ITAが「Internal Integration Testing（内部結合テスト）」を指す場合、運用管理スクリプトのテスト結果報告書は、内部モジュールやコンポーネント間の連携が正しく機能するかを検証する内容に焦点を当てます。以下に、内部結合テスト（ITA）に特化したテスト結果報告書のポイントとサンプルを、運用管理スクリプト向けに再構成して提示します。ポイントは前回の回答を基に、内部結合テストの特性を強調し、簡潔にまとめます。

**内部結合テスト（ITA）報告書の作成ポイント**

内部結合テストは、運用管理スクリプトの各モジュール（例: ログ収集モジュール、通知モジュール、データベース接続部など）が正しく連携し、期待通りの結果を出すかを検証します。以下のポイントを考慮します：

1. **テストの目的を明確化**
   * 内部結合テストの目的を明記。例: 「運用管理スクリプトの各モジュールが正しく連携し、エンドツーエンドの運用タスクを正確に実行することを検証する。」
   * 運用管理スクリプトの具体的な機能（例: サーバー監視、バックアップ実行）に焦点を当てる。
2. **テスト範囲と環境**
   * テスト対象のモジュール（例: Ansible Playbook内のタスク、シェルスクリプトの関数）とその依存関係を明確化。
   * テスト環境（OS、ミドルウェア、ITAツールのバージョンなど）を詳細に記載。
   * 例: 「テスト対象: ログ収集モジュールと通知モジュールの連携、環境: Ubuntu 20.04、Ansible 2.9.27」
3. **テスト観点**
   * 内部結合テスト特有の観点を設定：
     + **モジュール間通信**: データの受け渡しやAPI呼び出しが正しいか。
     + **データ整合性**: モジュール間で処理されたデータが期待通りか。
     + **エラーハンドリング**: モジュール間の異常時（例: 接続エラー、データ欠損）の動作。
     + **順序性**: モジュールの実行順序が正しいか。
   * 運用管理スクリプトの場合、運用タスクの連続性（例: 監視→通知→ログ記録）が重要。
4. **テスト結果のサマリ**
   * テストケースの総数、成功/失敗率、重大な不具合の有無を簡潔に記載。
   * 例: 「テストケース総数: 30、成功: 28、失敗: 2、不具合: 1件（軽微）」
5. **詳細なテスト結果**
   * テストケースごとに、ID、内容、入力値、期待値、実際の結果、不具合の詳細を表形式で記載。
   * 内部結合テストでは、モジュール間のデータフローや依存関係を明確に示す。
6. **不具合の分析と対応**
   * 不具合の原因（例: モジュール間のデータフォーマット不一致）と対応策を記載。
   * 例: 「通知モジュールがログモジュールからのデータを受け取れずエラー。原因: JSONパースエラー。対応: 次バージョンでデータフォーマット統一。」
7. **視覚的要素**
   * テスト結果の成功/失敗率やモジュール間の依存関係をグラフや表で視覚化。
   * 例: モジュール間のデータフローを図示、成功率を円グラフで表示。
8. **運用管理スクリプト特有の考慮**
   * 運用管理スクリプトは本番環境での安定性が重要。内部結合テストでは、本番シナリオ（例: 高負荷時、ネットワーク断）を模擬したテスト結果を強調。
   * モジュールの再利用性や保守性を考慮した提案を記載。

**テスト結果報告書のサンプル構成**

以下は、運用管理スクリプト（例: Ansibleを使用したサーバー監視スクリプト）の内部結合テスト（ITA）向け報告書のサンプルです。簡潔さと明確さを重視し、内部結合テストの観点を反映しています。

