CEDEC
CESA Developers Conference
2010

タダで始める
ゲーム開発自動化のススメ

株式会社 セガ
AM開発技術部
粉川貴至(Takashi KOKAWA)

どういうセッション？

自動ビルド

コーディング規約

ロギング

情報集約

CI

静的解析

オープンソース

バージョン管理

ソフトウェア開発環境改善

自動化

Hudson

自動テスト

コードメトリクス

ドキュメント自動生成

自動動作チェック

ゲーム作りの「環境」を良くするための話をします

※ゲームが自動でできるセッションではありません！

これから話す方法は・・・

- めんどくさくない？

  →簡単にできます！

  世の中にあるツールを使って楽に！

- 高いんじゃない？

  →タダで始められる方法があります！

  それをこれから紹介します

受講対象者

- 基本的にプログラマ向け
  - ツールの動かし方・活用事例まで紹介します！
  - ですが、
- デザイン
  - デザインデータのコンバート等で自動化している部分があれば、統合できれば便利
- 企画・マネジメント
  - 開発サイクルがスムーズに！

～目次～

- ゲーム開発環境
  - 開発環境診断
  - CI（継続的インテグレーション）
- CI導入の実例
  - CIツール「Hudson」の紹介
  - デモ、導入事例
    - 自動化ツール「AutoIt」の紹介
  - カスタム・他のツールとの連携
- タダで始めよう！
- まとめ

～目次～

- ゲーム開発環境
  - 開発環境診断
  - CI（継続的インテグレーション）
- CI導入の実例
  - CIツール「Hudson」の紹介
  - デモ、導入事例
    - 自動化ツール「AutoIt」の紹介
  - カスタム・他のツールとの連携
- タダで始めよう！
- まとめ

日頃こんな事起こっていませんか？

- 質問です

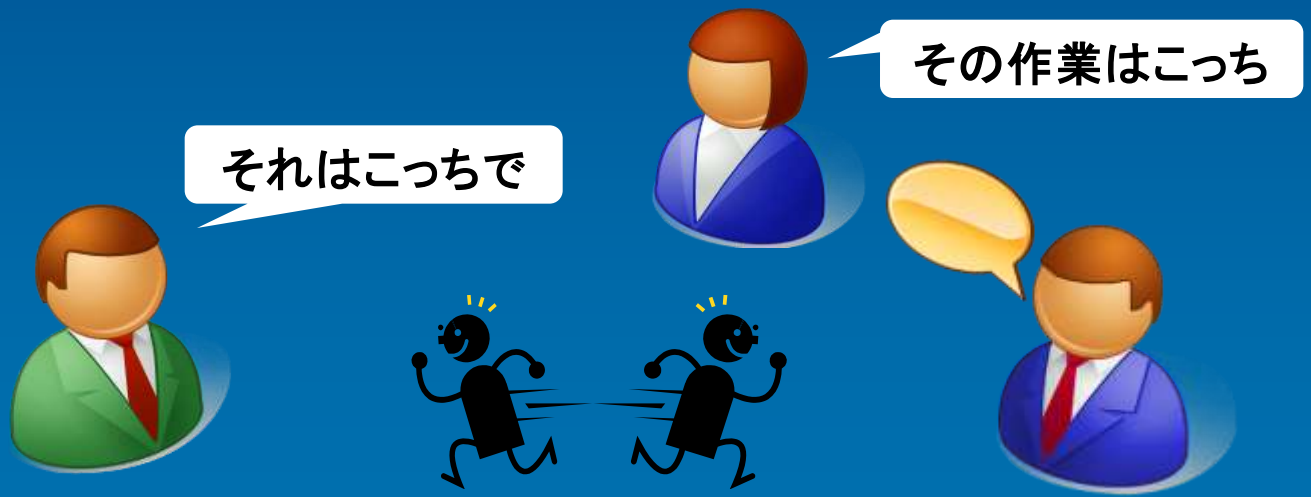
日頃こんな事起こっていませんか？

- バグが見つかり、さんざん振りまわされたが・・・
- 原因は・・・
  - コミット忘れ
  - 特定の人の環境設定ミス
  - バージョン違いミス

手元ではちゃんと動いたのに・・・

日頃こんな事起こっていませんか？

- いろんな人が作ったいろんな自動化の仕組み（バッチファイル、スクリプト）があちこちに散らばっている



日頃こんな事起こっていませんか？

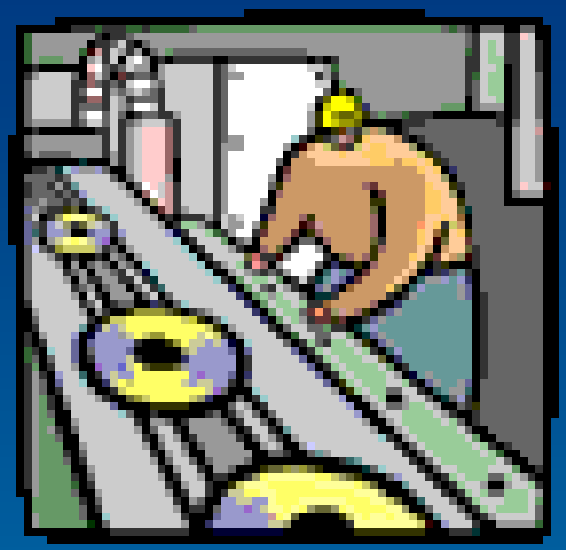
- えらい人がしょっちゅうスクリプトやエクセルのマクロをいじっている

問題点は・・・

- 人依存の作業
  - その作業は本当に人に任せるべきですか？
- 特定の人に作業とノウハウが集中している
  - その人が休んだり居なくなっても大丈夫ですか？
- 協同開発のための仕組みが不完全
  - 「コミット忘れ」「更新忘れ」
  - ビルドタイミングによる行き違い
  - 開発効率化のための技術が共有されない

