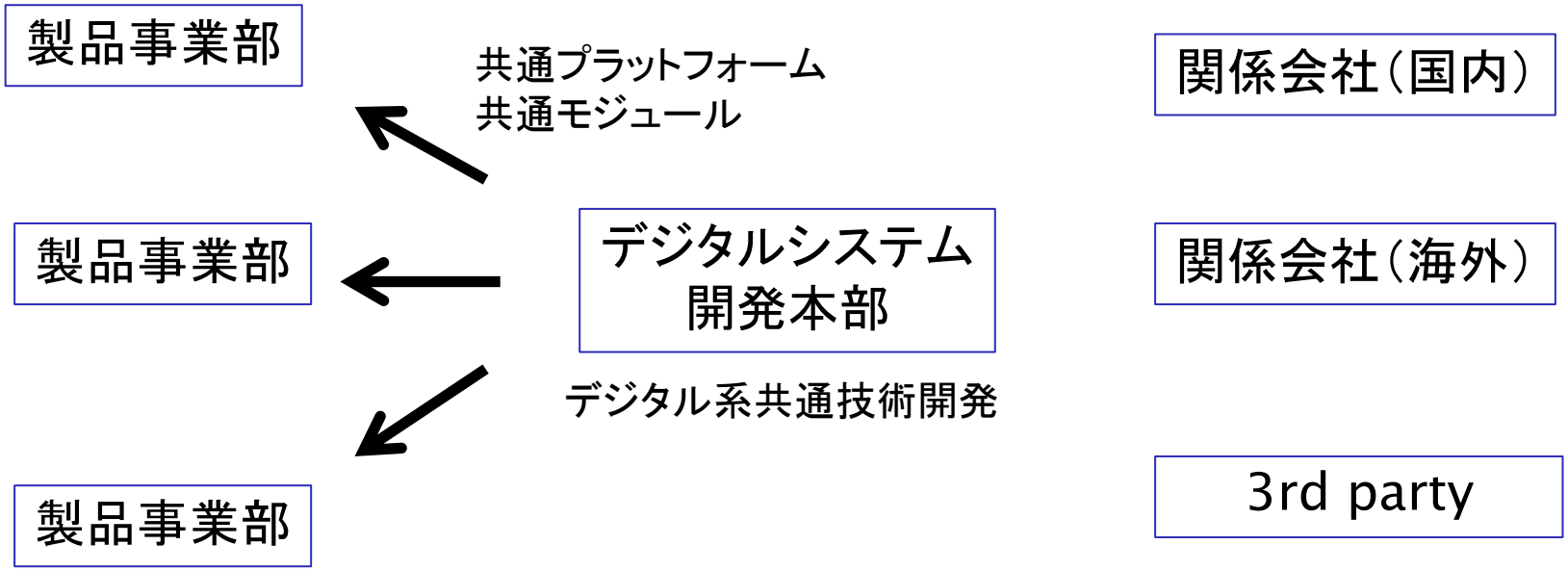
キヤノンにおける Jenkinsへの取り組み

キヤノン株式会社
デジタルシステム開発本部
ソフトウェア検証室

馬場 健

キヤノン デジタルシステム開発本部について



■お断り

- 本発表の内容はデジタルシステム開発本部に限定したものです

Jenkins導入の経緯

- 品質向上・テスト自動化は大きな課題

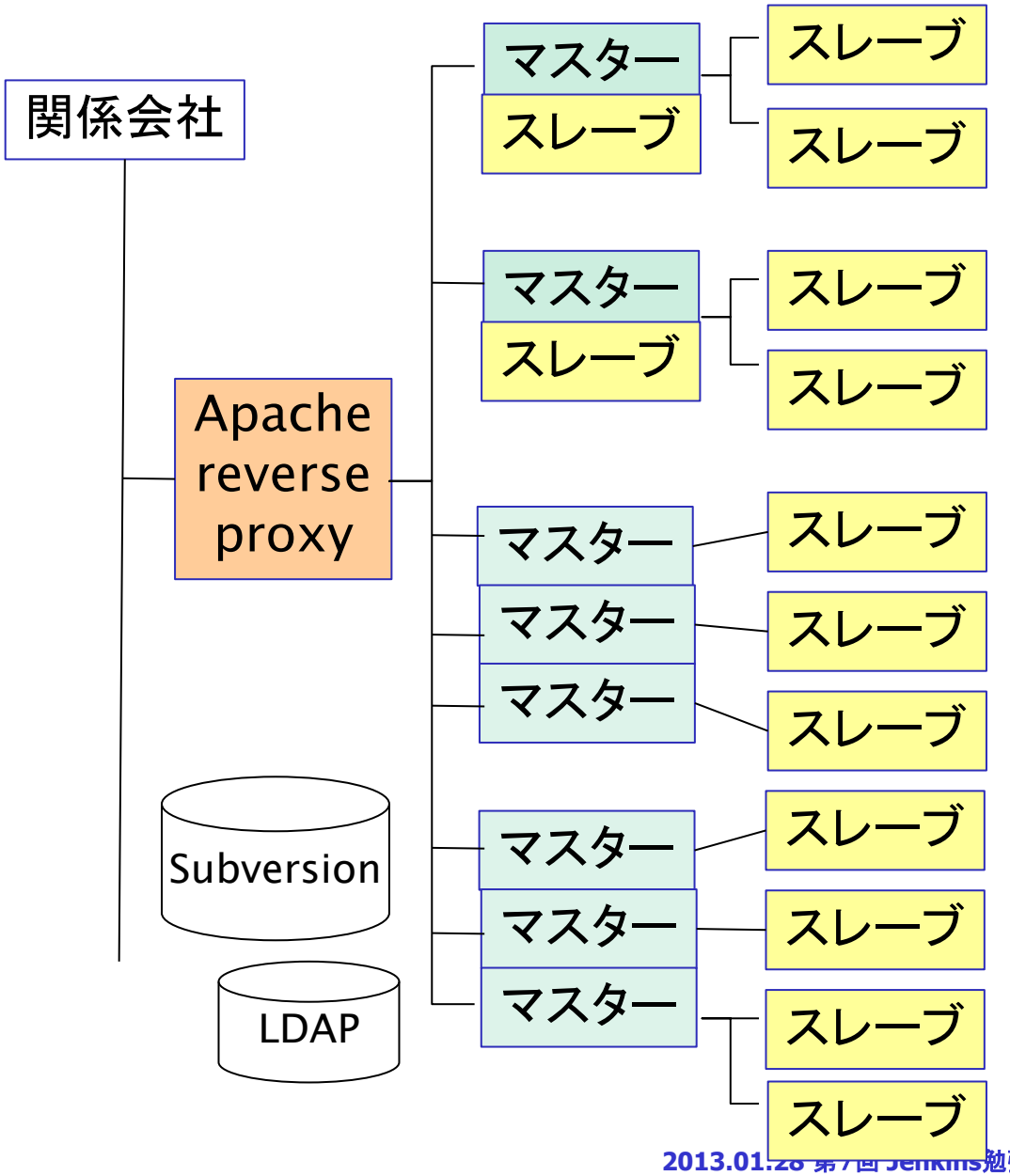
- 2010後半
  - CIツール調査・候補選定、Jenkins導入を決定（当時はHudson)
- 2011
  - 資料公開やデモサイト構築、本部内セミナー実施
  - 先行チームの立上げ、環境・標準など検討
  - C/C++系など: 既存プラグインへの変換など ⇒ 内製プラグイン開発へ
    - 10月 : Jenkins温泉参加
- 2012
  - 内製プラグイン公開・使用開始
  - 基本的な環境整備を終了（改良事項は多い）
  - 展開と有効利用をひたすら推進

本日の話題

- 当本部の事情・特徴
  - テーマ・チーム数が多い、大小さまざま、中身もいろいろ…
  - C/C++が大半。環境・ツールは標準化、でもやはりいろいろ…
- Jenkins 環境
  - 全体構成
  - Summary Report プラグイン
- Jenkins 利用状況
  - Jenkinsボード
  - 何をやっているか、結果はどうか
  - 全体傾向
- まとめ

Jenkins 環境

全体構成



- おおむねチームごとにマスター分散
  - セキュリティ、使い勝手優先
  - 現在約30マスター
- 同一マシンでもスレーブ化
- 最近は同一マシンに複数マスター（テストマシンの確保重点）
- 標準設定・プラグイン込の Jenkins テンプレート ⇒ 立上げは1分
- インフラ
  - SCM:　Subversion
  - LDAP：Svn, Jenkins,その他で共用
  - 関係会社アクセス用 Apache
  - KVMベースのVMプール
- 関連ツール含めて数名のサポートグループで運用

