

Spring 2023

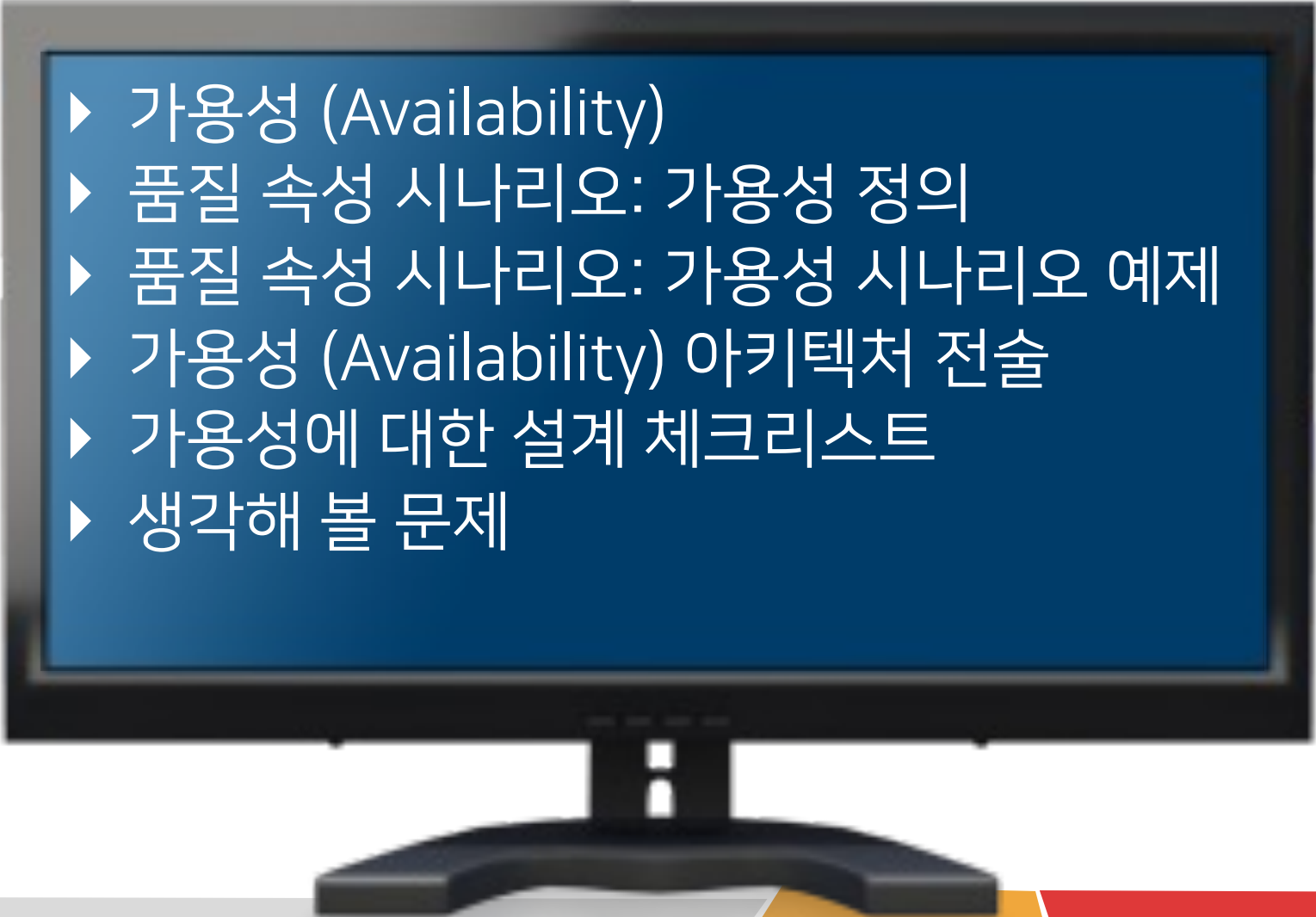


Availability Tactics

Seonah Lee

Gyeongsang National University

○○ 가용성 (Availability) 아키텍처 전술 ○○

- 
- ▶ 가용성 (Availability)
 - ▶ 품질 속성 시나리오: 가용성 정의
 - ▶ 품질 속성 시나리오: 가용성 시나리오 예제
 - ▶ 가용성 (Availability) 아키텍처 전술
 - ▶ 가용성에 대한 설계 체크리스트
 - ▶ 생각해 볼 문제

Availability

▶ 가용성 (Availability)

