	急いで	Example		2015/06/17 16:18
43	機能	新規	通常	REST TESTTT		2015/06/17 16:08
42	機能	新規	通常	REST TESTTT		2015/06/17 16:08
41	バグ	新規	通常	REST TESTTT		2015/06/17 16:07
40	バグ	新規	急いで	Example		2015/06/17 16:06
39	機能	新規	急いで	Example		2015/06/17 16:06
38	バグ	新規	急いで	Example		2015/06/17 16:06
37	バグ	新規	通常	REST TESTTT		2015/06/17 16:03

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:	神室可
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

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 (2/5)

- 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

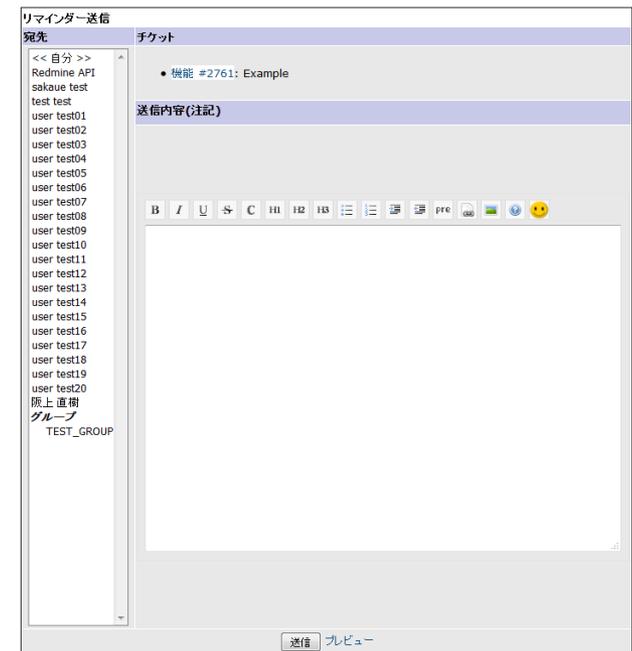
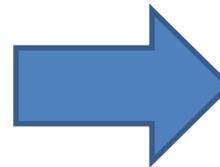
Mantisのリマインダー送信



The screenshot shows the 'Reminder Email' form in Mantis. It has two main sections: 'To' (宛先) and 'Message' (送信内容). The 'To' section is currently empty. The 'Message' section is a large text area. A 'Send' (送信) button is located at the bottom right of the form.

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 'Reminder Email' form in Redmine. It has two main sections: 'To' (宛先) and 'Message' (送信内容). The 'To' section contains a list of users (user test01 to user test20) and a group (TEST_GROUP). The 'Message' section is a large text area with a rich text editor toolbar above it. A 'Send' (送信) and 'Preview' (プレビュー) button are located at the bottom right of the form.

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。

Redmineでバグ管理 (3/5)

- メリット

- 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
- 有名なツールなので、学習コストを削減できる
- 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

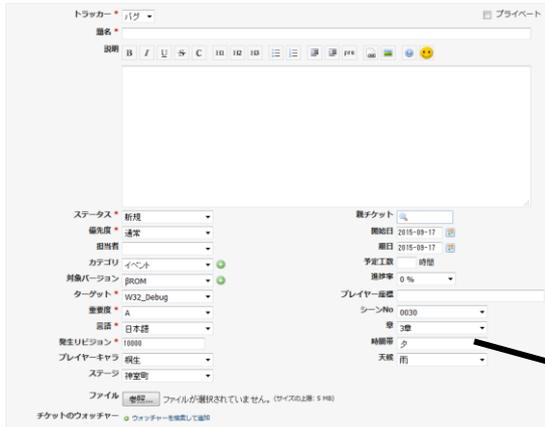
- デメリット

- 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 (4/5)

