IT システム運用サービスの構造化と品質向上

関 久根

Structural Definition for Quality Improvement of System Operation Service

Hisane Seki

本論文は，IT システムの運用に関する品質の向上を目的として，IT システム運用サービスの構造化を提案する．構造化によって定義された IT システム運用サービスによって，サービスの網羅性が保証され，結果として属人性の排除が可能になる．また，サービスの開発物における網羅性の高いテストが可能となり，初期段階および運用開始後の IT システム運用サービス全体の品質向上につながる．

This paper describes a new methodology to define operational service model with new structural modeling analysis method, and a technique to develop high quality documents and programs for operation service with object oriented development method. These techniques can assure the coverage of operational service definitions and tests of services, and improve the total quality of IT Operation Service.

Words & Phrases：サービス構造化定義 , 開発手法 , システム運用サービス , サービス定義の網羅性 , 属人性排除
Structural service definition, Development methodology, System operation service, Service definition coverage, Human dependency exclusion

1. はじめに

IT システム運用サービスの品質は，実際の運用実施者のスキルに頼ることが多く，さまざまな運用作業において属人性を生みやすい傾向がある．その多くは「サービス対象に対するサービス定義の網羅性の不備」が原因である．また，運用実施者のスキルに頼らざるを得ない部分の多くは文書化されておらず，確実な検証が行われていないため，サービス・イン後の IT システム運用サービスの品質保証を困難にしている．一方で，作業のアセット化［1］や IT システム基盤アーキテクチャーの網羅性を確保する方法論［2］は示されているものの，IT システム運用サービスの開発における最終成果物に対する適切な開発手法がないため，IT システム運用サービスの初期段階の品質の保証を困難なものにしている．

これらの課題は，大きく二つの要素が関連している。一つは IT システム運用サービスの開発者は，アプリケーション開発者ではなく，インフラストラクチャーのスペシャリストが担当することが多いことである．一般的にインフラストラクチャー・スペシャリストは，運用機能の開発において Java™ などの開発言語を用いることはまれであり，特定の開発手法を用いることに深い知識を持ち合わせていない場合が多い．開発手法では一般的な機能の構造化設計，文書化，テスト方法を適用することはほとんどなく，知識の範囲で対応してしまうことが多いのが現状である．もう一つは，開発された IT システム運用サービスの機能そのものの品質を維持，改善する方法が，アプリケーション保守で実施されている方法と同様に扱われていないことが挙げられる．「アプリケーションとそれを支える IT システム運用サービスの品質は，同時に維持，改善されていく必要がある」という考え方を実現させるには，多くの困難を伴っているのが現状である．

ここで論じている IT システム運用サービスとは，運用対象となる IT システムの稼働，維持を高い品質で保証するために必要なすべての要素を包含したものであり，運用管理サービスと運用機能サービスに分類できる．

運用管理サービスは，近年重要度が増している ITSM（IT Service Management）の領域であり，ITIL®（IT Infrastructure Library）に代表される IT サービスの維持，改善に重点を置いたサービスである．一方，運用機能サー

提出日：2007 年 9 月 3 日　再提出日：2008 年 2 月 27 日

ビスは，運用作業そのものをサービスとしており，SO（Strategic Outsourcing）などが代表的なサービス提供形態である．開発された運用機能サービスが運用管理サービスと連携することで，IT システム運用サービス全体が整合をもって実施可能であり，品質が保証されることを具体的に検証する．

本論文では，運用機能サービスを中心に，その構造化を検討する．以下，2 章で IT システム運用サービスの開発時および運用開始後の課題点を指摘し，課題克服のためのアプローチについて検討を行う．3 章では，IT システム運用サービスの構造化を検討することにより提供されるサービスの網羅性の保証と属人性の排除が可能であることを示す．最後に 4 章で IT システム運用サービスの開発において構造化アプリケーション開発手法を参照適用することにより，サービス全体の品質の維持，向上の可能性であることを論ずる．