- ▶ 사용자가 필요로 할 때 작업을 기꺼이 수행하는 시스템의 특성



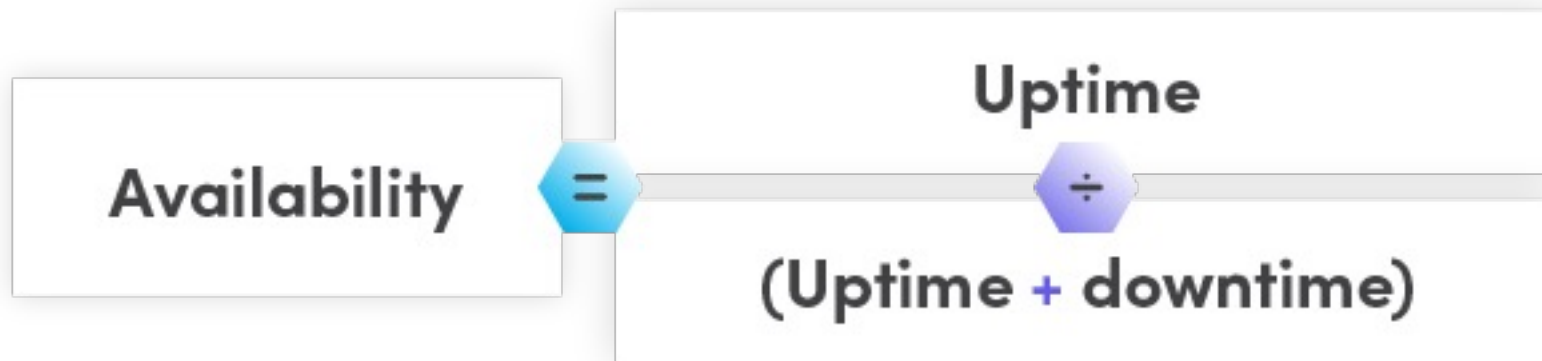
Availability

▶ 가용성 (Availability)

▶ 신뢰성과의 연관성

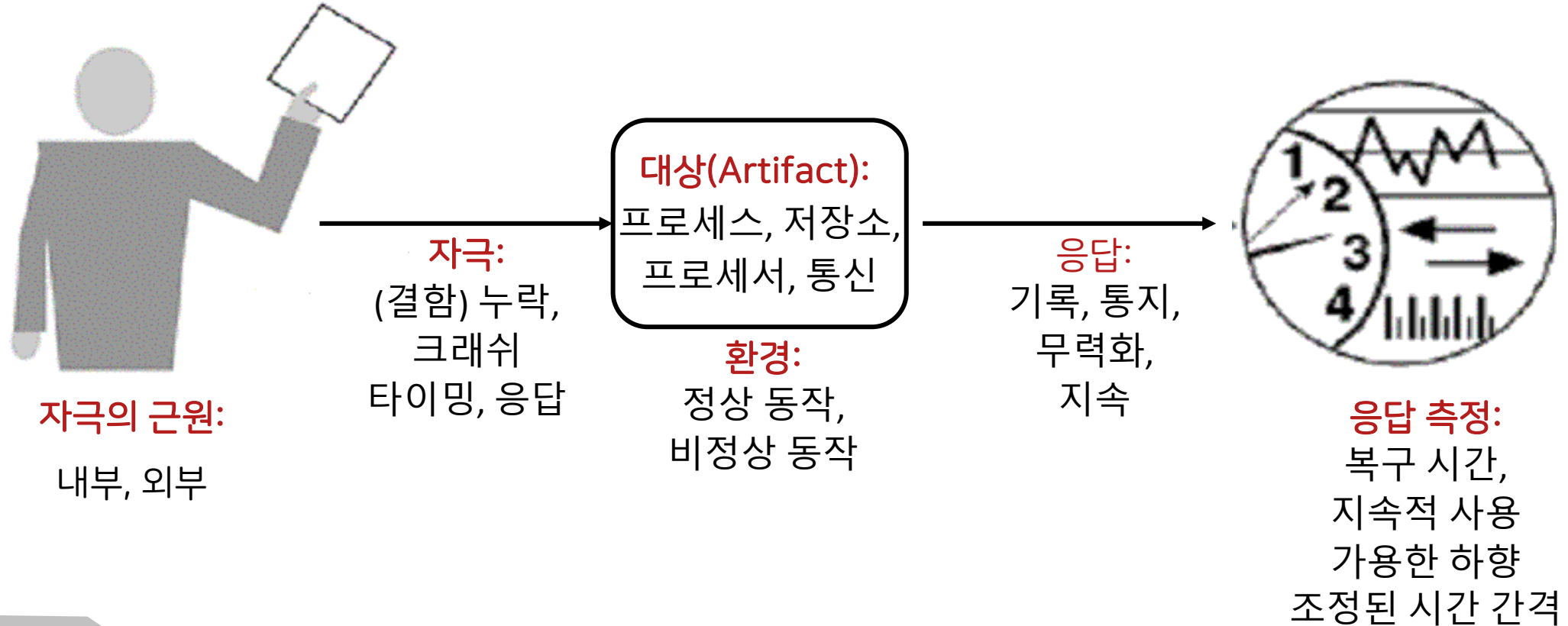
- ▶ 신뢰성 + 회복에 대한 개념을 포함
- ▶ 즉, 시스템이 문제가 생겼을 때, 스스로 수리하는 개념이 들어감

▶ 일정기간 제공되지 않은 축적된 서비스가 특정 시간 간격 내에 특정 값을 넘어서지 않도록 오류를 감추거나 수리하는 시스템의 능력



Quality Attribute Scenario for Availability

- ▶ 가용성: 소프트웨어 시스템 다운 상황으로 서비스를 더 지속하기 힘들 때, 이를 대처하는 동작들을 의미



Quality Attribute Scenario for Availability

Component	Description
자극의 근원 (Source)	시스템의 내부, 시스템의 외부
자극 (Stimulus)	결함(Fault)의 종류 <ul style="list-style-type: none">• 생략(Omission): 컴포넌트가 입력에 대한 응답에 실패• 충돌(Crash): 컴포넌트가 반복적으로 생략 결함을 경험• 타이밍(Timing): 컴포넌트가 너무 일찍 혹은 너무 늦게 응답• 응답(Response): 컴포넌트가 잘못된 값으로 응답
환경 (Environment)	결함이나 고장이 발생했을 때의 상태도 시스템의 예상 응답에 영향 미침 <ul style="list-style-type: none">• 정상 모드• Degrade 모드
대상 (Artifact)	가용성을 높이기 위해 필요한 자원으로 다음 포함 <ul style="list-style-type: none">• 프로세스• 통신채널• 저장소

