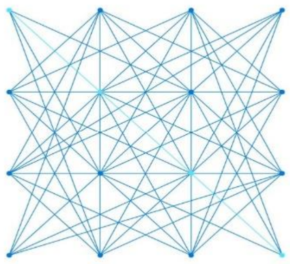

# Chaos Engineering Guiding Principles

Author: Rick Caudle

2/13/2023

## Change Record

| Date | Author | Version | Change Reference |
|---|---|---|---|
| 2/13/2023 | Rick Caudle | 1.00 | Document created |
| 2/13/2023 | Rick Caudle | 1.01 | Outline creation and Executive Summary |
| 2/20/2023 | Rick Caudle | 1.02 | Added detail |

## Document Properties

| Item | Details |
|---|---|
| Document Title | Chaos Engineering – Guiding Principles |
| Author | Rick Caudle,<br>Principal Cloud Solution Architect<br>Microsoft Corporation |
| Creation Date | 13 Feb 2023 |
| Last updated | 6 April 2023 |

## Table of Contents

## Executive Summary

Chaos Engineering is a disciplined approach to finding failures before they become outages. You literally "break things on purpose" to learn how to build more resilient systems.

The goal is to validate resilience through fault injections to confirm that your solution can withstand disruptions that will be encountered in production.

The pattern is to collect information, develop a hypothesis, conduct an experiment, then do analysis and improve. The process of observing, asking questions, and seeking answers through tests and experiments is not unique to any one field of science.

Today, we are going to cover some of the fundamental concepts of the scientific method and how they might be applied to Chaos Engineering. We are also going to identify patterns and establish guiding principles to help ensure we have a consistent and repeatable approach to conduct experiments.

## Scientific method

The scientific method is a mathematical and experimental technique employed in sciences. It is a technique used in the construction and testing of a scientific hypothesis and plays an integral role in Chaos Engineering.

A typical application of the scientific method is for the researcher to develop a hypothesis, test it through various means, and then modify on the basis of the outcome of the tests and experiments. The goal is to further modify, test again, until it becomes consistent with observed phenomena and testing outcomes.

## Guiding Principles

There are approximately ten phases or steps that should be followed for each experiment. When implementing a methodology that requires the team to complete each phase, you are creating a model that allows you to be consistent in your approach. By following a common approach, you are maximizing your potential to either prove that your solution is resilient or to uncover issues that need to be addressed. Regardless of the outcome, you end up either knowing the system is resilient or how the system responds to the fault and how it can be expected to behave.

Microsoft recommends leveraging a shift-left approach, meaning to validate in pre-production environments.

### *Nine steps that should be followed for each experiment:*

1. Understand the system end-to-end
2. Gain organizational agreement
3. Create a hypothesis and plan experiments
4. Enable Observability
5. Prepare your experiments
6. Run Chaos experiments
7. Analyze the results to prove or disprove hypothesis
8. Communicate findings and improve

9. Grow blast radius and repeat

These are the fundamental phases of conducting Chaos Engineering experiments and are proven to yield repeatable results to help ensure solutions are resilient. These steps allow you to uncover issues before it results in a major financial impact on the business.

## Information collection – Understand system end-to-end

This is the very first step to formulate a hypothesis. We need to collection information, make observations, and ask questions. This could be a combination of things i.e., observing past issues or problems, reading architecture diagrams, sequencing diagrams, interviewing stakeholders, developers, architects etc.

### *What exactly are you looking for?*

It's very important to collect solution architecture documents, sequencing diagrams, service dependencies maps, deployment documents etc. Are there any 3rd party dependencies? How are those integrated into the solution? Are there any known services or systems that have a history of causing major pain for the business? What are the areas of pain and how do those impact business?

Is there a specific region that is more prone to failure or seems to be problematic? Where and how are the solution artifacts deployed?