Summary Report プラグイン

- C/C++など多数ツールへの対応を目的に内製
  - ビルド結果のサマリー（ビルド全体のメトリクス）のみレポート
    - ex. 総テストケース数、総テスト成功数…
    - 詳細は各ツールの出力レポートを保存
  - 機能はシンプル～レポータの基本機能
    - 履歴グラフ、結果判定、ダッシュボードビューなど
  - 現在 19個 ～ ツール×レポート種類
    - 共通部：1，汎用：2，C/C++：9，Java：4，.NET：3
    - 単体テスト、エラー検出、コードカバレッジ、静的チェック、コードメトリクス
  - カスタムレポートプラグイン
    - 結果を所定フォーマットで出力すれば他ツール・独自ツールにも対応可
    - plot に比べて結果判定もできる

● ジョブ設定：表示項目設定、結果判定条件もサポート

● ビルド結果：各種レポートサマリを表示

単体テストサマリ（履歴）

- TestCases総数：5
- TestCases成功数：3
- TestCases失敗数：2
- TestCasesエラー数：0
- 結果レポート：report/cppunit/cppunit.xml

Lcov カバレジサマリ（履歴）

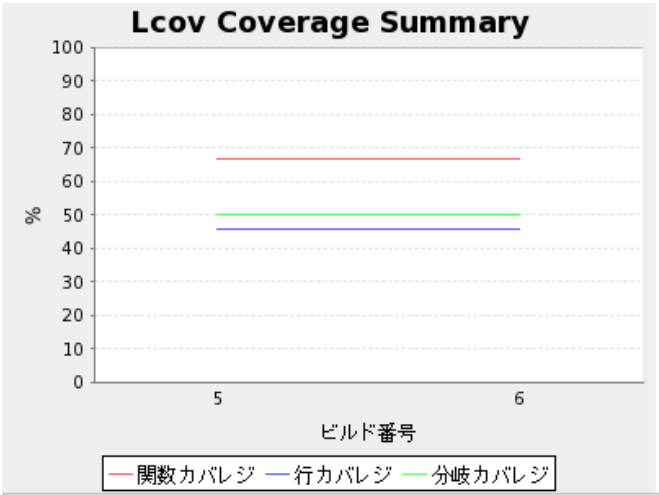
- 関数カバレジ：66.7
- 通過関数数：2
- 関数数：3
- 行カバレジ：45.5
- 通過行数：5
- 行数：11
- 分岐カバレジ：50.0
- 通過分岐数：1
- 分岐数：2
- 結果レポート：report/lcov/index.html

Valgrind MemCheckサマリ（履歴）

- トータルエラー数：0
- InvalidFree数：0
- MismatchedFree数：0
- InvalidRead数：0
- InvalidWrite数：0
- InvalidJump数：0
- Overlap数：0
- InvalidMemPool数：0
- UninitCondition数：0
- UninitValue数：0
- SyscallParam数：0
- ClientCheck数：0
- Leak_DefinitelyLost数：0
- Leak_IndirectlyLost数：0
- Leak_PossiblyLost数：0
- Leak_StillReachable数：0
- 結果レポート：report/valgrind/valgrind.xml

● 履歴レポート

Lcov カバレジ履歴



ビルド ↓	結果レポート	関数カバレジ
#6	report/lcov/index.html	66.7
#5	report/lcov/index.html	66.7
#4		
#3		

● ダッシュボードビュー

Lcov

ジョブ ↓	結果レポート	関数カバレジ	関数数	行カバレジ	行数
c-dynamictest-linux-sample01 #12	report/lcov/index.html	100.0	2	100.0	16
c-dynamictest-linux-sample02 #4	report/lcov/index.html	100.0	2	100.0	16
c-dynamictest-linux-sample03 #3	report/lcov/index.html	100.0	2	100.0	16
c-dynamictest-linux-sample04-summary #2	report/lcov/index.html	100.0	2	100.0	16
cpp-dynamictest-linux-sample01 #6	report/lcov/index.html	66.7	3	45.5	11
cpp-dynamictest-linux-sample02 #3	report/lcov/index.html	66.7	3	45.5	11
cpp-dynamictest-linux-sample03_summaryreport #2	report/lcov/index.html	66.7	3	45.5	11
総合結果	--	**82.4**	**17**	**81.4**	**97**

