

投稿日 2022/12/22

障害テスト計画ガイド

概要

本書は、システム開発プロジェクトの障害テスト計画で検討すべきトピック、および障害テストを推進・実施する上で考慮すべきポイントを解説するものです。障害テストを計画・実施する意義への理解を促進するとともに、障害テスト計画の属人化を回避することを目的としています。本書は主にウォーターフォールでWebアプリケーションを開発するプロジェクトで活用できます。なお、本書の一部内容は参考文献『[全体テスト計画ガイド](#)』と『[テスト種別&観点カタログ](#)』、『[非機能要求グレード](#)』を利用することを前提としています。

また、『[障害テスト計画書サンプル](#)』にて、本書をもとに作成した障害テスト計画書のサンプルを用意しています。こちらに合わせてご参照ください。

目次

- [1. 本書について](#)
- [2. 障害テストについて](#)
- [3. 検討が必要なトピックの解説](#)
- [4. 参考文献](#)

1. 本書について

本書（障害テスト計画ガイド）は、障害テスト作業全体の計画を検討するためのガイドです。全体テスト計画では取り扱われない、障害テスト計画固有のトピックを中心に解説するとともに、障害テストの実施におけるポイントについても触れていきます。本書で取り扱う「障害テスト計画」とは以下の作業を指します。

- プロジェクトにおける障害テストの方針を決定する。
- 障害テストを行うための環境の要件を決定する。

1.1. 目的

本書の目的は以下の通りです。

- 障害テストを計画・実施する意義が理解されること。
- 障害テスト計画で検討すべき事項、および障害テストで実施する作業内容を定義するとともに、それらについて考慮すべきポイントを解説することで、障害テスト計画の属人化を回避すること。

1.2. コンセプト

本書のコンセプトは以下の通りです。

- プロジェクト固有の事情に依存せず、横断的に利用できること。
- テストの中でも障害テスト計画固有のトピックを中心に解説すること。

1.3. 対象

1.3.1. 想定読者

主要なターゲットとして、下記を想定しています。

- 障害テスト計画の策定、またはレビューを担当する人。
- 障害テスト実施を担当する人。

ただし、テストケース作成やテスト実行の経験（障害テスト以外でも可）があることや、アプリケーションプログラムの実装、およびサーバ作業の経験があることを前提としています。

1.3.2. 対象とするプロジェクト

本書は、ウォーターフォールでクラウドまたはオンプレミス環境にWebアプリケーションを新規構築するプロジェクトを前提としています。とくに複数サーバや複数サービスが協調して動作する分散システムでは、耐障害性を作り込むために障害テストは重要であり本書が有効です。

本書は新規開発プロジェクトを前提として記載しますが、障害テストの実施方針等はローンチ後のシステムの変更やシステムを取り巻く環境の変化に対して障害耐性を維持・強化するためにもご活用いただけます。

1.3.3. 取り扱わない内容

本書では下記についてはスコープ外として取り扱わないものとします。

- 「プロジェクト計画書」に詳細が記載される傾向にあるトピックの詳細。たとえば、進捗管理、品質管理などの管理方針は「プロジェクト計画書」に記載されることが多いため、本書では説明しない。
- 見積りや品質の指標についてはプロジェクト毎に過去実績値などを参考に検討するものとし、本書には記載しない。
- 『[全体テスト計画ガイド](#)』に記載のある内容については冗長となるため、本書には記載しない。

1.4. 活用シーン

本書は以下のようなシーンで活用できます。

- 全体テスト計画で検討した障害テスト方針が問題ないか、実施可能かどうかを評価するとき。
- 実施可能な障害テスト計画を考えるために必要な検討事項や、あらかじめ収集が必要な情報を確認するとき。
- 障害テスト計画書の構成や内容を決定するとき。
- 障害テスト計画書の作成後、セルフチェックやレビューにて、記載内容の過不足を確認するとき。
- 障害テストに関わったことのない人が内容について学習するとき。

2. 障害テストについて

2.1. 障害テストとは

障害テストとは、広義の意味では言葉の通りシステムが障害に見舞われた際のふるまいを検証するテストです。しかし、障害と言ってもさまざまな要素があり、障害テストが何を指すのかはあいまいになりがちです。本書における障害テストとは、『[テスト種別&観点カタログ](#)』にて定義している狭義の意味での障害テストを指しています。具体的には、本番環境で発生し得る障害に対して以下により可用性要件を満たすかを検証し問題を発見することを指します。

- システムが正しく動作すること
- 自動復旧機能または手動復旧手順によりシステムが回復すること

以下の確認事項は含みません。確認を想定するテスト種別と合わせて記載します。

障害テストに含まない確認事項	確認を想定するテスト種別
・縮退環境で性能要件を満たすこと	性能テスト
・障害によるアラートの発報とそれに伴う関係者への伝達フロー ・メンテナンス等意図的なシステム停止	運用シナリオテスト
・関連サービスの異常レスポンス（400系、500系）に対するシステムの動作※1	機能テスト
・システムの限界を超えた負荷となるようにアクセスをした時の動作	ストレステスト
・システムに限界負荷となる量のデータを投入した時の動作	ボリュームテスト
・システムに長時間の負荷をかけた時の動作	ロングランテスト

※1 関連サービスを呼び出すケースにおいて、呼び出し先で障害が発生した時にその影響が自システムに及ばないことを確認することは非常に重要です。そのため、関連サービスのレスポンス遅延やタイムアウト時の自システムのふるまい確認は障害テストで行います。一方、異常レスポンスのステータスコードに対するふるまいの網羅的な確認は、テスト効率性観点から、スタブを利用し機能テストの中で実施するものとします。

