

最近の静的解析の立ち位置

コベリティ日本支社



静的解析が注目される背景

NASAによる日本車製乗用車の急加速問題調査

- 2011年2月に調査結果を公開
 - http://www.safercar.gov/ncap/reviews/2011/20110201_NASA_UA_Report.pdf
 - 電子制御システムの問題ではないことがわかり、疑いが晴れた

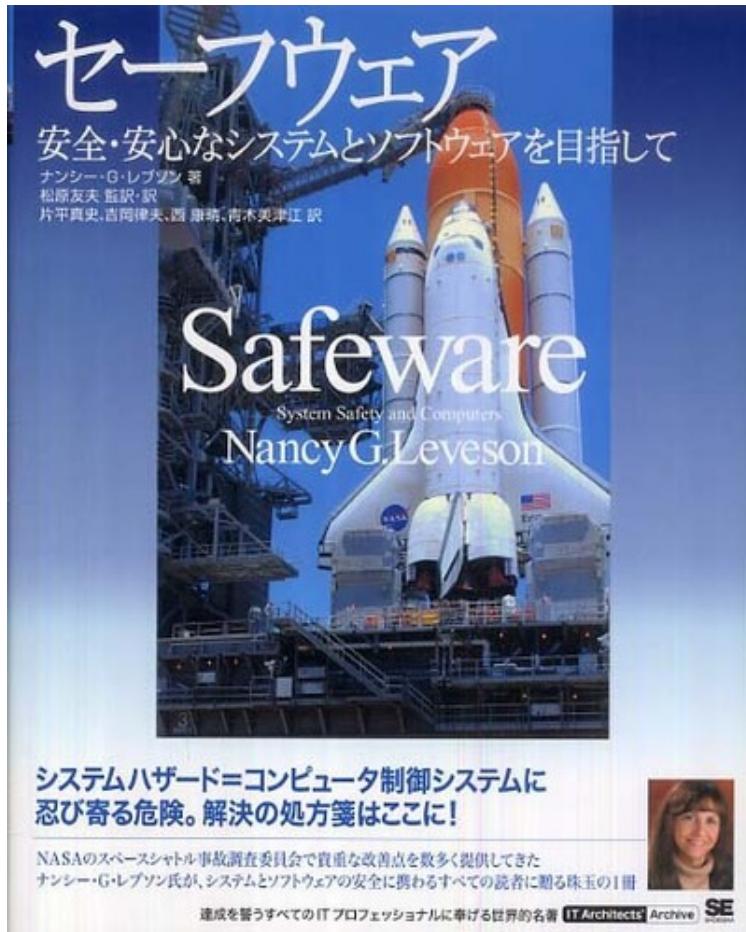
	NASA Engineering and Safety Center Technical Assessment Report	Version: 1.0
Title: National Highway Traffic Safety Administration Toyota Unintended Acceleration Investigation		Page #: 1 of 177

- ETC (Electric Throttle Control) のシステム設計・実装に意図していない不具合が含まれているかを調査

実装ソースコードの解析に
Coverity 静的解析を利用

静的解析が注目される背景

- ・ソフトウェアに起因する産業機器の不具合は少なくない



米国FDA（食品医薬局）は、
2010年前半に、

不具合による23件の
デバイスのリコールを行った

これらのリコールのうち、
少なくとも6件は、
ソフトウェアの不具合だった。

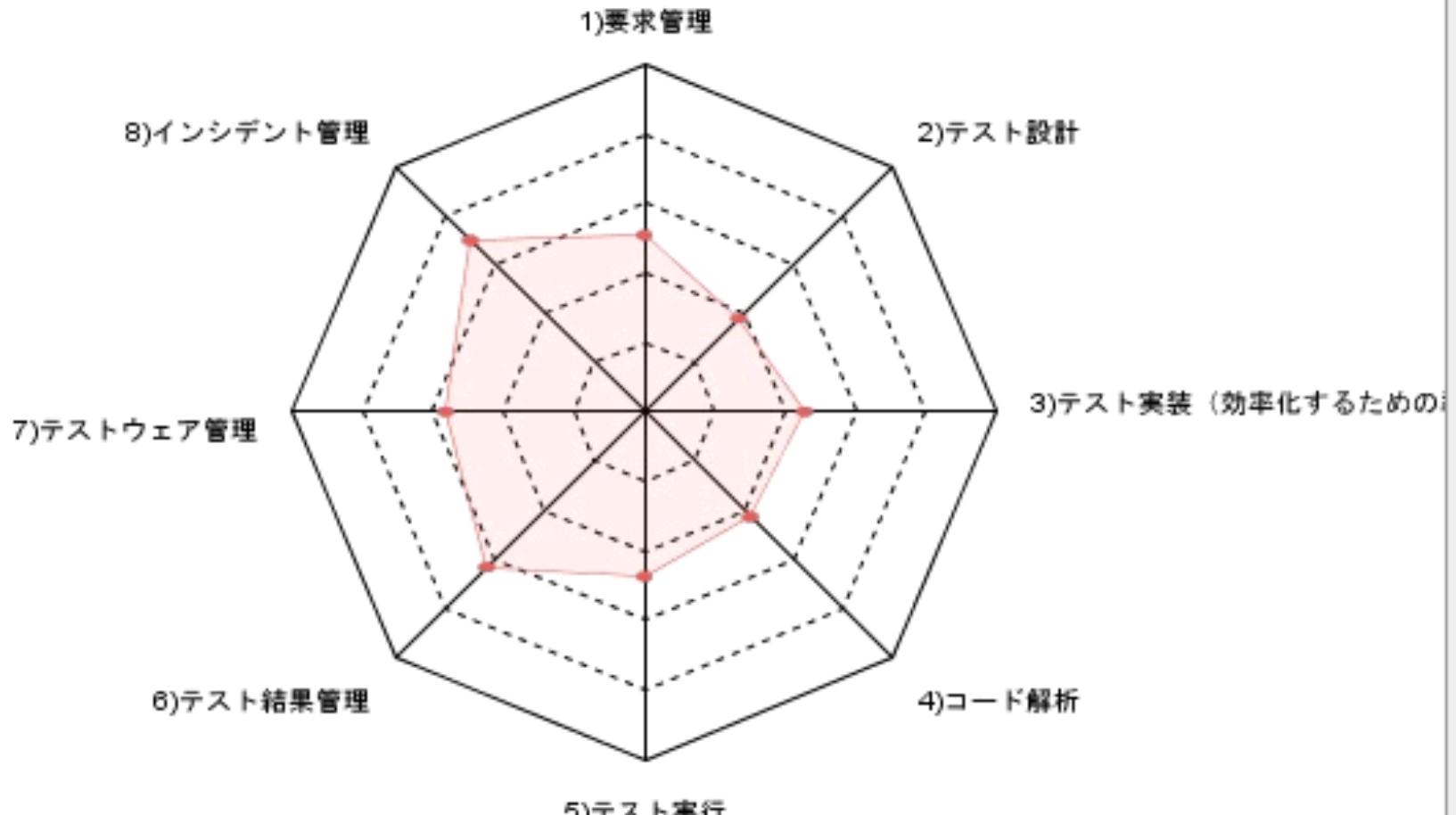
静的解析が注目される背景

- DISA STIG : Defense Information Systems Agency Security Technical Implementation Guide は防衛関連システムのセキュリティガイドとして利用
- DO-178B : Federal Aviation Administration (FAA) は、航空機のソフトウェアに対し DO-178B をコンプライアンスとして利用
- FDA Guidance: General Principle of Software Validation は、US 医療機器ソフトウェアに関するコンプライアンス
- ISO26262: 自動車業界向けのソフトウェア開発に対するコンプライアンス



でも....

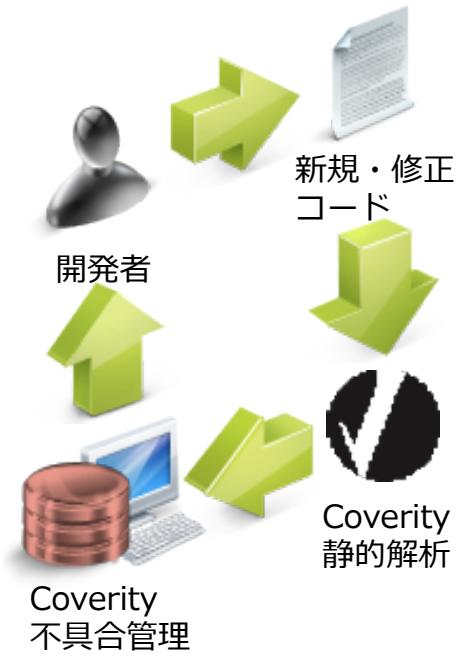
診断結果(平均値)