2. IT システム運用サービスの品質向上の課題と解決アプローチ

2.1 開発初期段階の品質保証

運用機能サービスを開発する初期段階は，IT システム運用サービス全体の品質の作りこみを行う重要な段階である．開発段階のサービス品質向上は，一般的なアプリケーション開発と同様の考え方で課題提起が可能である．

（1）運用対象のサービス定義の網羅性の保証

運用すべきすべての対象物に対する運用機能サービスの定義が不可欠．

（2）運用機能サービスの開発手法の確立

要件定義，分析手法，仕様書の整備が品質の作りこみ，維持に不可欠．

（3）運用機能サービスに対する網羅性の高いテストの実施

開発された運用手順，運用プログラムに対する単体／統合／システムテストの完全実施

2.2 サービス・イン後の品質維持

IT システム運用サービスがサービス・インした後のサービスの品質の維持については,以下の対応が課題となる．

（1）IT システム運用サービス自身のサービス・レベルの管理

IT システム運用サービスが対象としている IT システムと同様に，IT システム運用サービス自身のサービス・レベルの維持，向上のための厳格なサービス管理が不可欠である．本テーマと直接関わる部分としては，開発された IT システム運用サービス関連成果物をサービスの CI（Component Item）として扱い，その構成管理，変更管理，問題管理等を行う点が挙げられる．

（2）属人性の排除

運用対象の網羅性が保証されない場合，運用作業者の経験やスキルによって運用作業が行われる事態が散見されるが，その場合はスキルや経験の少ない運用作業者は，その具体的な手順が文書化されていないために対応することができない．結果として，一部の運用作業者でしか対応できないという属人性を助長してしまう．また，同時に文書化されていない作業手順は，完全なテストが行われることはなく，結果として作業の失敗，失敗時の場当たり的な対応を招くことになる．

2.3 課題克服のためのアプローチ案

前述した課題を克服するために，IT システム運用サービスの構造化定義には構造化分析手法を，運用機能サービスの開発にはオブジェクト指向をベースとした開発手法の適用を検討する．筆者がこのアプローチを選択した理由は，以下の仮説に基づいている．

（1）運用機能サービスの構造化定義

運用対象となる H/W（ハードウェア），S/W（ソフトウェア），サービスは，個々は運用する対象としては最小単位であり，構造化アプリケーション開発手法におけるモジュールとしてとらえることが可能である．また運用機能サービスは，複数の運用対象に対する作業をパッケージ化したものとして提供されるはずである．

（2）運用機能サービスの分析

運用対象に対する個別の手順は，運用作業者がアクターとなったユース・ケースとして定義可能である．また提供される運用機能サービスは，実際には複数の運用対象に対する連続作業としてとらえることが可能であり，そこには明確なシーケンス定義が可能である．開発および分析に際しては，オブジェクト指向開発におけるユース・ケース分析およびシーケンス図による表現が可能なはずである．

3. IT システム運用サービスの構造化

運用機能サービスは，サービス提供を受ける側（以下，ユーザー）の視点，運用管理者の視点，運用実施者の視点を考慮して体系化する必要がある．最終的な品質

を確認する場合，それぞれの視点から見て，サービスが成り立っているかを保証することが重要になるからである．

3.1 IT システム運用サービスのサブシステム化

ユーザーは，サービス提供者が提示するサービス・カタログを参照し，必要な運用機能を選択することになる．ここで選択されるのは，細かい運用作業（例：DB2® の起動）ではなく，あくまで運用目的に応じたサービス（例：長期連休明けのシステム起動）である．また，運用作業実施を行うことと，それらが円滑に行われるための運用管理は内容がまったく異なる．場合によっては運用管理者と運用実施者が異なるサービス会社によって提供されることも考えられる．

ここではまず，IT システム運用サービスを分類した運用管理サービスおよび運用機能サービスを，お互い補完しあうサブシステムとして定義付けする（**図 1** 参照）．最終的には 2 つのこれらのサブシステムは連携して実行可能であることを証明することが，IT システム運用サービスの品質の保証となることはいうまでもない．

