B-007

複数の開発拠点に跨る Build Verification Test における Automation Framework

The Automation Framework of Build Verification Test for a project which straddles world-wide development teams

菊地弘晶† 細野浩一†
Hiroaki Kikuchi Koichi Hosono

1. はじめに

複数の開発拠点に跨るソフトウェア開発プロジェクトでは、各拠点で開発されるコンポーネントを集約した build が利用可能となった時点で、その妥当性を検証できることが、その build が Functional Verification Test(FVT),System Verification Test(SVT)等を実施する対象として足るかどうかを判断するために望ましい。筆者らが属するミドルウェア開発プロジェクトにおいては、システム内の各コンポーネントを検証するテストケースを 1 拠点に結集しバッチ的に実行することによって妥当性を検証している。このような方針でテストが行われ、その過程においてテストケースが失敗した場合、その問題の原因を速やかに明らかにし、適切な拠点へ問題修正を依頼しなければならない。本論文では、複数拠点に跨るソフトウェア開発プロジェクトにおいて、効率的な問題判別を実現するテストフレームワークの提供について述べる。

2. 複数拠点に跨る Build Verification Test (BVT)

筆者らは、6 つの開発拠点による協業から成るミドルウェア開発プロジェクトに従事している。本章では、このプロジェクトにおいて実施された BVT の概要、およびその問題点について述べる。

2.1 Build Verification Test (BVT)

本プロジェクトにおけるテストでは、複数拠点が並行的にテストを行うために、ｂｕｉｌｄが利用可能になった段階で、その build が FVT,SVT 等の実行対象とするに足るものかどうかが判断でき、各拠点にその結果が迅速に通知されることが効率的なテストを行なうために必要であった。Build Verification Test(BVT)は、build の妥当性を検証するテストケースおよび FVT の Regression Test をバッチ的に検証するために導入されるテストフェーズである。各拠点で作成されたテストケースが BVT 実行拠点に結集され、build が利用可能になるとともに batch 的に実行される。複数拠点により並行的に加えられた変更・修正を含む build が正常に動作するかを随時検証する観点から有用なフェーズである。明示的に Regression Test フェーズを設定しなくても、常に各機能が正常であることを確認できる（図１）。本プロジェクトにおいては、Build Acceptance Test (BAT)と呼ばれる、build の簡易な End-to-End テストを行った後、より詳細な妥当性検証を BVT にて行う。BAT が build チームによって用意されたシステムの機能を簡易的に検証する End-to-End テストケースのみを含むのに対し、BVT は複数拠点によって用意された各コンポーネントに対する詳細なテストケースをも含む。

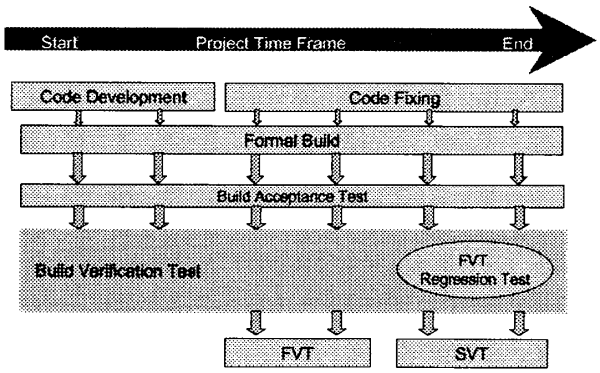

図 1. Build Verification Test

2.2 複数拠点に跨る Build Verification Test での問題点

筆者らの属するプロジェクトでは、世界 6 拠点に跨って開発・テストが行われている。

Unit Test(UT),FVT フェーズにおいては、各拠点で開発されたコンポーネントは開発元拠点によって検証されるため、各拠点が保有するコンポーネント固有のスキルに基づいてフェーズを完了することができた。しかし、BVT においては、テストを実行する拠点に全コンポーネントのテストケースが結集されるため、効率的な BVT プロセスを実現するため、各テストケースは以下の条件を満たす必要があった。

A) 実行の容易性
BVT はコンポーネント固有のスキルを持たないテスターによって設定、実行できる必要がある。

B) テストケースの可搬性
テストマシンの選定、およびテストケースの配置先ディレクトリはテストを実行する拠点に一任される。また、テストケースは製品がサポートする全てのプラットフォームで動作する必要がある。

†日本アイ・ビー・エム株式会社, ソフトウェア開発研究所

C)　問題判別の容易性

BVT の実行対象は、複数拠点において開発されたコンポーネントの集合体である build であり、多くのテストケースは、複数のコンポーネントに依存した機能を検証することになる。テストケースが失敗した際には、原因となったコンポーネントを特定し、問題報告、修正依頼を行う必要がある。

