OSS 開発支援ツールの利用技術に関する取り組み

Approach to Application Technology on OSS-based Development Supporting Tools

鈴　木　生　雄

要　約　社会・企業における情報システムの重要性が高まり，情報システム産業においてより一層の品質や生産性の向上が求められる中，ソフトウェア企業の開発現場では，開発環境の統合化や開発プロセスの標準化など，多面的な取り組みが進められてきている．OSS の開発環境においては，OSS の開発支援ツールも機能性，品質が向上しており，急速に普及してきている．こうしたツールの多様化，高度化に伴い，個々のシステム開発の特性や制約条件に適したツールの選択や，利用技術に関する知財の蓄積が望まれている．ツールの選定においては，開発プロセス全体を網羅したツールはなく，分野ごとに異なる開発コミュニティが開発した多様なツールが混在するため，ツール同士の連携を考慮することが重要である．ツールの運用においては，開発プロジェクトの形態が多様化する中，プロジェクト特性を考慮したプロセスの策定が不可欠であり，規模，期間といったプロジェクト特性や，分散性（集中/分散），流動性といった開発要員特性などによる考慮が不可欠となる．

本稿では，OSS 開発支援ツールの利用技術として，実プロジェクトでの適用評価を通し，ツール選定および開発プロセスにおける運用の視点で，開発環境に関して考察する．

Abstract　The more important the information system has become in the most of the organization, the more severe quality and productivity has been demanded in the information service industry. As a result, a multipronged approach, including the integration of development environment, the standardization of development process is indispensable in the development site of software developers.

The environment development tool of open source has improved in capability and quality, and has spread rapidly. System integrator needs to accumulate the knowledge of the useful tools and the method of using such tools. For selection of tools, there is no development community that develops the development support tools covering the entire system development process. Therefore, system integrator has to mix-and-match various tools which were developed by different communities and to study the compatibility among various tools. For utilization of tools, it is absolutely necessary to formulate the operational procedure in which the project properties including size and period of time, as well as the properties of development personnel including the dispersibility (concentration/ decentration) and fluidity are considered, based on the diversity of the project types.

This paper discusses the development environment as the application technology on the OSS-based development supporting tools through the evaluation process of application to the actual project, from the view point of the tool selection and the application to development process.

1. は　じ　め　に

社会・企業における情報システムの重要性が高まる一方で，情報サービス産業においてはよりいっそうの品質，生産性の向上が求められており，ソフトウェア企業は多面的な努力によっ

てそれに応える必要がある[1]．開発現場における開発プロセスの標準化や開発環境の統合化が進む中，オープンソースソフトウェア（Open Source Software，以下 OSS と表記）系の開発支援ツールは，その機能性や品質の向上と共に急速に普及してきている．開発対象システムやプロジェクトの持つ特性や制約条件に応じ，多様な OSS ツールから適切なものを選定し，その効果的な適用に向けて利用技術・ノウハウの蓄積を図ることは，ソフトウェア開発・保守環境を改善する上で重要な取り組みの一つである．

本稿では，OSS の開発支援ツールの利用技術として，実際の Java 言語を利用したソフトウェア開発プロジェクトへの適用を前提に，ツール選定，適用，並びに開発運用に関する取り組みを紹介するとともに，その適用評価結果と今後に向けた課題を述べる．

2. システム開発現場における開発支援ツールの現状と課題

急速に普及しつつある OSS 開発支援ツールをシステム開発に利用する際，多様なツールの中から適切なものを選定するための視点や，選定したツールの効果的な適用や開発運用が課題となる．

2.1 ツール選定の課題

システム構築プロジェクトでは，通常，プロジェクトごとに，構成管理，要件管理，障害管理，ソースコード管理等の複数の開発支援ツールを組み合わせて利用する．Microsoft 社製品・OSS のそれぞれについて，適用領域ごとの主要な開発支援ツールの例を表 1 に示す．

