OSSを活用したソフトウェア開発ツールチェーンの構築

Toolchain for Software Development Utilizing OSS

山元　和子　■YAMAMOTO Kazuko
山中　美穂　■YAMANAKA Miho
森　俊樹　■MORI Toshiki

ソフトウェア開発における品質向上，コスト削減，及び開発期間短縮を実現するには，開発を支援するツールの利用が必須である。開発を支援するツールを導入して，開発者が行う定型作業を自動化し，更に，プロジェクト管理者向けにプロジェクトの状況のリアルタイムな可視化を行う。

東芝は，OSS（オープンソースソフトウェア）と自社開発したツールを組み合わせたうえで，それを互いに連携させた実装～テスト工程向けツールチェーンを構築し，ツール連携による開発作業の自動化とプロジェクト状況の可視化を実現した。

The adoption of development support tools is essential in software development in order to improve quality, reduce costs, and shorten development periods. Development support tools automate the routine work of developers and visualize the project status in real time for project managers.

Toshiba has developed a toolchain consisting of open source software (OSS) and in-house-developed tools, for the processes from implementation to testing. The linkages between the tools in this toolchain realize both automation of development tasks and visualization of the project status.

1　まえがき

高品質なソフトウェアを納期内に効率よく開発することは，ソフトウェア業界共通の課題となっている。この課題を解決するためにはソフトウェア開発を支援するツールの導入が必須である。ツールを導入することで，定型作業の自動化による生産性の向上や，漏れ抜けがない作業による品質の確保が可能になる。近年は市販ツールに加えてオープンソースコミュニティでも，様々なツールが開発されており，そのうちのいくつかは多くのユーザーの支援を得て機能強化が行われ，製品開発に適用されるようになっている。

東芝は，このようなオープンソースのツールと自社開発したツールとを密に連携させた開発環境であるソフトウェア開発ツールチェーン（以下，ツールチェーンと呼ぶ）を構築し，ソフトウェア開発作業の自動化を実現した。更に，ツールチェーンの中に，各ツールに蓄積された開発情報を可視化する機能を組み込むことで，プロジェクト管理者が進捗や品質の状況を即時に把握できるようにした。

ここでは，ツール間の連携を含むツールチェーンの構成と導入事例について述べる。

2　ツールチェーンの構成

2.1　ツールチェーンの適用範囲

ソフトウェア開発を支援するツールには，開発の上流工程から下流工程にわたって様々なものが存在する。そのうち，要求仕様化から実装開始までの上流工程は，製品分野に適した設計方法及びツールが適用されることが多く，汎用的なツールによる自動化の効果は限定的である。一方で実装からテストまでの下流工程では，製品分野に関わらず基本的に実装，テスト，及び不具合修正といった作業が行われるため，これらの作業のうち定型的な部分を汎用的な環境によって自動化する

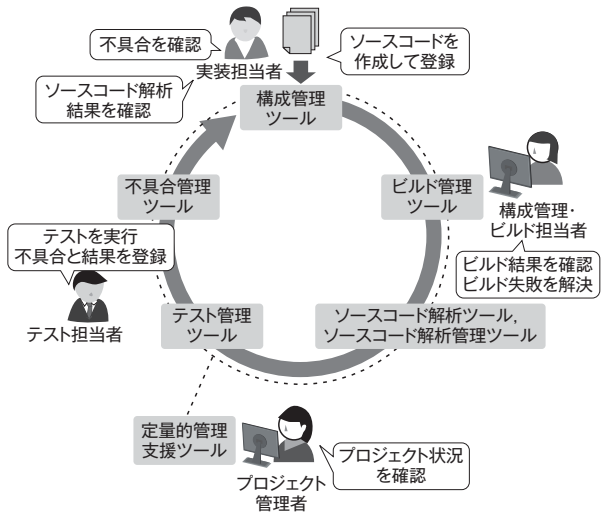

図1．ソフトウェア開発ツールチェーンの構成 — ソースコードを構成管理ツールに登録する作業を起点とし，ビルド，ソースコード解析，テストによって品質を確認する一連の作業をツールにより自動化する。この流れは一例であり，プロジェクト規模や開発プロセスによって作業の順番や実施担当は異なる場合がある。