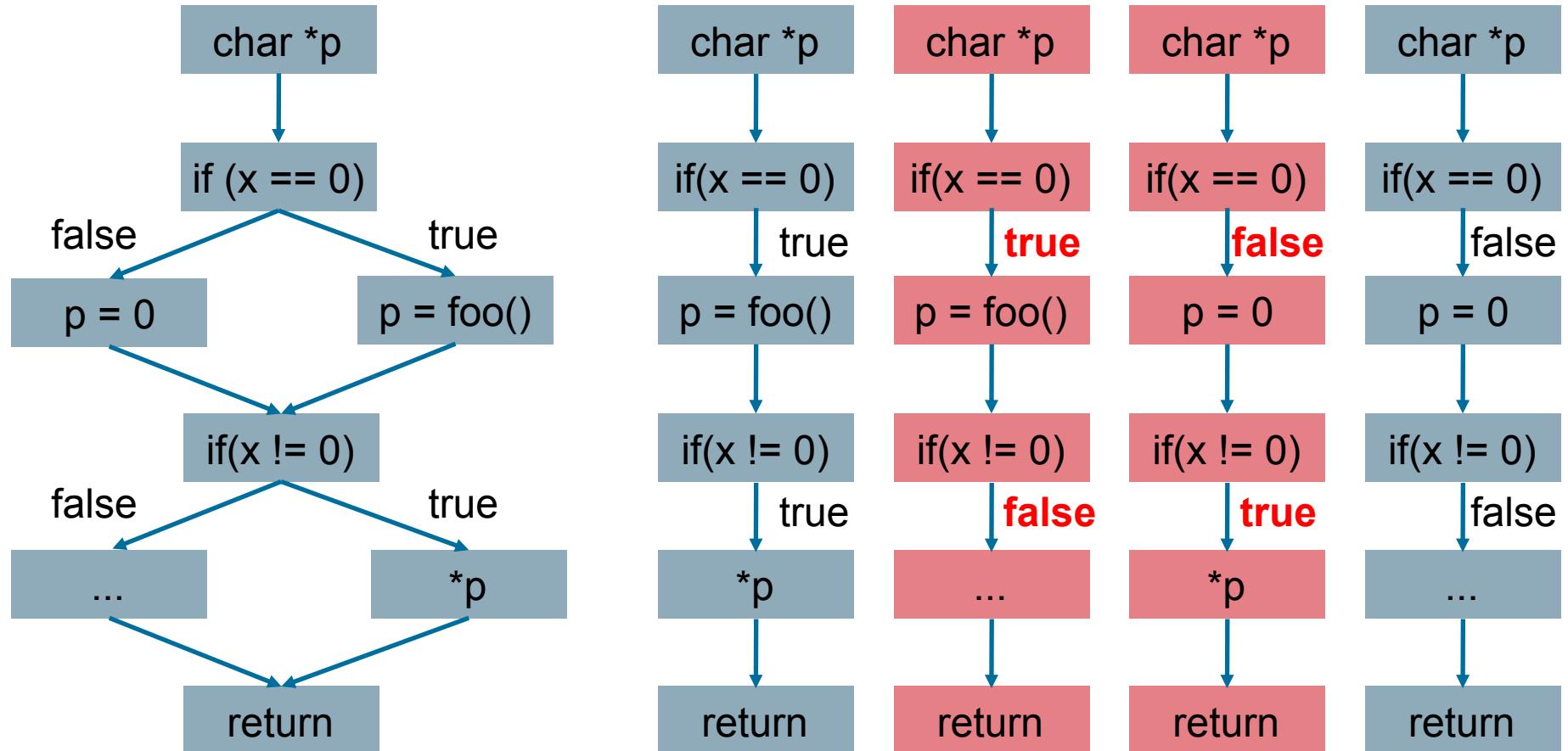
ツールWGの診断の結果。

コード解析の
利用自体が低そう？

というわけで
おさらいから



コードから検出される不具合って



検出される欠陥の種類（主なもの）

C/C++

並列処理の問題

- デッドロック
- 競合状態
- ブロック呼び出しの誤用

パフォーマンスの低下

- メモリ・リーク
- スタックの使用度合い

クラッシュの原因

- Null ポインタの間接参照
- ポインタの解放後のメモリ使用
- 二重解放
- 配列の新規作成と削除の不一致

セキュリティ上の脆弱性

- バッファ・オーバーフロー
- API エラー処理

API などの不適切な使用

- STL の誤使用
- API エラー処理

プログラムの不正な動作

- デッドコード
- 未初期化変数
- 負の変数の無効な使用
- MISRA-C キャスト違反

Java/C#

並列処理の問題

- デッドロック
- 競合状態
- ブロック呼び出しの誤用

パフォーマンス低下/スケーリングの制限

- リソース・リーク
- データベース接続リーク

クラッシュの原因

- null の間接参照
- リソースの解放後の操作

プログラムの不正な動作

- デッドコード
- 未初期化変数
- 負の変数の無効な使用