また、本書ではインフラの構築とインフラの各種テスト（単体・結合障害）が完了した状態でアプリケーションの動

作を確認する障害テストを対象とします。インフラの各種テストは取り扱いません。オンプレミス環境において、故障によるネットワーク機器やストレージ等のハードウェアの交換についても、インフラテストの範疇として、本書の対象外とします。

障害に対して、システムが自動で切り替わったり回復したりするふるまいの確認やあらかじめ準備した手順の実行によるシステムのリカバリまでを対象とします（ディザスタリカバリ環境への切り替えを含む）。サービス関係者への報知などシステム外の運用やふるまいの確認は取り扱いません。

2.2. 障害テストで実施すること

障害テストでは、大きく分けて以下の作業が発生します。

- 障害テスト計画策定
- テスト環境の準備
- 外部連携システムとの調整
- 障害テストシナリオ作成
- 障害テストケース作成
- 障害テストデータ作成・投入
- テストツールの準備
- 障害テスト実施
- 障害テスト結果の評価
- 障害テストの課題解消
- テストの再実施と再評価

上記作業には他のテストでも実施することや共用すること含まれますが、すべてにおいて障害テスト固有で考慮すべき観点があります。そのため、障害テスト計画では多くのトピックで障害テスト固有の観点が要求され、属人化しやすい傾向にあります。本書ではそういった観点を記載することで、属人化を回避することを目的としています。より具体的なタスクについては『[3.5. タスクと役割分担](#)』にて一覧化しています。

2.3. 障害テストの意義

障害テストを行う意義は、本番稼働時に近い状況下でシステムの動作確認をすることです。機能テストにより、機能要件通りにアプリケーションが動くことは保証されます。しかし、機能テストは障害に関するテストではないため、システムに障害が発生した時の動作保証まではできません。発生する障害の種類、障害が発生したときのアプリケーションの状態やアプリケーションが稼働するインフラ環境の状態によってシステムの動作は異なります。このため、障害を想定し設計することは非常に難易度が高い作業であり、機能テストでは検出できなかった不具合を検出することがあります。このような不具合は実際に本番相当の環境でテストを行い、本番で発生し得る障害をシミュレーションしなければ検出できません。また、このような不具合が本番稼働後に発生してしまうと、最悪の場合は長時間サービスが使えない状態となり、事業に多大な影響が出てしまいます。そのため、障害テストを実施し、本番稼働時にもシステムが正常に動くことを検証する必要があります。

耐障害性を作り込む意義とテクニックについては、「[システムへの耐障害性を高めるためのアプローチ](#)」を参照してください。

2.4. 障害テスト計画の意義

障害テストを計画的に進めていかないと以下のような問題を招く可能性があります。これらの問題により障害テストが十分に実施できなくなってしまうと、最悪の場合システムをリリースできなくなる可能性があります。また、リリースできたとしてもリリース後に発生する障害によって長時間のサービス停止などの大規模障害を招く可能性さえあります。

発生する問題	説明
障害テスト実施に必要な作業が漏れてしまう	障害テストの準備は時間がかかるものが多く、作業漏れの検知が遅れるとリカバリが効かなくなってしまうリスクが高い。
関連サービスで発生する障害がテスト出来ない	障害テストの観点として関連サービスにレスポンス遅延やタイムアウトが発生したときなど非機能面のふるまいを確認する必要がある。場合によっては関係会社との調整が必要になるため、実施方針・タイミング等を合意し計画的に進める必要がある。
障害テストの実施結果を評価する方針がぶれてしまう	障害テストの実施結果はサーバリソースの使用状況や、多数のリクエストに関する応答時間や応答結果等、膨大なデータとなる。これらを生データのままで評価することは難しく、統計をとったり、可視化したりすることが多い。ツールの導入や作成等の準備が必要となるため、計画的に進める必要がある。
障害テストの実施方針に関してステークホルダー間で認識齟齬が発生してしまう	障害テストで誰がどこまで検証するのか、どうやって検証するのか、実施や評価のタイミングで認識齟齬が発生してしまうと、テストの実施ができなくなったり、テストをやり直さなければならなくなったりといった問題が発生する。障害テストの実施にあたり、その意義と実施方針をステークホルダー間で合意する必要がある。

このような問題を未然に防ぐため、障害テストの計画を立案し、計画的に作業を進めるとともに、ステークホルダーと合意形成を行っていくことが重要となります。合意形成を行うステークホルダーは外部関係者（ベンダーなど）、外部接続のテストを行う際の調整先、テストチームなどプロジェクトにより異なります。

2.5. 検討時期

障害テスト計画は、各作業を実施するよりも早く立案する必要があります。障害テストで実施すべき作業は『[3.5. タスクと役割分担](#)』にて解説しますが、障害テストをするための環境構築が必要となるため、要件定義や設計工程から作業が発生します。そのため、それに伴って障害テスト計画も要件定義時点から始めていく必要があります。具体的には、可用性要件が決まった段階から計画の立案を始めるのが理想的です。

2.6. 検討事項

障害テスト計画の策定において必要な検討事項は以下のとおりです。

章番号	トピック	概要
-----	------	----

章番号	トピック	概要
3.1.	障害テスト方針	障害テスト計画を立案する上での、基本的な考え方・方針および前提事項の検討
3.2.	障害テストの実施方針	障害テストの実施や準備における方針の検討
3.3.	テスト環境	障害テストを実施する環境要件の検討
3.4.	体制	障害テストの推進体制・役割の検討
3.5.	タスクと役割分担	障害テストのタスクと役割分担の検討
3.6.	スケジュール	障害テストのスケジュールの検討