Quality Attribute Scenario for Availability

Component	Description
응답 (Response)	<ul style="list-style-type: none">• 시스템의 고장에 대한 반응으로 고장기록• 사용자나 타 시스템에 알림• 일부 기능만을 수행하는 Degrademod으로 변환• 외부 시스템의 동작 정지나 일시 정지 등의 작업 수행
응답 측정 (Response Measure)	<ul style="list-style-type: none">• 사용성의 비율• 고장이 수리되는데 소요되는 시간• 시스템이 사용가능해질 때까지 소요되는 시간

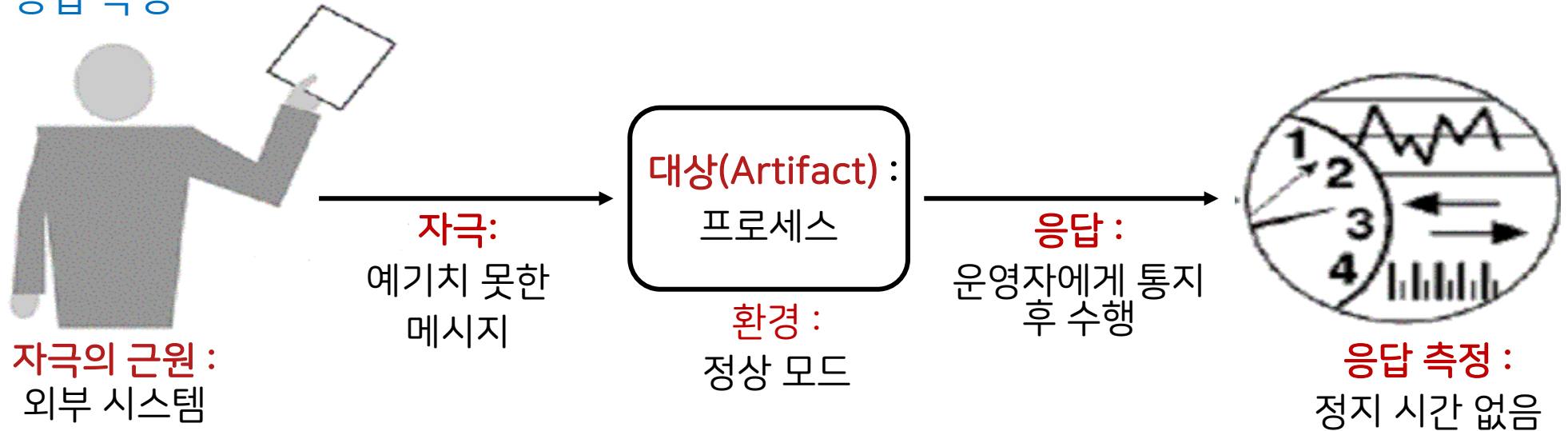
Quality Attribute Scenario

Example for Availability

▶ 가용성(Availability) 시나리오 예

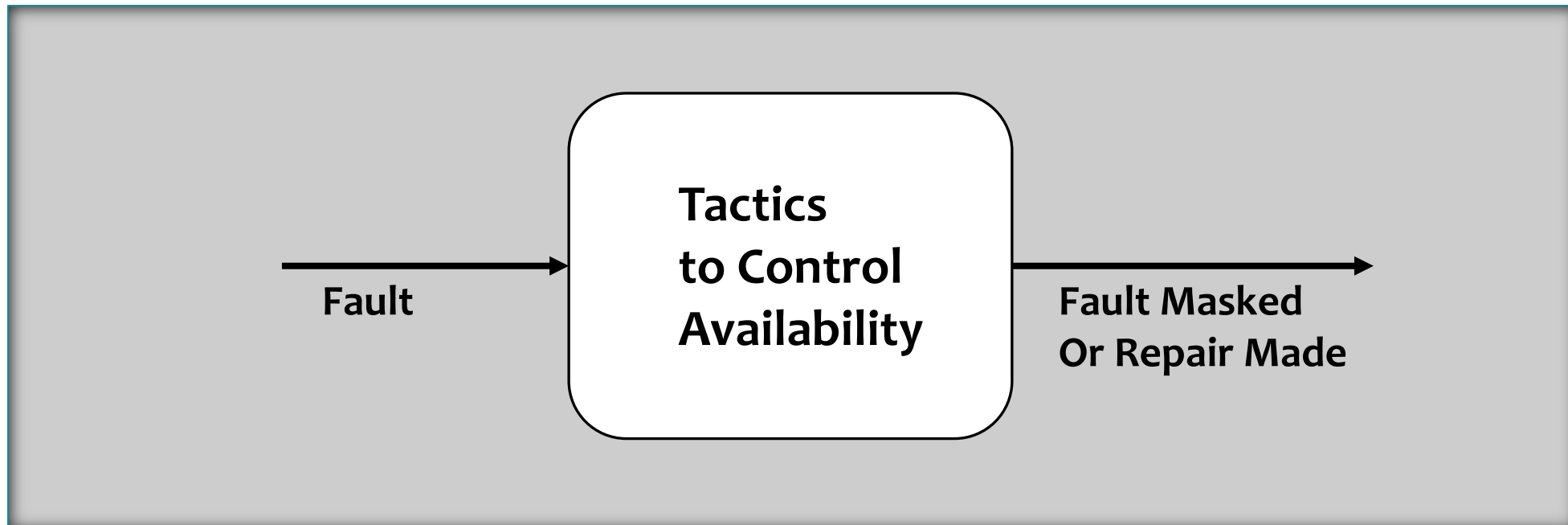
- ▶ 정상적인 동작 상태에서 예상치 못한 외부 메시지가 수신된다.
- ▶ 프로세스는 메시지가 수신됐음을 운영자에게 ^{자극의 근원}알리고 ^{자극}진행 중이던 동작을 시스템 중단 없이 계속 수행한다 ^{응답}