表 1　Microsoft 社の開発支援ツールと OSS の開発支援ツール

適用領域	Microsoft社	OSS
プロジェクト管理	Project	ProjectKeeper
プログラミング支援	Visual Studio	Eclipse NetBeans
ソースコード管理	Visual Source Safe Team Foundation Server	CVS Subversion Git
障害管理	Team Foundation Server	Mantis Trac Bugzilla
継続的インテグレーション	Team Foundation Server	Cruise Control Continuum Hudson
コミュニケーション支援	sharepoint server	phpbb IRC Jabber

Microsoft 社の開発支援ツールでは，例えば統合開発環境である Visual Studio からソースコード管理ツールである Visual Source Safe にアクセスできる等，複数のツール間の連携操作やデータ連携が考慮されている．他方，OSS の場合には，システム開発に必要となるツールを網羅的に開発しているような開発コミュニティは存在せず，かつ，適用領域ごとに複数のソフトウェアが存在するため，ツールの選定やツール間の連携性を利用者が考慮しなければならない．

2.2 ツール運用方式の策定に関する課題

開発支援ツールの運用に関しても，OSS の場合は，利用者の運用自由度がある半面，標準的な参照モデルが少なく，開発対象システムやプロジェクトの特性に応じた最適な運用方式をプロジェクトごとに検討し策定する必要がある．ソースコードのバージョン管理を例に取る

と，受託案件型の場合，標準ラインは単一に近くなるが，パッケージ型の場合は複数のバージョンを管理する必要がある．また，開発要員の流動性が高い場合は，仕様変更に伴うソースコードの改訂履歴の管理の良否が保守性に大きく影響を与えるが，開発要員が長期間固定するような場合は，厳密な改訂履歴管理のオーバーヘッドが逆に生産性を低下させる可能性もある．

3.　OSS 開発支援ツールの選定と適用に関する考慮点

システム開発における成果物の管理や，品質・保守性の向上に向けた情報管理・共有のためには，図 1 に示したような開発支援ツールが必要となる．

OSS では，適用領域ごとに複数の開発支援ツールが存在するが，主に Java 言語を利用したシステム構築プロジェクトを前提とした開発支援ツールを示す．本章では，前半で同図で網掛けとなっている適用領域/開発支援ツールについて，その特徴とツール選定に際しての視点を紹介し，後半でこれを開発現場で統合的に管理するための仕組みについて述べる．

図 1　開発支援ツール

3. 1　ツール選定の視点

管理系開発支援ツールの選定に際して最も重要な考慮点は，ツール間の連携である．個々のツールの機能にのみに着目してツール選定をおこなうと，統合的な開発環境としては十分な連携性が得られず，運用負荷を増加させることにもなりかねない．そのため，OSS 開発支援ツール選定時には，機能連携性やデータ連携性，操作感の統一性などの考慮が重要となる．

1）　ソースコード管理（採用ツール：Subversion）

一度作成されたソースコードは，仕様変更や不具合対応，リファクタリングなどの様々な理由で繰り返し変更される．ソースコード管理ツールは，ソースコードとその変更履歴を時系列で整理し管理するソフトウェアである．ソースコード管理ツールを選定する際の視点としては，システム開発保守の中核として長期間利用できること，拠点分散型の開発にも耐えうること，開発要員の変動を想定し利用障壁が低いツールであること，デファクトスタンダードに近いツールであること，等が挙げられる．

Subversion は，CVS（Current Version System）[*1] の後継として開発されたソースコード管理ツールであり，CVS に比べ様々な機能の追加と改良が施されている．ASF（Apache Software Foundation）[*2] を始めとする多くのオープンソースプロジェクトで採用され，OSS 開発支援ツールとしてデファクトスタンダードになりつつある．