3. 検討が必要なトピックの解説

3.1. 障害テスト方針

このトピックでは、障害テストに関わる要件・制約を踏まえて、障害テスト全体に関わる方針を検討します。検討においては、全体テスト計画で検討した内容をベースに、障害テスト固有の要件や制約を考慮します。このテスト方針をベースに詳細な内容を検討します。

3.1.1. 障害テストに関わる初期条件

プロジェクト計画書や非機能要件定義書、全体テスト計画書などから、障害テスト計画の立案に必要な可用性要件・制約事項を初期条件として収集・整理します。

障害テストでは要件定義で定められた可用性要件がインプットとして必要になってきます。プロジェクトによっては、SLO（Service Level Objective）やSLA（Service Level Agreements）として定義したサービスレベルもインプットとなります。具体的には、参考文献『[非機能要求グレード 2018 活用シート](#)』に大分類「可用性」として記載されているような項目が定義されている必要があります。可用性要件が要件定義にて決まっていないと障害テスト計画が立案できないため、早急に定義してください。

以下に『非機能要求グレード2018 活用シート』に大分類「可用性」中分類「継続性」として記載されている要求レベルの定義例を示します。

非機能要求グレードに対する要求レベルの定義例

小項目	小項目説明	メトリクス（指標）	要求レベル	項番※2
-----	-------	-----------	-------	------

小項目	小項目説明	メトリクス（指標）	要求レベル	項番※2
運用スケジュール	システムの稼働時間や停止運用に関する情報。	運用時間（通常）	24時間無停止	A.1.1.1
		運用時間（特定日）	2時間程度の停止有り（時間帯は都度調整）	A.1.1.2
		計画停止の有無	計画停止有り（運用スケジュールの変更可）	A.1.1.3
業務継続性	可用性を保証するにあたり、要求される業務の範囲とその条件。	対象業務範囲	外部向けオンライン系業務	A.1.2.1
		サービス切替時間	10分未満	A.1.2.2
		業務継続の要求度	単一障害時は業務停止を許容せず、処理を継続させる	A.1.2.3
目標復旧水準（業務停止時）	業務停止を伴う障害が発生した際、何をどこまで、どれ位で復旧させるかの目標。	RPO（目標復旧地点）	障害発生時点（日次バックアップ+アーカイブからの復旧）	A.1.3.1
		RTO（目標復旧時間）	1時間以内	A.1.3.2
		RLO（目標復旧レベル）	特定業務のみ	A.1.3.3
目標復旧水準（大規模災害時）	大規模災害が発生した際、どれ位で復旧させるかの目標。	システム再開目標	3日以内に再開	A.1.4.1
稼働率	明示された利用条件の下で、システムが要求されたサービスを提供できる割合。	稼働率	99.9%	A.1.5.1

※2「非機能要求グレード2018 活用シート」に記載されている項番。

これらの要件については、以下のようなパターンそれぞれについても一律で定義するのか、それともパターンごとに定義するのか、合わせて要件定義で決定する必要があります。

- 基盤のみとアプリケーションを含めたものを分けて定義するのか
- サービスごとに定義するかサービス全体で定義するのか
- 機能（重要な機能とそうでない機能）を分けて定義するのか

可用性として要求された項目をどこまで確認するかについても定義する必要があります。ここが明確になっていない場合、テスト実施範囲に関する方針の定義ができなくなります。

また、障害テストの制約事項として、どれだけのコストが割り当てられているのかについても確認する必要があります。「[2.2. 障害テストで実施すること](#)」に記載したとおり、障害テスト固有で考慮すべき観点は多いうえに、方針によるコスト変動が大きく、準備にも時間がかかります。そのため、障害テストのコストを正しく見積

度は高く注意が必要です。割り当てられたコストが現実的なものであるかを確認し、そのコスト内で障害テストを実施することが困難だと判断される場合には、プロジェクト内でのエスカレーションが必要となります。

他にも、後述の『[3.3. テスト環境](#)』にて障害テストに必要な環境の要件を検討する際のインプットとなる、テスト環境に関する制約事項も併せて確認するようにしてください。具体的には、たとえば以下について確認する必要があります。

- 外部システムを含めた各環境はどの時期から使えるようになるのか
- 各環境はどのような構成となっているのか

3.1.2. 品質・コスト・スケジュール・テスト実施範囲・網羅性に関する方針の定義

テストの網羅率は品質面からは100%が理想的ですが、コスト・スケジュールとの兼ね合いから、リスクの低いパターンについては目標とする網羅率を100%から下げることがあります。

品質・コスト・スケジュール・テスト実施範囲・網羅性に関する方針を検討します。基本的な考え方は『[全体テスト計画ガイド](#)』の『[4.1. テスト方針](#)』をご参照ください。ここでは、障害テスト固有の考慮点について解説します。システムの耐障害性品質や、どの機能についてテストするのかについてはすでに可用性要件として決まっているので、ここでは可用性要件に対して、どこまで保証するのかを検討します。テスト品質としては100%の網羅率とするのが理想的ですが、コスト・スケジュールとの兼ね合いから、リスクの低いパターンについては目標とする網羅率を100%から下げることがあります。

具体的には以下のような網羅性についてどこまでテストするのかを検討します。

テスト対象機能に対する網羅性

複雑なシステムでは、すべての機能に対して本番で発生し得る障害ケースをテストしようとするると多大なコストを費やすことになります。複数の機能が協調して動作することにより発生する状況は無数に存在し、それぞれのタイミングですべての障害ケースを想定することは非常に困難です。計画時点で、「想定する障害ケース」と「テストする機能」を決定します。それぞれの観点の例を示します。

