インフラ統合プロジェクトで開発した MQ 環境チェックツールの効果と他製品への展開

あおき　ひろふみ
青木　　啓文

株式会社エクサ　基盤第２ソリューション部
シニアＩＴアーキテクト

原稿量
本文　　１０，６９１字
要約　　１，１２３字
図表　　９枚

＜要　約＞

現在、マルチプラットホームのオープンシステムではJava Platform, Enterprise Edition（以下、Java EEと記す)をベースとしたWebシステムが主流である。1990年代後半までのクライアント・サーバーシステムから、Java Servletシステムを経て、2000年代初頭からJava EEシステムが各企業のシステムに根づいて現在に至る。つまり、この10数年間、Java EEのシステムが主流のままであり、かつてのクライアント・サーバーシステムからWebシステムへ置き換えのようなパラダイムシフトは起こっていない。

一方、インフラ基盤のほうはここ数年、仮想化技術がかなり浸透してきている状況にある。ハードウェアの老朽化やソフトウェアのライセンス切れに伴うシステム更改では、各企業は仮想化技術を積極的に取り込んでいる。仮想化環境の構築にコストをかけたとしても、Java EEが主流であることに変わりはないため、「既存の資産、ノウハウ」などをいかした構築及びランニングコストの削減により、総所有コスト(以下、TCOと記す）の削減ができる。

このような背景の中、弊社は、数百台規模のインフラ統合プロジェクトに基盤担当として携わった。仮想化技術を採用し、OS、ミドルウェアを現行踏襲としたシステム更改が特徴である。OS、ミドルウェアの現行踏襲については、前述の「既存の資産、ノウハウ」活用による、TCOの抑制が目的である。

このプロジェクトでは、既存設計書の移行も高い生産性、品質が要求された。特にWebSphere MQ（以下、MQと記す）については、定義量が非常に多く、アプリケーションチームへの引渡し後や、外部接続リハーサル時の不備の発覚はスケジュール遅延につながるリスクとなる。そのため、設計書の正確さと、環境構築の精度が要求された。
この要求をみたすため、MQに対しては、MQ環境チェックツールを開発することで対応した。このツールは、環境と設計書の差分をエクセルで「視覚化」することに特徴があり、環境差分をすぐに把握できて迅速な対応が可能となった。

開発したツールを活用することで生産性、品質が確保できたと考えている。本稿では、ツールを開発した経緯、概要、開発方法、及び使用効果について記載する。また、MQ以外の製品にもツール開発が適用できそうか考察を加えた。

弊社の案件引き合い状況からしても今後、今回のような規模の仮想化と現行踏襲を主体としたプロジェクト案件が多くなってくると思われる。また、ビッグデータ化に伴いデータベースの設計なども、より大きく複雑になっている。このようなケースにおいて本稿のツールによる取り組みが参考になれば幸いである。

目　次

1.はじめに

弊社は、お客様向けで数百台規模のインフラ統合プロジェクトに基盤担当として携わった。コスト削減が大前提であり、仮想化技術を採用してハードウェアのコストを削減する一方、OS、ミドルウェアを現行踏襲として、TCO を抑えることを目的とした点に特徴がある。

このプロジェクトでは、既存設計書の移行も高い生産性、品質が要求された。MQ は特に定義量が多く、アプリケーションチームへの引渡し後や、外部接続リハーサル時の不備の発覚は、スケジュールの大きな遅延につながるリスクとなる。そのため、設計書が正しく記載された上で、環境構築も正しく行うことが要求された。

この要求をみたすため、定義量が突出して多い MQ に対して対策を行うことが有効であり、MQ 環境チェックツールを開発することで対応した。本稿ではこのツールについて、開発した経緯、概要、開発方法、効果、及び他製品への展開について述べる。

2.インフラ統合プロジェクトの全体像

本章では、MQ 環境チェックツールについて言及する前に、インフラ統合プロジェクトの全体像を説明する。規模感、弊社の取り組み、成果を示し、インフラ統合プロジェクトにおける MQ 環境チェックツールの位置づけを示す。

2.1 概要

お客様のインフラ統合プロジェクト（オープン系、数百台、工期 2 年）において、弊社はインフラ構築メンバーとして、半数程度をしめる形で参画した。全期間を通してプロジェクトはほぼスケジュール通り進捗し、成功裡に完了することができた。

2.1 取り組み

最も重要なことは設計・構築・移行の各フェーズをより正確につなげてプロジェクトを推進していくことである。このため、弊社では早くから次の３つのテーマについてプロジェクト全体で共有するようにプロジェクトを推進した。

2.1.1 仮想化基盤のグランドデザインの共有

数百台規模のサーバーを支える仮想化基盤の設計を確立するためには、多方面からの要件を取り込んで設計を整合させる必要がある。このため、サーバー、ネットワーク、ストレージの各種・仮想化基盤、及び統合バックアップにおいては体系的に各種要件をみたすように強固なグランドデザインを確立するように努めた。

2.1.2 OS、ミドルウェアは生産性の向上、品質確保の実現

現行踏襲する方針で元の設計が正しければ、設計作業の難易度は低いと想定されたが、構築量は膨

大であり、生産性、品質を確保する必要があった。OS 構築ではマスターを展開して差分構築を行う方法で効率化を図ったほか、構築後のクロスチェックを行うなど品質の向上に努めた。更に、本稿のテーマである MQ 環境チェックツールの開発と適用により生産性の向上、品質確保を実現した。

2.1.3 確実な移行準備

ネットワーク、各種データ、データベース、外部接続を対象に移行方式を早期に確立して、移行がスムーズに実行できるように努めた。移行スケジュール、手順概要はチーム間で食い違いがないようにプロジェクト全体で共有して複数回のリハーサルで問題の洗い出しと対策を行った。最終的には時間測定、訓練を含めてじゅうぶんなリハーサルを実施できた。

2.2 成果

３つのテーマを早期に多くのメンバーで共有して、各メンバーがプロアクティブに業務を推進した結果、全期間を通して進捗は順調で、プロジェクトは成功裡に完了した。稼動後、１年以上たっているが、移行に伴うトラブルは発生していない。