**内部結合テスト（ITA）結果報告書: サーバー監視スクリプト**

**1. 概要**

* **テスト目的**: Ansibleを用いたサーバー監視スクリプトのモジュール（監視、通知、ログ記録）が正しく連携し、システムリソースの監視と異常時アラートを正確に実行することを検証する。
* **テスト対象**:
  + モジュール: 監視モジュール（CPU/ディスク監視）、通知モジュール（メール送信）、ログ記録モジュール
  + スクリプト: server\_monitor.yml（Ansible Playbook, v1.0）
* **テスト期間**: 2025年5月10日～2025年5月12日
* **テスト環境**:
  + OS: Ubuntu 20.04
  + Ansible: v2.9.27
  + サーバー: 仮想マシン（2CPU, 4GB RAM, 50GBディスク）
  + ネットワーク: 内部LAN（帯域1Gbps）
* **実行者**: テストチーム（山田一郎）

**2. テスト結果サマリ**

* テストケース総数: 30
* 成功: 28ケース（93.3%）
* 失敗: 2ケース（6.7%）
* 不具合検出: 1件（軽微）
* 結論: 軽微な不具合を修正後、運用移行可能

**3. テスト項目と結果**

| **テストケースID** | **テスト内容** | **入力値** | **期待値** | **結果** | **備考** |
| --- | --- | --- | --- | --- | --- |
| ITA001 | 監視→通知連携 | CPU使用率90% | メール通知送信 | 成功 | - |
| ITA002 | 監視→ログ連携 | ディスク使用率95% | ログに記録 | 失敗 | JSONパースエラー |
| ITA003 | 通知→ログ連携 | 通知実行 | 通知ログ記録 | 成功 | - |
| ... | ... | ... | ... | ... | ... |

**4. 不具合の詳細**

* **不具合ID**: ITA\_NG001
* **内容**: 監視モジュールからログ記録モジュールへのデータ受け渡しでJSONパースエラーが発生。
* **原因**: 監視モジュールの出力フォーマットがログモジュールの期待値と不一致。
* **対応**: 次バージョン（v1.1）でデータフォーマットを統一。
* **影響**: 軽微（本番環境では事前フォーマットチェックで回避可能）。

**5. 分析と推奨事項**

* **分析**: モジュール間の基本的な連携は正常。データフォーマットの標準化が必要。
* **推奨事項**:
  + モジュール間のデータフォーマットをJSON Schemaで定義。
  + テストケースに異常系シナリオ（例: ネットワーク断）を追加。
  + ログ記録の冗長性を向上（例: ファイルとDBへの同時記録）。

**6. 付録**

* モジュール間データフロー図（/docs/data\_flow.pdf）
* テストログ（/logs/ita\_20250512.log）

**視覚的要素の例（円グラフ）**

テスト結果の成功/失敗率を視覚化する円グラフを以下に示します。

Grokは間違いを犯す可能性があります。元のソースを常に確認してください。ダウンロード

この円グラフは、成功率（93.3%）と失敗率（6.7%）を視覚化し、緑（#4CAF50）と赤（#F44336）で視認性を確保しています。

**補足と注意点**

* **内部結合テストの特化**: 内部結合テストでは、モジュール間のインターフェースやデータフローに重点を置くため、テストケースはモジュール間の依存関係やデータ受け渡しを詳細に検証するように設計してください。
* **運用管理スクリプトの特性**: スクリプトが本番環境で安定動作することを保証するため、内部結合テストでは本番シナリオ（例: 高負荷、障害時）を模擬したテストを重視。
* **フォーマットのカスタマイズ**: プロジェクトの要件に応じて、報告書のフォーマットや詳細度を調整。例: ステークホルダーが非技術者の場合、専門用語を簡略化。
* **Web情報に基づく補強**: Web上のテスト報告書テンプレート（例: NotePM、テスト管理ツールのガイドライン）を参考に、モジュール間連携の検証に特化した項目を追加。

もし具体的なモジュール構成やテストケースの詳細、または他のグラフ（例: モジュール間依存関係図）を追加したい場合、詳細をお知らせください。また、特定のフォーマットやツール（例: Word、PDF）での出力が必要な場合もご指定いただければ対応します！