응답 측정

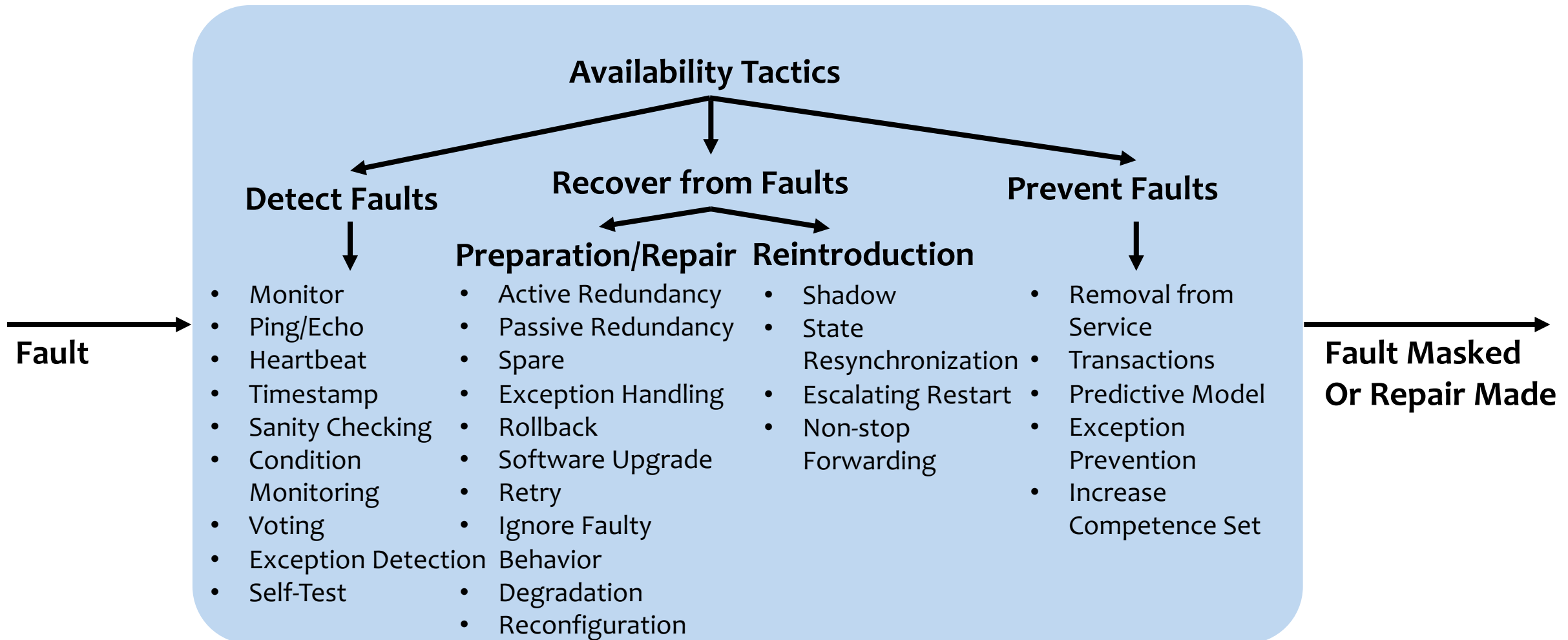


Availability Tactics

- ▶ 시스템이 원하는 서비스를 더 이상 제공할 수 없도록 하는 실패의 요인이 발생했을 때 이를 어떻게 대처할 것인가에 대해 고려할 수 있다.