バグのステータスの一例

新しいチケット



バグ報告

新規

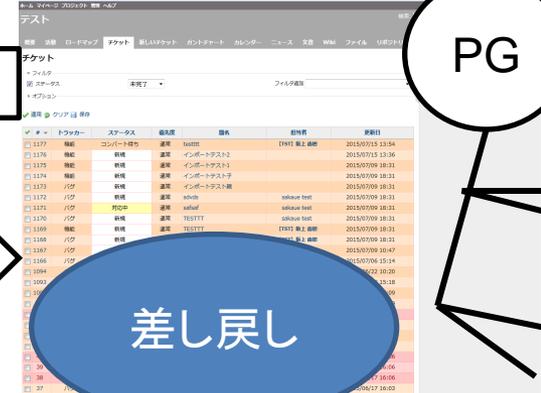


担当者に
振分

確認待ち

修正

PG



修正NG

差し戻し

確認済み

修正確認済

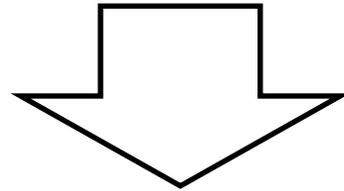
ワークフローは紙と同じ！

Redmineでバグ管理 (5/5)

- 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
 - Excel使いたいよ・・・の声
 - 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

バグ管理システムの問題と解決

- Excel使いたいよ・・・の声
- 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

Excel使いたいよ・・・の声 (1/2)

- RedmineにExcelのインポート&エクスポートを追加
- 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
 - Redmine 3.2で標準機能に追加
- 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

Excel使いたいよ・・・の声 (2/2)

- インポート用のテンプレートを作成
 - Redmine REST API経由でトラッカー別に自動生成
 - Jenkinsで定期的に更新
 - カテゴリやメンバーの追加に対応するため

	A	B	C	D	E	F	G	H	I
1	トラッカー	題名	説明	優先度	担当者	カテゴリ	対象バージョン	開始日	期日
2	タスク	タスクテスト1	タスクテスト1の説明です。	高め	阪上 直樹	イベント	α ROM	2016/1/1	2016/2/1
3	タスク	タスクテスト2	タスクテスト2の説明です。	高め	sakaue test	イベント	α ROM	2016/1/2	2016/2/2
4	タスク	タスクテスト3	タスクテスト3の説明です。	高め	阪上 直樹	プログラム	α ROM	2016/1/3	2016/2/3
5	タスク	タスクテスト4	タスクテスト4の説明です。	通常	阪上 直樹	プログラム	β ROM	2016/1/4	2016/2/4
6	タスク	タスクテスト5	タスクテスト5の説明です。	通常	阪上 直樹	プログラム	β ROM	2016/1/5	2016/2/5
7	タスク	タスクテスト6	タスクテスト6の説明です。	通常	阪上 直樹	プログラム	β ROM	2016/1/6	2016/2/6
8	タスク	タスクテスト7	タスクテスト7の説明です。	通常	阪上 直樹	プログラム	α ROM β ROM	2016/1/7	2016/2/7
9	タスク	タスクテスト8	タスクテスト8の説明です。	通常	阪上 直樹	プログラム	Final ROM	2016/1/8	2016/2/8
10	タスク	タスクテスト9	タスクテスト9の説明です。	通常	阪上 直樹	プログラム	Master ROM	2016/1/9	2016/2/9
11	タスク	タスクテスト10	タスクテスト10の説明です。	通常	阪上 直樹	プログラム	Final ROM	2016/1/10	2016/2/10
12	タスク	タスクテスト11	タスクテスト11の説明です。	通常	阪上 直樹	プログラム	Master ROM	2016/1/11	2016/2/11
13	タスク	タスクテスト12	タスクテスト12の説明です。	通常	阪上 直樹	プログラム	Master ROM	2016/1/12	2016/2/12
14	タスク	タスクテスト13	タスクテスト13の説明です。	通常	阪上 直樹	プログラム	Master ROM	2016/1/13	2016/2/13