3.MQの課題

本章では、インフラ統合プロジェクトの初期段階（要件定義フェーズ）で気づいた MQ 設計・構築に関する課題について記載する。

3.1 他のOS、ミドルウェアと比較して突出した定義量

OS、ミドルウェアについての方針は、現行踏襲及び、バージョンアップによるパラメーター見直しに限定され、特別に大きな負荷はかからないと予想された。負荷が大きくなる可能性としては、複雑な定義が多数あった場合、数百台規模のインフラ統合であるため、構築作業が大変になる場合が考えられる。そのため、事前に現行の設計書を確認し、MQ だけが群を抜いて不規則な定義量が多いことが判明し、何らかの対策が必要だと判断した。これはこのシステムの特徴であり、このような移行プロジェクトで一般的なこととは言えない。

3.2 MQ属性の概算定義数

MQ の定義量を把握するため、現行の設計書をもとに概算で定義量を確認した結果を表１に示す。

表１．MQ 属性の概算定義数

項目	概算数	補足
キューマネージャー	200	
オブジェクト	6,000	ローカル・キュー、リモート・キュー、チャネル、リスナーなどが対象、1 キューマネージャーあたり 30 オブジェクトとして算出
属性	60,000	1 オブジェクトにつき 10 属性として算出

おおよそ 60,000 箇所の設定が必要になることを把握した。

3.3 MQを正しく定義することの重要性

MQ に関しては、次の観点から構築を正確に行うことが重要である。

アプリケーションのテスト時などに不備が発覚した場合、大幅なスケジュール遅延につながるリスクがある。特に、外部接続は 10 箇所以上の接続先があり、夜間帯利用などの限られたリハーサルで接続確認する方針としたため、リハーサル時の不備の発覚はスケジュールにとって致命的となりかねない。

MQ クラスターを採用しているためメッセージ間の定義が複雑で、正確な構築を行うためには設計書も深く理解しておく必要がある。

3.4 MQの設計、構築に要求されること

現行の設計書に関して、更にもう少し調べると設計書に記載されている設定値に間違いがあり、オブジェクトの過不足があることが判明した。この時点で設計書に記載されている内容と定義されるべき情報が一致しないため、前述で述べたリスクを抱えてしまうことになる。

また、ソフトウェアのライセンス切れ対応のため、MQ のバージョンアップも合わせて実施する。このため、若干ではあるがパラメーターの見直しを含めた再設計を実施する必要がある。統合化対応でホスト名などの変更も設計への反映が必要となる。

つまり、MQ の設計に対しては、現行の設計書をもとにインフラ統合用の設計書を起こした上で、次のことが要求された。

- オブジェクト同士の依存関係を正確に把握する必要があるため、設計書が正しい値で記載されていること。
- バージョンアップ対応、統合化対応として、パラメーターの見直しを行うこと。
- 環境定義用のファイルからコマンドで環境構築を行えるようにするため、正しく環境定義用のファイルを作成すること。

これら課題への本プロジェクトでの対応について、説明する。

4.MQ課題対応

本章では、MQ の課題について、従来の手作業による環境差異のチェック量の問題点をどのように対応すべきか段階的に検討した結果を示す。最終的に MQ 環境チェックツールを考案して対応した内容を説明する。

4.1 従来の対応

従来の一般的な対応方法を、図１にまとめた。

① 抽出(saveqmgr)

MQ では saveqmgr というサポート・パックが提供されている。MQ が稼動するサーバー上で saveqmgr コマンドを実行すると MQ キューマネージャー属性、オブジェクトすべてがテキスト形式で定義されている MQ 定義ファイルが得られる。

② チェック、修正

MQ 定義ファイルと設計書で変更を加えている箇所を主体に目で比較して、違いがあれば設計書、又は MQ 定義ファイルを修正する。

③ バージョンアップ化対応設計

MQ 定義ファイルを修正して MQ バージョンアップ、統合化対応を行って、MQ 定義ファイルを作成する。この MQ 定義ファイルを環境構築用に使う。

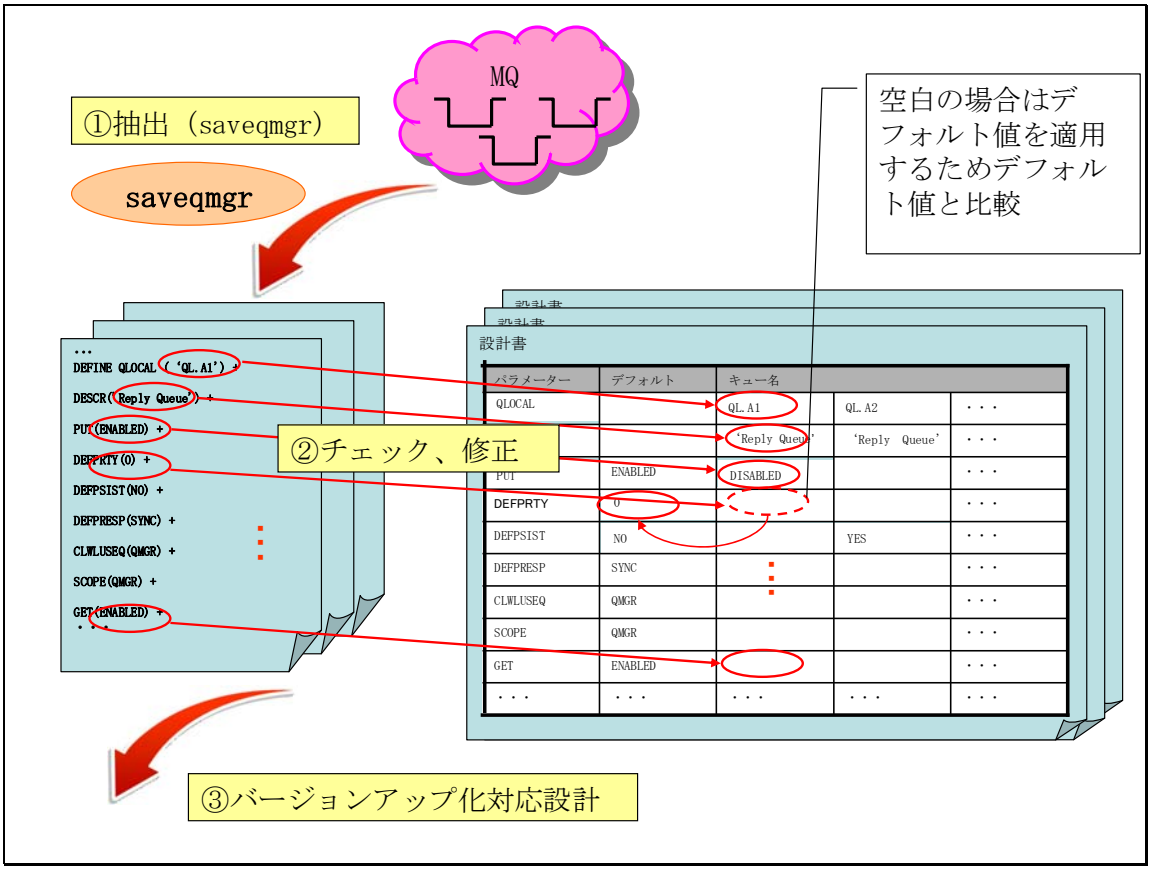


図１．従来の対応

残念ながら、この方法で 200 個のキューマネージャー、6,000 個のオブジェクト、60,000 箇所の属性を正しくチェックすることは容易ではない。

4.2 改善案

改善案として、を図２のような方法を考えた。

① 抽出(saveqmgr)