Availability Tactics



Availability Tactics

▶ 오류 감지(Detect Faults)

▶ Monitor

- ▶ 시스템의 다양한 건강 상태를 모니터링하는 컴포넌트를 둠
- ▶ 시스템의 실패 혹은 공유자원의 소진 등을 모니터링 할 수 있음

▶ Ping/Echo

- ▶ 노드 간에 비동기적으로 요청/반응하는 메시지
- ▶ 컴포넌트가 살아 있고 제대로 응답하는지 확인

▶ Heartbeat

- ▶ 시스템 모니터와 모니터링되는 프로세스 사이에 주기적인 메시지 교환으로 오류를 발견
- ▶ 지속적으로 오류가 없이 살아 있음을 모니터링되는 프로세스가 알림

Availability Tactics

▶ 오류 감지(Detect Faults)

▶ Timestamp

- ▶ 분산 메시지 전달 시스템에서의 잘못된 이벤트 순서를 발견
- ▶ 이벤트가 발생한 이후에 즉시 이벤트에 로컬 시간 상태를 기록

▶ Condition Monitoring

- ▶ 프로세스나 장치의 조건을 체크하거나 설계에서 기술한 가정을 확인
- ▶ 주로 **checksum**의 계산을 수행, 모니터링 자체는 오류가 없도록 간단해야 함

▶ Sanity Checking

- ▶ 컴포넌트의 특정 동작 혹은 결과를 확인; 내부적인 설계, 시스템 상태, 정보 속성에 기반함
- ▶ 인터페이스에서 자주 활용되며 정보의 흐름 조사함

Availability Tactics

▶ 오류 감지(Detect Faults)

▶ Voting

- ▶ 같은 입력을 서로 다른 노드들이 처리하여, 처리한 결과를 **Voter**에게 보냄
- ▶ **Voter**는 투표 결과로부터 오류를 감지

▶ Exception Detection

- ▶ 실행의 정상 흐름을 변경하는 시스템의 조건을 발견하는데 초점

▶ Self-Test

- ▶ 컴포넌트가 스스로 정확한 동작을 하는지를 테스트하기 위한 절차를 수행
- ▶ 컴포넌트 자체 혹은 시스템 모니터에 의해 실행 가능함

Availability Tactics

- ▶ 오류 복구: 준비/수리 (Recover from Faults: Preparation/Repair)
 - ▶ **Active Redundancy**
 - ▶ 모든 노드가 같은 입력을 받아 병행적으로 처리
 - ▶ 실패 발생 시 밀리 세컨드 내로 회복
 - ▶ **Passive Redundancy**
 - ▶ 액티브 멤버가 입력을 받아 처리하며
 - ▶ 나머지는 주기적으로 상태를 업데이트
 - ▶ **Spare**
 - ▶ 실패가 발생하기 전까지는 관여를 하지 않음
 - ▶ 실패가 발생하면 중복된 나머지가 실패 발생한 노드를 대체

Availability Tactics

▶ 오류 복구: 준비/수리 (Recover from Faults: Preparation/Repair)

▶ Rollback

- ▶ 시스템의 실패 발견 시 알려진 이전의 좋은 상태로 돌아갈 수 있도록 허용
- ▶ 체크포인트가 특정 장소에 저장되고 정해진 시간에 업데이트 되어야 함

▶ Exception Handling

- ▶ 예외 발생 시, 오류를 정정하는 방향으로 정보를 제공
- ▶ 해당 정보를 오류를 감추거나 수리하는데 활용될 수 있음

▶ Software Upgrade

- ▶ 서비스에 영향을 주지 않는 방법으로 서비스를 업그레이드하는 방법
- ▶ **Function**과 **Class** 수준에서 패치 제공, 혹은 중복 **spare**를 활용하는 방법도 있음

Availability Tactics

- ▶ 오류 복구: 준비/수리 (Recover from Faults: Preparation/Repair)
 - ▶ **Retry**
 - ▶ 실패를 발생시키는 오류가 일시적이라고 가정하고, 동작을 재시도
 - ▶ **Ignore Faulty Behavior**
 - ▶ 특정한 소스에서 오는 메시지가 문제가 있다고 판단하면 무시
 - ▶ **Degradation**
 - ▶ 실패 발생 시 가장 핵심 시스템 기능을 유지하고 나머지는 버림
 - ▶ **Reconfiguration**
 - ▶ 현재 제약되는 자원이나 컴포넌트에 책임을 재할당하여 실패로부터 회복

Availability Tactics

▶ 오류 복구: 재시작 (Recover from Faults: Reintroduction)

▶ Shadow

- ▶ 앞서 실패하거나 서비스 업그레이드하는 컴포넌트를 “**Shadow mode**”로 일정 시간 두다가 **Active**한 역할로 돌아오게 함
- ▶ 해당 기간 동안 행위가 정확한지 점검하고 상태를 정상화함

▶ State Resynchronization

- ▶ **Active redundancy**나 **Passive redundancy**에서 노드들은 동시에 같은 입력을 받아 처리
- ▶ **Active** 혹은 **standby** 노드들의 상태를 동기화를 확인하기 위해서 주기적으로 비교함
 - ▶ Checksum, hash function 등 사용

Availability Tactics

▶ 오류 복구: 재시작 (Recover from Faults: Reintroduction)

▶ Escalating Restart

- ▶ 서비스에 미치는 영향을 최소화하기 위하여 재시작 하는 수준을 정하여 재시작
 - ▶ **Level 0:** 서비스에 미치는 영향이 가장 적음
 - ▶ **Level 1:** 비보호 메모리를 모두 해제, 다시 초기화
 - ▶ **Level 2:** 모든 메모리를 모두 해제, 다시 초기화
 - ▶ **Level 3:** 관련 이미지와 데이터 세그먼트를 모두 다시 로딩, 다시 초기화