バグ管理システムの問題と解決

- Excel使いたいよ・・・の声
- **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

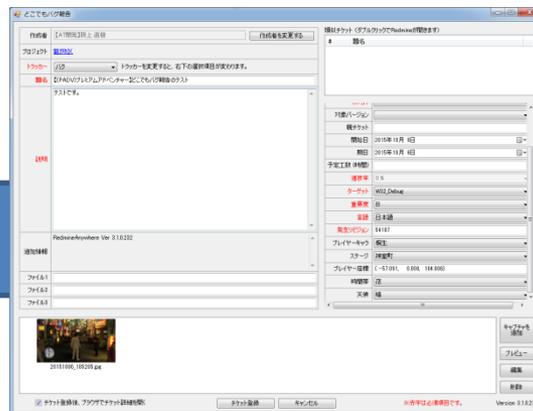
龍が如く独自

- 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
発生バージョン: 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.bxt (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
龍が如く 極	1,925件	4,799件

3タイトル(9,338件+6,307件+1,925件) × 10分(短縮時間/件)
= 約**3,000**時間を削減

バグ報告の質が均一化して、精度も向上

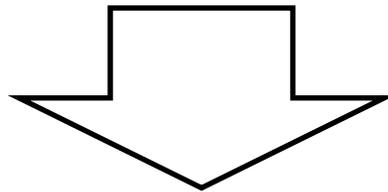
テストプレイヤーは、バグ探しに専念！
クリエイターは、バグの修正・バランス調整・クオリティの向上に専念！