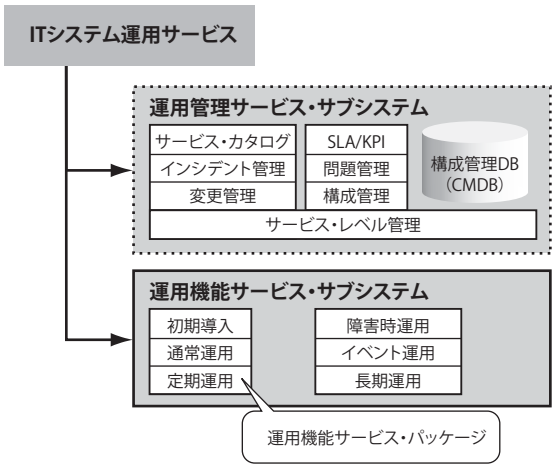

図 1. IT システム運用サービスの構造概要

(1) 運用管理サービス・サブシステム

IT システムの運用に関連した管理（インシデント管理，問題管理，構成管理，変更管理など），指示命令，サービス・レベルの維持・把握などを目的としたサブシステムである．サブシステムの提供物としては，運用管理サービス実施に必要な以下のものが構成要素となる．

- サービス・カタログ
- 運用管理業務フロー
- 体制・役割・責任定義
- 運用管理手順
- サービス・レベル

このサービス・サブシステムは，IT システム運用サービスの品質の維持管理を目的としており，主にシステム運用管理者がサービス提供者となる．成果物は具体的に示されることが多い［3］

(2) 運用機能サービス・サブシステム

システムの運用・維持の具体的な作業実施を目的としたサブシステムである．サブシステムの提供物としては，運用・維持に必要な以下のものが構成要素となる．

- 作業手順書
- プログラム（運用のために開発したシェル，バッチ，実行モジュール）

このサービス・サブシステムは，運用対象に対する運用作業そのものをサービスとしており，運用を実際に実施する運用作業者がサービス提供者となる．

運用管理サービス・サブシステムがサービス全体の方針，ルール，形式を決めているのに対して，運用機能サービス・サブシステムはシステム運用の具体的な手段を決めているといえる．

本論文では運用機能サービス・サブシステムにフォーカスして論じることとする．

3.2 運用機能サービスのパッケージ化

一般的に，IT システム運用サービスで提供されるサービスは，運用局面ごとに作業内容や実施頻度が異なる．一方で開発される運用機能は起動，停止，稼働確認など，運用作業者が実際に行う作業の単位であるため，サービス提供されるレベルとは乖離（かいり）がある．提供されるべきサービス形態は，目的に応じて必要な複数の運用機能をパッケージ化したものといえる．

運用機能サービス・パッケージは，ユーザーがサービスの構造を理解し，選択しやすくするための単なるカテゴリーであると同時に，運用機能サービス・サブシステムのサービス・レベルの確認単位である．（**表 1** 参照）

表 1. 運用機能サービス・パッケージ

運用機能サービス・パッケージ	運用局面	想定頻度	メモ
初期導入	構築時	1回	インフラ・運用設計，導入
通常運用	サービス期間中	毎日	決められた運用項目を毎日実施
定期運用		週次／月次／年次	決められた運用項目を決められたタイミングで実施
障害時運用		随時	随時実施
イベント時運用		中期計画	リクエスト・ベースで実施
長期		3〜5年ごと	製品ライフサイクル，リース切れを見越して計画して実施

3.3 サービス開発者および運用作業者の視点に立った運用機能サービスの構造化

IT システムの運用作業は，システムの構成要素に対して個別もしくは運用目的に応じて操作する作業の集合である．一般的な運用対象は，大きく H/W，S/W およびサービスに分類可能である（**図 2** 参照）．

