C-020

組込み機器向けの継続的ソフトウェア品質向上手法
A Continuous Quality Improvement Approch of Embedded System

天野 隆†
Takashi Amano

1. はじめに

迅速に低価格で組込み製品を提供できるようにするためには，開発時において効率良く品質維持・向上することが重要である。近年，組込み製品には汎用 OS である Linux が採用されている。Linux のソースコードは大規模であり，バグ発見時の調査範囲が広範囲に及ぶ。そのため，開発における品質維持・向上が求められる。本研究では，この中でもソフトウェアである組込み向け Linux の開発効率および品質維持の妨げになるビルドの属人化と低頻度なリアルタイム性能確認の問題を解決する自動ビルドと自動テストを提案する。試作により，非属人化ビルドと高頻度でリアルタイム性能を確認した。

2. 組込開発の課題

2.1 開発チームの前提

組込み向け製品は，コーディング，ビルド，テストなどを担当する複数の開発者がチームで開発を行う。組込み向け開発環境における開発工程を示す。例えば，あるチームの開発手順を下記(図 1)と想定する。

(1) ビルド担当者は，開発担当者からビルド対象ソースコードの場所情報を得る
(2) ビルド担当者は，ソースコードをビルド PC に集約する
(3) ビルド担当者は，ビルド PC でビルドする
(4) ビルド担当者は，ビルド結果の生成物を共有 NAS に保存する
(5) 開発担当者は，実機の OS を更新する
(6) 開発担当者は，実機をブートする
(7) 開発担当者は，リアルタイム性能テストをする
(8) 開発担当者は，テスト結果を確認する

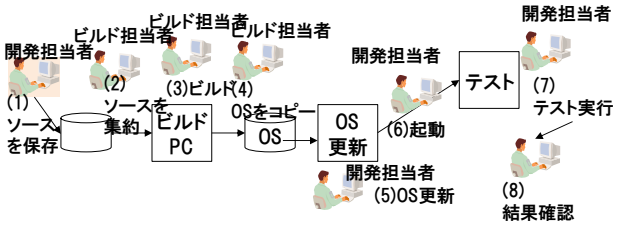

図 1 現状の開発環境例

2.2 組込み開発における問題

2.2.1 ビルドの属人化

ビルド担当者は，ソースコードの集約，コンパイルや OS イメージ作成などを実施している。ビルド作業は属人的であり，且つ多数の工程を実施するため，ミスが発生しやすい(問題 1)。例えば，毎週指定日に開発担当者にビルド対象のソースコードを保存してもらい，その翌日の午前中に各ソースコードを集約しビルドを実施している。開発担当者はビルド手順を把握していないため，ビルド担当者が不在の場合にはビルドやビルド後のリアルタイム性能テストも実施できない。

2.2.2 低頻度なリアルタイム性能確認

開発担当者は，コーディングや機能テストを集中して作業しており，非機能テストであるリアルタイム性能テストの実施間隔が広くなってしまうことがある(問題 2)。このため，リアルタイム性能テストで劣化が発見されたとき，前回の同テスト後から今までのどのコーディングが影響しているのか広い範囲を調査する必要があり非常に時間がかかる。

2.3 本研究の狙い

これら問題に対して，DevOps や継続的インテグレーションの分野では，Jenkins[1]等の OSS(Open Source Software)をベースとした自動化を取り入れた開発環境が利用されるようになってきている。本研究では，組込み向け開発のビルドとリアルタイム性能テストを自動実行する継続開発環境を構築し，ビルド担当者の工数を削減するとともに誰でもビルドできるようにすること，ビルド 1 回ごとにリアルタイム性能を確認できるようにする。組込み向け製品の開発期間を短縮し，ビルド工数(ビルドを実行した人が拘束される時間)を従来方式と比較して半減させることおよび品質維持・向上を目標とする。これにより，人件費を削減し適切な価格のソリューションを顧客に提供する。

3. 組込み向け継続開発環境の提案

問題解決として，次節以降で述べる自動ビルド方式と自動テスト方式を試作して評価した。問題 1 の解決には，自動ビルドにより，誰でもビルドを実行できるようにする。問題 2 の解決には，自動テストにより，ビルド後にリアルタイム性能テストを実施できるようにする。

本研究では，組込み向け継続開発環境のうち，ビルドとテストを対象範囲とした(図 2 の点線)。

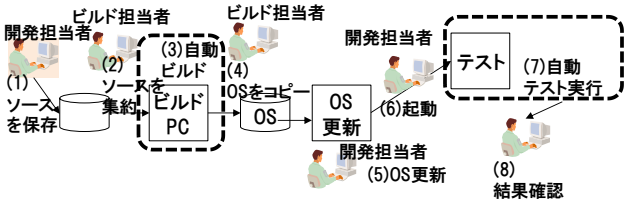

図 2 提案する開発環境

†株式会社 日立製作所 デジタルイノベーションセンタ
Hitachi, Ltd. Center for Technology Innovation-Digital Technology