そこで

- 自動化をしましょう！
  - 決められた環境
  - 決められた手順
  - 誰でも実行･メンテナンスできる



そのために･･･

CIをオススメします！

チラッ

CI（継続的インテグレーション）

- Integration → 統合、統一

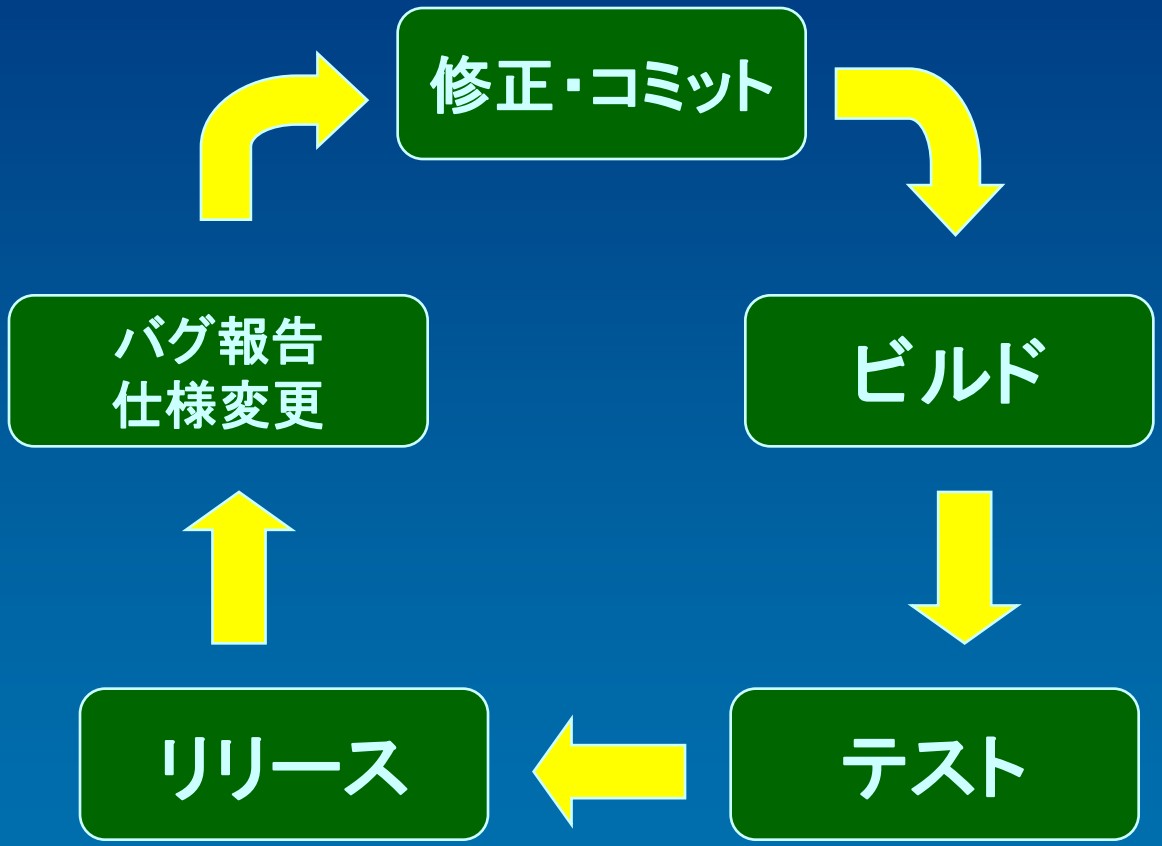
- 狭い意味では
  - 頻繁に（常に）ビルドが通る環境
    - ソースコードのコミットからの自動ビルド
    - スクリプトやバッチ等、作業の自動化
  - 常に最新の安定した成果物が維持される