2）　障害管理（採用ツール：Trac）

開発時のテスト工程や本番運用開始後に発生する障害を的確に管理し，これを品質保証に

繋げるためには，障害管理ツールの活用が不可欠である．障害管理ツールにて管理すべき主要なデータ項目としては，障害の発見日時や発見者，再現方法，修正担当者，修正履歴，修正方法，重要度，テスト状況などが挙げられる．開発要員が一定以上の数になると，障害に対する対応状況を開発要員全員で共有することが難しくなるが，障害管理ツールを利用することでこの問題を解決できるケースも多い．障害管理ツールの選定に際して重視すべき視点は，ソースコード管理ツールとの連携性である．

Trac は，プロジェクトのポータルページを作成することを目的とした Wiki 機能を備えた障害管理システムであり，ソースコード管理ツール Subversion との連携動作が可能である．そのため，ソースコードをウェブブラウザから閲覧する，不具合とソースコードの変更履歴を関連付けて管理する，といったことが実現できる．また，Trac 内部では，個々の障害を「チケット」として管理しているが，開発の最小単位を「チケット」として登録することで，システム改修や機能拡張等における要件管理のツールとしても利用可能である．

3） 継続的インテグレーション（採用ツール：Continuum）

継続的インテグレーション（Continuous Integration, CI）[2]ツールとは，ビルド，リグレッションテスト，デプロイといった作業を定期的かつ自動的に実行し，その結果を開発者にフィードバックすることで，開発中に生じる予期せぬ問題の早期検出を支援するためのツールである．具体的には，CI ツールはリポジトリを定期的にポーリングしてソースコードの変更有無を確認し，変更があった場合には開発者が事前に定義したビルドスクリプトを自動実行し，メールなどのコミュニケーションツールを通じて開発者にその結果を通知する．ビルドからデプロイまでのサイクルを継続的に繰り返すことで問題をいち早く検出し，品質の向上に寄与する．

Java ベースでのシステム構築プロジェクトにおける CI ツールの選定基準としては，Java 言語との親和性が重要である．Continuum は，Apache Maven プロジェクトのサブプロジェクトで開発されており，Java 用のビルドツールである Ant[*3] や Maven[*4] との親和性も高い．また，利用者向けドキュメントが充実している，設定が容易である，等の利点もある．

4） コミュニケーション支援（採用ツール：電子掲示板 phpbb, チャット IRC）

米国のプロジェクトマネジメント協会（Project Management Institute, PMI）の Vijay K. Verma が著作の中で“Communication provides the wings for flight to success.”と述べているように，開発要員間の良好なコミュニケーションはプロジェクト成功を左右する[3]．最もよく使われているコミュニケーションツールである電子メールは，過去のやり取りを追跡しにくい，送信相手がメールを読んでいるかどうかの判断がしにくい，といった課題がある．予見と予断で相方とコミュニケーションができていると勘違いをすることはプロジェクトの生産性を落とす原因になりかねない．コミュニケーション支援ツールの選定では，履歴の追跡を容易におこなえることが重要となるため，コミュニケーションの証跡がテーマごとに蓄積できる電子掲示板の選定が妥当だと考える．今回は OSS の掲示板ツールとして，世界中で広く利用され，その機能の豊富さ，使い勝手のよさで 2007 年度の SourceForge Community Choice Awards を受賞した phpbb[*5] を採用した．

また，相手が内容を読んでいるかどうかの判断の容易性という点では，リアルタイムコミ

ュニケーションを支援するチャット（IRC[*6]）との併用が望まれる．

3.2　開発支援ツールの統合

前節で選定した各開発支援ツールはそれぞれが独立したツールであるため，その導入・運用に際しては．ツールごとに利用者管理やバックアップ運用を考慮する必要がある．こうした作業負荷を軽減するためには，これらのツールを統合して利用者を管理する仕組みや，データを一括してバックアップする仕組みを構築する必要がある．

1）　利用者管理ツール

長期間に渡るシステム構築や保守作業フェーズでは，開発要員が流動するケースは少なくない．そのため，開発支援ツールが複数ある場合には，利用者アカウントの登録・削除作業を頻繁におこなう必要があるが，この煩雑な作業を回避するためには，統合的な利用者管理ツールの導入が望まれる．4 章で述べる実プロジェクト適用においては，独自に利用者管理ツールを開発し，各開発支援ツールの利用者の登録・削除・パスワード変更の操作をウェブブラウザから一括して行える機能を提供した．