saveqmgr から MQ 定義ファイルを出力する。

② バージョンアップ化対応設計

MQ バージョンアップのため、設計書を修正する。

③ 設計書からの抽出（WSH）

設計書はエクセルで書かれている。マイクロソフトから提供されている Windows Scripting Host（以下、WSH と記す）を使えば、saveqmgr から出力した MQ 定義ファイルと同等のものが、設計書から出力することが可能となる。WSH 用のプログラムは開発する必要がある。

④ UNIX コマンドによる比較

可能であれば、UNIX コマンドで比較を行う。

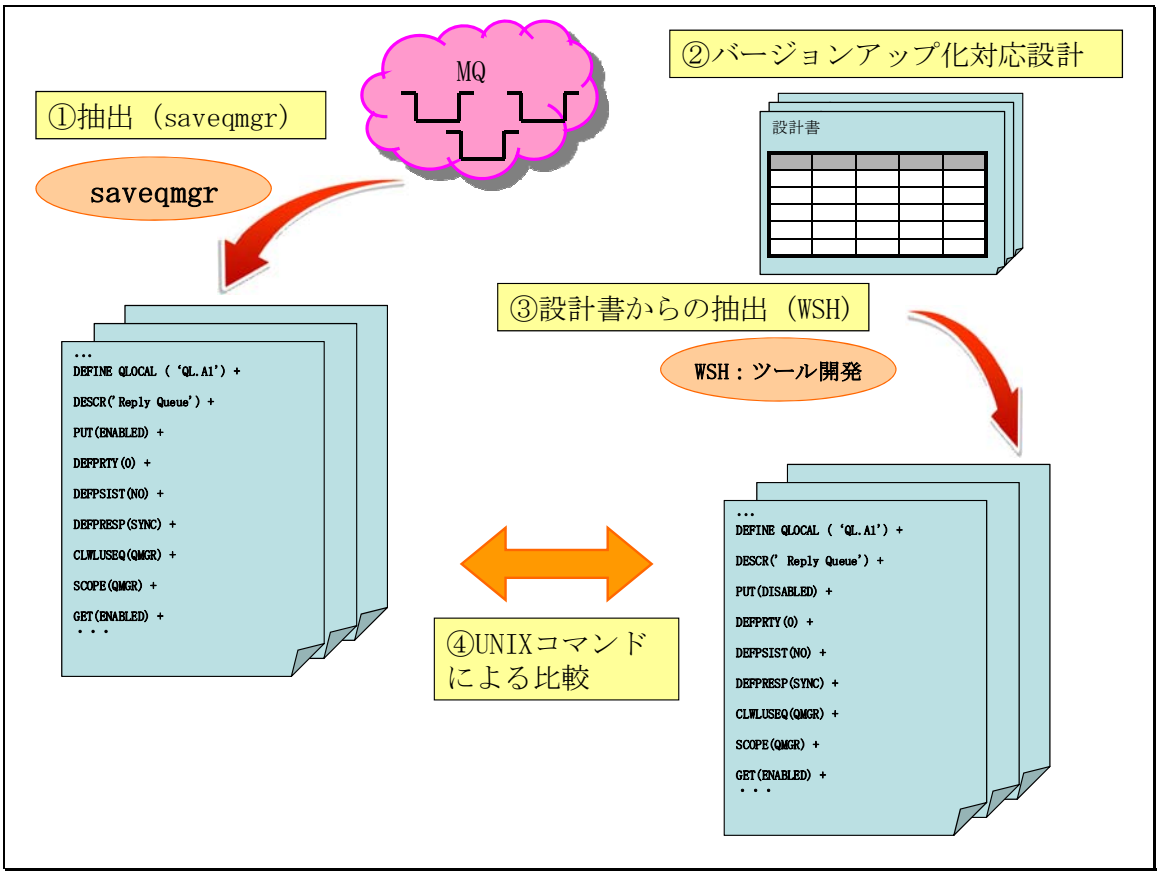


図２．改善案

UNIX コマンドでファイルの比較を行う場合、差分が少ない場合は有効であるが、多い場合、比較する手間が無視できない。また、バイト数単位でフォーマットを整合させる必要があるため、WSH で出力する処理に考慮が必要になる。このため、このままでは使えそうにないと判断したが、この考えを応用し発展させることにした。

4.3 MQチェックツール

4.2　改善案を応用して最終的には図３のような方法を採用し、MQ チェックツールとして実装

した。

① 設計書からの抽出(saveqmgr)

saveqmgr から MQ 定義ファイルを抽出する。

② バージョンアップ化対応設計

設計書を修正して MQ バージョンアップ、統合化対応を行う。

③ 設計書からの抽出 (WSH)

設計書から MQ 定義ファイルを抽出する。

④ 抽出ファイルの流し込み(WSH)

WSH を使えば、MQ 定義ファイルからあらかじめチェック用に準備した MQ パラメーターを、すべての情報を記載したエクセル(チェックシート)にデータを流し込むことが可能となる。WSH 用のプログラムは開発する必要がある。MQ 定義ファイルは saveqmgr と設計書からの抽出で同じフォーマットであるため、プログラムは１本で良い。ただし、見た目すぐ、区分けがつくように流し込む際に設計書からのものはエクセルのセルの背景色が黄色になるようにした。

⑤ チェック (VBA)

Visual Basic for Applications（以下、VBA と記す）を使えば、エクセル上で差分チェックを行い、違いをすぐに把握できるようにすることが可能となる。チェックシートに流し込んだデータをオブジェクト名でソートして、属性に違いがあれば該当箇所の文字を赤色にした。片方にしか定義がないものは全体の文字を赤色にした。