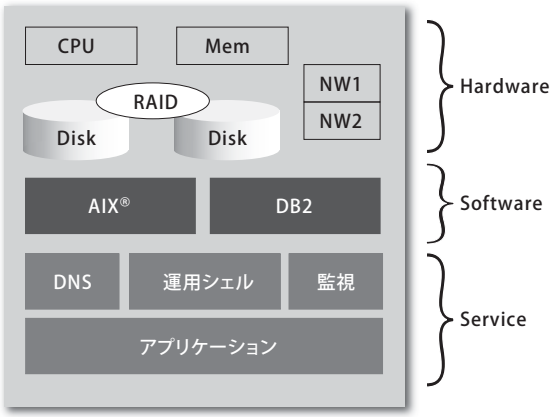

図 2．一般的な DB サーバーの構成

ここで H/W，S/W は，製品そのものを想定しており，その操作は製品マニュアル等に詳述されている．一方サービスは，アプリケーション以外に，バックアップや監視などの維持に関連した機能，外部インターフェース（DNS（Domain Name Service）や NTP（Network Time Protocol）など）を想定としており，運用手順は最初から開発しなければならない場合が多い．開発された運用手順は，それだけで独立して使用されることはほとんどなく，運用担当者の視点で適切に選択され使用されることが多い．運用機能サービスを開発者視点，運用作業者視点でそれぞれ分析すれば，適切なサービスの構造化が可能となる．

3.3.1 運用機能サービス・モジュールの定義

各運用対象には必ず「操作（Operation）」が存在する．例えば，DB2 という S/W に対しては，導入，起動，停止，設定変更，データ・メンテナンス，状況確認などの操作がある．これらの操作は，さまざまな目的に応じて，ある運用操作手順の一部として呼び出される．この「操作」を運用対象に対するユース・ケースととらえると，ある目的の運用操作手順を構成する最小作業単位であると考えることができる．これはサービス開発者の視点における開発単位であり，ここでは「運用機能サービス・モジュール」と定義する．運用機能サービス・モジュールは，さまざまな局面で呼び出されるが，それ単体では作業を行うことはほとんどないはずである．例えば「DB2 の起動」という作業は，システム全体の起動の一部，障害対応後の再起動の一部として呼び出されるのが一般的である．この特性は，構造化アプリケーション開発において，単一の機能を実行するために複数のモジュールが関連しあっている機能的強度の高いモジュールの構造に類似している．同時にサービスの構造化分析において，機能に重点をおいたモジュール強度を考慮して定義することで高い可搬性を実現するのと同様に，点在する手順をモジュール化することで，確実なテスト範囲の特定や変更容易性を得られるという大きなメリットがある．

運用機能サービス・モジュールは，手順（Sequence）に従った作業（Operation）になるが，実際には以下の実行形態を取る．

- 連続したコマンド実行
- GUI（Graphic User Interface）を使った対話式の操作

GUI 操作が中心の手順の場合には，画面イメージ等を含めた手順書を作成し，運用者は手順書に従って操作を進める．対話式操作が一切入らない手順の場合はコマンド呼び出しおよびエラー・ハンドリングを含むスクリプトとして定義可能であり，その場合の手順書はスクリプト呼び出し手順になる．このように考えると，運用機能サービス・モジュールの開発物は，手順書およびスクリプトとなる．

3.3.2 運用機能サービス・コンポーネントの定義

前述した運用機能サービス・モジュールは，単体で使用されることは想定していない．運用作業者が実際に運用作業を行う場合は，IT システムの構成要素に対して運用目的に応じて，必要な運用機能サービス・モジュールを手順（Sequence）に従って呼び出す形態を取る．

例えば電源が Off の状態の DB サーバーを，サービス可能な状態まで起動する「システム起動」という操作を運用作業者が行う場合，「p550 起動（電源 On）」「p550 正常稼働確認」「AIX 起動」「AIX 正常稼働確認」「DB2 起動」「DB2 正常稼働確認」「アプリケーション起動」「アプリケーション正常可能確認」という手順を実施する．