- 広い意味では
  - 開発サイクル、ワークフローの統合
  - ソフトウェア品質管理のバックボーン

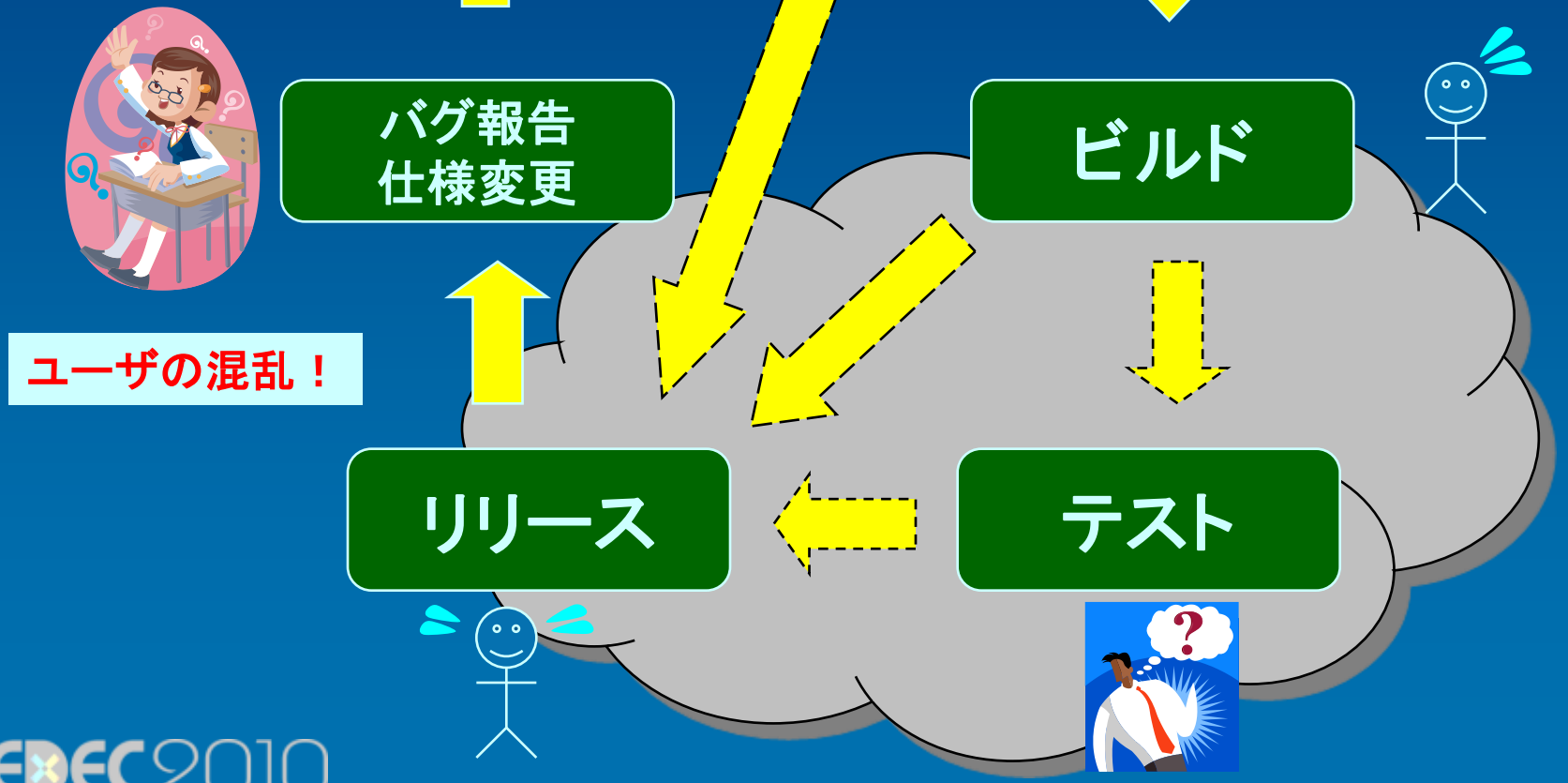
CIの位置づけ

- アジャイル開発
  - XP（エクストリーム・プログラミング）
    - CI（継続的インテグレーション）
- 適応的開発（変更に強い・速い）
  - アジャイル開発に必要な事（コミュニケーション・シンプルさ・フィードバック・勇気）
    - XPの実践の1つ（頻繁にビルドと結合を繰り返す）

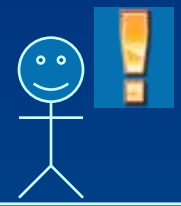
CIサイクル



CI前



CI後



修正・コミット

ビルド

テスト

リリース

バグ報告
仕様変更

自分でその環境を作るの？

専用のツールがあります！

CIツールを導入して
開発現場に秩序と速度を！

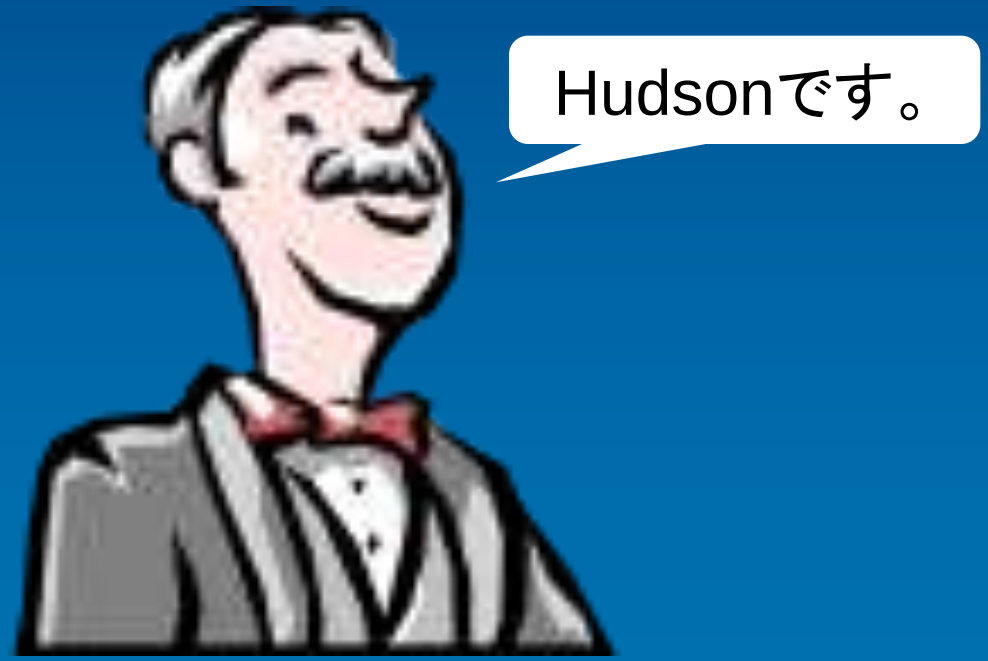
～目次～

- ゲーム開発環境
  - 開発環境診断
  - CI（継続的インテグレーション）
- CI導入の実例
  - CIツール「Hudson」の紹介
  - デモ、導入事例
    - 自動化ツール「AutoIt」の紹介
  - カスタム・他のツールとの連携
- タダで始めよう！
- まとめ