▶ Non-stop Forwarding

- ▶ 라우터 설계에 주로 사용; 실패가 발생했을 때 알려진 경로로 패킷을 계속해서 보냄
- ▶ 동시에 라우팅 정보를 회복하여 확인

Availability Tactics

▶ 오류 예방 (Prevent Faults)

▶ Removal from Service

- ▶ 일시적으로 시스템 컴포넌트를 서비스 상태에서 제외하여 잠재적인 시스템 실패를 완화
- ▶ 예: 메모리 누수가 있으면 해당 구성 요소를 메모리에서 제거했다가 다시 적재

▶ Transactions

- ▶ 분산 컴포넌트 간에 교환하는 비동기적 메시지에 대해서 트랜잭션의 **ACID** 성질을 적용 (Atomic, Consistent, Isolated, Durable)

Availability Tactics

▶ 오류 예방 (Prevent Faults)

▶ Exception Prevention

- ▶ 시스템의 예외 발생을 사전 예방.
- ▶ 에러 정정 코드; 잘못된 포인터를 예방하기 위한 추상 데이터 타입으로의 스마트 포인터 등

▶ Increase Competence Set

- ▶ 작동을 하는데 “**Competent**”한 상태들의 집합
- ▶ 예외 및 오류 발생 시에 정상 운영의 일부로 가능한 많은 케이스를 다루도록 설계

▶ Predictive Model

- ▶ 시스템의 건강상태를 모니터링을 하다가 어느 임계점이 넘어갈 때 정정 행위를 수행
- ▶ 서버에서의 세션 수립 비율, 메시지 큐 길이 통계 등을 활용



Design Checklist for Availability



- ▶ 책임 할당 **allocation of responsibilities**
 - ▶ 높은 **가용성**을 필요로 하는 시스템의 책임을 결정.
 - ▶ 이러한 책임하에 **omission, crash, incorrect timing, incorrect response time**을 발견하기 위한 추가적인 책임 할당을 확인.
 - ▶ 다음 책임도 확인:
 - ▶ 오류 **기록**
 - ▶ 적절한 사람 혹은 시스템에 오류 **통지**
 - ▶ 오류를 발생시키는 이벤트의 소스를 사용하지 **않도록** 함
 - ▶ 일시적으로 가용하지 않은 **상태**
 - ▶ 오류나 실패를 고치거나 **가림**
 - ▶ 저하된 상태에서의 **작동**



Design Checklist for Availability



▶ 조정 모델 **coordination model**

- ▶ 해당 메커니즘이 **omission, crash, incorrect timing, incorrect response time**을 발견할 수 있도록 하는가?
 - ▶ 예를 들어 전달을 **보증**할 수 있는지?
 - ▶ 저하된 상태에서 **협업**이 되는지?
 - ▶ 해당 메커니즘이 오류의 **기록** 및 상기의 책임을 **수행**할 수 있는지?
 - ▶ 해당 모델이 사용하는 대상(프로세서, 커뮤니케이션, 채널, 저장 매치, 프로세스)의 **대체**를 가능하도록 하는지?
 - ▶ 저하된 상태, **Startup/shutdown**, 수리 모드, 과부화 모드에서 **작동**하는지?
 - ▶ 예를 들어 해당 모델이 어느 정도의 정보 분실을 감내할 수 있는지, 결과가 어떤지 등의 질문 가능



Design Checklist for Availability



▶ 데이터 모델 **data model**

- ▶ 시스템의 어떤 부분이 **높은 가용성**을 가져야 하는지 결정.
- ▶ 해당 부분 안에서 어떤 데이터 모델이 특성 및 작동에 있어 **omission, crash, incorrect timing, incorrect response** 등의 오류를 발생시키는지 분석.
- ▶ 특성 및 작동을 일시적으로 가용하지 않도록 하거나, 오류를 수정하거나 가리기 위하여 사용하지 못하도록 할 수 있음을 확인.
- ▶ 예를 들어 서버가 일시적으로 사용하지 못하다가 서비스로 돌아올 때까지 요청을 캐시하는 작업이 있는지를 확인할 필요 있음

Design Checklist for Availability

- ▶ 아키텍처 요소 매핑 **mapping architectural elements**
 - ▶ 어떤 대상(프로세서, 커뮤니케이션, 채널, 저장 매치, 프로세스)가 **오류를 발생할 가능성**이 있는지 결정.
 - ▶ 아키텍처 요소의 맵핑과 재맵핑이 오류로부터 복구할 수 있도록 유연한지 확인:
 - ▶ 실패한 프로세서에서의 어떤 프로세스가 실시간에 **재할당**될 필요가 있는가
 - ▶ 프로세서, 데이터 저장소, 커뮤니케이션 채널을 활성화하고 **재할당**해야 하는가
 - ▶ 실패한 프로세서 및 저장소에서의 데이터를 어떻게 **대체**하는 단위에서 제공하는가
 - ▶ 얼마나 빨리 시스템이 제공된 단위에 **재설치**될 수 있는가
 - ▶ 실시간 요소를 프로세서, 커뮤니케이션 채널, 데이터 저장 장치에 **재할당**하는가
 - ▶ 기능의 중복성에 의존하는 기술을 채용할 때, 모듈을 **중복** 컴포넌트에 맵핑하는 것은 중요.
 - ▶ 예를 들어, 모듈이 활성화된 모듈과 백업 모듈 양쪽에 적절한 코드를 포함해야 함