想定する障害ケースを選ぶ観点

- 発生リスクが高い障害を選ぶ
- リカバリが困難な障害を選ぶ
- 影響範囲が大きい障害を選ぶ

対象とする機能を選ぶ観点

- 処理方式（登録、更新、検索、etc.）ごとに代表機能を選ぶ
- ユーザーからのアクセスが多い機能を選ぶ
- リカバリが困難な機能を選ぶ
- 障害による影響範囲が大きい機能を選ぶ

また、連携する外部システムが存在する場合、外部システムの障害をテスト実施範囲に含めるかどうか合わせて検討する必要があります。

外部システムがすでに稼働中のシステムで、テスト環境が存在しない、あるいはテスト環境で想定する障害をシミュ

レートできない場合、外部システムを含めた障害テストが実施できません。このような場合はシミュレーターによって外部システムとの処理をダミー化することも選択肢の1つとなります。 外部システムからのレスポンスに対して網羅的にテストする部分は機能テストで実施するなど、検討漏れのないよう他のテストとの棲み分けについても注意が必要です。

外部システムの障害により応答待ちのコネクションが滞留し呼び出し元システムに障害が波及することもあるため、テスト実施範囲は慎重な検討が必要です。

3.2. 障害テストの実施方針

このトピックでは障害テストの実施方針について検討します。 『[3.1. 障害テスト方針](#)』で検討した方針を踏まえ、どのように障害テストを実施していくのか検討します。 具体的には以下のトピックについて検討してきます。

- 障害テスト観点
- 障害テストシナリオの方針
- 障害テストデータ方針
- テストツールの導入

3.2.1. 障害テスト観点

障害テストでは、システムが可用性要件を満たしているかどうかという観点で検証します。 このトピックでは、どういった障害を想定し何を持って可用性要件を満たしていると評価するのか検討します。

障害が発生し得るレイヤー（障害発生箇所）ごとにシミュレーションする障害の種類を検討します。以下に例を示します。

障害が発生し得るレイヤーごとのシミュレーションする障害の例

障害が発生し得るレイヤー（障害発生箇所）	シミュレーションする障害	代替手段でシミュレーションする障害※3
ソフトウェア（アプリケーション・データベース・各種ミドルウェアなど）	プロセス停止	－
OS	OS停止	－
仮想サーバ	サーバ停止	OS停止
ハードウェア（機器、ストレージ、サーバ）	ハードウェア故障	OS停止
	CPU負荷高騰	－
	メモリ消費量高騰	－
	ファイルI/O負荷高騰	－
ネットワーク（回線）	ネットワーク故障	ネットワーク遮断

障害が発生し得るレイヤー（障害発生箇所）	シミュレーションする障害	代替手段でシミュレーションする障害※3
	ネットワーク遅延	－
	パケットロス	－
データセンター（AZ、リージョン）、ファシリティ	データセンター停止	OS停止
	ネットワーク故障	ネットワーク遮断
SaaS（アプリケーション・データベース）、PaaS（ミドルウェア・OS）、IaaS（仮想サーバ）	関連サービス停止	－

※3 想定障害を別の障害で代替する場合のシミュレーションする障害を記載。たとえば、仮想サーバ・ハードウェア・データセンター・ファシリティの停止はOS停止にて代替しシミュレーションする。

サーバやストレージ等のハードウェア障害は実際に障害を起こしてテストすることが困難な場合もあります。その場合は、1つ上のレイヤーで仮想的な障害をシミュレーションすることも検討します。たとえば、突然電源が落ちるなどの物理サーバの停止はサーバ上で稼働するOSの停止によってテストするなどです。その際、同じ筐体上の仮想サーバは同時に停止するなどの同時に起こりうる障害の組み合わせにも注意が必要です。

次に可用性要件に対する測定方法の例を示します。実際のプロジェクトでは以下の例を参考に、システムの構成や使用するテストツールに合わせて確認方法を検討してください。また、ログ等を生データのまま目視で確認することは難しく、確認漏れの原因となるため、評価方法として、測定したデータをグラフ等に可視化して評価すること推奨します。

可用性要件に対する障害テストでの評価方法の例

メトリクス（指標）	障害テストでの評価方法の例
サービス切替時間	アプリケーションへ継続的にリクエストを送信した状態でシステムへ想定障害をシミュレーションし、リクエスト・レスポンスのログより業務再開までに要する時間を算出する。
RTO（目標復旧時間）	サービス切替時間と同様
RPO（目標復旧地点）	アプリケーションへ継続的にリクエストを送信した状態でシステムへ想定障害をシミュレーションした後、あらかじめ用意した手順によりシステムを復旧する。復旧後のデータ状態とリクエストログを比較し、想定する地点（たとえば、障害が発生する直前のトランザクション）と一致するかを確認する。
データ整合性	測定する地点と確認対象のデータを明確にした上で、データの整合性を確認する。たとえば、確認対象データがリクエストに対するレスポンスとデータベースの状態の場合は、リクエストログから想定するデータの状態を導出しデータベースと比較する。
稼働率	年間動作可能時間／（年間動作可能時間＋年間停止時間）を算出する。たとえば、年間停止時間は障害発生箇所ごとにサービス切替時間を算出し障害ポイント数・障害影響範囲・想定する障害発生回数などから導出する。

メトリクス（指標）	障害テストでの評価方法の例
サービス影響範囲	想定障害をシミュレーションし各サービスにリクエストを送信することで障害時のサービス利用可否を確認する。

また、障害テストでは評価のための情報だけでなく、可用性要件を満たせないなどの問題が発生した際の解析に使う情報も収集し、確認する必要があります。主な確認項目と確認観点、測定方法の例を以下に示します。一定時間アプリケーションへ継続的にリクエストを送信し、以下の項目を測定します。一定時間経過したあと障害をシミュレーションし、障害シミュレーション前の定常状態と障害シミュレーション前後の状態を比較します。