System configuration of toolchain for software development

効果は大きい。

このため，ツールチェーンは，実装～テスト工程を適用範囲とし，製品ドメインに関わらず汎用的に適用可能なものとした。

2.2　ツールチェーンの全体像

実装～テスト工程をツールチェーンで自動化する場合の開発手順を**図1**に示し，図中の矢印に沿って説明する。

(1)　実装担当者がソースコードを作成し，構成管理ツールに登録する。構成管理ツールは登録されたソースコードのバージョンを管理する。

(2)　続いて，ビルド管理ツールが，構成管理ツールからソースコードを取得し，実行可能ファイルの作成（ビルド）を行う。ビルドが失敗した場合は関係者にメールを送付する。構成管理・ビルド担当者は，ビルド結果やビルド履歴をビルド管理ツールのWeb画面から確認することができる。

(3)　ビルド管理ツールは，ビルドに続いてソースコード解析ツールを起動する。ソースコード解析はコーディング規則違反のチェックや実行時エラーを引き起こすようなコーディングミスのチェックを行うものである。実装担当者はソースコード解析管理ツール上で解析結果を閲覧する。

(4)　テストを自動化している場合は，ビルド管理ツールから自動テストを起動する。テスト担当者は，手動のテストを実施するほか，テスト結果と発見された不具合をそれぞれテスト管理ツールと不具合管理ツールに登録する。

また，矢印の流れとは別に，プロジェクト管理者は，定量的管理支援ツール上で，最新のソースコード規模や不具合数，ソースコード解析による指摘項目数といった各種メトリクスを閲覧し，プロジェクトの進捗やソフトウェアの品質状況を確認する。

これらの作業手順では，ビルド以降の定型的な作業をビルド管理ツールによって自動化しており，ソースコードを作成してその品質を確認するまでを基本サイクルとした作業の時間を短縮することができる。また，定量的管理支援ツールは，各種ツールからデータを収集して可視化するまでの煩雑な作業の自動化を実現している。

2.3　ツールチェーンを構成するツール

実際にツールチェーンに組み込むツールは，機能性，ユーザビリティ，及び導入と保守に掛かるコストの観点でベンチマークを行い，必要十分なものが社外から導入できる場合はそれを利用し，独自技術を盛り込んだツール化が必要な場合は当社で開発した。ツールチェーンを構成するツールを**表1**に示す。

OSS（オープンソースソフトウェア）を導入する際には，特に以下の点を考慮した。

(1)　必要な機能を備えている

(2)　ユーザビリティに優れていてユーザーから受け入れられやすい

(3)　利用ユーザーが多く，要望や不具合報告を反映してスピーディなバージョンアップが行われており，品質が安定している

表1．ツールチェーンを構成するツール
Tools comprising toolchain

ツール種別	利用ツール
構成管理ツール	Subversion*1
ビルド管理ツール	Jenkins*1, *2
ソースコード解析ツール／ソースコード解析管理ツール	市販ツール／自社開発ツール
テスト管理ツール	自社開発ツール
不具合管理ツール	Redmine*1あるいは自社開発ツール
定量的管理支援ツール	自社開発ツール

＊1：Subversion[1]，Jenkins[2]，Redmine[3]はいずれもOSS
＊2：Jenkinsは継続的インテグレーションツールとも呼ばれる

特に，JenkinsとRedmineについては，ツールをユーザーが独自に拡張する仕組み（プラグイン機構）があり，これを利用してユーザーが自部門のプロジェクト向けにツールをカスタマイズしたり，後述するツール連携を実現したりできることがツール選択の重要な理由になった。