2）　バックアップツール

ソースコードや障害記録など開発支援ツールを運用することによって蓄積されるデータは，システム開発の品質や生産性向上のための要であり，その消失を防止するためにはバックアップが必須となるが，複数の開発支援ツールを利用している場合，ツールごとにバックアップの仕組みを準備する負荷が生じる．4 章で述べる実プロジェクト適用に際しては，複数の開発支援ツールで管理されるデータを一括してバックアップするツールを開発し，開発環境運用管理者の負荷を軽減している．

3.3　開発支援ツールの運用方式

本章で選定した管理系の開発支援ツールを活用し，生産性・品質面で効果を上げるためには，各開発プロジェクトで最適な運用方式を策定する必要がある．また，個々の開発プロジェクトでの運用方式策定の負荷を軽減するためには，標準的な運用ガイドラインがあることが望ましい．しかし，プロジェクトの持つ様々な特性や制約事項によって効果的な運用方式は異なり，各プロジェクトへの適用に当たっては次に示すような特性を考慮する必要がある．

- 開発要員の規模

数百人から時にはピーク時で千人を超える規模の大規模プロジェクトから，2 ～ 3 人の固定メンバーによる小規模プロジェクトまで，プロジェクトの開発要員規模やその流動性は様々であるが，プロジェクト内での確実な情報伝達に向けては，要員規模に応じた適切なコミュニケーション方式や管理ツールを導入する必要がある．

- プロジェクト形態（パッケージ開発・受託開発など）

長期間に渡り継続的にバージョンアップを繰り返すパッケージ開発のようなプロジェクトと，特定顧客向けのシステム受託開発プロジェクトとでは，ソースコードの取扱いが異なるこ

とが予想される．例えば，受託開発型のプロジェクトでは，多くの場合一つのソースコード管理ラインで間に合うのに対し，パッケージ開発型プロジェクトでは，バージョンごとに保守用ソースコード管理ラインを作成したほうが保守作業を効率的におこなうことができる．

● 開発拠点の分散性（集中/分散）

開発拠点が一箇所に集中している場合には，開発支援ツールも統一した環境で運用することが可能である．しかし，拠点分散型開発の場合，セキュリティやネットワーク環境などにより開発環境を集中的に管理することが難しく，開発拠点ごとに開発支援ツールを導入せざるを得ないケースも想定され，各拠点間での管理データの統合等に考慮が必要になる．

● 開発要員の流動性

仕様変更やこれに伴うソースコードの修正・改変に関する文書化の重要度は，開発要員が長期にわたり固定している場合と要員の流動性が高い場合とで異なる．開発要員の流動性が高い場合には，そうでない場合と比べ文書化や知財の可視化に関する運用を徹底する必要がある．

4. 適用プロジェクトによる評価

本章では，3章で述べた開発支援ツール群の適用事例とその評価について述べる．

4.1 適用対象プロジェクトの特性と課題

3章で選定した開発支援ツール群の適用対象としたのは，日本ユニシスグループのパッケージ・ソリューションであるRENANDI® 統合eラーニングシステム[*7]（以下，RENANDI）の機能拡張・保守プロジェクトである．同プロジェクトの特徴は，長期間に渡り小規模な機能拡張を行い，継続的にバージョンアップしている，商品化以来5年が経過し製品主管および外注先の開発要員の流動性が高い，開発拠点が2拠点に分散している，といった点である．

開発支援ツール適用前の製品開発・保守に関する課題は次のとおりであった．

1） ソースコード管理にかかる負荷

従来は，導入先の顧客ごとにリポジトリを作成しソースコードを管理していたが，顧客数の増加に伴い，標準ラインと顧客単位のラインの整合性を保持するためのソースコード・ライン管理に多大な負荷がかかっていた．