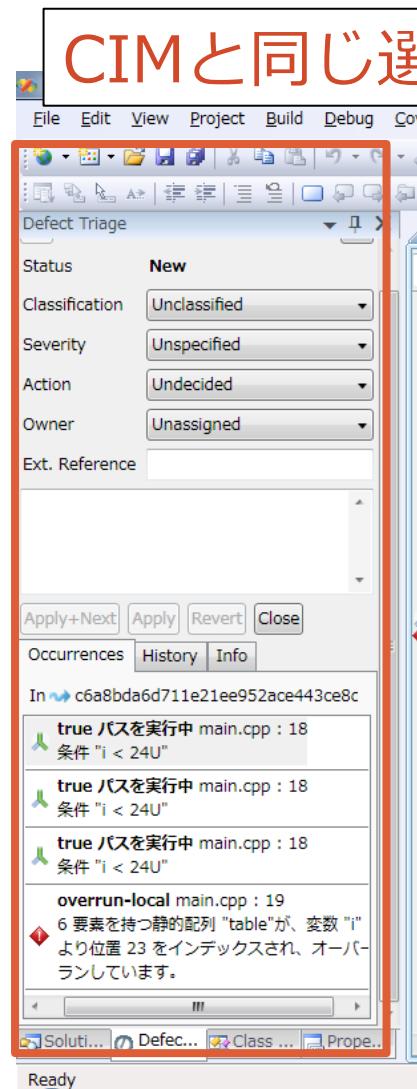
API の不適切な使用

- API エラー処理

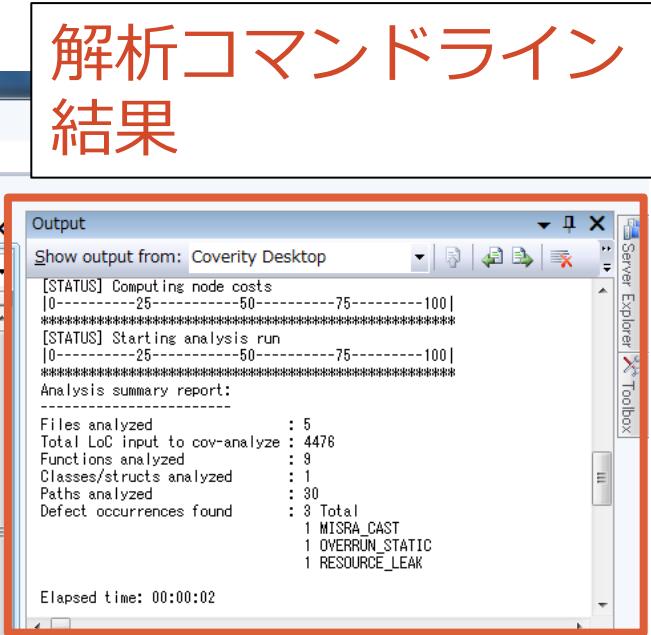
FindBugs で検知された不具合

Visual Studioの画面例

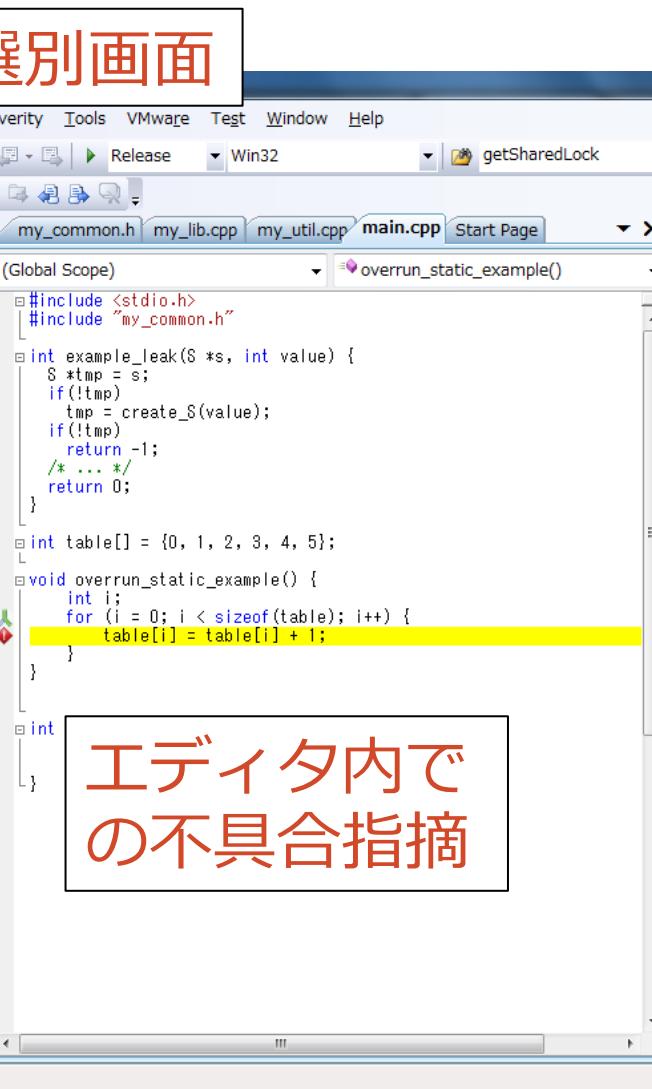
CIMと同じ選別画面



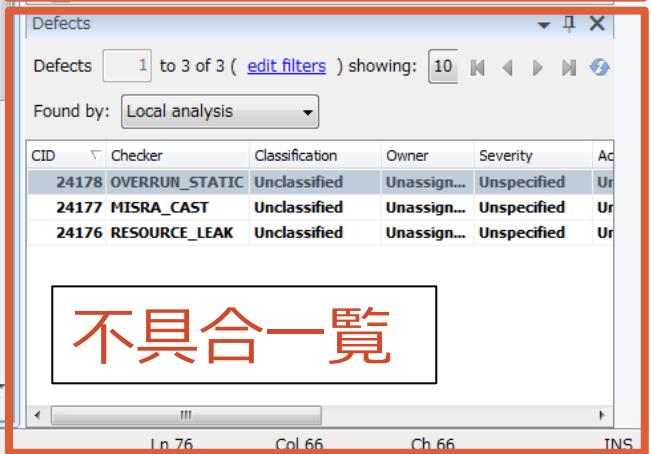
解析コマンドライン結果



**エディタ内で
の不具合指摘**

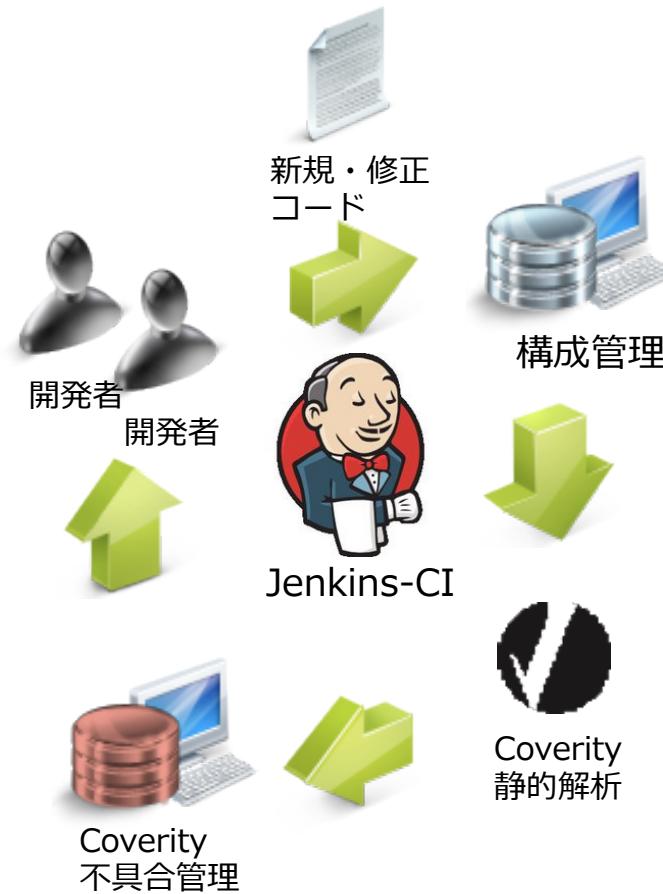


不具合一覧

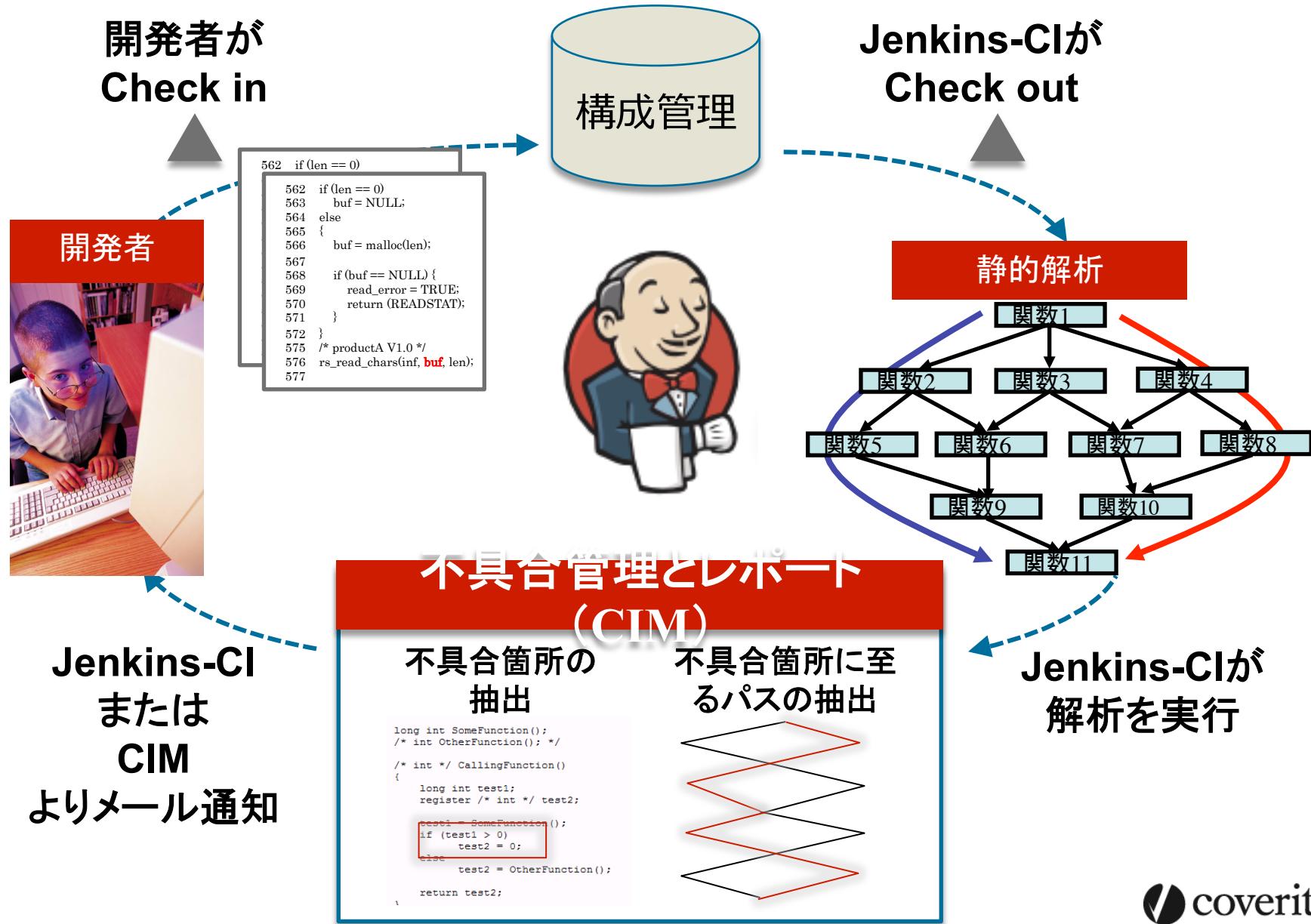


CID	Checker	Classification	Owner	Severity	Action
24178	OVERRUN_STATIC	Unclassified	Unassigned	Unspecified	Unresolved
24177	MISRA_CAST	Unclassified	Unassigned	Unspecified	Unresolved
24176	RESOURCE_LEAK	Unclassified	Unassigned	Unspecified	Unresolved