自社開発のツールに関しては，その特長を以下に示す。

2.3.1　ソースコード解析管理ツール TCANA$_{TM}$　ソースコード解析ツールは，ツールによってコーディングルール違反やコーディングミスの検出特性に違いがあるため，当社では複数のツールを相補的に組み合わせることにより，全体としての検出率を高める運用を行っている。しかし，ユーザーが複数のツールの検出結果をそれぞれ確認するのは煩雑で手間が掛かる。この問題を解決するため，ソースコード解析管理ツールTCANAは，複数のソースコード解析ツールによる結果をこれまでに蓄積された知見を活用して選別，集約しWeb上に表示することで，ユーザーが検出箇所をレビューする手間を削減する（**図2**）。

2.3.2　テスト管理ツール　テスト管理ツールは，テスト項目とテスト結果を管理し，Web上での情報共有を可能にする。自社開発のテスト管理ツール[4]はこれらの基本機能に加えて，不具合数の多いモジュールを可視化できるグラフや，機能デグレードを可視化できる一覧表，品質安定傾向を可視化できる信頼度成長曲線[5]，テスト進捗状況を可視化できるグラフといった各種表示機能を備え，品質向上及びテスト効率化を支援する。

2.3.3　不具合管理ツール PRISMY$_{TM}$[6]　不具合管理ツールは，不具合を，“新規作成→担当者に振分け→修正済み→対応完了”といったワークフローに従って処理する一連の作業を管理し，Web上で情報共有を可能にする。PRISMYは，こうした一般的な不具合管理機能に加えて，分散開発向けに複数拠点間に設置したPRISMYの間で不具合情報の同期を取る機能や，豊富な種類のグラフ表示による品質可視化機能

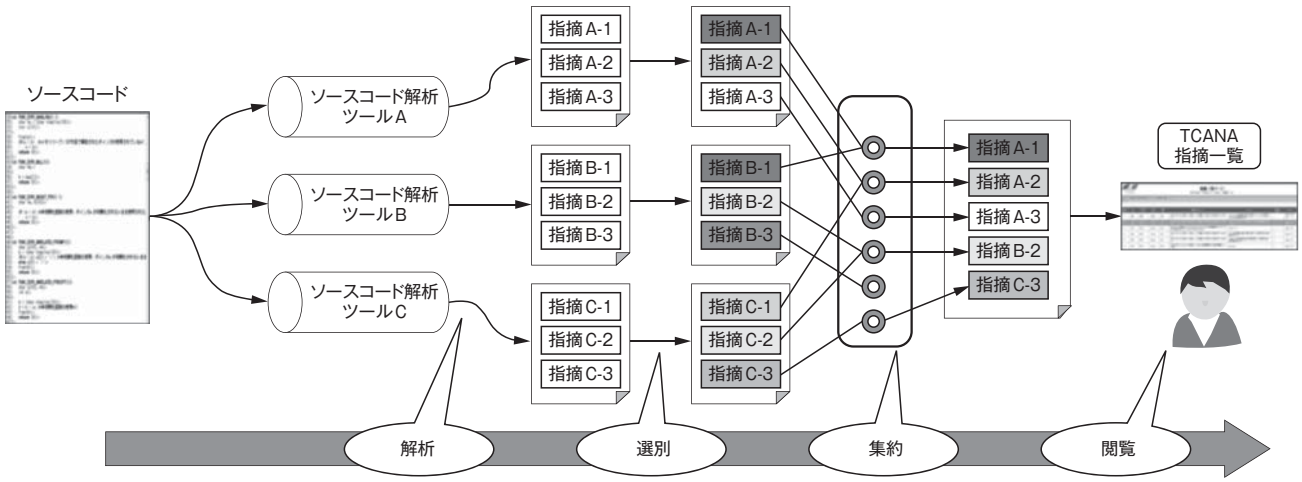

図2. TCANAによる複数のソースコード解析結果の集約表示 — 複数のソースコード解析ツールによる解析結果をTCANAが選別し集約して，Web上に一覧表示する。

Summary display of multiple source code analysis results using TCANA

を持っている。

2.3.4 定量的管理支援ツール　ツールチェーンに組み込んだツールやソフトウェア開発に利用しているその他ツールに蓄積された情報をあらかじめ定められたスケジュールに従って自動収集する。ユーザーは，収集したデータに基づく各種グラフをWeb上のダッシュボードと呼ぶ一覧表示画面（**図3**）で閲覧することができる。更に，データを属性で絞り込んだり，あらかじめ定めたしきい値から外れたデータがあれば警告を出力したり，といった分析支援機能を持っている。