⑥ 設計書修正

⑤ チェック結果、問題がなければチェックが完了し、③で抽出した MQ 定義ファイルを環境構築用に使う。問題があれば、設計書を修正し③～⑥の作業を問題がなくなるまで繰り返す。

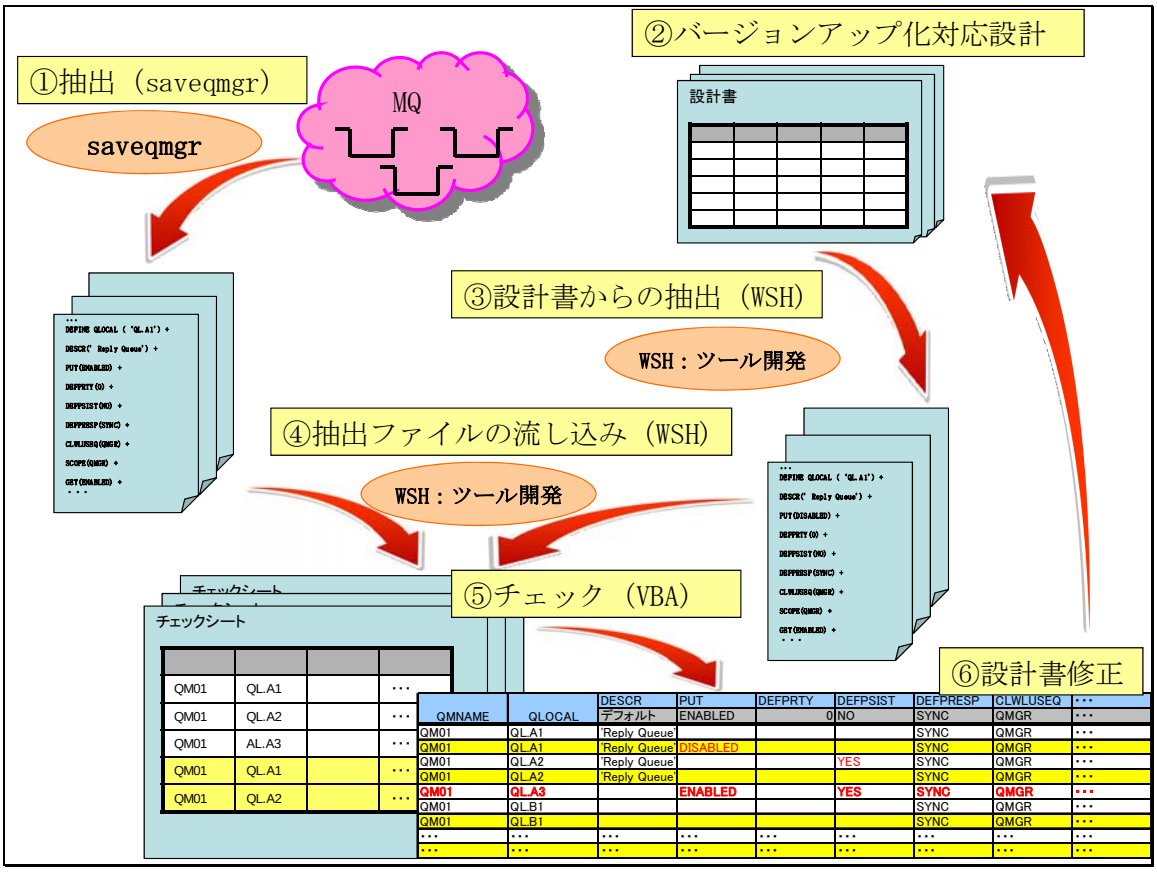


図３．最終的な対応

エクセル上で色を変更することにより、チェック結果した結果、差異がすぐに把握できるようになった。

補足であるが、③抽出（WSH）の替わりに VBA で開発することも可能である。WSH の場合は複数の設計書をバッチで処理できるメリットがある。一方、VBA の場合は外部のプログラムなしで MQ 定義ファイルを出力できるメリットがある。対象設計書が多いため、バッチ化により作業が効率化できる WSH 方式を採用した。コストを抑えることが前提であるため、WSH、VBA のプログラム開発コストは従来の手作業によるチェックからツールによるチェックへの置き換えの効率化で吸収できる前提とした。

4.4 視覚化

チェックシートは１つのキューマネージャーごとに１つのエクセルとして、キューマネージャー、ローカル・キュー、リモート・キュー、別名キュー、モデル・キュー、チャネル、リスナー、名前リストを各エクセル・シートとしている。違いがある場合はすぐに色分けしてわかるように「視覚化」している。saveqmgr から抽出したものが背景色・白色、設計書から抽出したものが背景色・黄色で属性が異なる場合は文字を赤色、片方にしか定義されていないオブジェクトは行全体の文字を赤色太字にしている。ローカル・キューの比較結果例を図４に示す。

QMNAME	QLOCAL	DESCR デフォルト	PUT ENABLED	DEFPRTY 0	DEFPSIST NO	DEFPRESP SYNC	CLWLUSEQ QMGR	・・・ ・・・
QM01	QL.A1	'Reply Queue'				SYNC	QMGR	・・・
QM01	QL.A1	'Reply Queue'	DISABLED			SYNC	QMGR	・・・
QM01	QL.A2	'Reply Queue'			YES	SYNC	QMGR	・・・
QM01	QL.A2	'Reply Queue'				SYNC	QMGR	・・・
QM01	QL.A3		ENABLED		YES	SYNC	QMGR	・・・
QM01	QL.B1					SYNC	QMGR	・・・
QM01	QL.B1					SYNC	QMGR	・・・
・・・	・・・	・・・	・・・	・・・	・・・	・・・	・・・	・・・
・・・	・・・	・・・	・・・	・・・	・・・	・・・	・・・	・・・

図４．比較結果例

5. ツールの開発概要

本章では、MQ 環境チェックツールの開発概要を説明する。詳細なロジックについては紙面の都合上、割愛する。

5.1 抽出（WSH）

WSH による設計書から MQ 定義ファイルへの出力の開発概要を示す。設計書はエクセルのシートのオブジェクトごとに記載されているとする。設計書はこの例で示すフォーマットを前提としている。

