B-012

マイクロサービス向けデータ整合性維持に関わるテスト支援フレームワークの提案と初期評価

Proposal and initial evaluation of a test support framework for maintaining data integrity for microservices

大越 淳平[1]
Jumpei Okoshi

1. はじめに

近年，FinTech の潮流を背景に，金融分野を中心にマイクロサービスと呼ばれる設計手法が注目を集めている [1]。この設計手法は，独立した小さなサービスの集合によりシステムを構成するアーキテクチャであり，開発の効率化やシステムの柔軟性の向上に寄与するとされる。

一方，マイクロサービスを採用したシステムにおいては，複数のサービスに跨る決済処理など，サービス間においてデータの整合性維持が重要となる。サービス品質の担保にはテストが有効となるが，分散システムや独立したサービス開発など，マイクロサービスの特性に起因した要因により，データの整合性維持に関わるテストに多大な工数を要する点が問題となっている。

本研究では，以上の状況を鑑み，マイクロサービス環境でのテスト工数増大の解消を目的としたテスト支援フレームワークを提案する。EC（Electronic Commerce）サービスを対象に，工数削減効果を評価し手法の有効性を明らかにする。

2. 提案フレームワーク

図 1 に提案するテスト支援フレームワークを示す。本フレームワークは，テスト設計や実行を支援するガイド，テスト実行支援機能，及び改善施策立案機能を有する。本フレームワークの設計思想として，設計から改善までのテストにおける各プロセスを，連携するサービスのテスト可能性・観測性の制約を加味して包括的に支援することが挙げられる。前述の通り，マイクロサービスは各サービスが独立に開発されるため，特に E2E（End-to-End）テストが難しい。本フレームワークでは，この制約を考慮したテストシナリオの生成や実行を支援するテスト実行支援機能と，テスト結果を解釈し改善施策を立案する改善施策立案機能によりフィードバック系を構築することで，包括的な支援を実現する。本フレームワークを用いた基本的なテストプロセスは以下の通りとなる。

1) 設計者は，Saga や 2PC（two-phase commit）[2]など各種データ整合性維持手法を用いたビジネストランザクションを設計し，設計データや必要に応じたテスト項目を出力する（図 1 の設計フェイズ）。
2) 開発者は，設計データを受け取り，フレームワークの支援により生成されたテストシナリオに基づき，一連のテストを実施する（図 1 の開発フェイズ）。本テストは，特に単体テストなど，開発と一体で実施される単一サービスに閉じるテストが対象とされる。
3) テスタは，同じく設計データを受け取り，フレームワークの支援を受けて生成されたテス

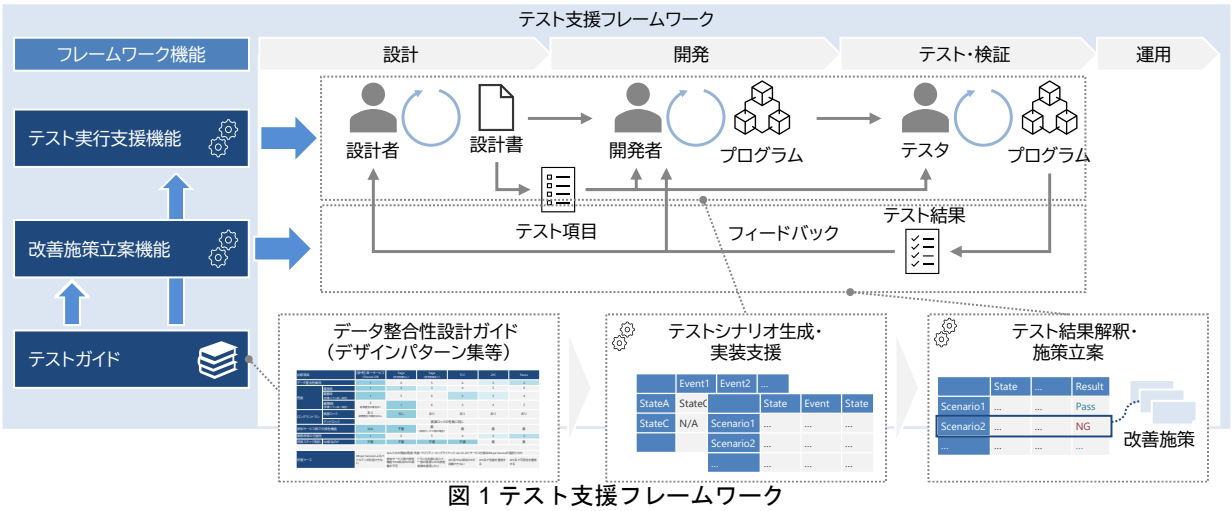

図 1 テスト支援フレームワーク

[1] 株式会社 日立製作所 研究開発グループ Hitachi, Ltd. Research & Development Group
[2] いずれも分散環境下におけるサービス間のデータ整合性維持に関わる代表的な手法