データ収集や分析の機能はプラグインとして本体に追加できる構造となっており，ユーザーが独自にプラグインを作成して，任意のツールからデータ収集を行い，新たなグラフを追加することができる。

2.4 ツール間の連携による自動化

ソフトウェア開発ツールチェーンを構成する六つのツールは，それぞれ独立して利用するのではなく連携させて利用する。ツールを互いに連携させることで，別々のツールに格納された情報間のトレースが自動化できる。例えば，“不具合Aを修正したソースコード箇所の特定”を行いたい場合に，不具合管理ツールと構成管理ツールが連携していなければ，人手で二つのツールを操作しなければならないが，ツールが連携していれば，不具合管理ツールから自動的に修正したソースコード箇所を特定することができる。

トレースしたい情報の観点から，ツール間に必要な連携を整理した結果を図3に示す。関連付けは，ツールが持つ機能やプラグインで実現する。今回利用するOSSでは，ツール間を連携するための機能やプラグインが充実しており，これらを利用するか，必要な連携機能を当社で開発した。OSSへの必要な連携機能の追加は，ソースコードが公開されているため，

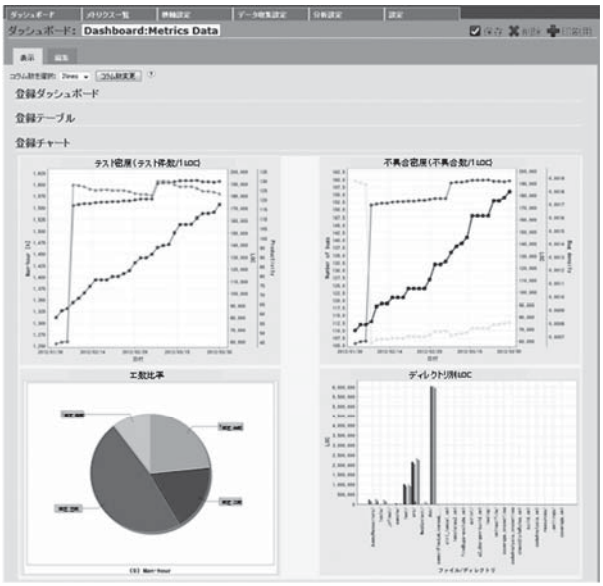

図3. 定量的管理支援ツールによるプロジェクト状況の可視化 — ダッシュボード上でソースコード行数（LOC：Lines of Code）や，不具合数，工数といった各種情報を可視化できる。

Visualization of project status by quantitative project management support tool

比較的容易に行うことができる。

3 ツールチェーン導入事例と普及展開のための施策

3.1 ツールチェーン導入事例

ツールチェーンを導入した組込みソフトウェアの開発プロジェクトでは，図1に示した作業が基本的に1日1サイクル行われるように運用ルールを定めた。実装担当者は，1日に1回以

上，編集したソースコードを構成管理ツールに登録する。登録時には，ソースコード編集に関連する不具合番号と編集に関わる作業時間を登録するルールにした。ツールチェーンをこのような運用ルールで利用することで，以下の導入効果が確認できた。

(1) ツールチェーン導入前は数日以上ビルドの間隔が空くことがあったが，1日1回以上ビルドとソースコード解析を行うことで，デグレードを早期に発見し，品質を維持することができた。

(2) **図4**に示す各種ツール連携を利用することで，作業効率が向上した。そのような作業の代表例としては，ビルド失敗の原因となったソースコード箇所の特定や，不具合を発見した際のテスト情報の取得などが挙げられる。

(3) ツールチェーン導入前は，各種ツールからデータを収集して可視化するのに半日ほどを要したため，プロジェクト管理者は，週1回程度の頻度でしかプロジェクトの状況が確認できなかったのが，まったく人手を掛けることなくリアルタイムにデータ収集と可視化を行えるようになった。