キューマネージャー属性、及び、（ローカル・キュー、リモート・キュー、チャネル、リスナーなどの）シートごとに次の処理を行う。

（１）エクセルをオープンする

（２）該当エクセルシートをワークシートとする

（３）オブジェクトの属性値を読み取り、MQ 定義ファイルへ出力していく

対象オブジェクトがなくなるまで処理する。MQ 定義ファイルはキューマネージャーごとに出力する。

当該処理の概要を図５に示す。

① 起点位置の取得

エクセル・セルのデータ取得はセルの何行、何列目がわかると容易に取得可能である。設計書の属性が記載されている表の左上と右下の何行、何列目がわかると２重ループの処理で各属性が取得可能となる。起点位置の取得は表左上の「パラメーター」という文字列はエクセル内で１つであれば、それをキーとして何行、何列目を特定できる。このようにすれば、設計書の編集によって、起点位置が変更になってもプログラムの修正は不要である。

② 終点位置の取得

終点位置の取得は表の左下に属性値が入るため、キーワード検索ができない。その代わりとして「パラメーター」の列に対して、文字列が空白に達するまでの区間行のカウントを加算、同様に QLOCAL の行に対して、文字列が空白に達するまでの区間列のカウン

トを加算して特定できる。
出力用の MQ 定義ファイルをオープンしておく。

③ 項目名、属性値取得、出力
（後で設計書の場所がすぐに特定できるようにするため）最初にエクセルのブック名、シート名、オブジェクトのセルの行列番号をコメントで出力しておく。
以降は、該当する列の項目名と属性値をループ処理で順次取得しながら、出力データを MQ 定義ファイルのフォーマットに加工して出力していく。

④ 繰り返し
次の列に移動し③を実行する。ローカル・キューの対象があるまでループ処理で繰り返し実行する。
出力用の MQ 定義ファイルをクローズする。

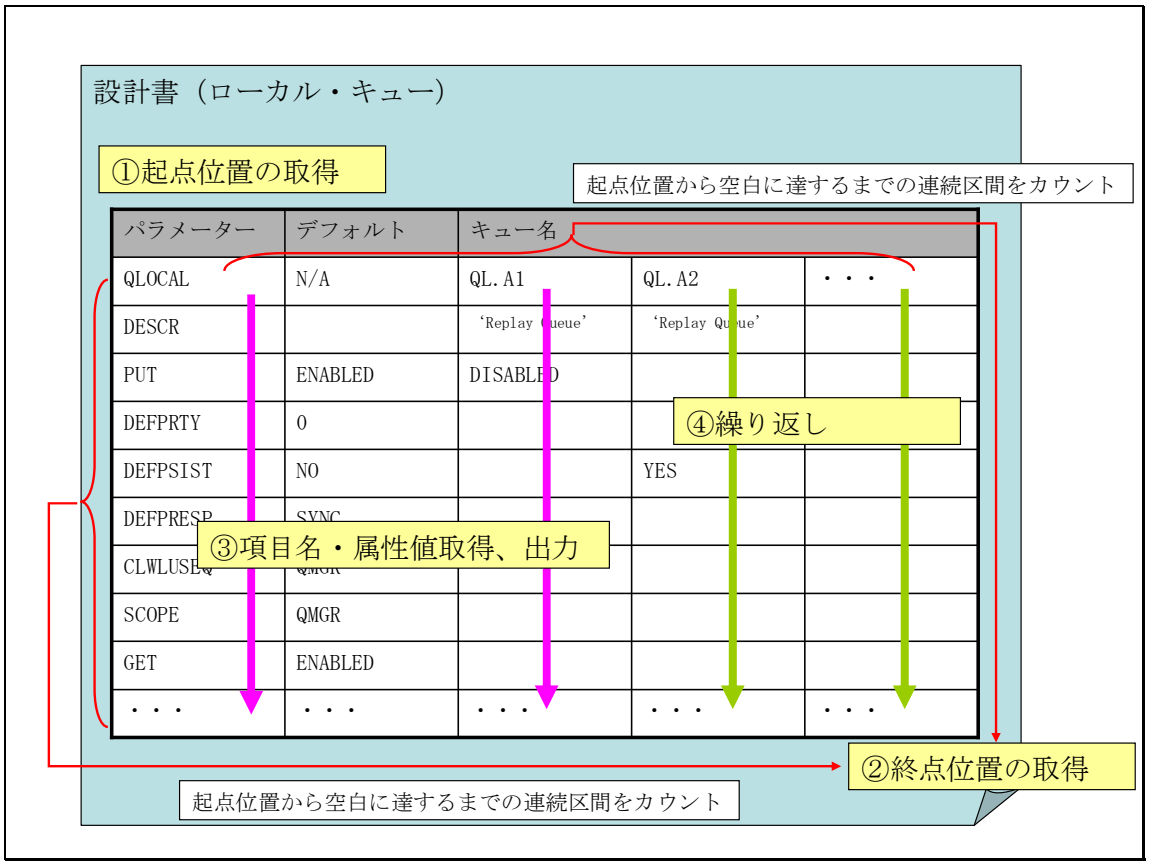


図５．抽出（WSH）エクセルのアクセス処理概要

（４）エクセルをクローズする。

5.2 WSH(流し込み)

WSH による MQ 定義ファイルからテンプレートへの出力の開発概要を示す。

MQ 定義ファイルの書式はオブジェクトの最初の行が「DEFINE オブジェクト・タイプ（'オブジェクト名')」で以降は「属性名（値)」となる。改行する場合は、行の最後に「+」を付与する。以下参照とする。ただし、キューマネージャーの属性値の場合は最初の行を「ALTER QMGR」と置き換える。

```
DEFINE QLOCAL ('QL.A1') +
DESCR('Reply Queue') +
PUT(ENABLED) +
DEFPRTY(2) +
DEFPSIST(YES) +
DEFPRESP(SYNC) +
CLWLUSEQ(QMGR) +
SCOPE(QMGR) +
GET(ENABLED) +
・・・
USAGE(NORMAL)
```

テンプレートはエクセルで MQ のオブジェクトごとにシート分けしたもので、すべての属性を網羅したものを用意しておく。

キューマネージャーごとに次の処理を行う。
（１）MQ 定義ファイル、テンプレートをオープンする
（２）MQ 定義ファイルを１行ずつ読み込みながら、流し込み処理を行う
当該処理の概要を図６に示す。