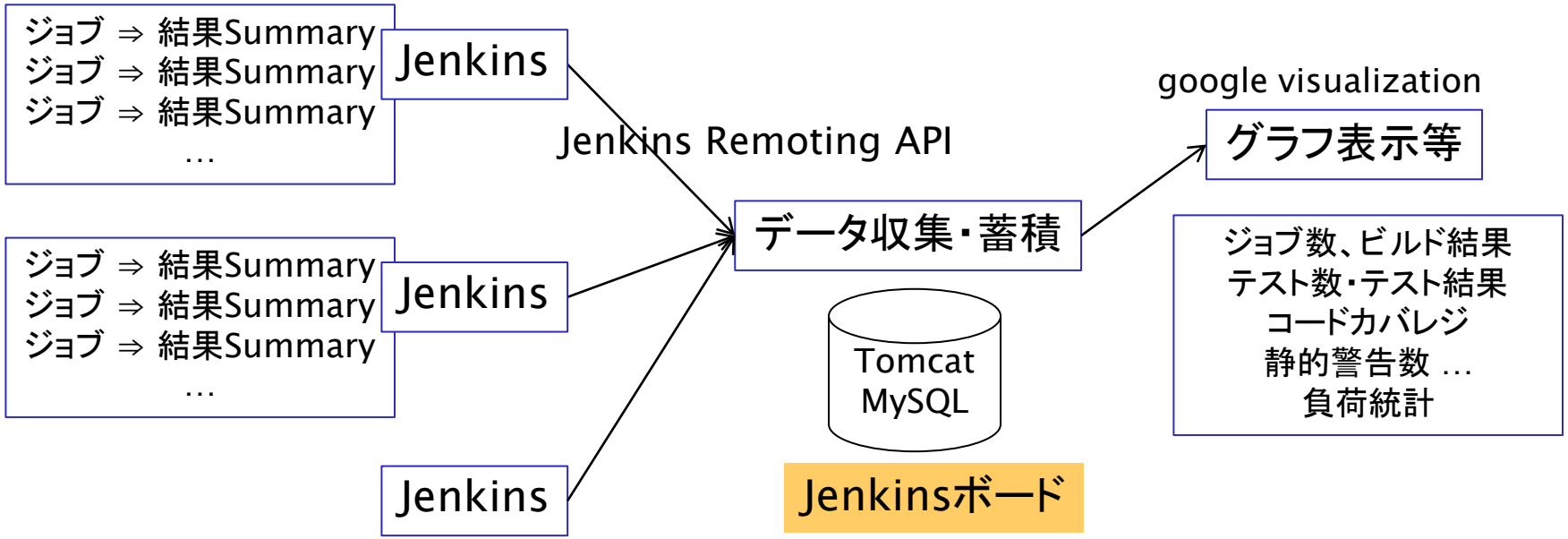
Cpptest static

ジョブ ↓	結果レポート		抑制数	チェック行数	行数
c-static-cpptest-linux-sample00_summmaryreport #6			0	0	0
c-static-cpptest-linux-sample01_summmaryreport #1			0	0	0
test-cpptest-parser #38		250	109	3315	3315
総合結果	--	--	**109**	**3315**	**3315**

Jenkins利用状況

Jenkinsボード

- 各チームのJenkinsからビルド情報を REST API で自動収集
  - 利用状況把握 ＋ 多少のトラブル監視



- お断り
  - 粗いデータ ～ 各ジョブの最新ビルド結果のみ、同様ツールは同じデータに集計
  - ジョブの粒度などまちまち、ジョブ作成によっては重複も
  - 取れていないデータも（大半のデータは前述のSummaryReportが必要）
  - これだけで「良し悪し」は判断不可