～目次～

- ゲーム開発環境
  - 開発環境診断
  - CI（継続的インテグレーション）
- CI導入の実例
  - CIツール「Hudson」の紹介
  - デモ、導入事例
    - 自動化ツール「AutoIt」の紹介
  - カスタム・他のツールとの連携
- タダで始めよう！
- まとめ

CIツール「Hudson」



CIツール「Hudson」

- Java製のオープンソースソフトウェア
- インストール・セットアップが簡単！
- Webサーバ機能もセット！
- 日本語標準対応！
- 拡張性が高い！
- 豊富なプラグイン！

CIツール「Hudson」

しかもタダ！

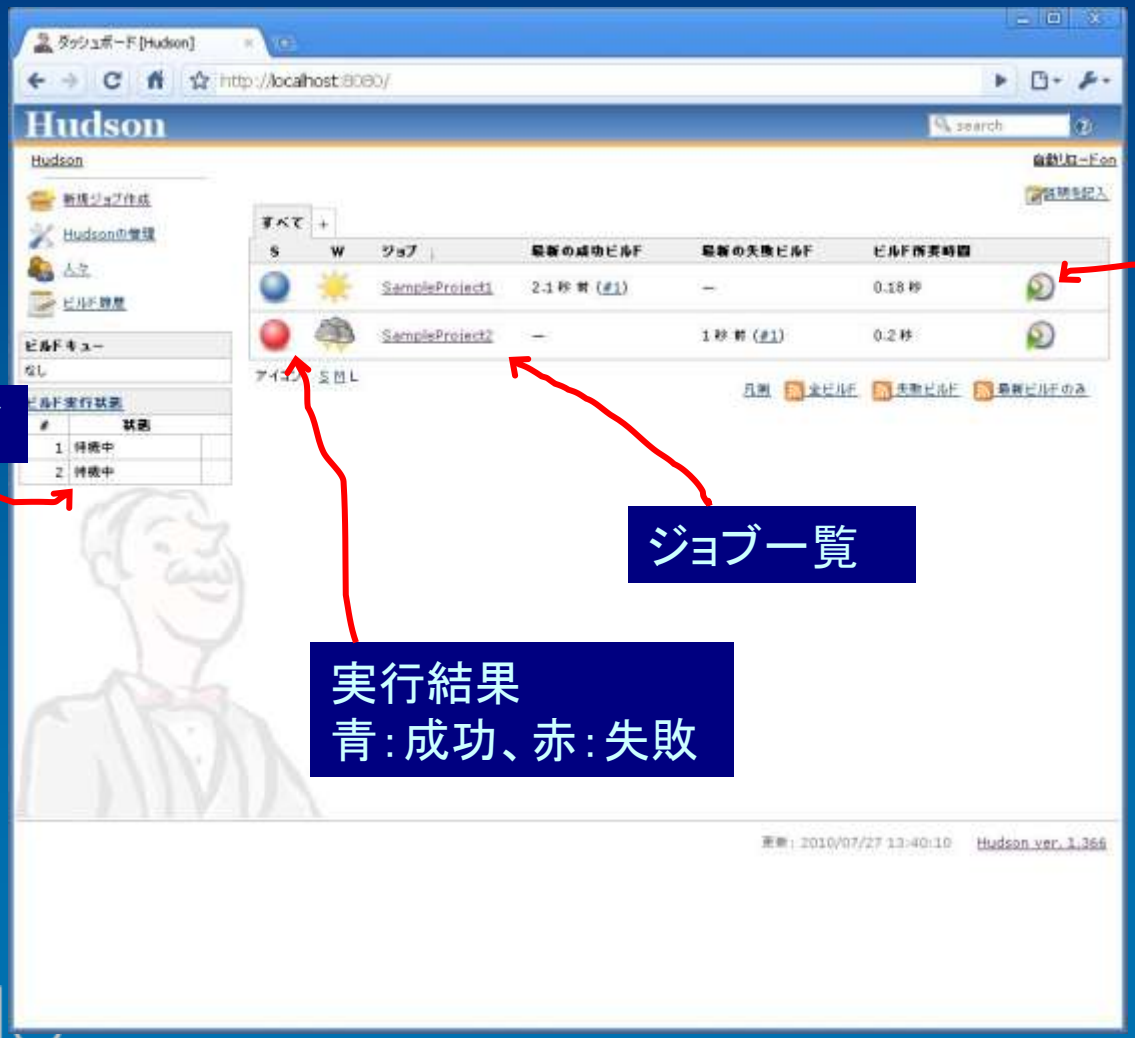
CIツール「Hudson」

- 自動ビルド
  - ビルドプロセス（自動化手順）を「ジョブ」として登録
  - 「ジョブ」を自動・任意に実行
- ソースコード管理システム（SCM）と連動
  - 最新のソースコードを取得してビルド
  - 標準でCVSとSubversionに対応
  - プラグインで様々なSCMに対応
    - Git、Perforce、StarTeam…

チェックアウト⇒ビルド

CIツール「Hudson」

- 設定・操作はほぼ全てWebから可能！



その他、自動化可能な作業

- 自動テスト

チェックアウト⇒ビルド（結合）⇒テスト

  - ビルド同様「ジョブ」として登録→実行
  - 一般的な単体テストツールで出力されるXML（xUnit形式）を読み込み、グラフで表示
- スクリプト、コマンド実行
  - Windowsならバッチ、Unix系ならシェルで表現可能な処理を「ジョブ」として登録→実行
  - プラグインで様々な処理を登録可能
    - Python、PowerShell…
  - Exeの実行も ←後で