これら一つひとつは H/W（p550），S/W（AIX,DB2），サービス（アプリケーション）それぞれの運用対象に対して定義された運用機能サービス・モジュールの連続呼び出しであり，そこには明確な手順（Sequence）が存在する．運用作業者の立場からみれば，運用作業の目的は「システムの起動」であるため，この作業目的に対して手順書は 1 つであるべきである．運用目的を実現するために必要な複数の運用機能サービス・モジュールを，手順（Sequence）に従って作業実施する運用作業者の視点

における作業単位であり，ここでは「運用機能サービス・コンポーネント」と定義する．

運用機能サービス・コンポーネントは，運用作業者が実際に作業を行う単位に着目して洗い出しを行う．また，運用局面と関連性が強いため，最終的には前述した運用機能サービス・パッケージで分類される．従って，運用機能サービス・パッケージに必要な運用作業を列挙すれば，比較的容易に洗い出しが可能である（**表 2** 参照）．また，運用機能サービス・コンポーネントは，実際に運用作業が想定されるものについてはすべて定義する必要がある．例えば「稼働状況確認（Health Check）」について，「システム全体」，「S/W のみ」，「アプリケーションのみ」それぞれの運用作業が想定されるなら，別々に定義する．これによって運用作業者の作業内容が確定することになり，結果的に属人性の排除に貢献できる．

例えば，DB サーバーの障害発生時に最初に運用作業者が行う「稼働状況確認（Health Check）」は，「p550 稼働状況確認」「AIX 稼働状況確認」「DB2 稼働状況確認」「アプリケーション稼働状況確認」という運用機能サービス・モジュールを，この手順（Sequence）どおりに連続して作業実施すればよいことになる．当然この運用機能サービス･コンポーネントには，「稼働状況確認」という運用目的に必要なすべての操作が含まれる．

表 2. 運用機能サービス・パッケージと運用機能サービス・コンポーネント

運用機能サービス・パッケージ	運用機能サービス・コンポーネントの例
初期導入	H/W設置，設定，S/W導入，設定，チューニング，サービス導入
通常運用	サービス監視，パフォーマンス監視，リソース監視
定期運用	H/W起動，停止，構成保全，S/W起動，停止データメンテ，データ・バックアップ，サービス起動，停止
障害時運用	稼働状況確認，情報収集，障害対応，修正適用
イベント運用	JOB更新，アプリ更新，計画停止

3.4 運用機能サービス構造化のまとめ

これまで定義したサービス定義をエンティティと考えて，その関連を ER 図（Entity Relation Diagram）によってモデリングを行うと以下のように表現可能である（**図 3** 参照）．なお，このモデルでは，後述する運用機能サービスの開発過程で作成されるドキュメントとの関係も表している．

運用機能サービス・コンポーネントは，運用作業者が必要作業を実施するのに必要な複数の運用サービス・モジュールを参照利用している．一方で，一つの運用サービス・モジュールは可搬性の高い機能として提供されているので，多くの運用サービス・コンポーネントから参照