①シートの選定

読み込んだ行の構文解析を行い、キューマネージャー定義、又は、MQ オブジェクトを定義する最初の行であれば、流し込むシートの選定を行う。

構文解析は文字列に「DEFINE QLOCAL」「DEFINE QALIAS」など何が含まれるかを判別する。

②オブジェクト名の書き込み

選定したシートの最終行をフォーカスする。該当するセルにキューマネージャー名、オブジェクト名を書き込む。

saveqmgr から抽出した場合はキューマネージャーを作成する際に（オブジェクト名が「SYSTEM.」で始まる）システム系のオブジェクトが作成される。これらは設計書の管理外としているため、書き込みの対象外とする。

設計書から抽出した場合はエクセルの行の背景色を黄色で塗りつぶす。

また、設計書からの MQ 定義ファイルであれば、オブジェクトの最初の定義にあらかじめ、

エクセルのブック名、シート名、オブジェクトのセルの行列番号がコメントで出力されているので、この情報を一番右側の備考欄に記入しておく。これは、設計書の修正が必要になる際に設計書と箇所の特定がすぐにできるようにするためである。

③属性値の書き込み

読み込んだ行の構文解析を行い、属性値の出力であれば、最終行をフォーカスする。該当するセルに属性値を書き込む。

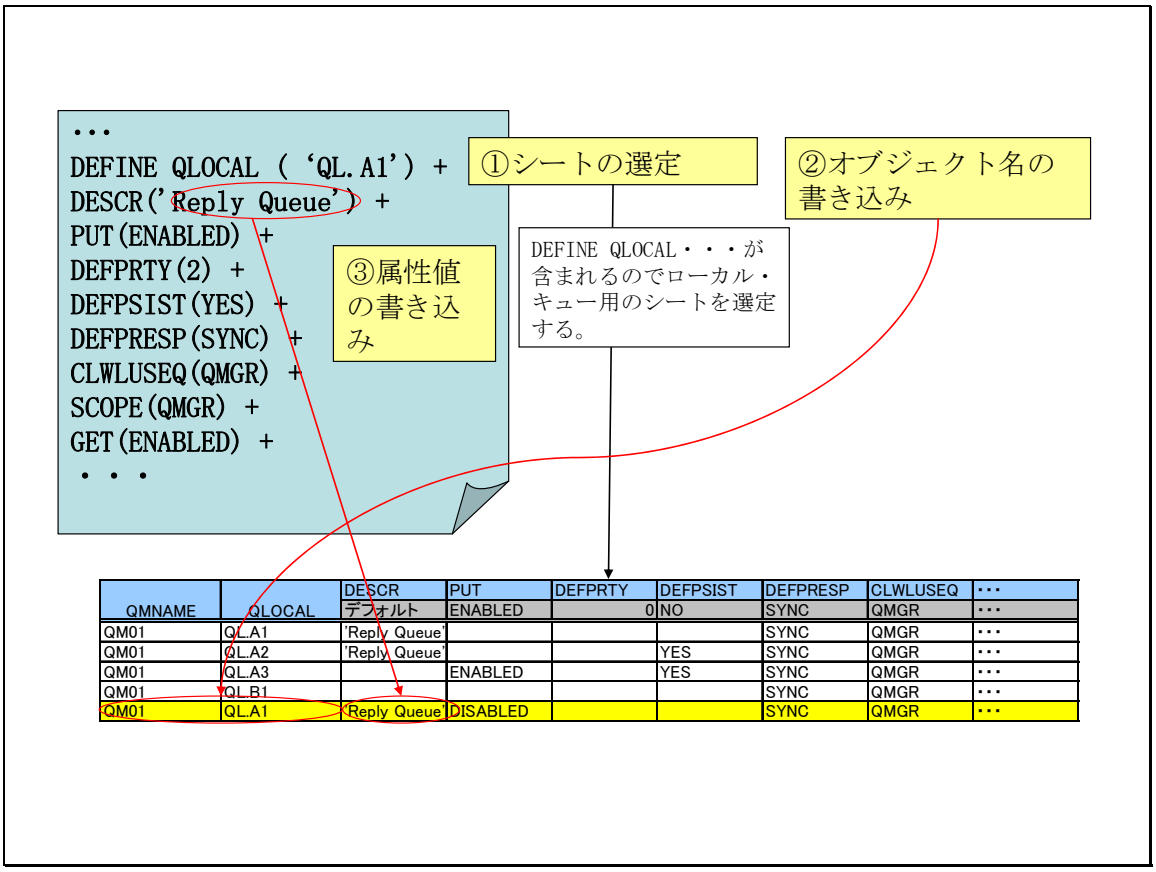


図６．WSH（流し込み）処理概要

（３）MQ 定義ファイル、テンプレートをクローズする

5.2 チェック(VBA)

VBA によるチェック機能の開発概要を示す。

当該処理の概要を図７に示す。

① ソート

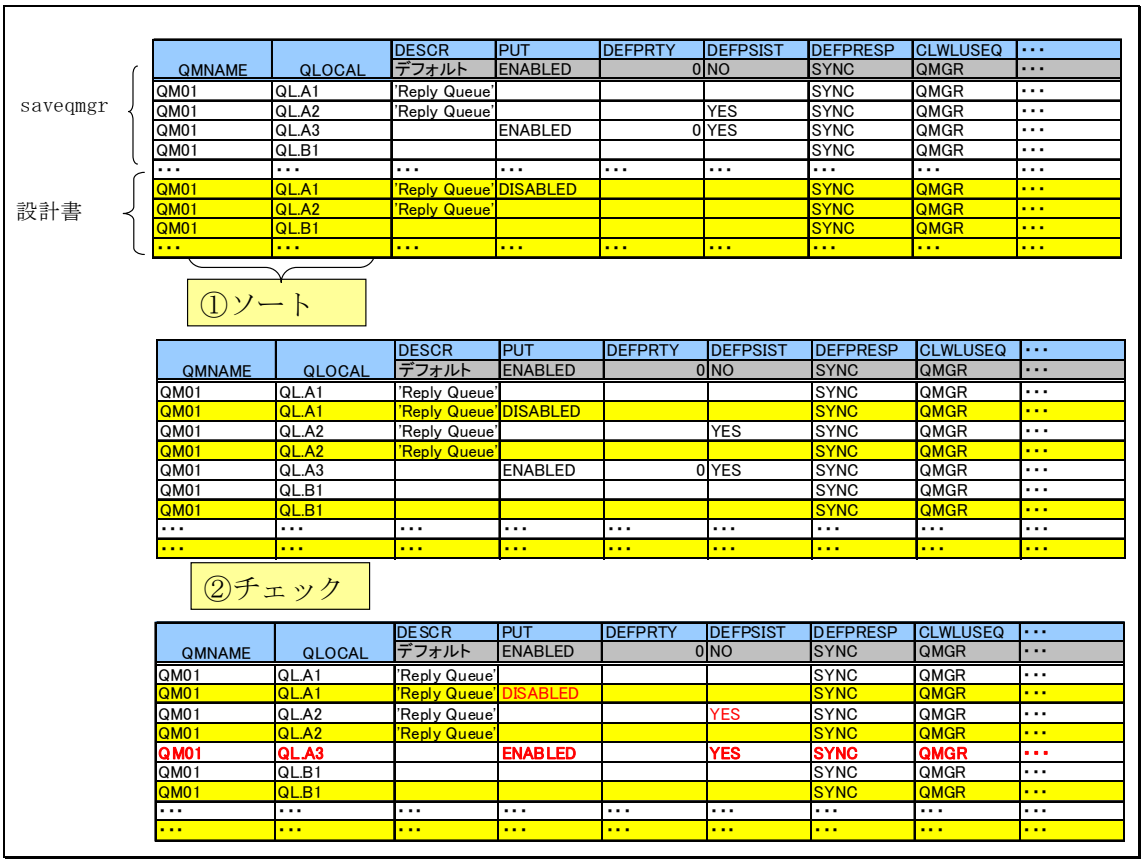
「QMNAME」を第１優先、「QLOCAL」を第２優先として、ソート（並び替え）を行う。