成果物の管理

- ビルド結果の保存
  - いつ実行されたか
  - 実行手順のログ（コンソール出力）
- 成果物の保存
  - いつビルドされた成果か（MD5チェックサム）
  - 依存関係の管理
    - ライブラリとアプリの関係…

チェックアウト⇒ビルド（結合）⇒成果物の生成⇒テスト

情報共有

- ビルド結果の通知
  - メール通知
  - プラグインでIRC、Jabber、Twitterなども
- バグ管理システム（BTS）との連動
  - バグがどの成果物で起こったのか
  - 知られているBTSにはほぼプラグインで対応
    - Redmine、Trac、Mantis、Bugzilla…

チェックアウト⇒ビルド（結合）⇒成果物の生成⇒テスト
⇒結果の通知・関連付け

まさにインテグレーション！

～目次～

- ゲーム開発環境
  - 開発環境診断
  - CI（継続的インテグレーション）
- CI導入の実例
  - CIツール「Hudson」の紹介
  - デモ、導入事例
    - 自動化ツール「AutoIt」の紹介
  - カスタム・他のツールとの連携
- タダで始めよう！
- まとめ

STEP1　動かしてみよう！

- Hudsonは導入が簡単！

デモ

- インストール
  1. hudson.war をダウンロード
  2. java –jar hudson.war (Java実行環境が必要)
  3. http://localhost:8080/

- ジョブの作成
  - ジョブの実行
  - 結果の成否

簡単でしたね！

STEP2 ゲーム開発でのビルド自動化

- SCMの設定
- C++ソースコードのビルド
  - 実はVC++のビルドはプラグイン・・・
- ビルド実行・結果通知・コンソール出力・履歴

デモ

- Subversionリポジトリの設定
- VC++ビルド（MSBuild）
- 結果表示
- 成果物の保存

これが・・・CI・・・！

STEP3　CIツールを使いこなす

- チームでの導入事例
- アプリケーションの実行テスト
- さらに品質向上に向けて

アーケードゲームライブラリ開発での事例

- 社内向けの汎用ゲームライブラリ
- ターゲット
  - Windows OSベースのボード向け
  - Linux OSベースのボード向け
- 機能
  - OS抽象化、スレッド管理、ファイル管理、算術、描画、フォント、アニメーション、入力、アーケード機能
  - DCCツールのエクスポータ、コンバータ、ビューア
  - 内製ツールのランタイム、ビューア
- 開発人員（約6名）
  - 1人の担当1モジュール以上

ある程度の環境はあるが混乱

- 手元の環境でビルド
  - コミット・更新忘れ
  - ターゲット・構成のビルドチェック抜け
- テスト代わりに手元でサンプルを実行
  - 実行忘れ
- リリース管理が無いので、バグが起こった時の追跡が困難
- 抽象的なバグ報告が来る
- ドキュメントの保守まで手が回らない

まずは自動ビルドで安定を

- ビルドマシンを用意
  - ターゲット毎に分けるのがベスト
  - 仮想マシンはゲーム開発環境では▲
- リポジトリへのコミット→自動ビルド
  - 既にあるビルドバッチを実行させる
  - MSBuildプラグインでVC++プロジェクトを指定
- 失敗したらメールで通知
  - コミッター名@hoge.co.jp
  - カスタマイズで必要な人にも通知

第一段階完了！

～目次～

- ゲーム開発環境
  - 開発環境診断
  - CI（継続的インテグレーション）
- CI導入の実例
  - CIツール「Hudson」の紹介
  - デモ、導入事例
    - 自動化ツール「AutoIt」の紹介
  - カスタム・他のツールとの連携
- タダで始めよう！
- まとめ

アプリケーションの実行テスト

- サンプルが「正常に」動くか確認したい
  - 起動時、一定時間動かす、終了時にエラーが出ないか
  - 終了後メモリがリークしていないか
- 実行画面を確認したい
  - 表示がおかしくないか
  - エラー時の画像が見たい

画面操作の自動化ツール「AutoIt」を導入！

AutoIt

- フリーツール
- WindowsのGUI操作を自動化できる、BASICライクなスクリプト言語
- キー入力・マウス操作の自動化
- WindowsAPIをダイレクトに呼ぶ
  - ウインドウハンドルを直接操作
  - 操作の誤爆を防ぐ
- 単独で動作するexeに変換可能

AutoItを使ったアプリテスト自動化

- AutoItからアプリを起動
- ウインドウのスクリーンキャプチャ
- 入力テスト
- ウインドウの終了
- コンソール出力のチェック
  - プロセスの標準（エラー）出力を取得可能
  - ライブラリで用意しているメモリリークメッセージ（デバッグ用メッセージ）をキャッチ
- アクティブウインドウのチェック
  - アプリのウインドウが最前面にきているか

デモ

- AutoItのデモ
- HudsonからAutoIt経由でアプリのテストをするデモ

これはベンリ！

その他、AutoItで試した事

- 実機でのテストは？
  - Windows上でデバッグできる範囲であれば有効？
- ゲーム画面を判断しながらの処理には工夫が必要
  - 一応、AutoItを使ってできます
  - ライブラリサンプルのテストなのでそこまで突っ込まず

Hudsonの話に
戻ります！

～目次～

- ゲーム開発環境
  - 開発環境診断
  - CI（継続的インテグレーション）
- CI導入の実例
  - CIツール「Hudson」の紹介
  - デモ、導入事例
    - 自動化ツール「AutoIt」の紹介
  - カスタム・他のツールとの連携
- タダで始めよう！
- まとめ