障害テストでの確認項目の例

確認項目	測定方法の例
レイテンシ	「リクエストの処理にかかる時間（ms: millisecond）」を測定する。 ・平均・最小・最大・パーセンタイル・標準偏差等で評価する。 ・必要に応じて失敗と成功のリクエストを区別して評価する。
トラフィック	「1秒間に処理を行ったリクエスト数（rps: Request Per Second）」を測定する 必要に応じて機能特性（静的／動的コンテンツ、外部接続有無など）ごとに分類して計測する。
エラー	リクエストに対するレスポンスから「失敗リクエストの割合（％）」を算出する。 エラーは以下に分類できる ・明示的错误：HTTPステータ500系（400系はクライアント側の問題としてエラーに含まない） ・暗黙的错误：HTTPステータス 200系 だがBody部が不正 ・ポリシー違反：レスポンス時間が目標時間オーバー
サチュレーション	テスト対象サーバから以下のようなメトリクスを取得・測定する。※とくにクリティカルなリソースにフォーカスすると良い。 ・「CPU使用率」 ・「メモリ使用率」 ・「ネットワーク帯域使用量」 ・「ディスクI/O」

これらの情報は問題が発生した際の解析だけでなく、問題の予兆検出や障害に対して「予測が難しいシステムのふるまい」の理解にも役立ちます。

3.2.2. 障害テストシナリオの方針

障害テストのシナリオは「障害発生レイヤー」「障害発生箇所」と発生する「障害の種類」を変数として検討します。以下に例を示します。

障害テストシナリオの例

N o	障害発生レイヤー	障害発生箇所	障害の種類	シナリオ
1	ソフトウェア：	アプリケーション	プロセス停止	オンライン処理（データベース登録）

N o	障害発生レイヤ ー	障害発生箇所	障害の種類	シナリオ
	アプリケーション	ン#1		中にアプリケーション#1上で稼働するアプリケーションのプロセスを停止する。
2	ソフトウェア： データベース	データベース マスター	DBインスタンス停止	オンライン処理（データベース照会系）を実行中にデータベース マスターを停止する。

各障害シナリオに対してユーザーがどの機能を利用しているときに障害が発生することを想定するかを示す「対象機能」と、テストの「実施手順」「合格条件（影響）」を定義します。テストの実施手順は障害のシミュレーション方法と障害が発生したときの復旧方法を具体的に記述します。復旧方法は運用設計および障害設計に定義されるシステムの自動的な回復および、自動的な回復が不可能な場合の回復手順を含みます。合格条件は、可用性要件とそれに対する耐障害性設計を元に具体的なシステムへの影響とシステムのふるまいを想定し条件設定する必要があります。 これらを定義できない場合は、障害に対する対策や耐障害性設計を十分にできていない可能性がありますので再度検討が必要です。

最後に、テスト実施結果を入力する欄も準備しておきます。以下に例を示します。

障害テストケースの例

シ ナ リ オ No	ケ ー ス No	障害 発生 レイ ヤー	障害 発生 箇所	障 害 の 種 類	対 象 機 能	実施手順（障害シミュレーションと復旧方法）※4	合格条件（影響）	実 施 結 果	実 施 者	実 施 日	検 証 結 果	検 証 者	検 証 日
1	1-1	ソフトウェア： アプリケーション	アプリケーション#1	プロセス停止	会員登録	Chaos Toolkitのシナリオ「app_process_down_1.json」により以下を実行 ・JMeterのシナリオ起動 ・5分経過後、障害シミュレーション	<input type="checkbox"/> サービス切替時間が10秒以内であること <input type="checkbox"/> 影響範囲がサービス切替中のリクエストの3分の1程度であること						
2	2-1	ソフトウェア： データベース	データベースマスター	インスタンス停止	会員照会	Chaos Toolkitのシナリオ「db_process_down_1.json」により以下を実行 ・JMeterのシナリオ起動 ・5分経過後、障害シミュレーション	<input type="checkbox"/> サービス切替時間が10秒以内であること <input type="checkbox"/> 影響範囲がサービス切替中のリクエストの3分の1程度であること						
3	3-1	データセンター：	東京リージョン	リージョ	ポイ ント	Chaos Toolkitのシナリオ「region_down_1.json」により以下を実行	<input type="checkbox"/> 2時間以内に復旧できること <input type="checkbox"/> データベースのテーブル間の整合性						

シナリオ No	ケース No	障害発生 レイヤー	障害発生 箇所	障害の 種類	対象機能	実施手順（障害シミュレーションと復旧方法）※4	合格条件（影響）	実施結果	実施者	実施日	検証結果	検証者	検証日
		リージョン		ン 停止	登録	・ JMeterのシナリオ起動 ・ 5分経過後、障害シミュレーション 「障害復旧手順書」に従い以下を手動で実施 ・ データ整合性チェック ・ 東京リージョン復旧	が保たれていること						
4	4-1	ハードウェア：ストレージ	DBストレージ	ディスク故障	ポイント登録	Chaos Toolkitのシナリオ「disc_failure_1.json」により以下を実行 ・ JMeterのシナリオ起動 ・ 5分経過後、障害シミュレーション 「障害復旧手順書」に従い以下を手動で実施 ・ データリカバリ ・ データ整合性チェック	<input type="checkbox"/> 2時間以内で復旧できること <input type="checkbox"/> 直前のトランザクションまで復旧できること <input type="checkbox"/> データリカバリ後データベースのテーブル間の整合性が保たれていること						