Design Checklist for Availability

▶ 리소스 관리 **resource management**

- ▶ 오류에도 불구하고 지속적으로 운영되어야 할 **중요한 자원**이 무엇인지 결정.
- ▶ 다음의 작업을 수행할 때 남아있는 자원이 있는지 확인.
 - ▶ 오류가 발생했을 때 오류를 **기록**,
 - ▶ 적절한 개체의 **통지**,
 - ▶ 오류를 초래하는 이벤트의 소스를 **비활성화**,
 - ▶ 오류 발생 시 오류와 실패를 수정하고 **가리는** 작업.
- ▶ 다음 결정
 - ▶ 중요한 자원에서 가용한 시간
 - ▶ 명시한 시간 간격 동안 가용해야 할 중요한 자원,
 - ▶ 저하된 상태에서 중요한 자원이 있게 되는 시간 간격
 - ▶ 수리 시간 등
- ▶ 이러한 시간 간격에 사용 가능한 중요한 자원을 확인.
 - ▶ 예: 서버가 실패했을 때 메시지를 분실하지 않기 위해 입력 큐가 충분히 큰지 확인



Design Checklist for Availability



- ▶ 바인딩 시간 결정 **binding time decisions**
 - ▶ 언제 어떻게 아키텍처 요소를 바인딩할지 결정.
 - ▶ **오류난 컴포넌트를** 대체하기 위하여 늦은 바인딩을 사용하려면, 선택한 가용성 전략이 모든 소스에서 발생할 수 있는 오류를 커버하는지 확인.
 - ▶ 늦은 바인딩 시간을 사용하여 오류를 받는 대상의 프로세서를 대체하는 경우, 오류 발견 및 회복 메커니즘이 모든 가능한 바인딩에서 작성할 것인가?
 - ▶ 늦은 바인딩 시간을 사용하여 오류를 구성하는 경우, 복구 전략이 모든 경우에 적용가능한가?
 - ▶ 늦은 바인딩에 가용성 특성이 무엇이고, 실패할 확률이 있는지?



Design Checklist for Availability



▶ 기술 선택 **choice of technology**

- ▶ 오류를 발견, 복구, 실패한 컴포넌트의 재도입을 도울 수 있는 기술을 결정. 이벤트 기록과 같이 오류에 대한 대응을 도울 수 있는 기술 결정.
- ▶ 선택한 기술의 가용성 특성을 결정.
 - ▶ 어떤 오류를 회복할 수 있는지?
 - ▶ 어떤 오류를 시스템에 가져오는지?



Questions



Q1. 결함 탐지 기술(핑/에코, **Heartbeat**, 시스템 모니터, 투표, 예외 탐지)를 검토한다.