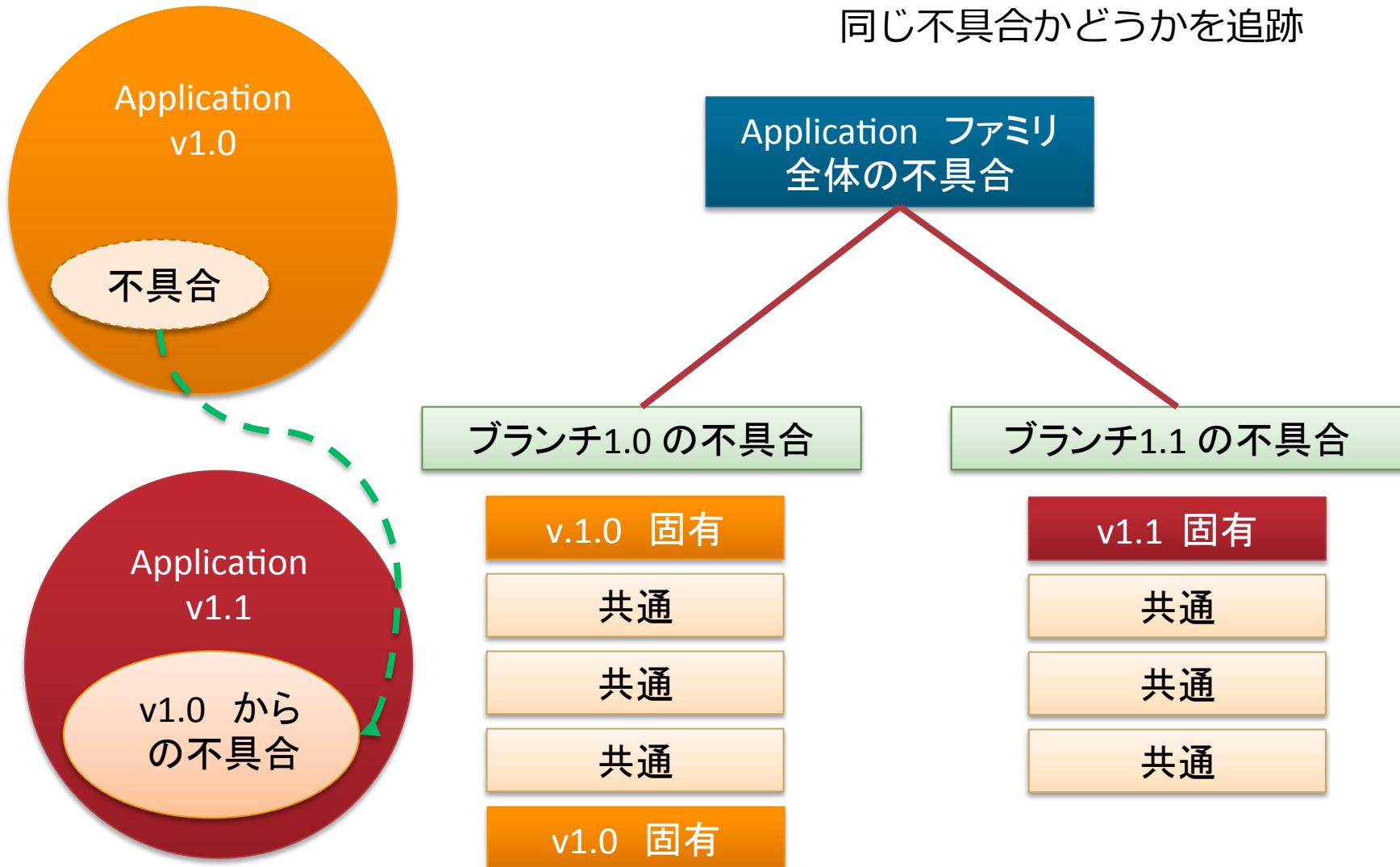
チーム全体の
最新のコード品質を
得るには？



継続的な統合 + 静的解析

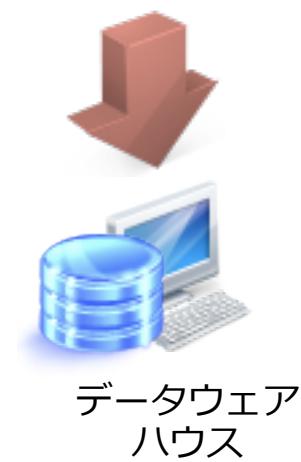
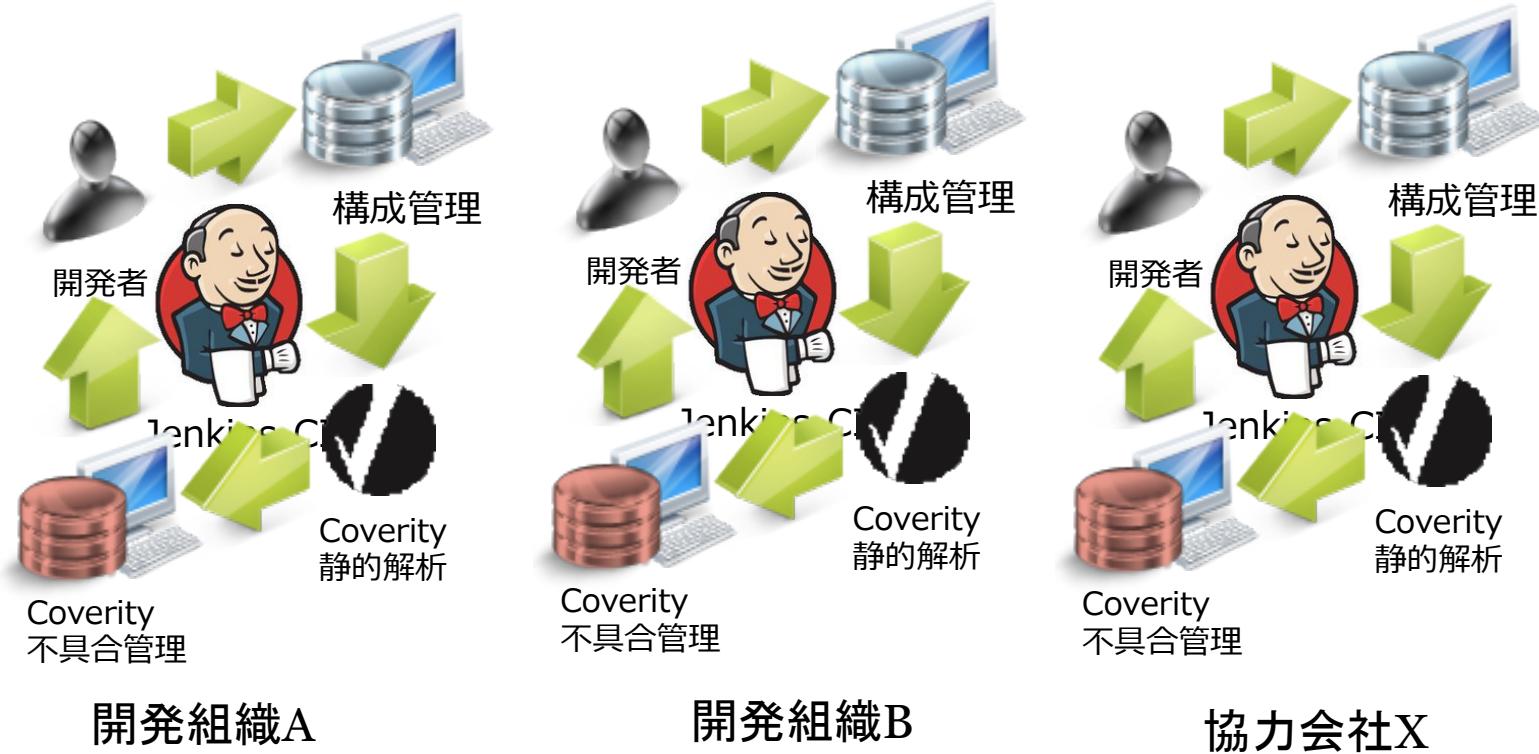


派生開発、第三者コードを考慮したインパクト分析



継続的にコード品質
を確認できる～

このデータを
組織運営に活かす



開発状況の可視化と ソフトウェアコードのガバナンス

ソフトウェアコードガバナンスの3つのステップ

定義

ポリシー、
標準規約
および
閾値の設定

テスト

開発中の
ソースコード
品質の多角的な
モニタリング

コントロール

開発チーム、
サプライチェーンの
管理

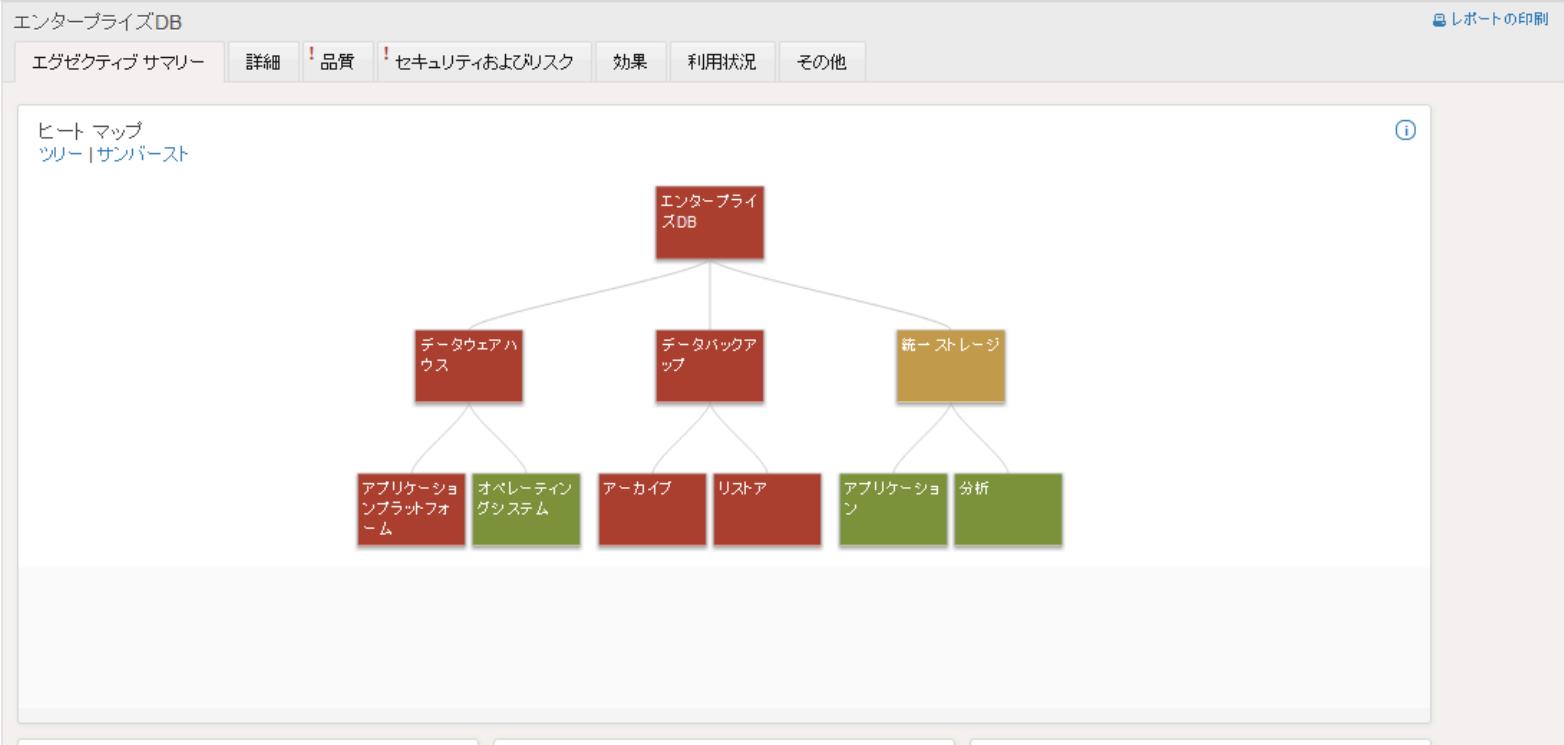
CIM（複数も可）からの情報の収集とモニタリングを行い
企業レベルでのソフトウェアコードのガバナンスを実現