ジョブの数・内容～何をやっているか

		内容（出力レポート種類）					
開発チーム	ジョブ数	動的テスト	メモリリーク等	カバレッジ	静的チェック	コードメトリクス	
A	50	34	0	34	0	0	
B	36	31	0	1	1	0	
B'	129	9	0	1	31	0	⇒ カバレッジはこれから
C	26	26	0	0	0	0	
D	27	15	0	9	11	0	
E	14	11	8	8	2	1	
E'	38	14	8	8	10	5	⇒ ほぼ網羅的に実施
F	3	0	0	0	0	0	⇒ 実際は動的、まだログのみ
G	62	0	0	0	0	0	⇒ 実際は動的、まだログのみ
H	7	2	0	3	0	0	⇒ ビルドと動的のみ
I	202	3	0	0	0	0	⇒ テストマシン25台でフル活用、ただ大半のデータが取れていない
J	6	0	0	0	0	6	⇒ コードメトリクスのみ
K	9	4	0	0	1	1	⇒ 立上げ中、GUIロボット対応中
N	4	2	0	0	0	0	⇒ 実際はカバレッジ、静的あり
L	2	0	0	0	0	0	⇒ 実際は動的、カバレッジ、静的あり
M	30	18	18	8	0	0	⇒ 動的は充実
N	5	0	0	0	0	0	⇒ コードメトリクスとリリース作業のみ
O	5	0	0	0	2	3	⇒ 最近静的可能に、動的も取組中

赤：やっていない
緑：やっているがデータが取れていない

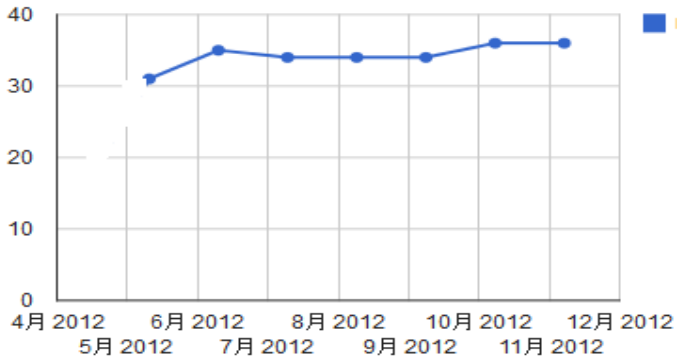
ビルド・動的テスト状況～結果はどうか

開発チーム	ジョブ数	ビルド安定率	テスト成功率	テストケース数	行カバレッジ	カバレッジ対象行数	
A	50	20%	86%	1,635	53%	185,104	⇒ メンテされていないジョブ多数？
B	36	83%	100%	4,712	50%	19,924	⇒ カバレッジは最近投入の1ジョブのみ？
B'	129	75%	100%	2,809	47%	25,191	
C	26	69%	100%	1,976	N/A	N/A	
D	27	73%	80%	7,171	80%	43,144	
E	14	64%	94%	1,041	83%	2,495	⇒ 検証済みマージを実施
E'	38	61%	86%	1,373	84%	2,495	（上が trunk、下が委託先ブランチ）
F	3	100%	N/A	N/A	N/A	N/A	⇒ 結果判定できていない
G	62	100%	N/A	N/A	N/A	N/A	⇒ 結果判定できていない
H	7	71%	100%	136	80%	14,245	⇒ ジョブ・テスト数は少ないがカバレッジはそこそこ
I	202	74%	100%	6,597	N/A	N/A	⇒ データはごく一部しか取れていない
J	6	100%	N/A	N/A	N/A	N/A	
K	9	56%	74%	35	N/A	N/A	⇒ 立上げ中
N	4	100%	100%	2,911	N/A	N/A	⇒ 実際はカバレッジ、静的、doxygen あり
L	2	100%	N/A	N/A	N/A	N/A	⇒ 実際は動的、カバレッジ、静的あり
M	30	27%	75%	7,918	74%	114,125	⇒ テスト失敗残存
N	5	100%	N/A	N/A	N/A	N/A	
O	7	71%	N/A	N/A	N/A	N/A	⇒ 動的は取組中