CIツールをさらに使いこなす

- Hudsonの機能やプラグインを使って品質向上のためのソフトウェア開発環境を統合！

Hudsonのカスタム

- 分散ビルド
  - 総ビルド時間の短縮
  - クリーンなビルド環境
    - ターゲット毎に環境を構築
    - テストを利用者環境で実行
  - 構成毎（Debug,Release）に分けてもいいかも？
- 分散環境の構築も簡単！

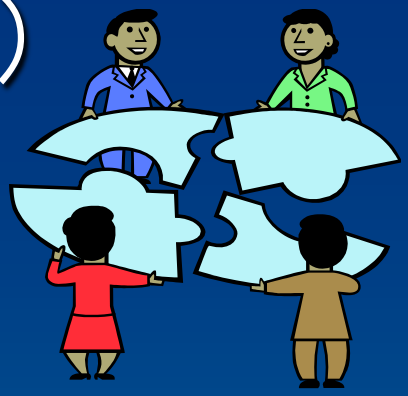
Hudsonのカスタム

- リリース管理
  - UnixサーバにHudsonの分散クライアントを置いてコマンドを実行
    - 「最新の安定したビルドの成果物」をwget
    - 更新情報の管理はWiki等でWebAPIがあれば自動化可能（研究中）

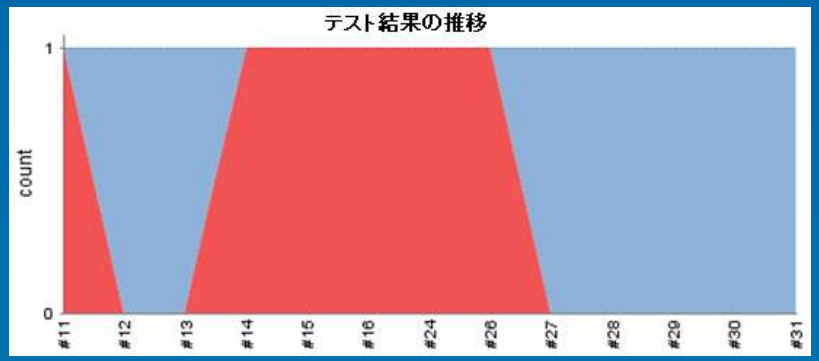
Hudsonのカスタム

- デザイン素材など（バージョン管理外のデータ）の管理
  - ファイル共有+チェックサム
    1. ファイル共有⇒ワークスペースにコピー
    2. コンバートやデータのチェック
    3. 追跡したいデータを成果物に指定
  - 問題点
    - ファイルが多い（大きい）とチェックサムを計算する時間がかかる
    - 成果物/ワークスペースそれぞれ、ビルド毎にデータを保存しておけるが、ディスク容量を圧迫

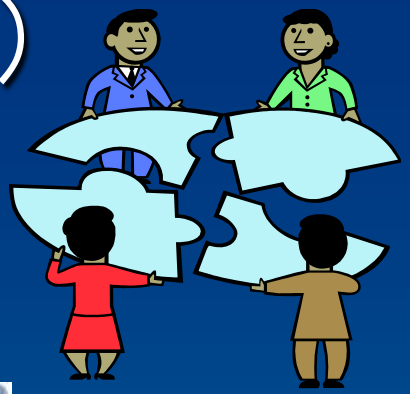
他のツールとの連携（プラグイン）



- ユニットテスト
  - CppUnit
  - Googletest
    - 失敗を確認するテスト（DeathTest）がイイ！
  - MSTest（有償）
    - VS2008（Professional Edition以上）からは単体テスト機能が標準装備
    - データドリブンテスト（cvs、XML）がイイ！



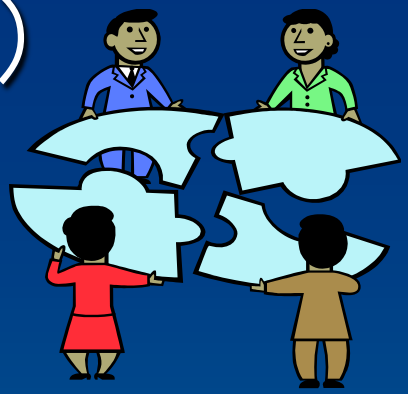
他のツールとの連携(プラグイン)