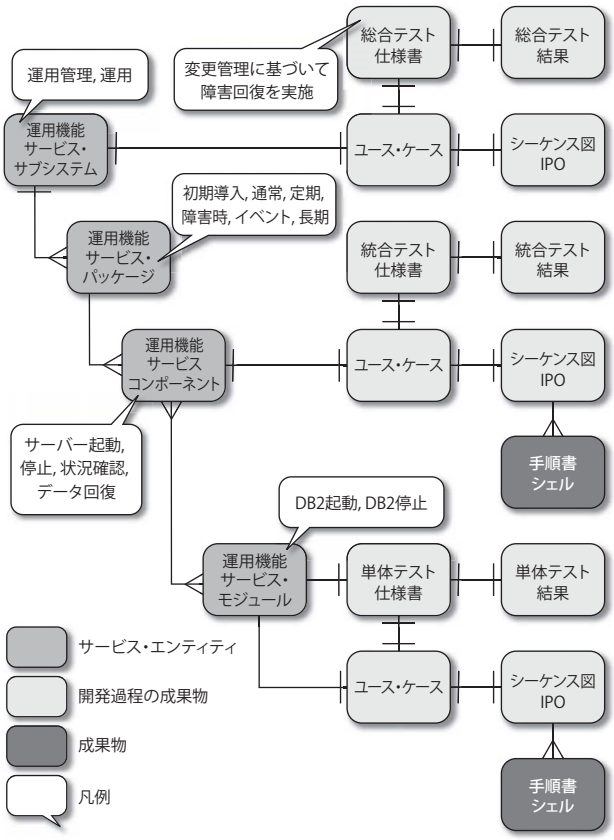

図 3. 運用機能サービスのモデリング（ER 図）

利用されている．このように運用機能サービス・コンポーネントと運用機能サービス・モジュールは M 対 N の関係を持つ．

一つの運用機能サービス・モジュールに変更が加わった場合，単体テストを実施して運用機能サービス･モジュールとしての品質を確認した後に，参照使用している運用機能サービス・コンポーネントの統合テストのみを実施すれば，最小の工数で確実な品質の保証が可能となる．これは運用機能サービス･モジュールの高い可搬性のメリットといえる．別の視点で考えると，定義された運用機能サービス・モジュールが，結果としてどの運用機能サービス・コンポーネントからも参照されていない場合は，運用機能サービス・コンポーネントの洗い出しに不備がある可能性がある．このようにサービスのモデル化は，運用対象に対するサービス定義の網羅性の向上に役立つ．

4. IT システム運用サービスの開発と品質保証

IT システム運用サービスの品質は開発物の確実なテストによって保証される．ここでは，前段で定義された運用機能サービスの開発方法および品質保証の考え方について論ずる．