3.2 普及展開の施策

ツールチェーンを利用するには，六つのツールとその間の連携機能をインストールする必要がある。これには時間が掛かるうえ，OS（オペレーティングシステム）やミドルウェア，ツール設定の知識とスキルが必要になる。この問題を解決するため，ツールチェーンを仮想化ソフトウェア(注1)上に構築して，オールインワンパッケージとして提供できるようにしている。オールインワンパッケージには，統合的なユーザー認証の仕組みやマルウェアのチェックツールも組み込みセキュリティを確保した。仮想化ソフトウェア上に構築することでクラウドからの提供にも容易に対応することができる。

また導入部門で，ツールチェーンを効果的に利用できるように，ユーザー向けの教育や，ベストプラクティスに基づく活用ガイドの充実も図っている。

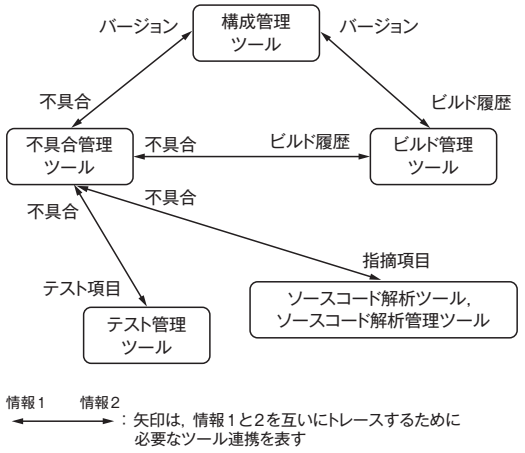

図4．実装～テスト工程で必要なツールの連携 ― 異なるツールが格納する情報をトレースするためにはツールを連携させる必要がある。
Linkages between tools necessary in processes from implementation to testing

(注1) 1台のコンピュータの上で複数のOSやアプリケーションを動作させるソフトウェア。

4 あとがき

実装～テスト工程で汎用的に利用できるツールチェーンを構築した。ツールチェーンに組み込むツールをベンチマークにより決定し，機能性に優れるSubversionやRedmine，JenkinsといったOSSと自社開発ツールを組み合わせた構成とした。Jenkinsでビルドを中心とした作業を自動化したのに加えて，ツール間の連携による高度な自動化を実現した。

今後は，ツールチェーンの機能を拡充して作業の自動化範囲を広げるとともに，社内のプロジェクトへの展開普及を図っていく。

文 献

(1) Apache Software Foundation. "SUBVERSION". <http://subversion.apache.org/>, (accessed 2012-07-19).
(2) Jenkins CI community. "Jenkins". <http://jenkins-ci.or.g/>, (accessed 2012-07-19).
(3) Jean-Philipp Lang. "Redmine". <http://www.redmine.org/>, (accessed 2012-07-19).
(4) 河村 透 他. ソフトウェアのテスト管理システム. 東芝レビュー. **66**, 1, 2011, p.32－36.
(5) 山田 茂. ソフトウェア信頼性モデル－基礎と応用. 東京，日科技連出版，1994，202p.
(6) 東芝情報システム（株）. “不具合情報管理システム PRISMY”. <http://www.tjsys.co.jp/page.jsp?id=742. >,（参照 2012-07-19）.

山元 和子 YAMAMOTO Kazuko
ソフトウェア技術センター ソフトウェア設計技術開発担当参事。ソフトウェア開発インフラ構築技術の研究・開発に従事。
Corporate Software Engineering Center

山中 美穂 YAMANAKA Miho
ソフトウェア技術センター ソフトウェア設計技術開発担当主務。ソフトウェア開発インフラ構築技術の研究・開発に従事。
Corporate Software Engineering Center

森 俊樹 MORI Toshiki
ソフトウェア技術センター プロセス・品質技術開発担当参事。ソフトウェア品質技術及び定量的プロジェクト管理技術の研究・開発に従事。
Corporate Software Engineering Center