ソフトウェアコードガバナンス

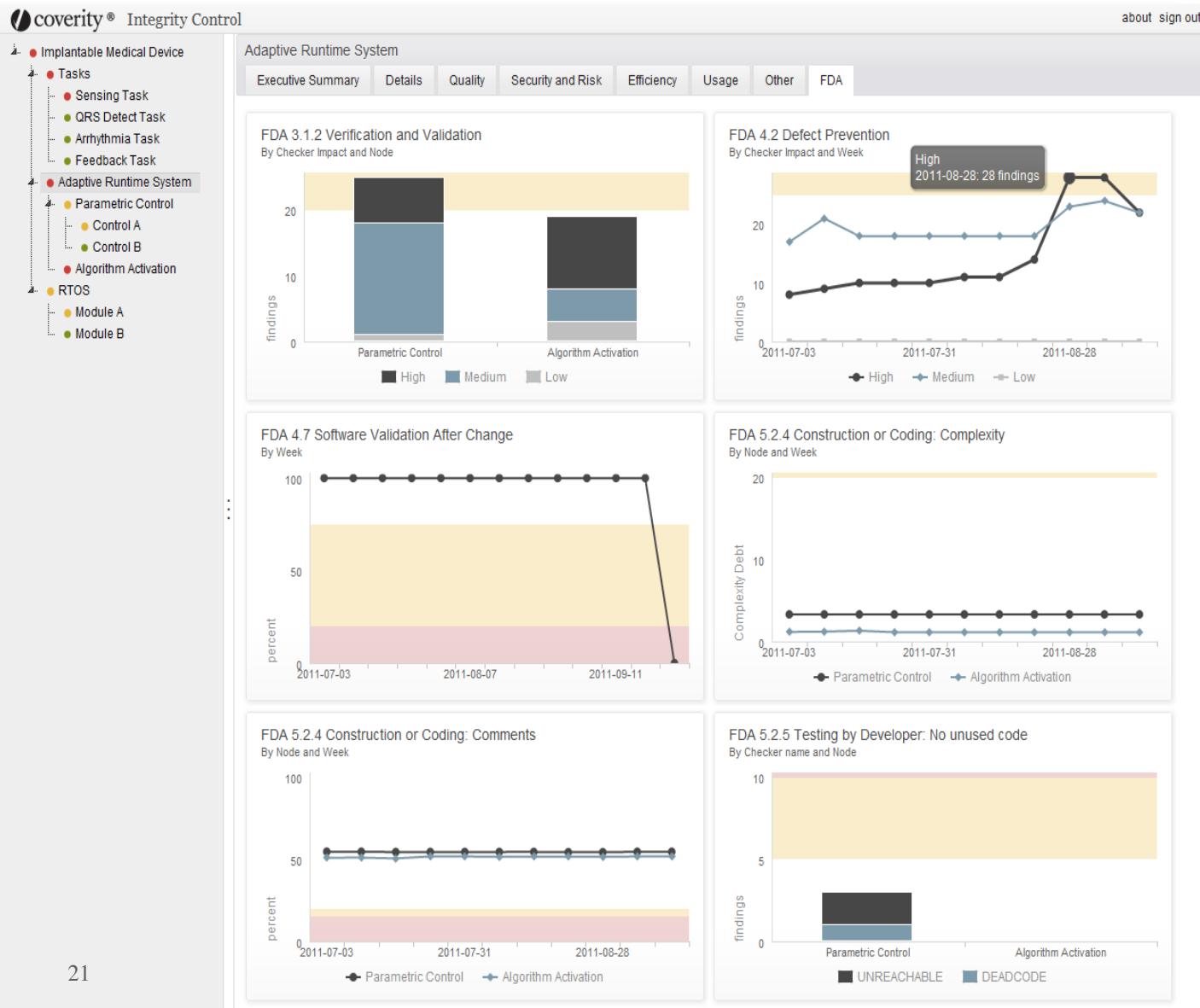
シナリオ1：プロジェクトレベルのリスクマネージメント
プロジェクト内の各コンポーネントがどのような状態にあるか
を隨時監視