トシナリオに基づいてテストを実施する（図 1 のテストフェイズ）。本テストは，特に E2E テストや受け入れテストに相当したテストなど，複数サービスが連携したテストが対象とされる。

4) 設計者，開発者，テスタは，フレームワークの支援を受け，テスト結果から得られた改善施策（典型的にはデータ整合性維持に関するデザインパターン）を用いて各種フィードバックを実施する（図 1 のフィードバックフェイズ）。

以降の節では，本フレームワークの特徴機能であるテスト実行支援機能と改善施策立案機能の詳細について，特に，データ整合性維持の観点での詳細について述べる。

2.1 テスト実行支援機能

テスト実行支援機能は，開発者やテスタからサービス仕様を入力として受け付け，状態遷移を加味した契約[1]であるステートフル契約を生成する。具体的には，サービス仕様書等のドキュメントから生成されるテスト対象のサービスと，接続先サービスの状態遷移表に基づき，API コールやサービス間で伝搬されるイベントの対応からビジネストランザクションごとの遷移経路をテストシナリオとして抽出する。このテストシナリオの各状態遷移に対応した契約（典型的には API コールに対応した応答の定義）を紐づけて，一連の契約をステートフル契約として定義することでステートフル契約テストを実現する。契約やステートフル契約の代表的な実装形態としては，Spring Cloud Contract[2] や Mock/Stub の活用が考えらえれる。このテスト実行支援機能により，ビジネストランザクションの状態遷移に対応したステートフル契約テスト（E2E テストに相当）を，サービス単体に閉じた形で実行可能となる。

2.2 改善施策立案機能

改善施策立案機能は，テスト結果を受け取り改善施策をユーザに提示する。具体的には，テストをパスできなかったテストシナリオの諸条件（発生した異常など）とテスト結果の組み合わせから，対処法をデザインパターン（体系としては[2]が詳しい）として提示する。例えば，イベントのロストを模したテストにおいて結果が失敗となった場合，イベントを再送するデザインパターンである Retry パターンを改善施策として開発者に提示するなどが挙げられる。

3. 評価

決済サービスと連携した EC サービスをユースケースとし，提案するテスト支援フレームワークの適用によるテスト関連工数の削減効果を机上にて評価した。具体的には，EC サービス，決済サービス，商品管理サービスの 3 サービスに跨る発注処理において，正常シナリオと，決済サービスもしくは商品管理サービスの処理が失敗する 3 シナリオを想定した。

表 1 評価結果

プロセス	削減率	Before	After	Before-After
状態遷移図作成	0.0%	25.0	25.0	0.0
シナリオ生成	100.0%	25.0	0.0	25.0
シナリオ実装	66.7%	16.7	5.6	11.1
テスト結果解釈	50.0%	16.7	8.3	8.3
テスト再設計	50.0%	16.7	8.3	8.3
Total		100.0	47.2	52.8

表 1 に提案手法の適用による各テストプロセスの工数の削減効果を示す。なお，本試算においては，以下の 3 つの想定を置いた。

- 各プロセスの比率はマイクロサービス開発者に対するヒアリングにより算出した。
- 状態遷移図作成については手動での作成が必要なため削減効果なし，シナリオ生成については本フレームワークにより 100%の削減効果，シナリオ実装については Mock/Stub の注入により支援可能な 2 サービスを考慮し全体で 2/3 の削減効果とした。
- テスト結果解釈とテスト再設計については改善施策の網羅率 50%とし，削減効果も同様とした。

提案手法により，テスト関連工数の 52.8%が削減されると試算された。特に，テストシナリオ生成や実装に要する削減効果が大きいものと期待される。一方で，状態遷移図の作成は全体の比率も大きいことに加え，依然として人手による作成が必要となっており，当該プロセスの自動化がさらなる工数削減の観点で有効であると考えられる。

4. 結言

本研究では，マイクロサービスにおけるテスト工数の増大に対し，テスト実行支援機能と改善施策立案機能を持つテスト支援フレームワークを提案した。また，EC サービスを想定した机上評価により，テスト工数の 52.8%の削減を確認した。

今後の課題として，実環境における検証，及びテスト削減効果の評価が挙げられる。

参考文献

[1] M. Fowler, Microservices, https://martinfowler.com/articles/microservices.html (Accessed: 2024/3/17).
[2] C. Richardson, Microservice Architecture, https://microservices.io/. (Accessed: 2021/3/19).

[1] 契約テストにおける契約に相当

[2] https://spring.io/projects/spring-cloud-contract