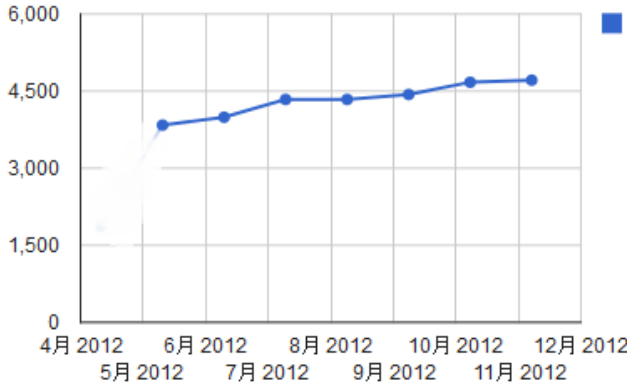
赤：問題あり？
緑：ちゃんとデータが取れていない

あるチームの推移グラフ

ジョブ数

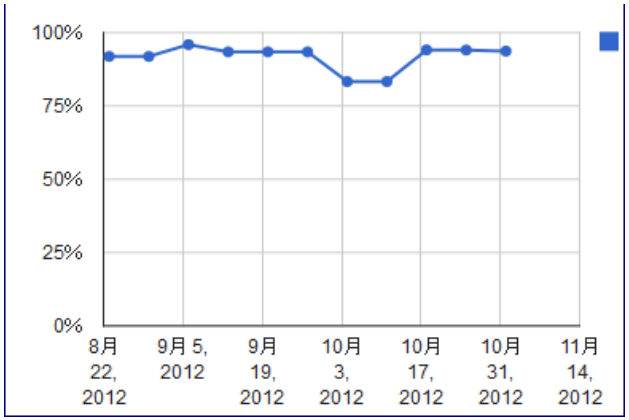


テストケース数



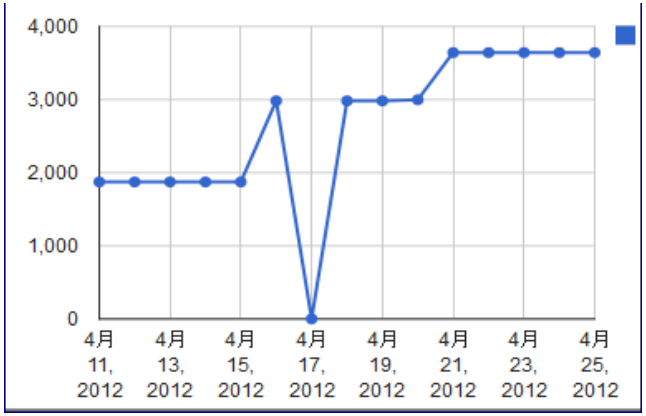
◎ ジョブやテストケース数が徐々に増加

テスト成功率



◎ 多少の増減あるもテストを維持

テストケース数



△ 稀にはトラブルも

全体傾向

- チームでのバラつきは大きいが、おおむね有効活用できている
  - テスト作成、Jenkinsジョブ化、その拡張維持はほぼ定着
  - Jenkinsでの手動ビルドも人気～簡単実行、結果共有
  - 一部だが、検証済みマージ（手動）やリリース自動化なども
  - 関係会社とのジョブ共有も徐々に進展

- 一部チームは苦戦…
  - そもそも単体テスト等できるように作られていない…
  - 特殊なコンパイラで静的チェックも通らない…
  - ⇒ 時間はかかっているが、改善傾向

ちょっと問題…

- 「カスタムワークスペース」利用がけっこう多い（同じ場所で異なるビルド）
  - クリーンビルドが確保できているか心配
- 静的チェックの活用はいまいち？
- たくさんビルド、でも結果をちゃんと見ていない、
  SummaryReportや結果判定を活用できていないチームも

トラブル・珍プレー…

- 複雑なマルチ構成プロジェクト ⇒ 数万ジョブが生成、動かない…
- コンソールログが 160 GiB ⇒ 共有Jenkinsマシンが満杯に
- 深夜ビルドが流行 ⇒ 商用ツールのライセンスが取れない…

まとめ

- ゴール：「継続的テスト」による開発スピードと品質の確保
  - テストセカンド！ コードと同時にテストを書く、それをどんどん増やす
  - 実証的ソフト開発 ～ 再現性＋見える化＋データを残す
- 現状
  - まだまだ基本レベルだが、継続的テストと Jenkins活用は着実に進展
  - 自動化・省力化だけでなく、状況の共有・見える化は大きい
- 課題・今後
  - 事業部・関係会社含めた活用
    - 当社特有のテスト環境構築なども
  - 環境面ではテストマシンの有効活用・クラウド化
  - ユーザーの指導、良い事例の共有、検証済みマージの普及などなど

ご清聴ありがとうございました

Jenkins の有効活用・安定運用に向けて、
今後ともよろしくご指導・ご支援お願いします