※4 障害のシミュレーションやJMeterシナリオの起動等は、Chaos Toolkitを使用する想定です。Chaos Toolkitについては「[3.2.4. テストツールの導入](#)」を参照してください。

3.2.3. 障害テストデータ方針

データコンディションによってもシステムのふるまいは異なります。データのマス킹等セキュリティ面に十分考慮した上で、できるだけ本番に近いデータを準備します。準備にかかる時間を省力化するため性能テスト等のデータの活用を検討します。

障害が起こった後に整合性が保たれているか、あるいは、バックアップからリカバリした後に期待したデータの状態になっているか、などの観点は正しいデータでテストしないと正しく評価できません。

3.2.4. テストツールの導入

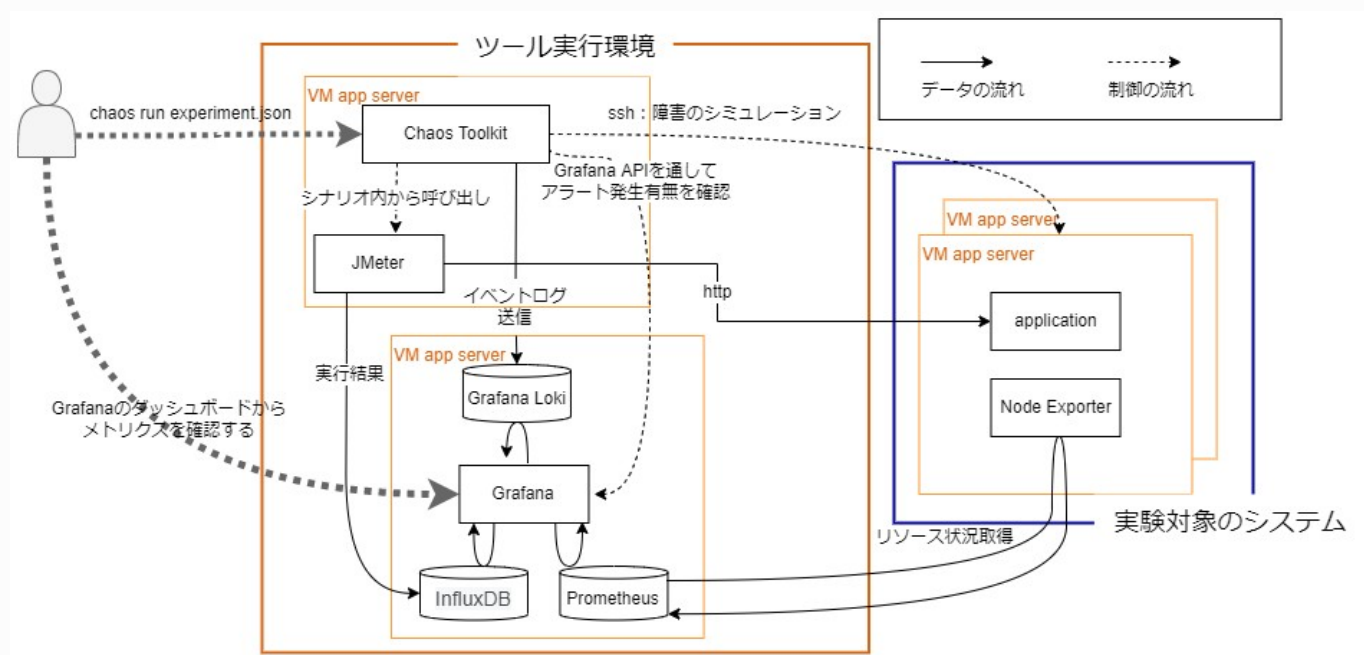
障害テストは一度実施すれば終わりではありません。想定通りの結果が得られなければ合格するまで繰り返しテストが必要になります。また、人手でテストを実施するとオペレーションミスにより正しくテストできないリスクも発生します。そのため、できる限りテストを自動化し効率的に実施すべきです。自動化戦略を検討するにあたり障害テストでは3種類のツールを検討します。

役割	内容	ツール候補	検討ポイント
障害のシ	・ 本番で発生し得る障害を	・ Chaos Toolkit	・ 導入のしやすさ、拡張

役割	内容	ツール候補	検討ポイント
ミュレーション	シミュレート	<ul style="list-style-type: none"> ・ Gremlin ・ AWS Fault Injection Simulator など 	の観点で環境に応じたツールを選定する
ユーザーからのリクエスト送信	<ul style="list-style-type: none"> ・ 本番と同様のトラフィックを再現（外部サービスに関わる場合はスタブ適用を検討） 	<ul style="list-style-type: none"> ・ Apache JMeter ・ Gatling など 	<ul style="list-style-type: none"> ・ 実行シナリオを含めた環境をそのまま利用できるため性能テストと同じツールを利用すると良い ※性能テスト計画ガイドの負荷ツールの選定 参照
システムの状態の可視化	<ul style="list-style-type: none"> ・ システムの状態を測定し可視化する仕組み ・ データの取得、蓄積、可視化が必要 	<ul style="list-style-type: none"> ・ Grafana ・ Zabbix など 	<ul style="list-style-type: none"> ・ 評価・確認項目に対するシステムの状態を把握するため可視化が必須 ・ 本番と同等の監視環境を構築する

これらの環境をまとめて構築できる「[障害テストツール](#)」を公開しています。無償で利用可能です。効率的に障害テストを実施するため、導入するツールの候補として検討してください。

ツール導入環境イメージ



※ カオスエンジニアリングでは、仮説を立て障害をシミュレーションすることで反証を試みる一連の流れを「実験」と呼ぶため、テスト対象のシステムを「実験」対象のシステムと記載しています。