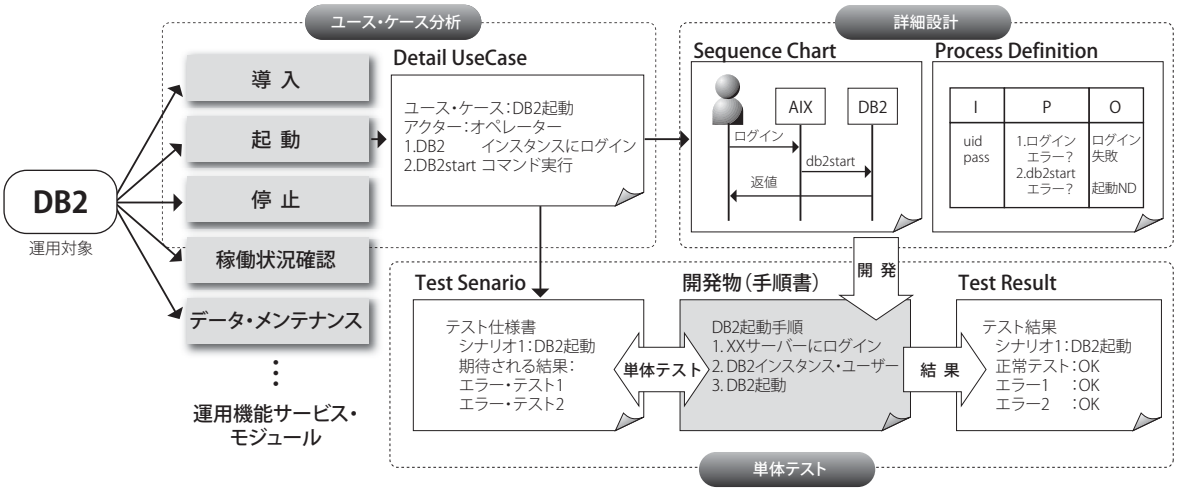

図 4. 運用機能サービス・モジュールの分析、開発およびテスト

4.1 運用機能サービス・モジュールの開発

運用機能サービス･モジュールの開発は,まず「操作」に着目してユース･ケースを洗い出すことから始める. ユース・ケースの洗い出し作業は，今後開発が想定される目的別の一連の運用操作を考慮する必要はなく，製品やサービス単体に対する基本的な操作なので，運用対象の知識を持った開発者なら，比較的容易に洗い出すことが可能である. ユース・ケースは，これから開発される運用機能サービス・モジュールの基本仕様であり，手順（Sequence）が定義可能である. また単体テストの単位としてとらえれば, 品質の作りこみの基礎であると言える. 運用機能サービス・モジュール開発の成果物は，開発成果物（手順書，スクリプトなど）の他，分析段階で記述されるユース・ケース，シーケンス図，プロシージャー定義（IPO），単体テスト仕様書，テスト結果なども含まれる（**図 4** 参照）.

4.2 運用機能サービス・コンポーネントの開発

運用機能サービス・コンポーネントの開発を検討するにあたり，構造化アプリケーション開発手法の参照適用を試みる. 複数の運用機能サービス・モジュールを集めて，運用作業者の作業単位である運用機能サービス・コンポーネントを形作ることは，必要な基本モジュールを呼び出すことで，特定の機能を実現するプログラムを開発するのに類似している. 運用作業者が行う運用作業は，ユース・ケースとしてとらえることが可能であり，運用機能サービス・モジュールの開発と同様に，シーケンス図，プロシージャー定義が可能である. シーケンス図で定義されるのは，必要な運用機能サービス・モジュールの呼び出し順と，エラー・ハンドリングとなる. これらの開発成果物の記述方法は，一般的な UML などのノーテーションが参照可能である［5］. 運用機能サービス・コンポーネントのテストは,単体テストされた運用機能サービス・モジュールの呼び出しテストを行うということになるため，統合テストの位置付けとなる.

4.3 IT システム運用サービスの品質保証

運用機能サービス・サブシステムは，前述した運用機能サービス・パッケージでユーザーに提供されることを想定しており，運用管理サブシステムとの連携において密接に関連している. つまり，運用管理サブシステムと運用サブシステムの連携ができることが確認されて初めて成立すると言える. したがって，サブシステム間の連携テストこそが最終的なシステムテストの位置付けとなる.

例えば障害発生時には，運用管理サブシステムにおいて，インシデントの起票，障害インパクトの評価，優先順位付け等が行われた後にサービスの回復作業に入ると同時に問題管理プロセスが発動することになる. インシデント管理におけるサービス回復においては，必要な変更管理プロセスを経て障害の原因となっている運用対象に対して，「稼働状況確認」,「回復のための障害対応」の運用機能サービス・コンポーネントの実行を指示する. サービス回復後は，問題管理プロセスにおける調査のために,「情報収集」の運用機能サービス･コンポーネントが実行され，収集された情報をもとに解決方法の確立が図られる. 最終的には,変更管理プロセスを経て,「修正適用」の運用機能サービス・コンポーネントの実行によって，障害原因の完全な排除が行われる.

このように，開発された運用機能サブシステムの運用機能サービス・コンポーネントが，運用管理サブシステムで提供される管理プロセス，体制（必要スキルを含む），評価，改善プロセス等との連携が可能であること

が確認できれば，IT システム運用サービスとしてユーザーに提供可能であることが保証されると同時に，全体のサービス品質を高く維持することが可能になる．

4.4 サービス・イン後の IT システム運用サービスの品質の維持

IT システム運用サービスの開発で作成された成果物は，それ自身が IT サービスの CI（Component Item）であり，構成管理，変更管理，問題管理等を行う必要がある．この点に関しては，並行して開発されるアプリケーションと管理方法はまったく同じである．運用管理サブシステムの管理対象がインフラストラクチャーやアプリケーション・サービスが中心であることはいうまでもないが，同時に IT システム運用サービスそのものも同様に ITIL で定義される品質基準による継続的評価を行う［4］ことで，サービス・イン後の IT システム全体の品質保証が可能となる．