### *Identifying the business-critical use cases*

During this phase it's important to first focus on the business critical use cases and having the stakeholders agree on what those are. Once you have defined hypotheses for the use cases you can begin to identify the types of experiments that are needed to vet the hypothesis.

## Hypothesis

A hypothesis is an assumption, an idea that is proposed for the sake of argument so that it can be tested to see if it might be true.

The hypothesis plays a very important role within the scientific method as it is constructed before any applicable research has been done, apart from a basic background review.

Ask a question, then read up on what has been studied before and then form a hypothesis. A hypothesis is usually tentative; it's an assumption or suggestion made strictly for the objective of being tested.

Hypotheses allow us to develop broad general explanations, or scientific theories.

## Theory

A theory, in contrast, is a principle that has been formed as an attempt to explain things that have already been substantiated by data. It is understood to be more likely to be true than a hypothesis is.

It's important to note the hypothesis and theory are often used interchangeably in non-scientific uses to mean an idea, speculation or hunch, which is not the case when used in the context of the scientific method. This is very important as hypothesis and theory are prone to being wrongly interpreted even when they are encountered in scientific contexts.

## Hypothesis, Experiment, Analysis, Improve

Building a hypothesis, creating the experiment to test the hypothesis and then analyzing the results and modify until the results are consistent. This ensures that our hypothesis is very well defined, and the results can be proven consistently.

Below you find some fundamental guidelines to help define a hypothesis, the types of failure and disruption scenarios you may need to consider as well as some examples Hypotheses.

### *Getting started*

As a rule of thumb, you can start by building a list of ***scenarios and hypotheses to validate based on either your DR plans*** or ***past outages*** that you have been impacted by. Coupling these details along with the steps below should get off to a great start.

1. Collection information about the system/solution, understanding the dependencies and architecture.
2. Start by asking questions.
    a. What could go wrong if? (For example, what happens when the third-party services go down?)
3. Develop a hypothesis based on our questions.
4. Build a controlled experiment simulating failure and measure the impact.
    a. You will have one of two outcomes. Either you've verified that your system/solution is resilient to the failure you introduced, or you've found a problem you need to fix.

After running your experiment, you'll have one of two outcomes. Either you've verified that your system/solution is resilient to the failure you introduced, or you've found a problem you need to fix. Both are good outcomes. On one hand, you've increased your confidence in the system and its behavior, on the other you can fix the problem before it causes an outage.

### *Example failure scenarios*

- Availability Zone Down
- DNS outage
- AAD outage
- Region isolation
- Region failover
- Dependency Disruption

## *Example Use Cases*

- Ad-hoc experimentation
- Drill events or game days
- Automated validation in CI/CD or release pipeline

## *Disruption types*

- Chaos Agent for Windows & Linux VMs i.e. resource pressure faults (CPU, memory, disk, network), Dependency disruption faults (process/service, network)
- Service Direct – Agentless and mock/proxy faults against Azure services (Dependency disruptions, Configuration changes, Latency)
- Service container failure

## *Example Hypotheses for Chaos Engineering*

1. **VM Failure:** If we randomly shut down a portion of our virtual machines, our system will continue to function normally.
2. **Network Failure:** If we simulate a network partition, our system will continue to function normally.
3. **Disk Failure:** If we simulate a disk failure, our system will continue to function normally.
4. **DB Failure:** If we simulate database failure, our system will continue to function normally
5. **Latency:** If the network latency between two services increases, the performance of the system will be negatively impacted.
6. **Memory leaks:** Hypothesis: If a service allocates more memory than it requires, it will eventually lead to a memory leak.
7. **Data corruption:** Hypothesis: If a service fails to write to a database correctly, it will lead to data corruption.
8. **Infrastructure failures:** Hypothesis: If a service is unable to access the resources it needs, it will lead to an infrastructure failure.
9. **Container failures:** Hypothesis: If a container is unable to start or run correctly, it will lead to a container failure.

✓ Keep in mind, that leveraging current DR plans and past outages will have you create scenarios and hypotheses what need to be validated.