シナリオ2：サプライチェーンリスクマネージメント
外部組織からの成果物に対する、品質チェック、セキュリティ規準の
厳守チェックなど
複数の組織を階層的に管理可能

シナリオ3：中央管理体制の確立
全開発組織の組織単位でのソフトウェアコード品質の状況を、VPレ
ベルの視点からレポート



FDA用テンプレート

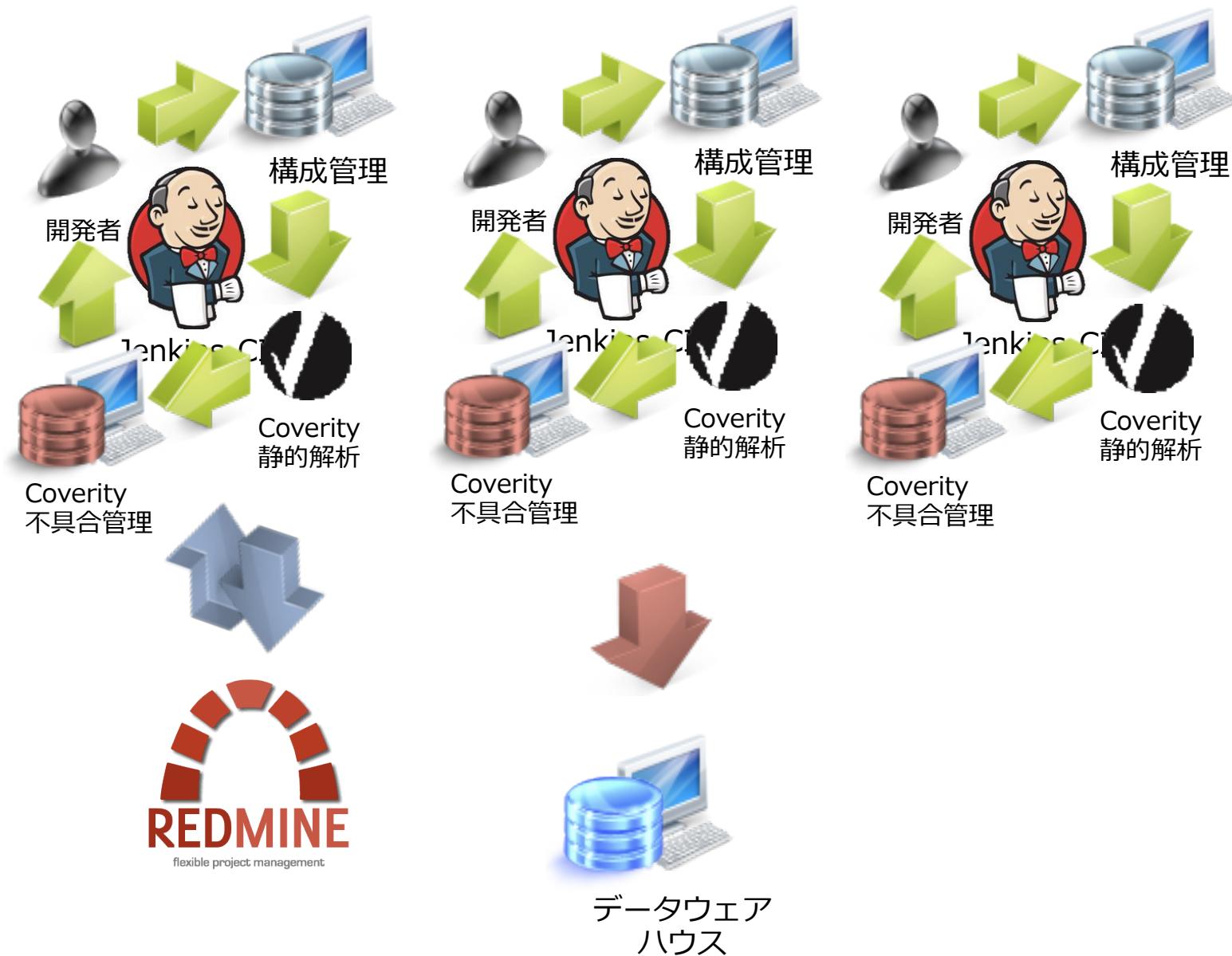


医療機器
プロジェクトであれ
ば、FDA ガイドライ
ンに則したポリシー
テンプレートを
利用できる

ポリシー設定に
より、コンプライ
アンスに関するメ
トリクスのトラッ
キングを容易にす
る

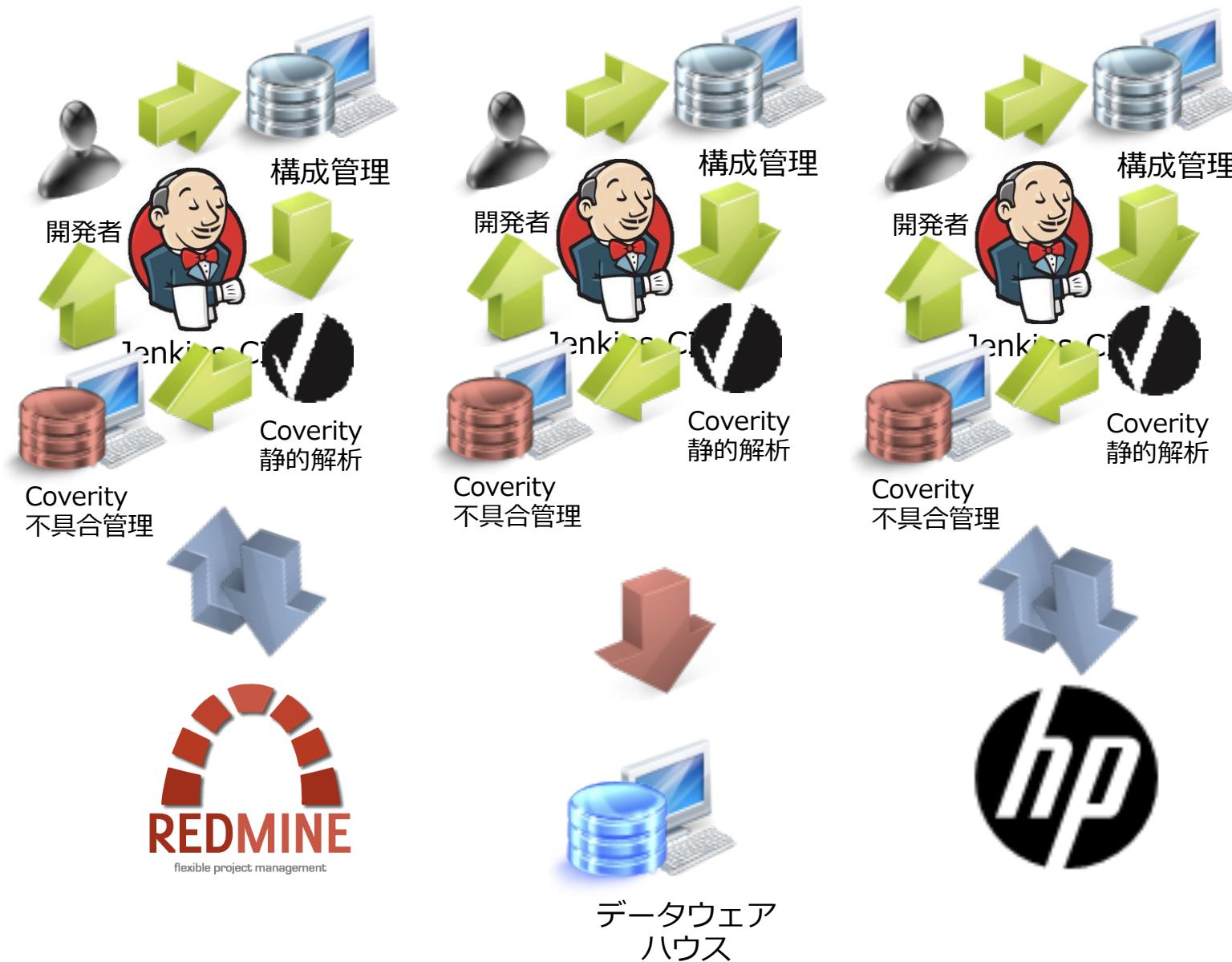
コードレベルなら
モニタリングできるけど

他の不具合も含めた
管理は？



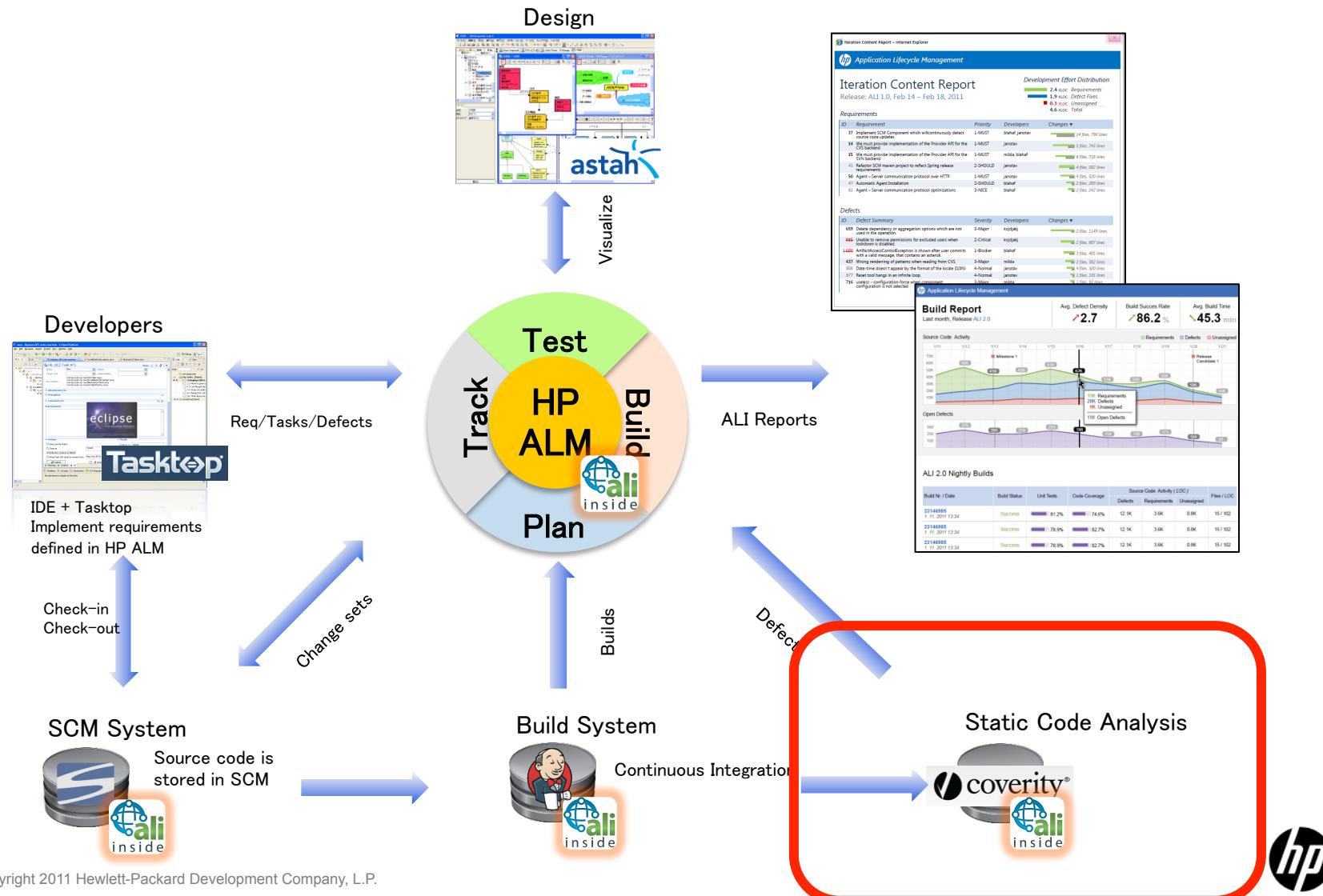
不具合だけでは足りない.....