バグ管理システムの問題と解決

- Excel使いたいよ・・・の声
- 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

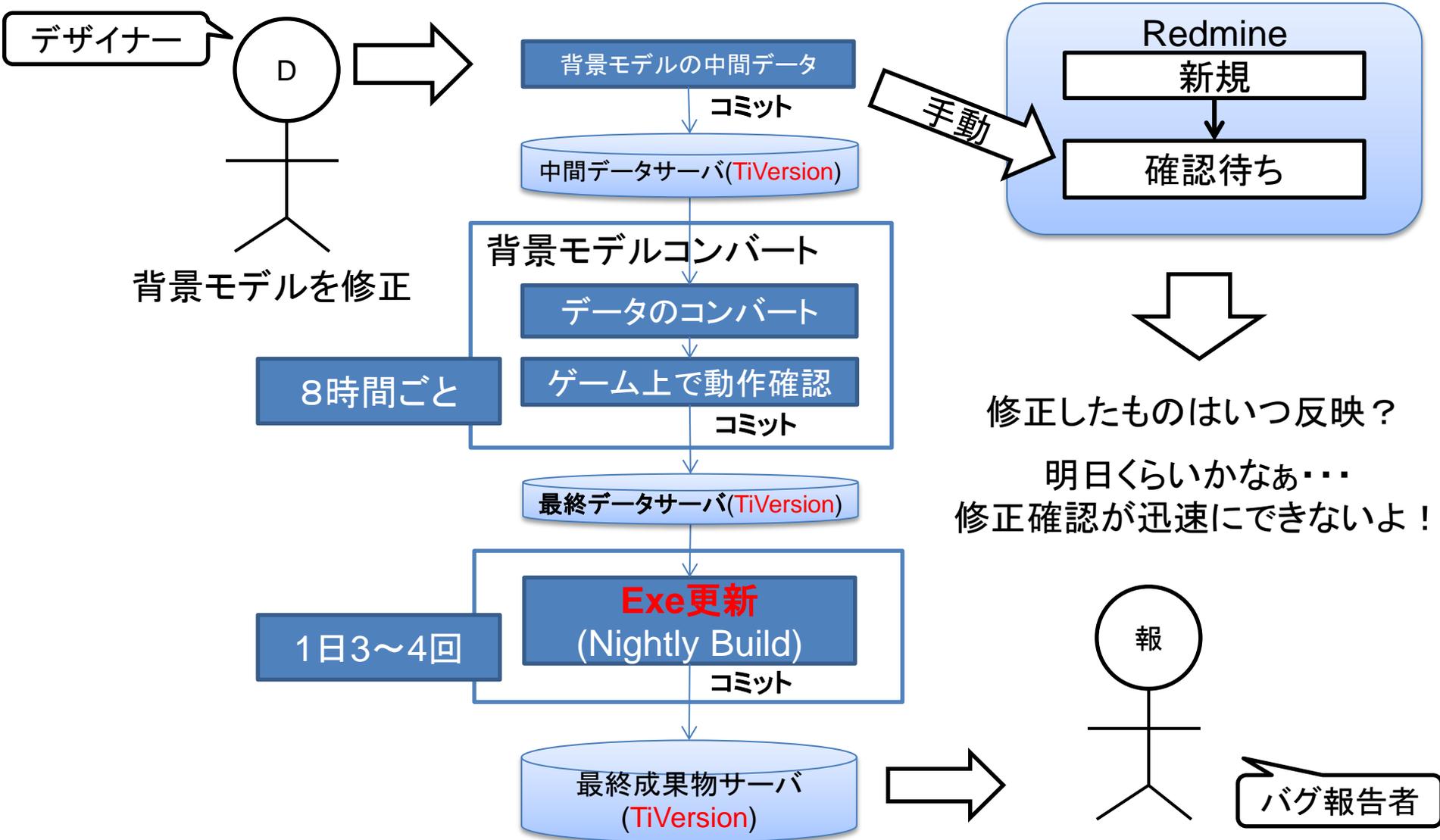
バグのワークフローの問題点

- 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

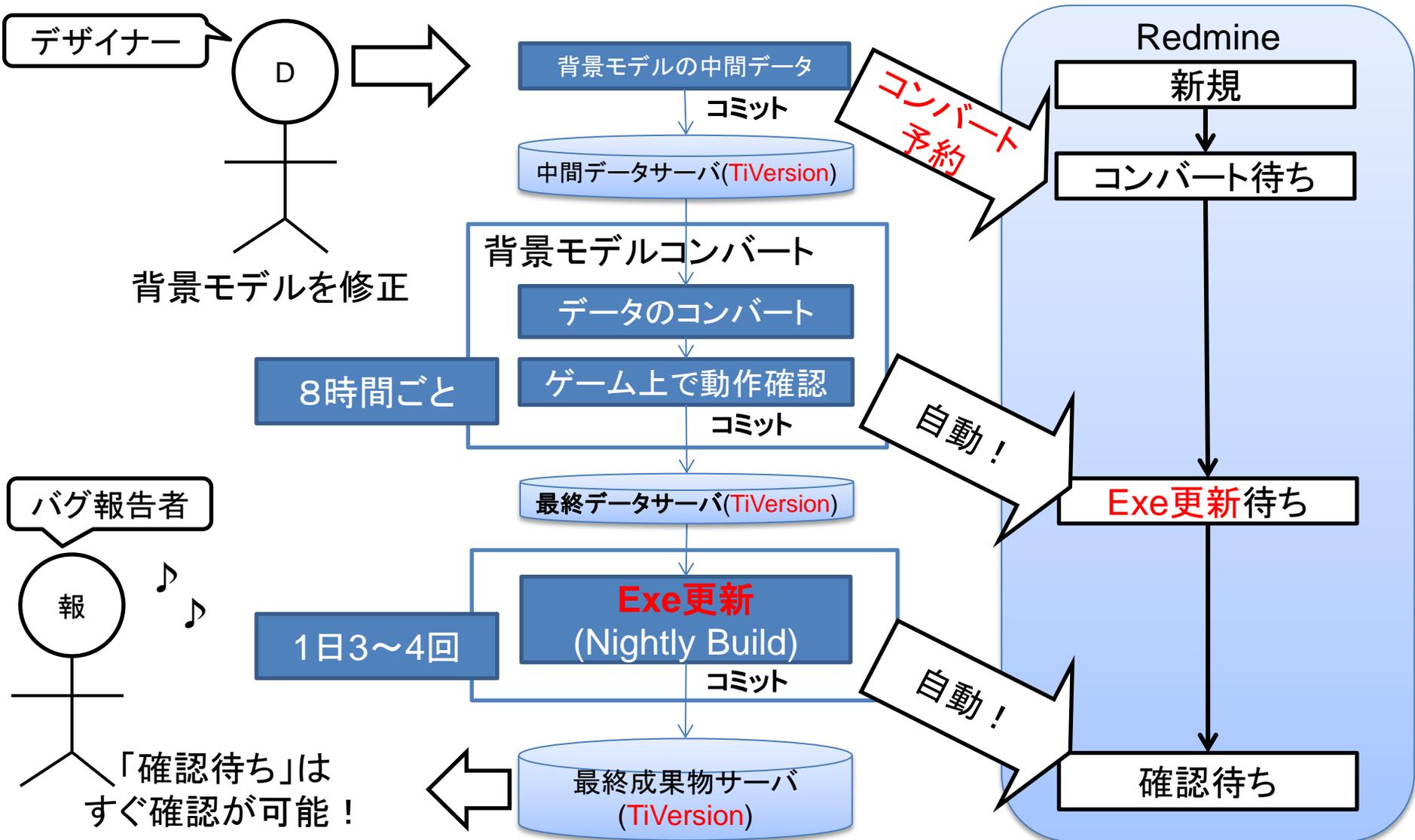