② チェック

ソートしたデータで背景色が白と黄色でペアになれば saveqmgr と設計書どちらもオブジェクトの定義が存在することになる。このペアに対して各属性値を比べて違う場合は文字を赤色にして差異があることがわかるようにする。ただし、デフォルト値と同じものは

赤色に変更しない。

ペアにならない場合、saveqmgr か設計書の片方にしかないため行全体の文字を赤太字に変更して片方にしか定義がないことがすぐにわかるようにする。同一オブジェクト複数定義が存在する場合は、実環境から抽出した saveqmgr では起こらない。設計書からのデータは起こりうる。この場合は、同一オブジェクト複数定義でペアからあふれたオブジェクトに対して、行全体の文字を赤太字に変更する。



QMNAME	QLOCAL	DESCR	PUT	DEFPRTY	DEFPSIST	DEFPRESP	CLWLUSEQ	･･･
		デフォルト	ENABLED	0	NO	SYNC	QMGR	･･･
QM01	QL.A1	'Reply Queue'				SYNC	QMGR	･･･
QM01	QL.A2	'Reply Queue'			YES	SYNC	QMGR	･･･
QM01	QL.A3		ENABLED	0	YES	SYNC	QMGR	･･･
QM01	QL.B1					SYNC	QMGR	･･･
･･･	･･･	･･･	･･･	･･･	･･･	･･･	･･･	･･･
QM01	QL.A1	'Reply Queue'	DISABLED			SYNC	QMGR	･･･
QM01	QL.A2	'Reply Queue'				SYNC	QMGR	･･･
QM01	QL.B1					SYNC	QMGR	･･･
･･･	･･･	･･･	･･･	･･･	･･･	･･･	･･･	･･･

QMNAME	QLOCAL	DESCR	PUT	DEFPRTY	DEFPSIST	DEFPRESP	CLWLUSEQ	･･･
		デフォルト	ENABLED	0	NO	SYNC	QMGR	･･･
QM01	QL.A1	'Reply Queue'				SYNC	QMGR	･･･
QM01	QL.A1	'Reply Queue'	DISABLED			SYNC	QMGR	･･･
QM01	QL.A2	'Reply Queue'			YES	SYNC	QMGR	･･･
QM01	QL.A2	'Reply Queue'				SYNC	QMGR	･･･
QM01	QL.A3		ENABLED	0	YES	SYNC	QMGR	･･･
QM01	QL.B1					SYNC	QMGR	･･･
QM01	QL.B1					SYNC	QMGR	･･･
･･･	･･･	･･･	･･･	･･･	･･･	･･･	･･･	･･･
･･･	･･･	･･･	･･･	･･･	･･･	･･･	･･･	･･･

QMNAME	QLOCAL	DESCR	PUT	DEFPRTY	DEFPSIST	DEFPRESP	CLWLUSEQ	･･･
		デフォルト	ENABLED	0	NO	SYNC	QMGR	･･･
QM01	QL.A1	'Reply Queue'				SYNC	QMGR	･･･
QM01	QL.A1	'Reply Queue'	DISABLED			SYNC	QMGR	･･･
QM01	QL.A2	'Reply Queue'			YES	SYNC	QMGR	･･･
QM01	QL.A2	'Reply Queue'				SYNC	QMGR	･･･
QM01	**QL.A3**		**ENABLED**		**YES**	**SYNC**	**QMGR**	**･･･**
QM01	QL.B1					SYNC	QMGR	･･･
QM01	QL.B1					SYNC	QMGR	･･･
･･･	･･･	･･･	･･･	･･･	･･･	･･･	･･･	･･･
･･･	･･･	･･･	･･･	･･･	･･･	･･･	･･･	･･･

図７．チェック（VBA）処理概要

6. ツールの効果

本章では、MQ 環境チェックツールを適用した結果、修正に対する量・早期対応・正確性についての効果を記載する。また、単体テストへの転用を含めた生産性についても記載する。

6.1 定義項目の修正量、及びパターン

全オブジェクト 6,000 個の内、１割程度は修正した。どのような間違いがあったかを表２に示す。

表２．パターン

パターン	概算比率	補足
属性値の間違い	50%	設計書と属性値が違う。
オブジェクト定義の過不足	30%	長い運用期間の間に実態と設計書が乖離したと考えられる。
オブジェクト名などのスペルミス	20%	設計書のオブジェクト名の区切り文字が「.」１つのところが「..」 と２つになっているパターン、英字でシングルバイトであるべきところが、マルチバイトになっているパターンを含む。 これらはエクセル上では、問題なさそうに見えたが、実際には違っていた。

ツール化により、差異を見落とすことなく間違いを修正することができたと考える。

6.2 単体テストでの作業効果

MQ 構築後の単体テストで構築した MQ が正しく定義されているかどうか、確認する項目を設けたが、ツールを使えば容易に単体テストの一部に置き換えることが可能となった。