3.1 自動ビルド方式

自動ビルド方式は，継続開発ツールである Jenkins を利用する。Jenkins は，Web ベースの GUI(Graphical User Interface)を用いて，外部の git リポジトリに格納されたソースコードをビルドする機能が標準でサポートされている。本研究では，非属人化の確認のため，専用ハードウェアの OS 更新方法に合わせた組込み向け OS インストールイメージまで作成する GUI と連携したスクリプトを開発してビルドすることにした。

Jenkins で git 連携機能を用いたビルドを行うには，リポジトリを含むすべてのファイル実行権限を Jenkins ユーザに変更する必要がある。加えて，git レポジトリと通信できない隔離された開発環境では場合は，別の方法でソースコードをダウンロードしておく必要がある。

Jenkins は，ビルド処理として設定されたスクリプトを実行することができる。ビルド作業を処理する開発したスクリプトをビルド処理として実行できるように Jenkins に設定する。開発したスクリプトは下記を処理する。

(1)カーネルのコンパイル
(2)各種コマンドのコンパイル
(3)ブートイメージの作成と保存

3.2 自動テスト方式

自動テスト方式は，自動ビルド方式と同様に Jenkins を利用する。Jenkins は，ビルド後処理として設定されたスクリプトを実行することができる。テスト作業を処理する開発したスクリプトをビルド後処理として実行できるように Jenkins に設定する。開発したスクリプトは下記を処理する。

(1)テストプログラムを実機へ送付
(2)テストプログラムを実行
(3)テスト結果を取得し，グラフ描画用にデータを加工

リアルタイム性能の劣化を迅速に確認できるようするために，Jenkins のプラグインを使用し，ビルド後のリアルタイム性能テストの結果を過去の結果と共にグラフ表示する。操作画面の「ビルド実行」をクリックすることで，テストを実行できる。グラフを用いて過去のリアルタイム性能テストの結果と見比べることによって，リアルタイム性能劣化の原因特定を早める。

4. 提案方式の評価結果

4.1 組込み向け継続開発の開発工数

組込み向け OS の開発工数(作業を実行した人が拘束される時間)を従来方式と，提案方式である自動ビルド方式と自動テスト方式で試算して比較した。従来方式でビルドした場合は，従来の工程にかかる時間は 230 分である。一方，提案方式で実施した場合は，同じ工程を 115 分で実行できる。よって，提案方式は，従来方式にかかる時間を 50%削減できる(表 1)。

表 1 開発時間 [単位：分]

主な開発ステップ	従来手法	本手法
ビルド	90	10
テスト実行	10	0
テスト結果取得	10	0
その他	120	105
全ステップ合計	230	115

4.2 リアルタイム性能の確認頻度

OS のバージョンを変更して，評価用ハード(表 2)[2]でそれぞれ継続的なテスト(Cyclictest 使用)を実施した。スリープから起きる予定時刻と実際に起きた時刻との差分(遅延時間)を測定したリアルタイム性能テストの結果を表 3 に示す。T_n は，n 回目のビルド後のテストである。自動テスト方式により，ビルドごとにリアルタイム性能を確認できる。よって，自動テスト方式は，従来よりも高頻度でリアルタイム性能を確認できる。

表 2 ハード情報

ハード	仕様
製品	BeagleBone Black
CPU	ARM v7l
メモリ	512MB

表 3 リアルタイム性能テストの結果

[単位：マイクロ秒]

カーネルバージョン	項目	T_1	T_2	T_3	T_4	T_5	T_6	T_7
4.19.14	最大	62	51	44	42	44	51	52
	平均	16	16	16	16	16	16	16
	最小	13	13	13	13	13	13	13
5.0.3	最大	51	48	50	53	38	52	51
	平均	15	15	15	15	15	15	15
	最小	12	12	12	12	12	12	12

4.3 提案方式の効果

自動ビルド方式により，ビルド担当者が集約したソースコードを用いてビルド担当者以外でも容易にビルドを自動的に実行できるようになり，従来方式よりビルド工数を 50%削減できた。

自動テスト方式により，ビルド後に連続して実行することで，ビルドごとにリアルタイム性能を確認できるようになった。これにより，品質劣化を引き起こすバグの迅速な原因特定およびバグ修正ができるようになった。

以上より，組込み機器の開発期間短縮および品質維持・向上を実現でき，人件費削減による適切な価格のソリューションを顧客に提供できる。

5. おわりに

組込み開発における効率および品質維持・向上の問題を解決する自動ビルドと自動テストを提案し，非属人化ビルドと高頻度でリアルタイム性能を確認した。

本研究の今後の課題を次に示す。

(1) 組込み向け継続開発環境の運用体制検討
(2) ソースコードの静的自動解析による品質確認
(3) ビルドエラーレポートの保存

謝辞

職場の関係者各位には，本研究に関する検討内容について議論頂くとともに，貴重なご意見を頂いた。

参考文献

[1] Jenkins，Jenkins ホームページ，https://jenkins.io/，2019 年 6 月現在.
[2] BeagleBone Black , BeagleBone Black ホームページ, https://github.com/beagleboard/beaglebone-black/wiki, 2019 年 6 月現在