serverspec: 宣言的記述でサーバの状態をテスト可能な汎用性の高いテストフレームワーク

宮下 剛輔[†1,†2,a)]　栗林 健太郎[†1,b)]　松本 亮介[†3,c)]

概要：システムの大規模・複雑化に伴い，サーバの構築・運用を効率化するために，サーバの状態をコードで記述する手法が数多く提案されている．それらの手法を効率良く扱うプロセスとして，テスト駆動開発の手法をサーバ構築に応用した Test-Driven Infrastructure が提案されている．このプロセスを支援するテストフレームワークもいくつか登場しているが，あるものは特定の構成管理ツールに依存，またあるものは OS 毎の違いを自ら吸収しなければならないなど，汎用性に難がある．そこで，本論文では，特定の構成管理ツールや OS に依存することなく，サーバの状態を汎用的かつ可読性の高いコードでテスト可能なテストフレームワークを提案する．提案手法では，汎用性を高めるために，これまでの OS や構成管理ツール固有の振る舞いを整理して一般化し，汎用コマンド実行フレームワークとして定義する．続いて，テストコード記述の抽象度を高め可読性を上げるために宣言的な記法で汎用コマンド実行フレームワークを操作できる制御テストフレームワークを定義する．これにより，管理者が OS や構成管理ツールの違いを気にすることなくサーバの状態を容易にテストできるようになり，サーバの運用・管理コストを低減できる．また，フレームワークを用途別に分離して定義することにより，制御テストフレームワークを独自の記述に変更する事も容易である．提案するテストフレームワークを serverspec と名付けた．

1. はじめに

サーバの構築・運用を効率化するために，サーバの構成管理方針を宣言的なコードで記述するという手法が提案され，その実装として構成管理ツール CFEngine[1] が 1993 年に登場している．また，システムの大規模化・複雑化の流れや，IaaS のようなサーバの構築・破棄が手軽にできるような環境の登場を背景に，CFEngine を更に発展させた実装として，2006 年に Puppet[2] が登場，その後も様々な実装が登場している [3][4][5].

2006 年の Puppet 登場以降，サーバの構成管理方針をコードで記述するという “Infrastructure as Code” という概念が台頭し，“Agile infrastructure and operations”[6] という流れが生まれている．これはアジャイル開発の手法をサーバの構築や運用に適用しよう，という潮流である．（ここで言う “Infrastructure” はアプリケーションを載せるための土台を意味し，OS やミドルウェアといったソフトウェアレイヤーを含む.）

この流れの中で，“Agile infrastructures and operations” を効率よく実践するためのプロセスとして，テスト駆動開発の手法をサーバインフラの構築・運用に応用した “Test-Driven Infrastructure”[7] というプロセスが提案されている．

このプロセスを支援するテストフレームワークがいくつも登場している [8][9][10][11][12][13].

これらのうち，ChefSpec，rspec-puppet は，構成管理ツール固有の言語で書かれたコードの内容をテストするのみで，実際にコードをサーバに適用した結果はテストしない．そのため，単体テストしては利用できるが結合テスト用途には利用できない.

Cucumber-chef，minitest-chef-handler は Chef という特定の構成管理ツールに完全に依存している．そのため，Chef 以外の構成管理ツールでは利用することができない.

Test Kitchen や rspec-system は，テスト用 VM の作成，テスト用 VM への構成管理ツールの適用，テストの実行をトータルで行う統合テストスイートであるが，組み込みのテスト機構は汎用性に乏しく，特定の構成管理ツールに依存していたり，OS 毎の違いを意識したテストコードを書く必要がある.

我々は，テストフレームワークの汎用性を高めるために，構成管理ツール特有の振る舞い（例えば，パッケージのイ

†1 現在，株式会社 paperboy&co.
†2 現在，帝京大学
†3 現在，京都大学
a) gosukenator@gmail.com
b) kentarok@gmail.com
c) matsumoto_r@net.ist.i.kyoto-u.ac.jp

ンストールやシステムユーザの作成など）を抽出・一般化し，それらをテストするためのコマンドを OS 毎に分離，その上で OS 毎のコマンドの違いを吸収するレイヤーを設けることにより，汎用コマンド実行フレームワークを定義した．

続いて，テストコードの記述の抽象度を高め可読性を上げるために，宣言的な記法で汎用コマンド実行フレームワークを操作できる制御テストフレームワークを定義した．これにより，テストコードのメンテナンス性を高め，サーバの運用・管理コストを低減することができる．また，フレームワークを用途別に分離して定義することにより，制御テストフレームワークを独自の記法に変更することも容易である．例えば，本論文で提案するテストフレームワークでは，テスト記法して RSpec[14] を採用しているが，minitest[15] に差し替えたり，あるいはまったく独自の記法に差し替えたりすることも可能である．このテストフレームワークを serverspec[16] と名付けた．

本論文の構成について述べる．2 章ではサーバの構成管理とテスト手法について更に詳しく述べる．3 章では提案するサーバのテスト手法を詳細に述べ，4 章では提案手法によりもたらされた成果を論じ，5 章で結びとする．

参考文献

[1] Mark Burgess, “CFEngine: a system configuration engine”, University of Oslo report 1993, http://cfengine.com/markburgess/papers/cfengine_history.pdf
[2] Luke Kanies, “Puppet: Next-Generation Configuration Management”, USENIX ;login, February 2006, Volume 31, Number 1 https://c59951.ssl.cf2.rackcdn.com/807-kanies_0.pdf
[3] “Chef”, http://www.getchef.com/.
[4] “SaltStack”, http://www.saltstack.com/.
[5] “Ansible”, http://www.ansibleworks.com/.
[6] Patrick Debois, “Agile infrastructure and operations: how infra-gile are you?”, Agile, 2008. AGILE ’08. Conference, http://www.jedi.be/presentations/IEEE-Agile-Infrastructure.pdf.
[7] Stephen Nelson-Smith, “Test-Driven Infrastructure with Chef, 2nd Edition Bring Behavior-Driven Development to Infrastructure as Code” ISBN: 978-1449372200.
[8] “ChefSpec”, http://code.sethvargo.com/chefspec/.
[9] “rspec-puppet”, http://rspec-puppet.com/.
[10] “Cucumber-Chef”, http://www.cucumber-chef.org/.
[11] “minitest-chef-handler”, https://github.com/calavera/minitest-chef-handler.
[12] “Test Kitchen”, http://kitchen.ci/.
[13] “rspec-system”, https://github.com/puppetlabs/rspec-system.
[14] “RSpec“, http://rspec.info/.
[15] “minitest”, http://docs.seattlerb.org/minitest/.
[16] “serverspec”, http://serverspec.org/.