真の変更管理は？
ALMとの連携

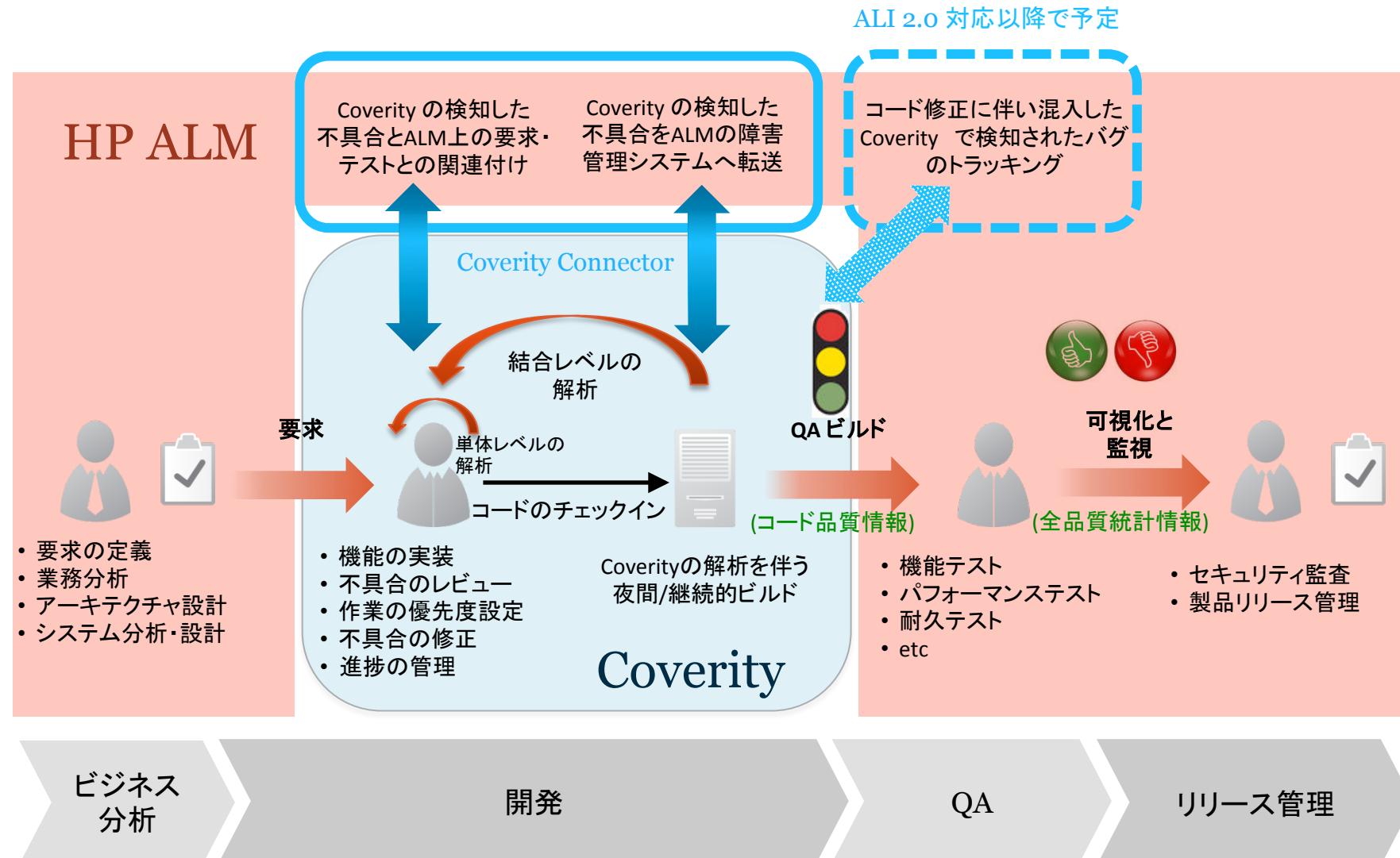


静的解析ツール連携

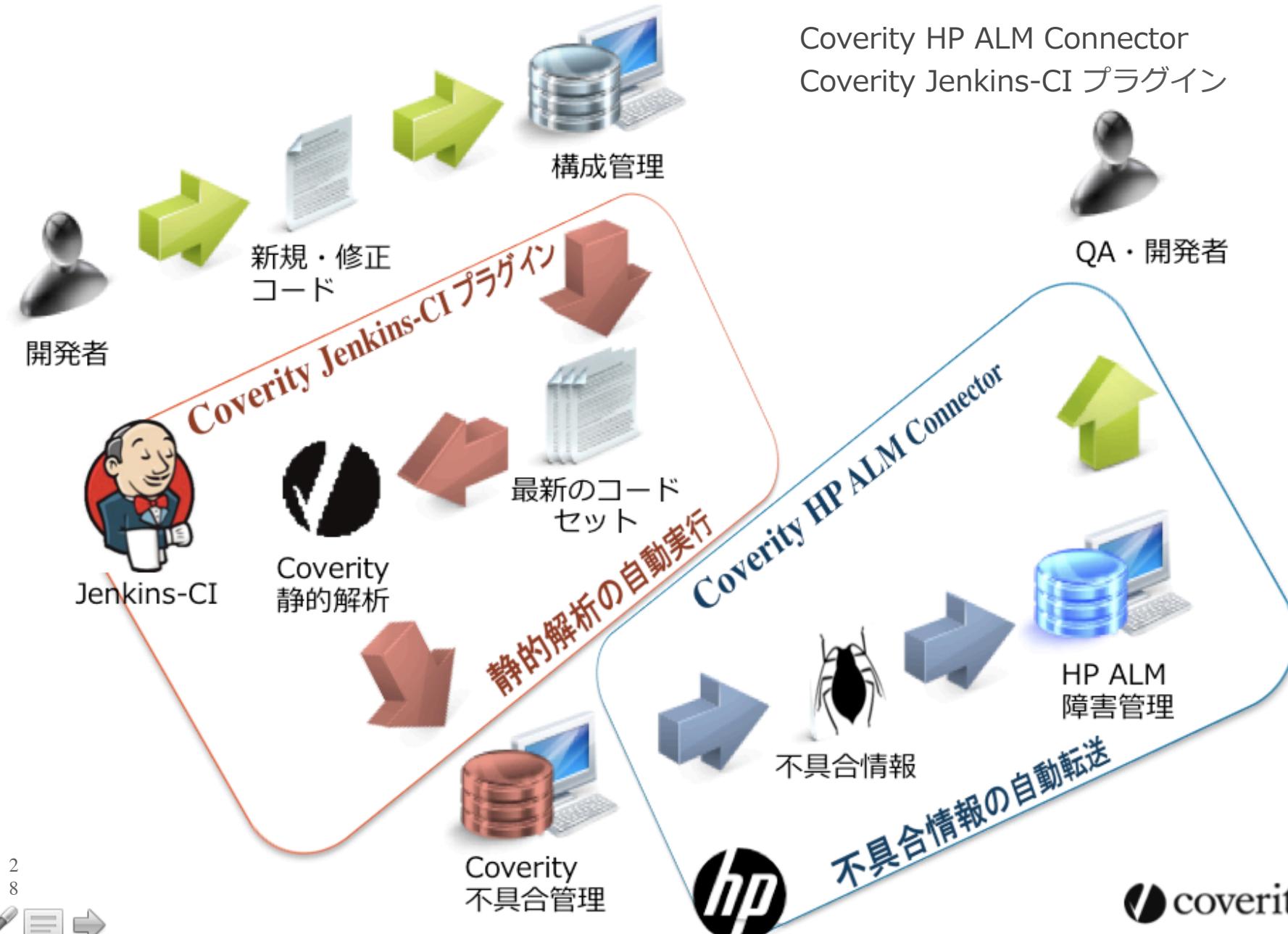
Coverity社 Static Analysis



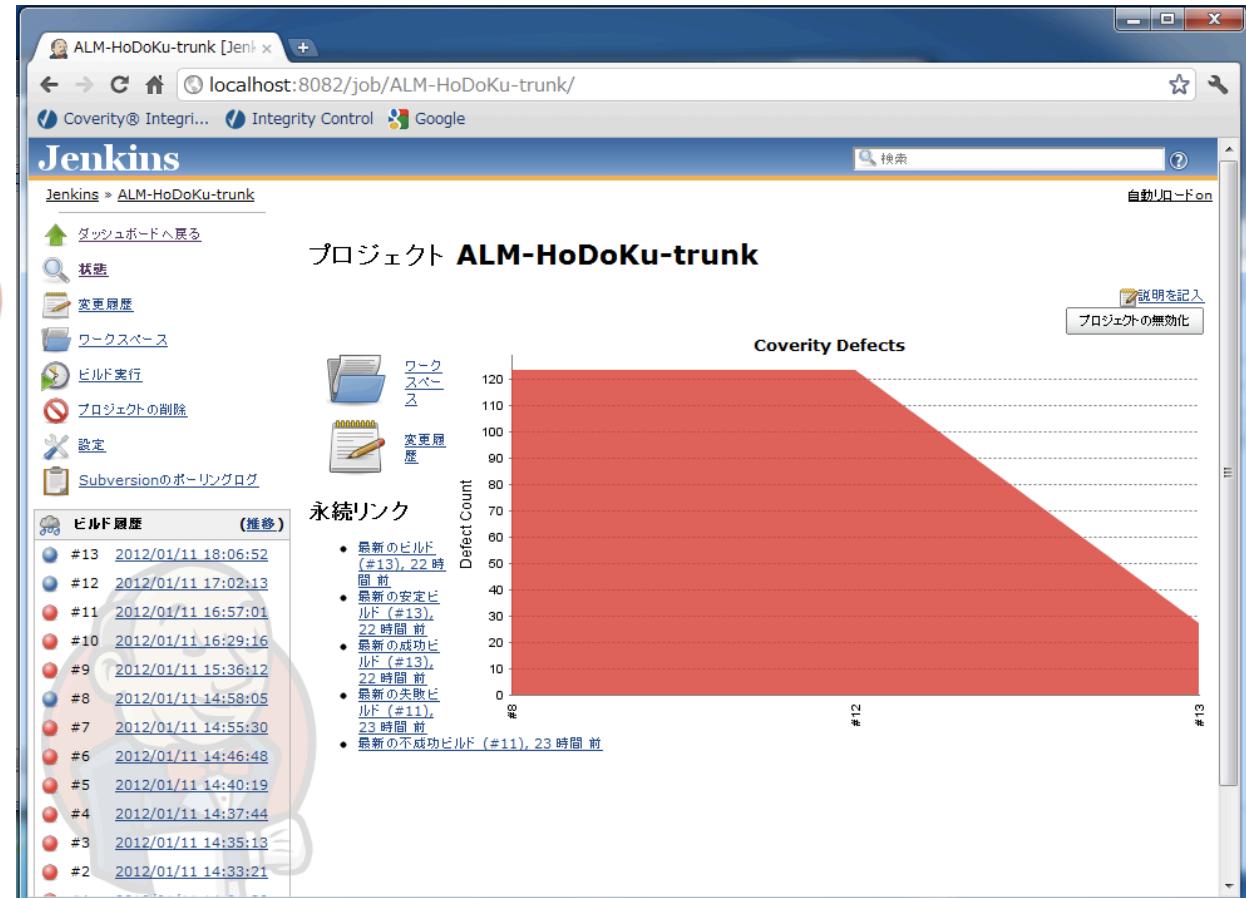
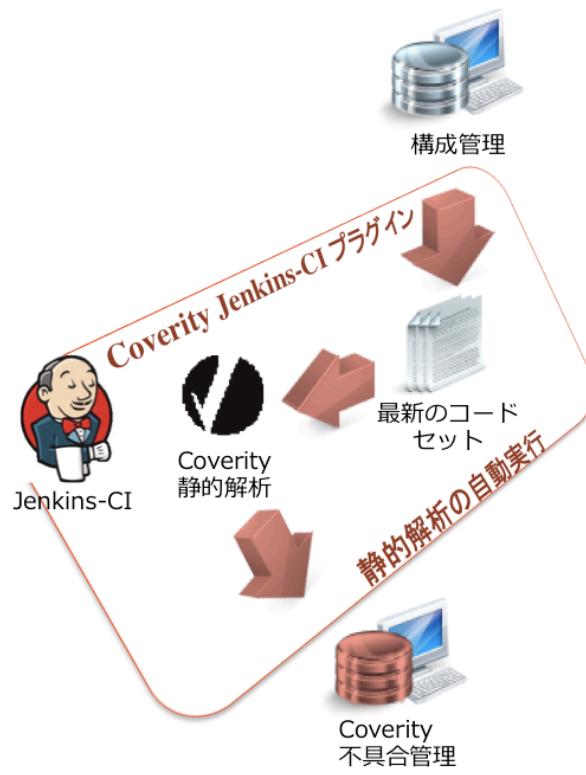
Coverity HP ALM Connector