上記の条件のうち、C)については全てのテストケースに共通の framework を確立することが必要であった。

3. 複数拠点に跨る Build Verification Test における Automation Framework

筆者らの属する拠点においては、図 1 で示した BVT フェーズにおいて、効果的な Regression を毎 build に対して行うことで、FVT Regression に要する作業量を大きく軽減させることを実現した。すでにＵＴとして、コンポーネント単体を対象に実行されたテストケースのうち、システムとしての機能を検証するテストケース、すなわち FVT としても実行されるべきテストケースを選定、自動化して BVT へ組み込んだ。

複数の拠点が並行的に開発しているため、テストケースが失敗した場合、その原因がどの拠点が開発したコンポーネントに依存しているのかを判別することにコストがかかる。筆者らは、同一のテストケースを以下に挙げる複数のモードで実行し、原因特定を容易にする仕組みを実現した(図 2)。BVT が行なわれるとき、コマンドライン・モードとシステム・モードで動作するテストケースを実行する。失敗が報告された場合には、該当するテストケースをスタンドアロン・モードで実行する。

- システム・モード

  Build をインストールすることによって構成されるシステム内のコンポーネントに対して機能呼び出しを直接行うモード。インストール済みシステムの制御層が用意する API に対して機能呼出を実行し、システム上でのコンポーネントの動作確認を行う。ユーザがシステムを操作するためのインターフェースとして提供される管理コンソールを含む、外部アプリケーションから実行される動作を検証することと等価である。FVT の Regression Test に相当する検証を行うことができる。

- スタンドアロン・モード

  コンポーネントを独立的に起動させ、機能呼び出しを行う。テスト対象のコンポーネントのみを独立的に起動し、実行する。他コンポーネントに依存せず、UT で行なわれた検証内容と基本的に等価となる。このモードでの実行は、コンポーネント単体の動作の妥当性を確認できる。問題を持つコンポーネントの特定が容易となる。

- コマンドライン・モード

  デバッグや機能検証のためにプロジェクト内の開発者向けに提供されるコマンドラインインターフェースを経由し、テストケースを実行する。ユーザからの操作を簡易的に検証することが出来る。ユーザへ提供される管理コンソールを経由した実行に比べ、発行された操作は単純な経路でテスト対象の開発コンポーネントに到達する。コンポーネントのシステム上での基本的な一連の動作の妥当性を検証するテストケースを実行する。

4. 今後の課題

筆者らの属する拠点で開発されたコンポーネントは、コンポーネントとしての独立性が非常に高く、他のコンポーネントに依存しないため、今回の framework は容易に実現することができた。今後、他のコンポーネントとの依存度の高いコンポーネントに対してこの framework を適用するためには、スタンドアロン・モードにおいて対象のコンポーネントを仮想的に独立させるための driver/stub の整備等が必要となる。また、現時点では、システム・モードの実行後、その実行結果を目視した後にスタンドアロン・モードでの実行を人手によって行う必要があるが、今後はこの処理を自動化することにより、半自動的な問題判別・報告が可能となると考えられる。

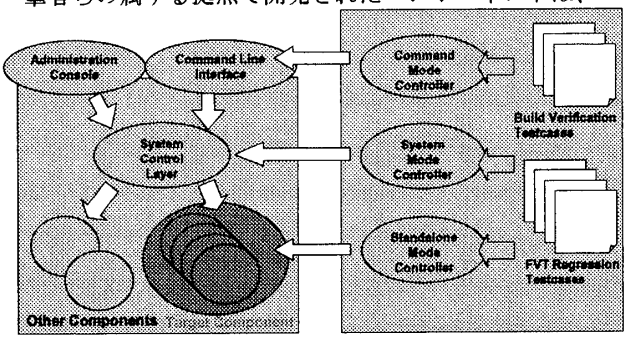

図 2.　問題判別を容易にする Automation Framework

5. まとめ

本論文では、複数拠点に跨る開発プロジェクトにおける Build Verification Test での Automation Framework について述べた。大規模で複数拠点に跨るソフトウェア開発プロジェクトにおいては、テストプロセスの効率化は非常に重要な課題である。本 Framework は、筆者らの属するミドルウェア開発プロジェクトにおける、ＢＶＴによる build 検証に対して適用されたものである。しかし、この考え方は他のプロジェクト、また FVT,SVT 等のフェーズがテストケース開発元以外の拠点において実行される場合に一般的に有用であると考えられる。テストケースが作成元以外の拠点で実行されるとき、効率的な実行が可能であるだけでなく、効率的な問題判別に優れた Framework を確立することは重要であり、筆者らは実際のプロジェクトにおいてこの１例を示すことができた。

謝辞

Framework の開発に携わった、日本アイ・ビー・エム、ソフトウェア開発研究所の三瓶光司さん、また濱田誠司さんをはじめとする、ミドルウェア開発プロジェクトのメンバーの皆様に心より感謝いたします。