2） 過去の仕様の追跡の負荷

RENANDIは，商品化以来継続的な機能拡張・バージョンアップがおこなわれているが，その間に開発・保守要員も流動しており，過去の仕様決定の経緯やソースコードの変更動機を追跡することが困難となりつつあった．

3） 統合テストでの不具合発見の負荷

RENANDIの開発では，開発者は担当した開発項目に応じて各自のPC上でリポジトリからチェックアウトしたソースコードを編集し，テストをおこなった上で，プロジェクトで共通のリポジトリに修正済みのソースコードを登録する．そのため個人の開発範囲での不具合

は早期に発見することが可能であるが，複数の開発要員による作業結果の不整合に起因した不具合に関しては，統合テストまで発見が遅延し，これが負荷となっていた．

4. 2　適用プロジェクトにおける開発支援ツール運用

前節の課題を解決するために策定した，開発支援ツールの運用方式の概要は，次のとおりである．

4. 2. 1　ソースコード管理ライン

Subversion では，一般に trunk・branches・tags というネーミングで管理ラインを分け運用することが推奨されている．

- trunk … メイン開発用の管理ライン．通常はこのラインを使って開発を行う．
- branches … trunk とは別に変更履歴を管理したい場合に利用する管理ライン．この管理ラインに対しておこなわれた変更は通常は trunk にマージ（併合）する．
- tags … リリース済みのコードを管理する開発ライン．原則としてこの管理ラインには変更を加えない．

パッケージソフトウェアの開発では，長期間に渡りバージョンアップを繰り返すことから，新規バージョンに向けた機能拡張作業と過去にリリースしたバージョンの保守作業との整合性を保持するため，バージョンごとに管理ラインを分岐させる運用方式を選択した．

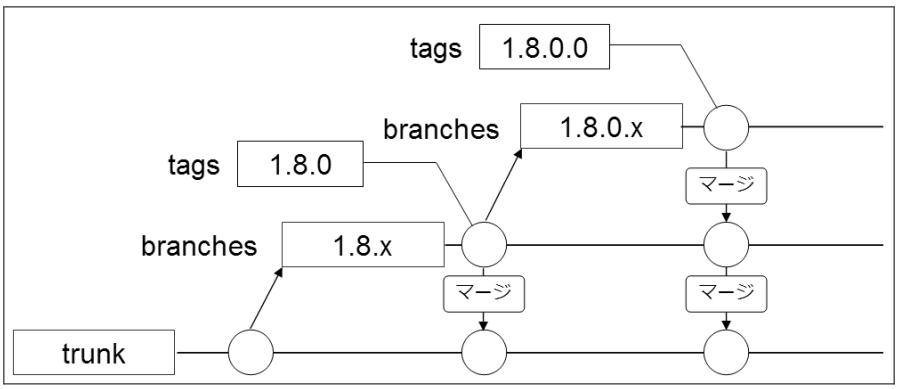

図 2　RENANDI のソースコード管理ライン

具体的には，機能拡張/バージョンアップ版の開発は trunk でおこない，リリース前のテストやリリース後の保守は branches（1.8.x など）とし，客先リリースしたソースコードは tags で管理するという運用方式を策定した（図 2）．このように目的に応じて管理ラインを分けることで，機能拡張開発をおこないながら，保守作業を同時並行でおこなうことが可能となる．

4. 2. 2　ソースコードとタスクの関連付け

機能拡張や仕様変更，不具合の修正，リファクタリングなどの様々な理由でソースコードの変更が繰り返しおこなわれるが，過去のソースコード修正の経緯を踏まえずに目先の修正に捉われ，これがデグレードに繋がる場合がある．特に開発要員の流動性が高い場合は，変更動機