具体的な連携フロー（不具合情報の例）



Jenkins-CI Coverity プラグイン



構成管理リポジトリをポーリングし、
自動的にビルド／静的解析を実行

Coverity HP ALM Connector 不具合情報の共有

The screenshot shows the HP Application Lifecycle Management (ALM) interface. The left sidebar includes links for Dashboard, Management, Requirements, Testing, Defects (which is highlighted in orange), and Code Changes. The main content area has a 'Defects' tab selected. A table lists two defects assigned to 'james':

Def...	Assigned To	Category	Summary	Subject	Detected By	Severity	Detected on...	Status	Product /
130	james	Coverity Defect	Coverity CID 10012: FB.DLS_DEAD_LOCAL_STORE f...	Book Flight	coverity	4-Very High	2011/09/17	Open	Reservation
131	james	Coverity Defect	Coverity CID 10013: RESOURCE_LEAK found in com...	Book Flight	coverity	3-High	2011/09/17	Open	Booking Sys

Clicking on the second defect (CID 10013) opens a detailed view. The summary states: 'Coverity CID 10013: RESOURCE_LEAK found in com.mtours.MercuryTours.CustomerAddress.validateAddress(java.lang.String, java.lang.String, java.lang.String, java.lang.String, java.lang.String)'. The description panel shows the code snippet:

```
File: /com/mtours/MercuryTours/CustomerAddress.java
Function: com.mtours.MercuryTours.CustomerAddress.validateAddress(java.lang.String, java.lang.String, java.lang.String, java.lang.String, java.lang.String)
First detected on: 2011-09-17 06:38:39 PM

There are 1 occurrences of this defect.

For full defect report please see: http://localhost:5580/sourcebrowser.htm?projectId=10002&mergedDefectId=10013
```

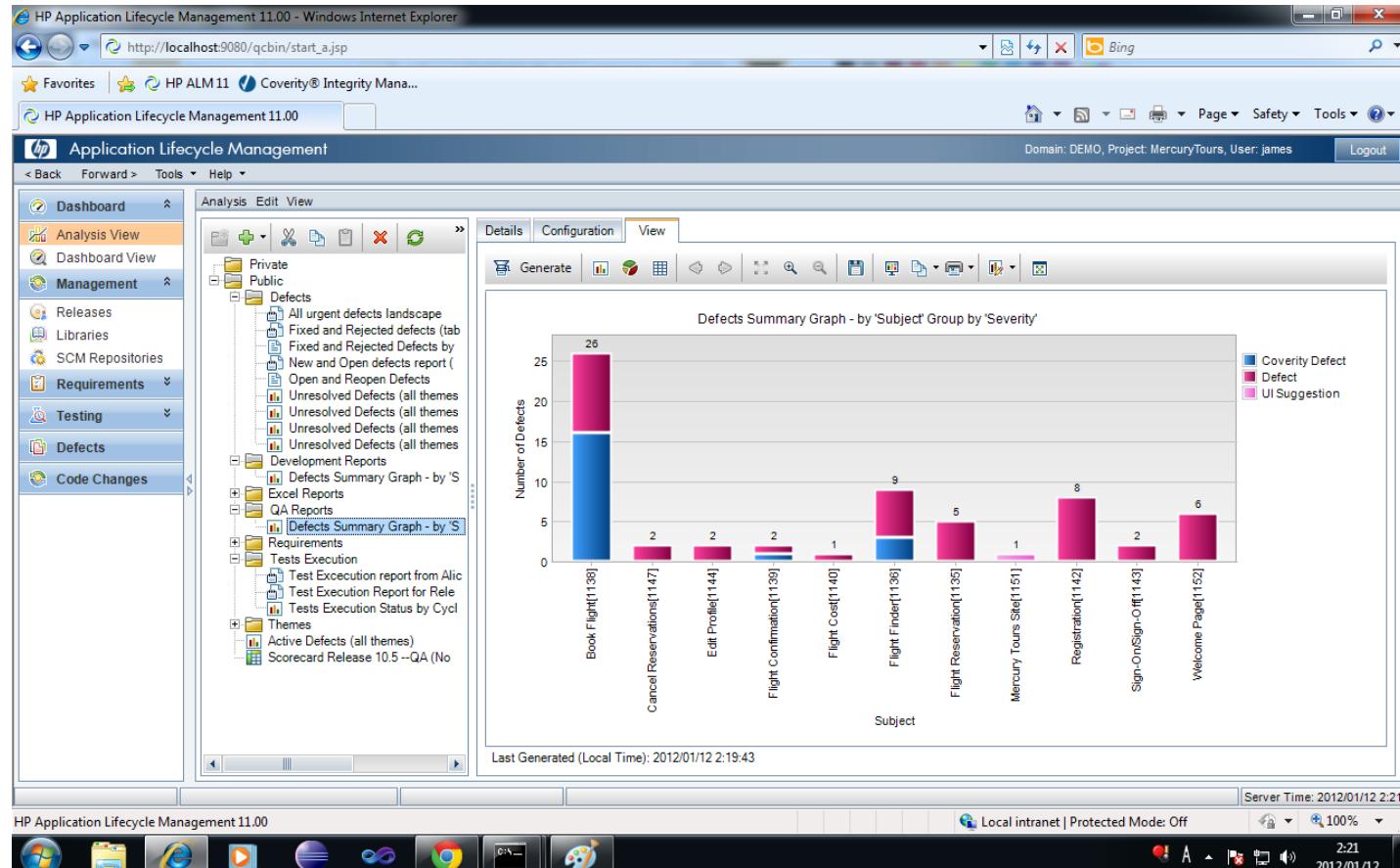
From /com/mtours/MercuryTours/CustomerAddress.java:

```
...
12 String postal)
```

不具合の詳細情報とともに、
HP ALM上に不具合除法を転送



Coverity HP ALM Connector コード品質に関するKPIをALMで一括管理



The screenshot shows the HP Application Lifecycle Management 11.00 interface. The left sidebar has a 'Defects' section under 'Testing'. The main area displays a bar chart titled 'Defects Summary Graph - by 'Subject' Group by 'Severity''. The Y-axis is 'Number of defects' (0 to 25) and the X-axis is 'Subject'. The chart shows the following data:

Subject	Coverity Defect	Defect	UI Suggestion
Block Flight[1138]	15	10	1
Cancel Reservations[1147]	0	2	0
Edit Profile[1144]	0	2	0
Flight Confirmation[1139]	1	0	0
Flight Cost[1140]	0	1	0
Flight Finder[1136]	9	0	0
Flight Reservation[1135]	0	5	0
Mercury Tours Site[1151]	0	1	0
Registration[1142]	0	8	0
Sign-On/Sign-Off[1143]	0	2	0
Welcome Page[1152]	0	6	0

Bottom left: 'Last Generated (Local Time): 2012/01/12 2:19:43' and 'Server Time: 2012/01/12 2:21'.

コード品質に関するKPIは
静的解析ツールから自動収集し
モニタリング可能



今年はどんな立ち位置？



coverity®

無料お試しできます！

展示ブースにお立ちよりください