Redmineと各システム・ツールを連携して
ワークフローを自動化

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

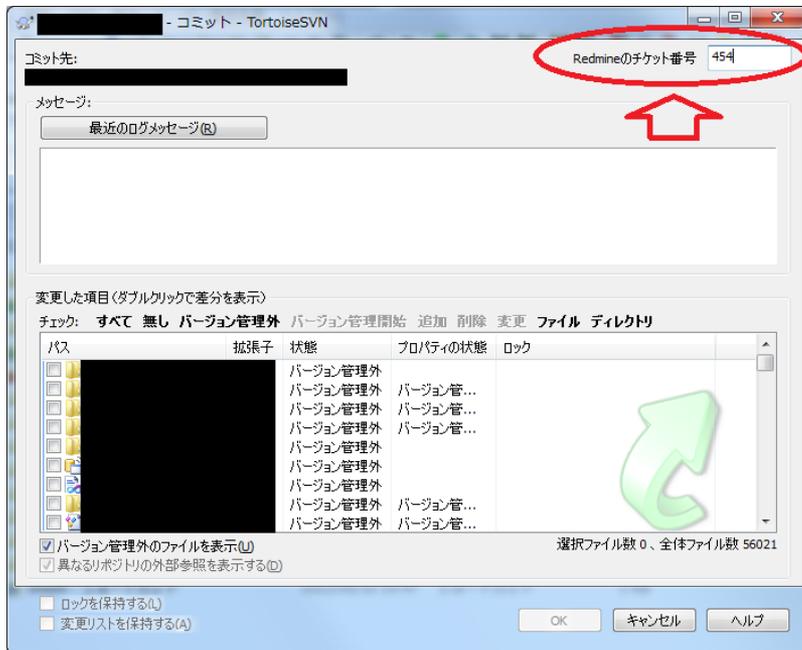
• ステータスを [exe更新待ち] から [確認待ち] に変更

• コンバート完了: exe更新

RedmineとSubversionの連携 (1/2)