これまでの説明では saveqmgr は現行環境から MQ 定義ファイルを抽出する前提としていたが、この処理を環境構築後に saveqmgr で MQ 定義ファイルを抽出する。この置き換えによってツールによる差分チェックを行うと設計書と環境構築した差分がすぐにわかる。そのため、エビデンスとして残せるメリットもあった。

6.3 全オブジェクト定義の正確性

ツールを運用して環境差分チェックした結果、単体テスト後のフェーズでは修正が発生しなかった。アプリケーションチームへ環境を引渡しした後、外部接続の試験の際も修正はなかった。

エクセルにより、視覚的に違いがすぐにわかるメリットがあり、仮に差異が発生した際にも、正しい値を、より早く導けるため、現行の運用担当者などに判断を仰ぐ際に有効な手段となった。

6.4 生産性

ツール開発に関しては、比較的ロジックが単純であったが、３つのシステムで異なる様式があり、同一システムの設計書に関しても微妙に様式が異なっていた。このため、WSH による設計書からの抽出処理は、次の対応を行った。

- ３つの異なるシステム用にオリジナルを開発する。
- 同一システムでの微妙な違いに関しては、汎用的に開発できないレベルだったため、オリジナルをコピーしてカスタマイズする。

WSH による MQ 定義ファイルからの流し込み処理と VBA によるチェック処理はオリジナルのみで良い。

このような開発方法を用い設計フェーズ中にツールを開発したが、ツール開発を想定していない本来の設計フェーズのスケジュール内で吸収できた。

ツールの運用によるチェックは構築フェーズの初期に実施した。ツールを運用してチェックが終了するとMQの環境構築用のMQ定義ファイルもツールから自動生成されたもので構築を行うことにより生産性は上がった。

単体テストの一部もツールを利用することができた。その後のフェーズではMQの環境が正しく構築されていたため、不備を修正するなどの作業は発生しなかった。このためトータルでの生産性は上がったと実感した。

6.5 当初課題は解決

当初課題は、MQの設計書をバージョンアップした上で正しい値で記載することと、正しい環境構築用のMQ定義ファイルを作成することであったが、これまでに述べてきた内容で解決済みである。また、ツールによりトータルで効率化が図れた。

7.他製品への展開

本章では、多くの読者諸氏に本稿で述べたアイデアを応用・活用していただきたいため、MQ環境チェックツールの仕組みが他の製品について展開できるか考察する。技術的な観点で方向性を示すにとどめ、ツールの採否にあたっては、別途、ツールの効果・開発・運用コストについて考慮する前提としている。

7.1 DB2への展開

DB2にはデータベース・オブジェクトを抽出することができるdb2lookコマンドが標準で提供されている。このコマンドは本稿で説明したチェックツールにおいて、MQのsaveqmgrの位置づけに相当する。その他の部分もDB2用に置き換えれば、そのままの仕組みでDB2へ展開できると考える。

7.2 WASへの展開

WebSphere Application Server（以下、WASと記す）には、構成情報を取得できるwsadminというツールが提供されている。saveqmgr、db2lookはコマンドを実行すると一度にオブジェクト定義情報を出力してくれるが、wsadminはプログラムを開発する必要がある。

プログラムの開発については、エクセルのチェックシートに流し込みやすい独自のフォーマットでテキストファイルに出力してテンプレートに流し込むようにすれば良い。補足すると saveqmgr、db2lookの出力結果はそのまま環境構築に使うことが可能であるが、WASのwsadminの場合、出力結果が複雑なため、独自のフォーマットでテンプレートに流し込んだほうが良いと考える。

このような考えを適用すればツール化は可能であると考える。少なくとも過去、別のプロジェクトで、WASの旧バージョンであるが、環境からwsadminコマンドで定義情報を抽出して、テンプレートへ流し込んだ類似の実績はある。この時は、多くのWAS環境で環境間の差異をチェックするため開発

した。設計値を記載したテンプレートへ複数の WAS 環境の設定値を流し込み、差分があると色分けする機能を設けることにより、環境差異がすぐに把握でき、品質面で効果があった。

デメリットとしてツールの開発コストがかかるため、いままで述べた方法に対して、より、コスト見合いを考慮する必要がある。

7.3 UNIX、Linuxへの展開

UNIX、Linux では設計書に記載されている情報の抽出はコマンドの出力結果、及び、ファイルの中身を表示するコマンドの出力結果の組合せであることが考えられる。この結果をいままで述べた方法に適用すれば展開可能であると考える。

7.4 その他について

環境からの情報の抽出と設計書に記載されている情報の抽出ができれば、展開可能であると考える。

8.おわりに

MQ 構築量が突出して多いと意識したのは要件定義フェーズの終盤であった。次フェーズの設計フェーズを意識し始めたころであった。その際に何とかしなければと思い始め、ツールを開発して対応することにしたが、プロジェクトを終えて、早期に気づきアクションを起こしたことが功を奏したと考える。

環境差異は当初想定していたよりも多く、ツールであるからこそ、確実に差異を洗い出し、正しい修正と環境構築が行えた。アプリケーションチームからの環境問い合わせ、外部接続リハーサルの際にすぐに対応できるようにしていたが、結果的に杞憂に終わった。ツール化の一番のメリットは差分チェックが、やったつもりではなく、確実にチェックできたことである。

今後のプロジェクトにおいては、今回のような差分チェックのニーズがある可能性もあると考える。また、ビッグデータ化に伴いデータベースの設計なども、より大きく複雑化している。このようなケースにおいても、今回、提示したツールの構築方法の一部で VBA などを使って、データベースの物理設計支援ツールと構築用のデータベース・オブジェクト作成ツールなどにも応用できると考える。

ツールの課題として、設計書から抽出するロジックでは、プロジェクトごとに設計書の書式が異なるため、そのまま再利用できない。このような場合はツール自体の資産、ノウハウの蓄積により、カバーすることもできると考える。

どのみち、今後はツールを積極的に取り込む姿勢と開発するノウハウの必要性が迫っていると考える。なぜなら、昨今、仮想化、クラウド化が本格化してきている状況で幅広い技術が要求される一方で、既存の定着した技術はコストを抑えて品質が要求されてきている。さまざまなケースに対応する必要があり、本稿で示したようなニーズがでてくる場合があると考える。こういった状況の中、今回示した内容が参考になれば幸いである。