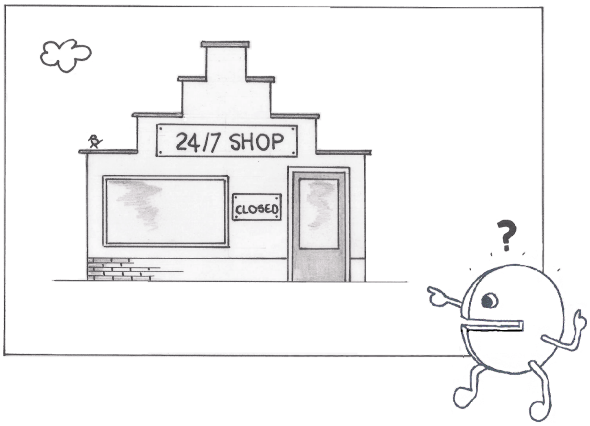
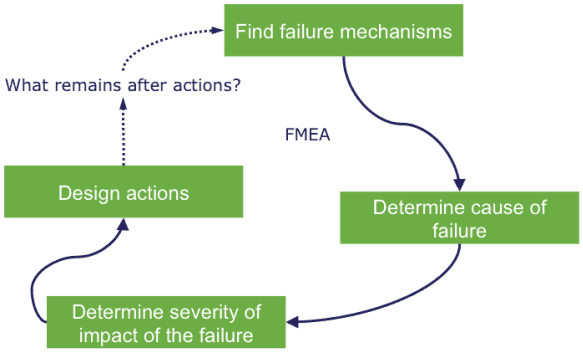
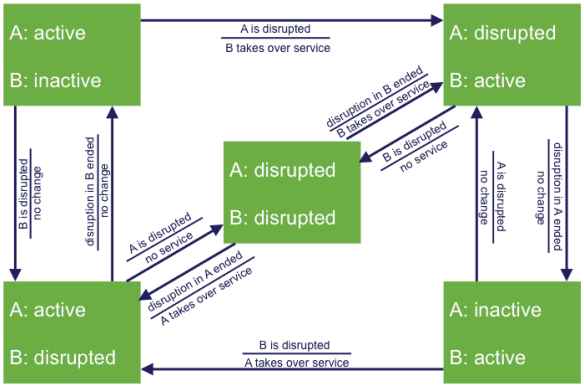
Availability and continuity are terms that can cause confusion when used next to each other. Availability is about a part of the IT landscape, such as a service. Continuity is about the process that uses it. Availability is a precondition for continuity. For instance, the continuity of the invoicing process depends on the availability of the email service to send invoices.
The continuity of processes is key (also known as business continuity and business reliability), and the following questions need to be answered: How often does a disruption occur, how fast is it resolved, and what damage has this disruption caused? To reach high service availability, a duplicated setup is needed so when a failure occurs, a spare part can take over the function of the failing part (failover or fallback). These mechanisms are (luckily) rarely used, so it is uncertain whether they actually work. However, the impact is large, so testing is crucial.
When there is a continuity failure, it is usually the service failure that is considered. However, the continuity of business processes can also be disrupted when a supplier alters the behavior of the service. Sometimes these can be predicted by previously announced changes.
In addition to changes in the service, other events can occur that cause continuity to be in jeopardy. In the selection and implementation stages, what-if scenarios can be worked out during risk analysis. For example, what happens to the data when a supplier or the customer goes bankrupt or when there is a business conflict? With major business risks, testing or simulating what-if scenarios is a measure to be considered.




## Checklist test measures ‘availability/continuity testing’




- 5.5.1 Failure Mode and Effect Analysis
  - Find possible failure modes (failing mechanisms) in the production situation.
  - Determine possible causes of failing mechanisms.
  - Determine the severity of possible (damaging) impact of the failure.
  - Design measures to discover and prevent the failure.
- 5.5.2 The role of architecture
- 5.5.3 Hardware reliability
- 5.5.4 Software reliability; examples:
  - Unexpected input (syntax test)
  - Closing user interface without logging off
  - Not working according to standard workflow (skipping steps, re-executing steps)
  - Simultaneously performing conflicting actions on different resources
  - Stopping actions in different places
  - (Paas) Rolling out the wrong software (can it be reversed?)
  - (Paas) Accidentally throwing away files
  - (IaaS) Accidentally changing infrastructure
  - (IaaS) Rolling out the wrong operating system
- 5.5.5 Guarantees and SLAsGuarantees
  - SLAs on up&down time
    - Mean time between failures (MTBF): how often a failure occurs.
    - Mean time to repair/recovery (MTTR): how long it takes, on average, to resolve the failure.
- 5.5.6 Impact of availability mechanisms
- 5.5.7 Internet and Internet connection
- 5.5.8 Testing failover
  - The time it takes to detect the failure
  - The time it takes to end the failure
  - The time it takes to repair possible damage
- 5.5.9 Testing working offline




---

Terug naar Testing cloud services | Terug naar Test Measures