が不明になる危険性はさらに高い．そのためソースコードとタスク（あるいは不具合）を関連付けて管理し，変更動機を即座に追跡できる仕組みを作っておくことが重要である．

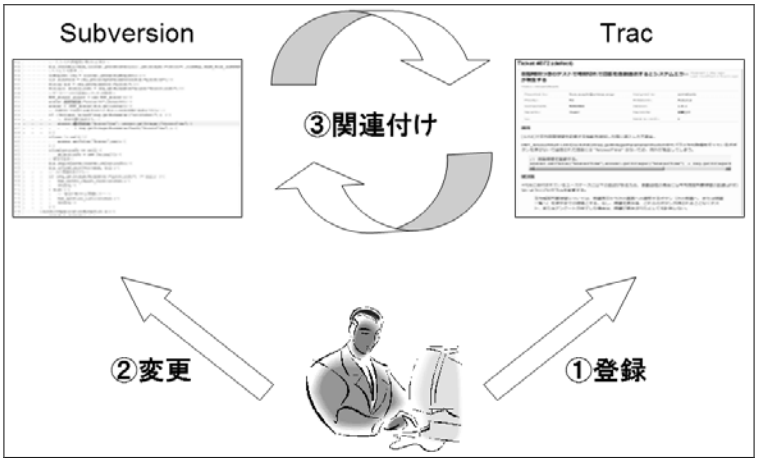

図3　ソースコードとタスクの関連付け

RENANDIでは，開発者がソースコードに変更を加える際には，まず最初に不具合（タスク）管理システム（Trac）に作業内容をタスクとして登録し，次にソースコードを変更しその変更内容をソースコード管理システム（Subversion）に反映させる（図3）．このとき，作業内容がタスクとして登録されていない場合にはソースコード管理システムに変更を反映しないように規約を定め，変更動機が明確になっていないソースコードのリポジトリへの反映を排除する．最後に，不具合（タスク）管理システムの機能を用いて，ソースコードの変更履歴とタスクを相互参照できるように関連付ける．具体的にはTracのリンク作成機能を利用して関連付けを実現している．リンク作成機能とは，定められた記法を用いることで，特定のチケットやチェンジセットへのリンクを作成する機能である．例えば，図4では，コミットログ[*8]に#331というチケット番号を記述することでチェンジセットからチケットへのリンクを作成している．

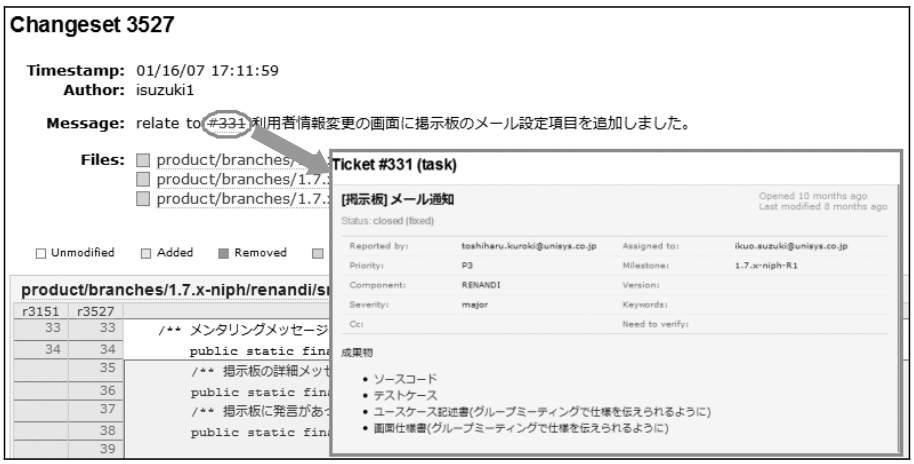

図4　チェンジセットからチケットへのリンク

このようにして，ソースコードとタスクを関連付けて管理することで変更内容の追跡が容易になり，開発保守の生産性を高めることが可能になる．

4. 2. 3　ビルド結果の通知

プロジェクトによっては，不具合のあるソースコードのリポジトリへの混入を防ぐために，開発者のローカル環境でテストをおこない，他の開発者のレビューを受けたソースコードのみチェックインを許可する，といった対策を講じるケースがある．しかし，不具合が，一人の開発者の担当範囲内ではなく，複数の開発保守作業の相互作用から引き起こされる場合には，統合テストまでこれが顕著化しないリスクがある．さらにこうした相互作用による不具合は，時間の経過とともに発見が困難なものとなる．これを回避するためには，リポジトリに混入した不具合を極力早期に検出するための仕組みが有効である．