3.3. テスト環境

このトピックでは、『[3.1. 障害テスト方針](#)』や『[3.2. 障害テストの実施方針](#)』にて検討した内容を踏まえ、障害テストに必要なテスト環境の要件を検討します。基本的な考え方については『[全体テスト計画ガイド](#)』の『[4.5. テスト環境](#)』をご参照ください。

ここでは障害テストの環境へ求められる要件について解説します。

検討方法

障害テストを実施するためには、テスト対象システムの環境と障害をシミュレーションするための環境（ツール実行環境）が必要です。

テスト対象システムの環境

障害テストでは本番同等のテスト環境が必要です。本番と同等の環境で検証しなければ、本番運用時に可用性要件を満たすことを確認出来ないためです。具体的には以下の要件を満たしている必要があります。

- 本番と同等のシステム構成（災対環境を含む）であること
- 本番と同等のサーバリソースであること
- 本番と同等のデータ量を投入できること
- 本番と同等の通信帯域であること

新規開発の場合は一般ローンチ前の本番環境を利用するのがオススメです。そうでなければ上記条件を満たしたステージング環境が必要です。

これらは連携する関連サービスについても同じです。他ベンダーの協力が得られない場合など準備できない場合は代替環境の構築を検討してください。

ツール実行環境

障害テストツールがシミュレーションした障害の影響を受けず安全にテストするために、テスト対象システムの環境とは別に、障害シミュレーションのための環境（サーバ）を準備します。障害シミュレーションにより、ユーザーリクエストの送信やシステム状況の監視がストップしてしまえば、障害発生時のシステムのふるまいを正しく評価できないためです。ネットワーク分断等ネットワークに関連する障害や、データセンター停止等の広域障害のテストをする場合はとくに注意が必要です。

3.4. 体制

このトピックでは、テスト実施に関与するステークホルダーと、各ステークホルダーの責務、指示系統、トレーニングについて検討します。基本的な考え方については『[全体テスト計画ガイド](#)』の『[4.6.体制](#)』をご参照ください。

ここでは障害テストの体制を考える上で重要となる観点について解説します。

検討方法

障害テストの実施では、以下のメンバーが体制に含まれている必要があります。

- アプリケーションの開発メンバー
- システムのインフラ開発メンバー
- システムの運用メンバー（開発メンバーと異なる場合）

障害テストに関わるタスクは、障害テスト計画の策定にはじまり、テスト環境の設計・構築、テストシナリオ作成、実行、結果の評価、不具合解消など多種多様です。これらには、インフラにかかわるタスクとアプリケーションにか

かわるタスクが混在します。タスクとチームの役割を明確にした上で、必要な要員をアサインしなければなりません。

とくに、障害テストで発生する不具合の多くは、アプリケーションとインフラの双方に起因することがあり得ます。そのため、アプリケーションとインフラの双方の観点による解析・解消をすることになるため、不具合解析・解消時にもアプリケーション開発メンバーとインフラ開発メンバーの協力が必要となります。

また、システムの運用メンバーが開発メンバーと異なる場合は、ローンチから運用への移行をスムーズに行うため運用メンバーも体制に含みます。

3.5. タスクと役割分担

このトピックでは障害テストに必要なタスクを洗い出し、その役割分担を検討します。基本的な考え方については『[全体テスト計画ガイド](#)』の『[4.7.タスクと役割分担](#)』をご参照ください。

検討方法

以下に障害テストで発生するタスクの例を記載します。

分類	タスク
障害テスト計画策定	・ 本ガイド記載事項の実施
プレ障害テストの準備	・ 障害テストシナリオの作成 ・ 障害テストケースの作成 ・ 障害テストデータ作成・投入 ・ テストツールの準備
プレ障害テストの実施	・ 障害テストの実施 ・ 障害テスト結果の評価 ・ 障害テストの課題解消
テスト環境の準備	・ 障害テスト実施環境の設計 ・ 障害テスト環境の構築
外部連携システムとの調整	・ 障害テスト時に利用できる環境の調整 ・ 障害シミュレーション方式の調整
障害テストシナリオ作成	・ 障害テストシナリオの作成
障害テストケース作成	・ テストケースの作成
障害テストデータ作成・投入	・ テストデータの作成 ・ テストデータの投入 ・ 障害シミュレーションシナリオの作成
テストツールの準備	・ 障害テストツールの導入 ・ 障害テストツールの実行シナリオ作成 ・ 負荷ツールの導入 ・ 負荷ツールの実行シナリオ作成

分類	タスク
	<ul style="list-style-type: none"> ・ 監視ツールの導入 ・ 各ツールの疎通確認
障害テスト実施	<ul style="list-style-type: none"> ・ 障害テストツールの実行 ・ 実行ログの取得 ・ サーバリソースの監視・取得
障害テスト結果の評価	<ul style="list-style-type: none"> ・ 評価対象となる情報の収集 ・ 収集した情報の可視化
障害テストの課題解消	<ul style="list-style-type: none"> ・ 問題の原因調査・特定 ・ アプリケーションやインフラの改修
テストの再実施と再評価	<ul style="list-style-type: none"> ・ テスト再実施 ・ テスト結果の再評価

システムやプロジェクトの特性にあわせ、必要なタスクを追加してください。

3.6. スケジュール

このトピックでは、『[3.5. タスクと役割分担](#)』にて検討したタスクの実施スケジュールを検討します。基本的な考え方については『[全体テスト計画ガイド](#)』の『[4.8. スケジュール](#)』をご参照ください。