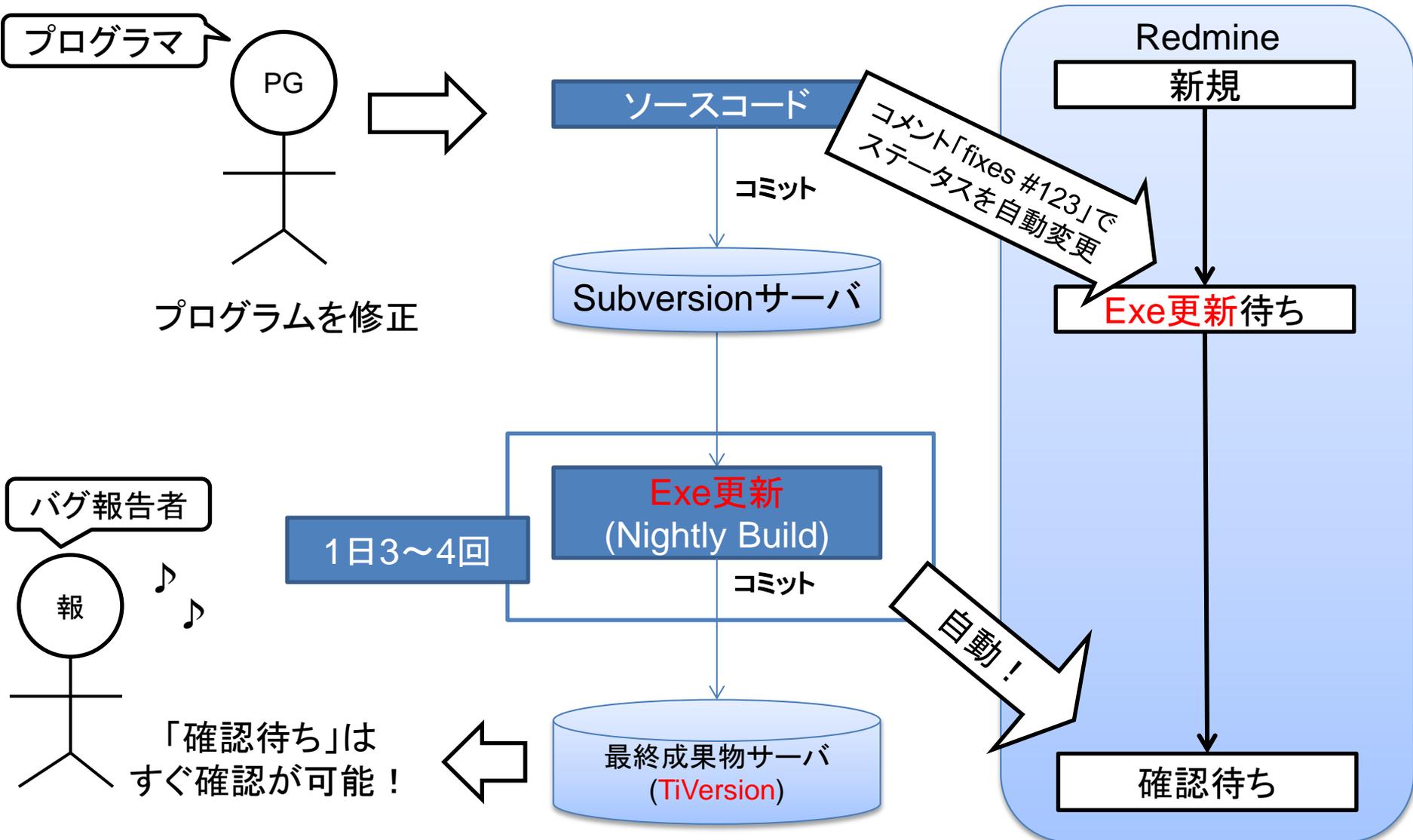
- バグチケットに対して、どういう修正が入ったのか分かりにくい

→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)