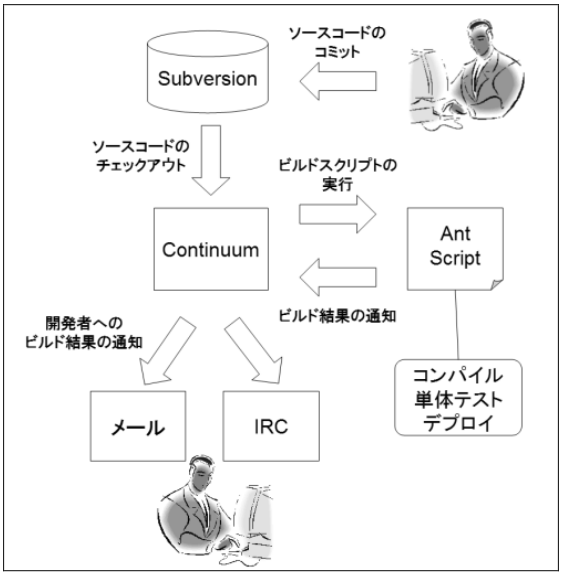

図 5　ビルド結果が開発者に通知されるまで

RENANDI では，Continuum を使って統合環境でビルドを実施することでリポジトリに混入した不具合の検出をおこなっている（図 5）．開発者がソースコードをコミットすると，Continuum がリポジトリからソースコードを自動的にチェックアウトし，ビルドスクリプトを実行する．ビルドスクリプトでは，コンパイル・単体テスト・デプロイをおこなうように設定されており，実行結果を Continuum に通知し，Continuum がメールやチャット（IRC）などのコミュニケーションツールを介して自動的に開発者に通知する仕組みを構築した．

このように，ソースコードに変更が加えられるたびに自動ビルドがおこなわれ，その結果がプロジェクトの全開発要員へ通知されることで，問題を早期に検出することができる．

4. 3　適用評価

本節では，オープンソースの開発支援ツールを RENANDI プロジェクトで導入・運用した結果を評価する．

4.3.1　ソースコード管理にかかる負荷の軽減

ソースコードの管理ラインを分岐させ，trunk = 機能拡張開発用，branches = 保守用・特定顧客用，tags = リリース用，といった形で各管理ラインの役割を明確にすることでバージョンアップ作業と保守作業の並行実施が可能となった．RENANDI プロジェクトの場合，顧客ごとに導入バージョンが異なる場合もあり，複数の異なる保守用の管理ラインに対して作業をすることも多いため，これらの工夫は生産性の向上に大きく貢献している．

4.3.2　過去の仕様の追跡にかかる負荷の軽減

「作業内容をタスクとして登録していない場合には関連するソースコードの変更をソースコードリポジトリに反映させてはならない」という規約を定めることにより，変更動機が不明なソースコードの変更が激減した．そのため，新機能の開発や不具合対応におけるソースコードの修正時に，過去の修正内容に対する作業内容の影響範囲の測定が従来と比べ容易となった．障害（タスク）管理システムに作業内容を登録する作業コストに比べても，得られる便益が明らかに大きく，総コストの削減に繋がっている．

4.3.3　不具合の早期発見

継続的インテグレーションの導入により，ソースコードが変更される都度，本番環境に近い環境でコンパイルと単体テストが自動実行されることで，作業の省力化が可能となり，開発者の個別の環境では検出できなかった不具合を検出できる確率が上がった．また，ステージング環境へのデプロイも都度実施することで，開発途上のバージョンの最新状態を常に開発メンバーがチェックできるようになり，人間系によるテストも早い段階でおこなえるようになったことも品質確保に大きく貢献している．