検討方法

障害テストのスケジュールを検討する際は以下の点に注意が必要です。

実績のない方式を適用する場合はプレ障害テストの実施を検討する

実績のない方式を適用する場合は、システムテスト工程で実施する障害テストとは別に前工程でプレ障害テストを実施することを検討してください。非機能要件を満たすべく設計した方式に顕在化する問題点を早期に検出し、設計を見直すためです。時にはクラウドサービス等の再選定に迫られることも想定しておくべきです。アプリケーションの開発が完了していない段階でテストを行う場合は、プロトタイプやPoCで開発したアプリケーションの利用を検討します。

テスト環境を専有できるスケジュールを検討する

障害テストを他のテストと並列で実施すると、障害の種類によっては他のテストに影響を与える可能性があります。たとえば、機能テストや性能テストと同じサーバやネットワーク上で実施する場合、シミュレーションした障害によってはこれらのテストへ影響が出てしまう可能性があります。そのため、障害テスト実施時はテスト環境を占有するスケジュールにすることが望ましいです。

繰り返しテストするため自動化を検討する


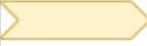


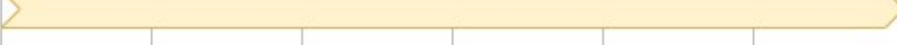
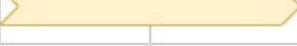


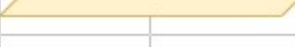




障害テストはシステムに潜在する問題を検出し是正するために実施します。一度きりで全ケースに合

りません。テスト対象のアプリケーションやインフラ環境の不具合だけでなくテスト環境構築の不備、実施手順のミスなどさまざまな問題に対処していくために繰り返しテストを実施する必要があります。そのため、テスト環境を構築する際には「障害のシミュレーション」「アプリケーションへのリクエスト送信」「システムが正常であることの確認」などの自動化を図るための準備をスケジュール上で考慮します。

外部連携システムが存在する場合は、テスト方式・実施時期等を合意する

外部連携システムが存在する場合は、自チームだけ、あるいは自社だけでテストが完結しません。他チーム・他社と協力してテストする必要があるため、実施方式や実施スケジュールは前もって調整し合意しておきます。とくに、前述の通り障害テストには十分な準備が必要な場合や、他のテストと並行実施できない場合も多く、計画的に実施する必要があります。

上記を踏まえ、以下に『[3.5. タスクと役割分担](#)』のタスク例に対し、どのタスクをどの工程で実施するかを検討した例を示します。

タスク	要件定義	外部設計	内部設計	プログラミング・単体テスト	結合テスト	システムテスト
障害テスト計画策定						
プレ障害テストの準備						
プレ障害テスト実施						
テスト環境の準備						
外部連携システムとの調整						
障害テストシナリオ作成						
障害テストケース作成						
障害テストデータ作成・投入						
テストツールの準備						
障害テスト実施						
障害テスト結果の評価						
障害テストの課題解消						
テストの再実施と再評価						

例として挙げている各タスクの実施スケジュールを設定した意図は以下の通りです。

- プレ障害テストの準備と実施

プレ障害テストは早期問題検出のため、設計と並行して進めるスケジュールとしています。

- テスト環境の準備

プレ障害テストに向けた準備、障害テストに向けた改善を含め長い期間を設定しています。

- 外部連携システムとの調整

外部システムとの連携部分も障害テスト範囲に含める場合、外部システム側でもテスト環境の準備が必要であるため、要件定義時点から調整を始めるスケジュールとしています。

- 障害テストシナリオ作成、障害テストケース作成

外部設計が終わった機能のテストシナリオから着手するスケジュールとしています。

- 障害テストデータ作成・投入

性能テストで作成したデータを利用する想定のため、システムテスト直前に実施するスケジュールとしています。

- テストツール準備

テストツールのテストシナリオ作成はアプリケーションを動かしながら実施すると効率が良いため、プログラミング・単体テストからの着手としています。

- 障害テスト実施、障害テスト結果の評価、障害テストの課題解消、障害テストの再実施と再評価

いずれも障害テストの工程で実施する作業となります。

4. 参考文献

- [全体テスト計画ガイド](#)
- [テスト種別&観点カタログ](#)
- [障害テスト計画書サンプル](#)
- 独立行政法人情報処理推進機構（IPA），“システム構築の上流工程強化（非機能要求グレード）”，IPA 独立行政法人 情報処理推進機構, 2019, <https://www.ipa.go.jp/sec/softwareengineering/std/ent03-b.html>(参照 2022-03-10).
- [カオスエンジニアリング - O'Reilly Japan](#)
- [カオスエンジニアリングの試み](#)

/* Recommend */

「ソフトウェアテスト」のおすすめ記事はこちら

この記事に関連する記事もお読みください。



ソフトウェアテスト

システムの耐障害性を高めるためのアプローチ

2022/12/22 6



ソフトウェアテスト

障害テスト計画書サンプル

2022/12/22 7



ソフトウェアテスト

性能テスト計画書サンプル

2022/04/15 11

Cookie利用について

最近投稿された記事も用意しました。



ソフトウェアテスト

TIS AIChatLab：MagicPodをつ
かった自動テスト導入戦略

2025/02/26 8



ソフトウェアテスト

DialogPlay：AIテスト自動化プ
ットフォーム『MagicPod』の活
用

2024/10/30 22



ソフトウェアテスト

Sales Driveの安定リリースを支
える自動テスト：MagicPodの活
用状況

2023/12/22 14

「ソフトウェアテスト」で最も読まれている記事を以下にまとめています。



ソフトウェアテスト

全体テスト計画ガイド

2018/10/01 35



ソフトウェアテスト

テスト種別&テスト観点カタログ

2018/10/01 19



ソフトウェアテスト

性能テスト計画ガイド

2022/04/05 21