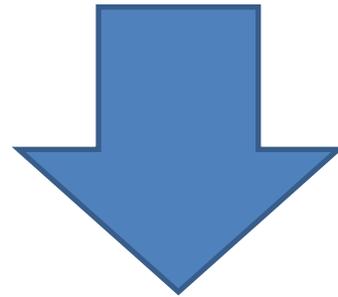


バグ管理システムの問題と解決

- Excel使いたいよ・・・の声
- 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

各ツールにプロジェクトの情報が分散

- 様々なツールを行き来して、作業が煩雑になる
- バグとタスクの管理を分けると、チケット全体の優先順位が把握できなくなる

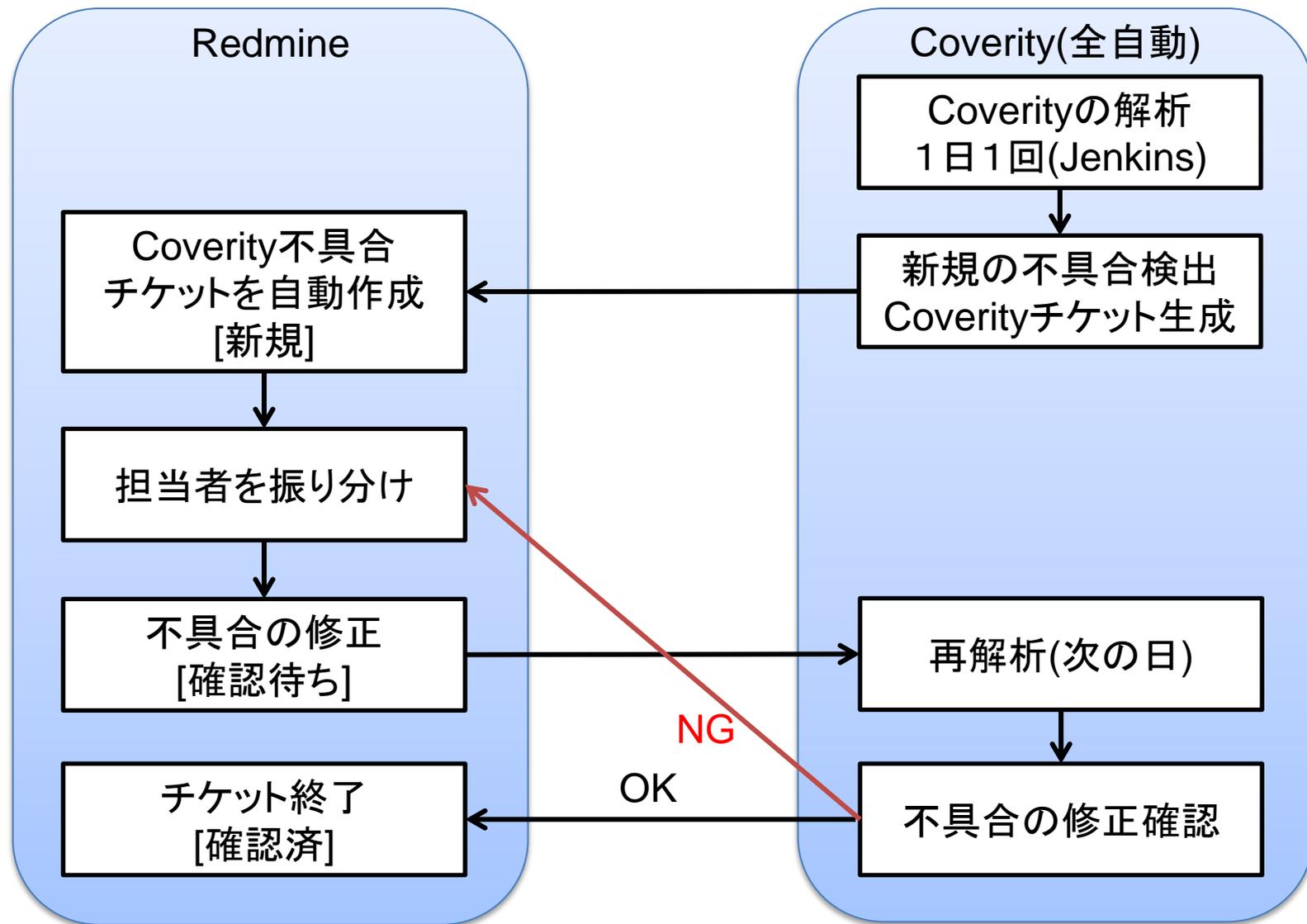


Redmineに情報を集約

RedmineとCoverityの連携 (1/2)

- 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
 - 講演資料(CEDiL)
 - https://cedil.cesa.or.jp/cedil_sessions/view/324