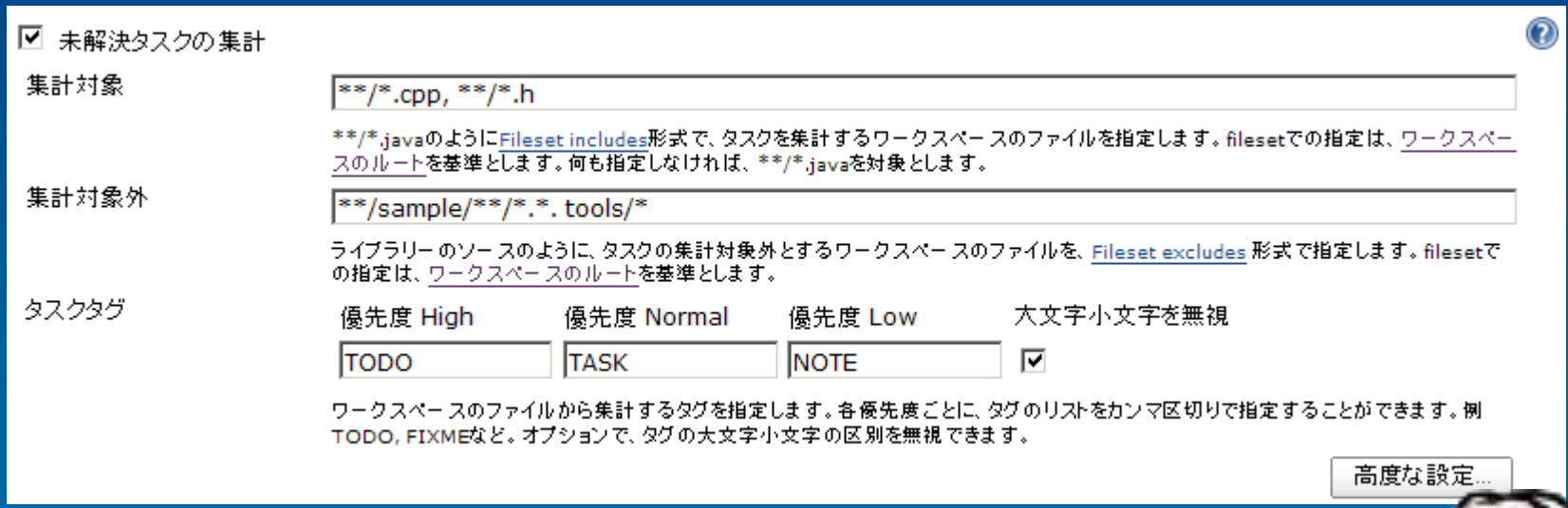


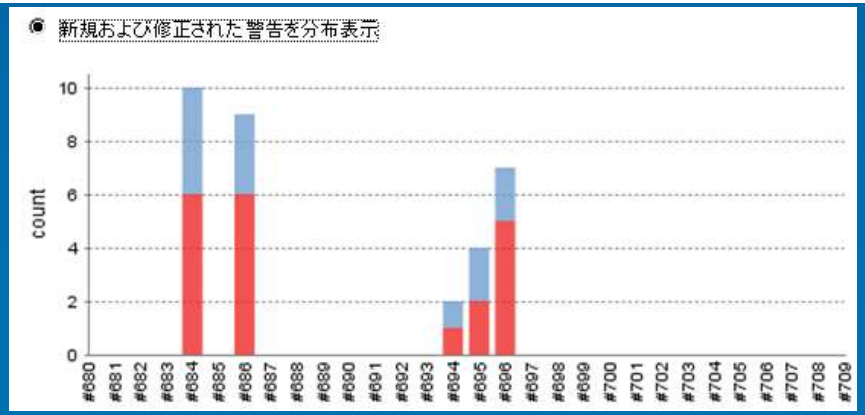
- 警告
  - Warningsプラグイン

他のツールとの連携(プラグイン)

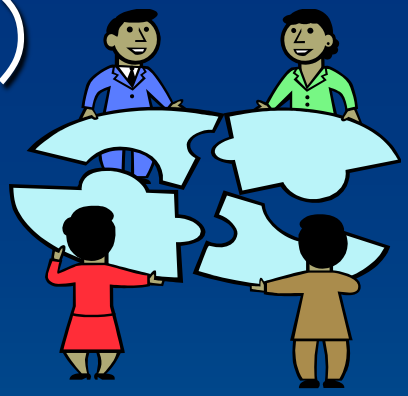
- タスク検知
  - TaskScannerプラグイン







他のツールとの連携(プラグイン)



- 静的解析
  - cppcheck、Coverity(有償)

プロジェクトへ戻る
状態
変更
コンソール出力
Cppcheck Result
前のビルド

Cppcheck Result

Errors Trend

All errors	New errors
68	0

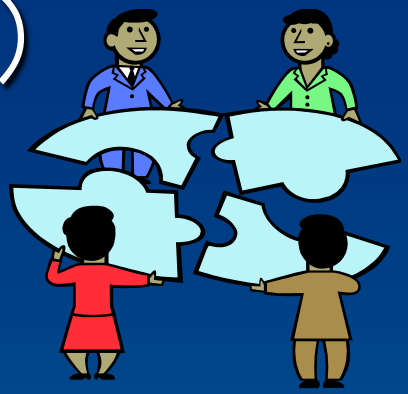
Summary

Total	Severity 'error'	Severity 'possible error'	Severity 'style'	Severity 'possible style'	no category
68	0	0	66	2	0

Details

Filename	LineNumber	CppCheckId	Severity	Message
[redacted].h	36	noConstructor	style	The class 'Application' has no constructor. Member variables not initialized.
[redacted].h	36	noConstructor	style	The class 'Application' has no constructor. Member variables not initialized.
[redacted].h	249	uninitVar	style	Member variable not initialized in the constructor [redacted]
[redacted].h	249	uninitVar	style	Member variable not initialized in the constructor [redacted]
[redacted].h	249	uninitVar	style	Member variable not initialized in the constructor [redacted]
[redacted].h	249	uninitVar	style	Member variable not initialized in the constructor [redacted]
[redacted].h	249	uninitVar	style	Member variable not initialized in the constructor [redacted]
[redacted].h	48	uninitVar	style	Member variable not initialized in the constructor [redacted]
[redacted].h	48	uninitVar	style	Member variable not initialized in the constructor [redacted]
[redacted].h	48	uninitVar	style	Member variable not initialized in the constructor [redacted]
[redacted].h	48	uninitVar	style	Member variable not initialized in the constructor [redacted]
[redacted].h	48	uninitVar	style	Member variable not initialized in the constructor [redacted]
[redacted].h	48	uninitVar	style	Member variable not initialized in the constructor [redacted]

他のツールとの連携(プラグイン)