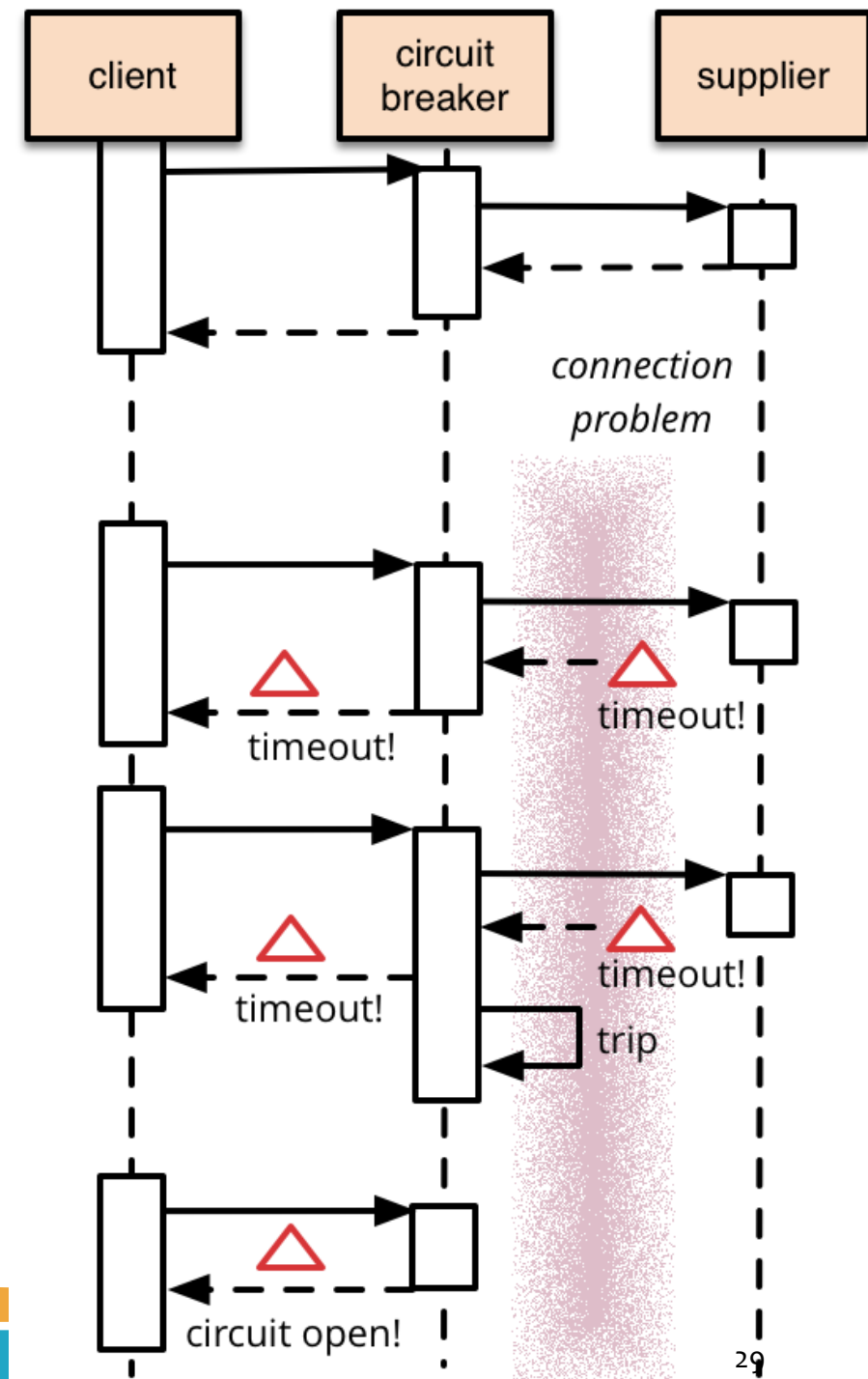
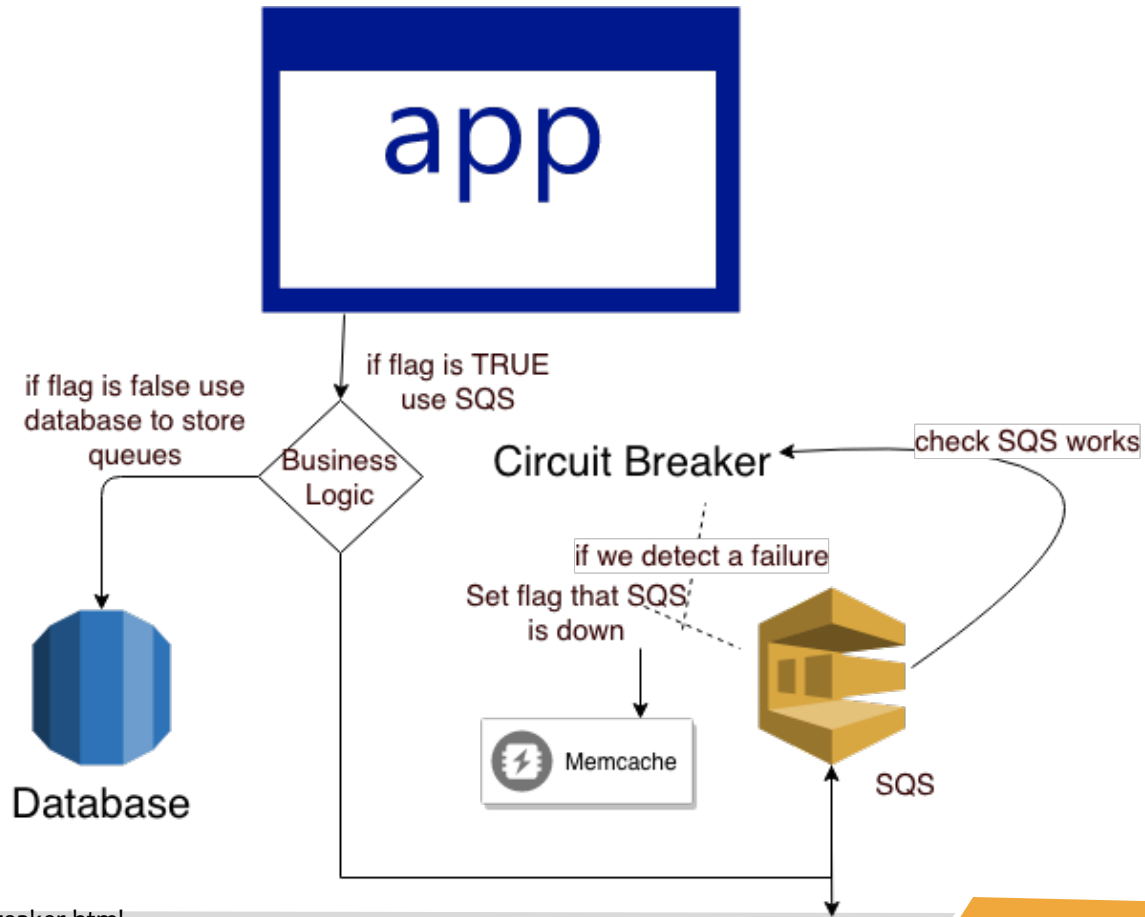
▶ 이들 기술을 사용할 때 성능에 미치는 영향은?

Q2. 가용성은 변경용이성과 어떻게 트레이드오프를 하는가?

▶ “**24/7**” 가용성(예정된 또는 예정되지 않은 중단 시간은 없음)을 요구하는 시스템은 어떻게 변경해야 할까?

분산서비스 환경에 대한 **CirCuit Breker** 적용

- ▶ LINE의 서버에 실제로 도입한 Circuit Breaker 시스템





Question?



Seonah Lee
saleese@gmail.com