5.　お　わ　り　に

Java 言語を利用したシステム構築プロジェクトへの適用を前提に，OSS の開発支援ツール選定のための視点や開発運用に関する課題を整理し，実際に選定したツールの適用事例とその評価を述べた．OSS の開発支援ツール群は，各開発コミュニティによる機能拡張や品質向上に向けた努力の結果，実用に耐え得るレベルとなってきており，実プロジェクトへの適用を通じ，費用面でのメリットだけではなく，システム開発保守の品質および生産性の向上に寄与することが確認できた．本稿で紹介した適用事例と類似した特性をもつシステム開発プロジェクトにおいては，同様の効果が期待できるものと考える．

しかし，システム開発プロジェクトの特性によって効果的な運用方式は異なることから，様々なプロジェクトへの適用を通じ，プロジェクト特性に応じた運用方式の策定や有効性の検証をおこなう必要がある．また，統合的な開発環境構築に向けては，本稿でとりあげた開発支援ツール以外に，テスト関連の管理ツールやリリースに関する管理ツールなどの整備が課題である．

日本ユニシスでは，OSS 開発支援ツールの動向調査および評価を継続的に実施しつつ，実プロジェクトへの適用を通じて得られる知財を蓄積・整理し，更なる品質や生産性の向上に貢献できるツール選定と運用方式の策定を目指す予定である．本稿が，様々なシステム構築プロジェクトに対する OSS 開発支援ツール適用の一助になれば幸いである．

＊ 1　オープンソースのソースコード管理ツールの一つ.
＊ 2　米国で登録されたオープンソースソフトウェアプロジェクトを支援する団体.
＊ 3　Apache Software Foundation で開発されている Java ベースのビルドツール.
＊ 4　Apache Software Foundation で開発されている Java ベースのビルドツール．Ant に比べて，使える機能が豊富なことが特徴である.
＊ 5　PHP で作られたフォーラム型掲示板ツール．トピックの作成や権限の管理が柔軟におこなえる．2007 年度は SourceForge が主催する優れたオープンソースを選ぶコンテスト Community Choice Award のコミュニケーション向けプロジェクト部門最優秀賞を受賞した.
＊ 6　チャットシステム．専用クライアントをインストールしてサーバを介してチャットをする．ブラウザを使った Web でのチャットに比べて高速なのが特長．利用者同士で自分の状態を通知できるため，在席状況の確認にも利用できる.
＊ 7　RENANDI は，講義型の集合学習形態や自己学習型，協調学習型など高等教育機関における様々な学習スタイルに対応したコース管理機能を持ち，シラバス管理，出欠管理，教材管理，レポート管理，Web テスト管理，グループ管理，学習進捗管理，成績管理といった学習支援機能や，オンライン質問箱，FAQ，Web アンケート，掲示板，お知らせなどネットワーク・コミュニケーション機能を有し，時間や場所に制約されない学習環境の構築が可能なインターネットを活用した学習管理システムである.[4]
＊ 8　ソースコードをレポジトリリポジトリに反映する際には，変更内容の要約を同時に記録することができる.

参考文献　［1］ 吉野 良成，システム構築プロジェクトの現状と課題，技報 93 号，日本ユニシス，p.7
［2］ Martin Fowler, “Continuous Integration”, http://www.martinfowler.com/articles/continuousIntegration.html
［3］ Projects For Success, Vijay K. Verma, PMI, p.34, 1995
［4］ 倉田 菜生子，高等教育における ICT を活用した産学連携による取り組み，技報 91 号，日本ユニシス，p.108

執筆者紹介　鈴　木　生　雄（Ikuo Suzuki）
2005 年日本ユニシス(株)入社．教育研究機関ビジネス室にて自社製品の RENANDI，東京大学との共同開発製品 CFIVE という二つの e ラーニングシステムの開発に従事し，OSS を使った Java 開発経験を積む．現在は，その経験を活かし，OSS センター ソリューション開発室に所属し，開発環境関連の OSS の調査・適用業務を担当している.