- コードメトリクス
  - ソースコードの統計情報を計測
  - CCCC

Cccc results

Project Summary

This table presents summary values of various measures over the body of source code submitte

Metric	Tag	Overall	PerModule
Number of modules	MOM	1181	
Lines of Code	LOC	191405	162.07
McCabe's Cyclomatic Number	MVG	25001	21.169
Lines of Comment	COM	80314	68.005
LOC/COM	L_C	2.383	
LOC/COM	M_C	0.311	
Information Flow measure (inclusive)	IF4	5250427	
Information Flow measure (visible)	IF4v	1706218	
Information Flow measure (concrete)	IF4c	0	0.0
Lines of Code rejected by parser	REJ	25782	

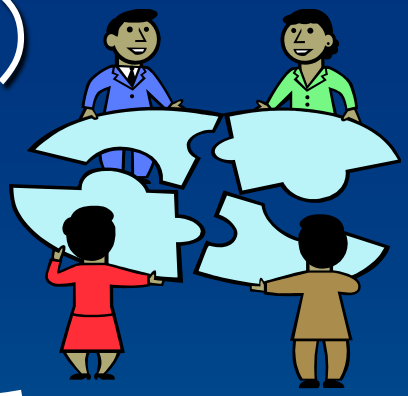
モジュール数

コード行数

複雑度

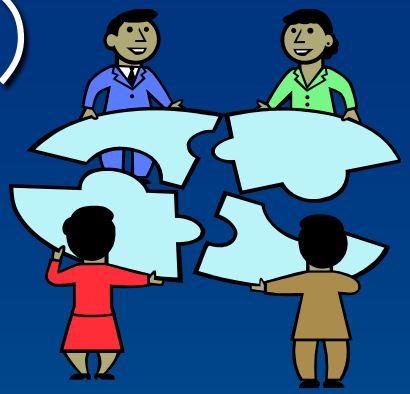
コメント行数

他のツールとの連携（プラグイン）



- コーディングスタイルチェック
  - cpplint（google-styleguide）
  - 「Google C++ スタイルガイド」に則っているかをチェックできるスクリプト
  - Pythonスクリプトを、Hudsonから実行
    - プラグインはまだ無いらしい・・・
  - IDEと統合するのもあり

他のツールとの連携（プラグイン）



- ドキュメント生成
  - Doxygen
- プロジェクト管理ツール
  - BTS（ Trac、Redmine ）
  - オススメ構成！
    - Redmine：Hudson プラグイン+ TortoiseSVN Plugin
- その他
  - プラグイン情報は公式WikiかHudsonの設定画面からチェック！（英語）

Hudsonを使って・・・

- プラグインの導入
  - 使えるツール、注目されている技術はプラグインが作成される
    - →良さそうなものはどんどん試用・統合！
- CIツールをバックボーンとする事で世界中で利用されている要素技術が分かります！
- 成長していく品質管理環境！
  - 最初はデイリービルドからスタート！

CIツールを導入した成果

- ビルドエラーはリリース前に解決！
- サンプルの自動テストで早期バグ発見！
- エラーは起こった時に分かるので手戻り無し！
- 週1回の安定したライブラリリリース！
- 1年間の継続運用！

今すぐCIツールを入れましょう！

～目次～

- ゲーム開発環境
  - 開発環境診断
  - CI（継続的インテグレーション）
- CI導入の実例
  - CIツール「Hudson」の紹介
  - デモ、導入事例
    - 自動化ツール「AutoIt」の紹介
  - カスタム・他のツールとの連携
- タダで始めよう！
- まとめ

タダで始めよう！

※なんでも無料が良いという意味ではありません

- タダでいいの・・・？

ライセンスについて

- HudsonはMITライセンス
  - タダ！
  - ただし無保証
  - 有償サポートあり
- AutoItはライセンス無しのフリーツール

その他のCIツール

- Buildbot（GPLライセンス）
- CruiseControl（BSDライセンス）
- Apache Continuum（Apacheライセンス）
- TeamCity（有償）
  - 20configrations、3agents、20users → 制限ありでタダ！
- Visual Studio Team Foundation Server（有償）

まずは

- 世の中に目を向ける
  - 最新技術を知っておく
  - 開発環境についても同じです
- とにかく、試してみる
  - 「車輪の再発明」はしない
- タダから始める
  - まずはチームの文化に合うかどうか
  - 欠かせないようになってきたら有償ツールも視野に

～目次～

- ゲーム開発環境
  - 開発環境診断
  - CI（継続的インテグレーション）
- CI導入の実例
  - CIツール「Hudson」の紹介
  - デモ、導入事例
    - 自動化ツール「AutoIt」の紹介
  - カスタム・他のツールとの連携
- タダで始めよう！
- まとめ

CIで「見通しの良い」開発環境を作ろう！

- プロジェクトの進捗の管理
  - プロジェクト管理ツール
  - バグ管理システム（BTS）
- 成果物の管理
  - CIツール ←ここ
- 開発状況の管理
  - ソースコード管理システム（SCM）

成果物の管理

- 安定した成果物を頻繁に作成
- ソースコード管理システム（SCM）と成果物
  - 一定以上の品質を保証
  - （例:ビルドは通る、コミット漏れが無い）
- プロジェクトの進捗管理と成果物
  - 課題を成果物に関連付け
  - 問題の範囲を特定できる

まとめ

- CIで、より良い開発環境づくりを！

- ツールはツールです！
  - 使い方次第
  - 使う人次第
  - まずはチームに溶けこむように

- いきなりがんばりすぎない←

ありがとうございました！

質問･ご意見は

Kokawa_Takashi@sega.co.jp

Twitter id: Kokawa_Takashi