また今回検討されたサービスの開発手法は，一般的なアプリケーション開発と考え方は同じである．そのためアプリケーション開発で多く利用，採用されているソース管理やバージョン管理のツール類と親和性が高く，IT システム運用サービス自身の改変が必要になったとしても，開発されたアプリケーションと同様の品質維持が可能である．

5. おわりに

IT システム運用サービスの体系的なサービスの構造化において，構造化分析手法の考え方を適用することにより，IT システム運用サービスの定義の網羅性の確保が容易に可能となる．この事が確実な運用手順の開発につながり，結果として手順書の不備が最大限抑制され，特定運用者でなくては作業できないという属人性が排除可能になる．IT システム運用サービスの開発においてオブジェクト指向をベースとした開発手法を採用することにより，開発手法がもたらすメリットが享受可能となった．

構造化された IT システム運用サービスは検証単位が明確になり，結果として変更にも柔軟に対応でき，高いレベルで品質の維持が可能となる．

プロトタイプを実施した IT システムは，典型的な Web アプリケーション・システムであり，H/W（IBM System p，外部ディスク），S/W（AIX，DB2，WebSphere® Application Server），サービス（バックアップ，ジョブ管理，監視）で構成されている．サービス定義の結果，1 運用機能サービス・サブシステム，6 運用機能サービス・パッケージ，200 運用機能サービス・パッケージ，460 運用機能サービス･モジュールが洗い出し可能であった．IT システム運用サービスの構造化分析の実施によって高い網羅性が確保可能であることが確認されている．

定義された運用機能サービス・モジュール，運用機能サービス・コンポーネントの開発を行うことが可能であることを実証する点が今後の課題である．同時に，既存の運用マニュアルとのベンチマーキングを行うことにより，運用作業項目の網羅性についても検証をしていく必要がある．

謝辞

当論文の執筆にあたっては，大手自動車会社様の生産工程管理システム開発のプロジェクト・メンバーより多くの助言をいただきました．

久保篤史氏には，現在の IT システム運用が抱えるさまざまな課題について，強力な問題提起をしていただきました．大井健祐氏には，開発メソドロジーのスペシャリストの立場から，IT システム運用サービスの構造化定義および品質の保証に関して多くの助言をいただきました．吉岡崇氏，和田全弘氏には，運用担当者の経験をもとに，洗い出された IT システム運用サービスの網羅性の検証をしていただきました．あらためて深謝いたします．

参考文献

［1］加藤 洋, 三谷 隆, “サーバー統合基準の確立と作業アセット化によるTCO削減,” IBM PROVISION, No.47, pp.62-67（2005）.
［2］大嶽 隆児, “表形式テンプレートにより網羅性と一貫性を確保したITシステム基盤アーキテクチャ設計手法,” IBM PROVISION, No.47, pp.83-90（2005）.
［3］itSMF: IT サービスマネジメント －ITIL入門, Van Haren Publishing, ISBN:90-77212-23-X（2002）.
［4］Randy A. Steinberg: Measuring ITIL, Trafford Publishing, ISBN: 141209392-9（2006）.
［5］長瀬 嘉秀: UML2ハンドブック, 翔泳社, ISBN:4-7981-0590-2（2004）.

日本アイ・ビー・エム株式会社
GBS 自動車第二プロジェクト・第二開発部
主任 IT スペシャリスト

関 久根 Hisane Seki

［プロフィール］
1992 年日本 IBM 入社．製造業（自動車）の研究所および生産系システムのお客様担当．Unix® 上のクライアント・サーバーシステムの開発，オープン系大規模インフラストラクチャの提案，設計，構築を経て現在はグローバルに展開する生産工程管理システムの提案，システム設計，運用支援に従事．
hisane@jp.ibm.com