中国科学技术大学
University of Science and Technology of China

Software Architecture

SSE USTC Qing Ding
dingqing@ustc.edu.cn
http://staff.ustc.edu.cn/~dingqing

Quality Attributes of Architecture I Availability

Agenda

- Quality Attributes of Architecture

- Availability
  - Source of stimulus
  - Stimulus
  - Environment
  - Artifact
  - Response
  - Response measure
  - Tactics

Why do we should consider QA?

- Business considerations determine qualities that must be accommodated in a system's architecture.
  - These qualities are over and above that of functionality, which is the basic statement of the system's capabilities, services, and behavior

- Systems are frequently redesigned not because they are functionally deficient,
  - but because they are difficult to maintain, port, or scale, or are too slow, or have been compromised by network hackers.
  - the replacements are often functionally identical

Functionality and Architecture

- Functionality and quality attributes are orthogonal
  - this is not to say that any level of any quality attribute is achievable with any function.
  - any of functions your choices as an architect will determine the relative level of quality

- What is functionality?
  - It is the ability of the system to do the work for which it was intended.

- Functionality may be achieved through the use of any of a number of possible structures.

Architecture and Quality Attributes

- Achieving quality attributes must be considered throughout design, implementation, and deployment.
  - No quality attribute is entirely dependent on design, nor is it entirely dependent on implementation or deployment.

- Satisfactory results are a matter of getting the big picture (architecture) as well as the details (implementation) correct. For example:
  - Usability
  - Modifiability
  - Performance

Architecture and Quality Attributes

- The message of this section is twofold:
  - Architecture is critical to the realization of many qualities of interest in a system, and these qualities should be designed in and can be evaluated at the architectural level.
  - Architecture, by itself, is unable to achieve qualities. It provides the foundation for achieving quality, but this foundation will be to no avail if attention is not paid to the details.

Architecture and Quality Attributes

- Within complex systems, quality attributes can never be achieved in isolation. The achievement of any one will have an effect, sometimes positive and sometimes negative, on the achievement of others.
  - security and reliability
  - almost every quality attribute negatively affects performance

- We will examine the following three classes:
  - Qualities of the system. We will focus on availability, modifiability, performance, security, testability, and usability.
  - Business qualities (such as time to market) that are affected by the architecture.
  - Architecture qualities, such as conceptual integrity, that are about the architecture itself although they indirectly affect other qualities, such as modifiability.

Quality Attribute Scenarios

质量属性场景

- A quality attribute scenario is a quality---attribute---specific requirement. It consists of six parts.
  - Source of stimulus. This is some entity (a human, a computer system, or any other actuator) that generated the stimulus.
  - Stimulus. The stimulus is a condition that needs to be considered when it arrives at a system.
  - Environment. The stimulus occurs within certain conditions. The system may be in an overload condition or may be running when the stimulus occurs, or some other condition may be true.
  - Artifact. Some artifact is stimulated. This may be the whole system or some pieces of it.
  - Response. The response is the activity undertaken after the arrival of the stimulus.
  - Response measure. When the response occurs, it should be measurable in some fashion so that the requirement can be tested.

AVAILABILITY

- Availability is concerned with system failure and its associated consequences.
  - system failure occurs when the system no longer delivers a service consistent with its specification. Such a failure is observable by the system's users—either humans or other systems.

- Among the areas of concern are
  - how system failure is detected
  - how frequently system failure may occur
  - what happens when a failure occurs
  - how long a system is allowed to be out of operation
  - when failures may occur safely
  - how failures can be prevented
  - what kinds of notifications are required when a failure occurs

AVAILABILITY

- We need to differentiate between failures and faults. 一定能观察到的是failure
  - A fault may become a failure if not corrected or masked. That is, a failure is observable by the system's user and a fault is not. When a fault does become observable, it becomes a failure.
  - For example, a fault can be choosing the wrong algorithm for a computation, resulting in a miscalculation that causes the system to fail.

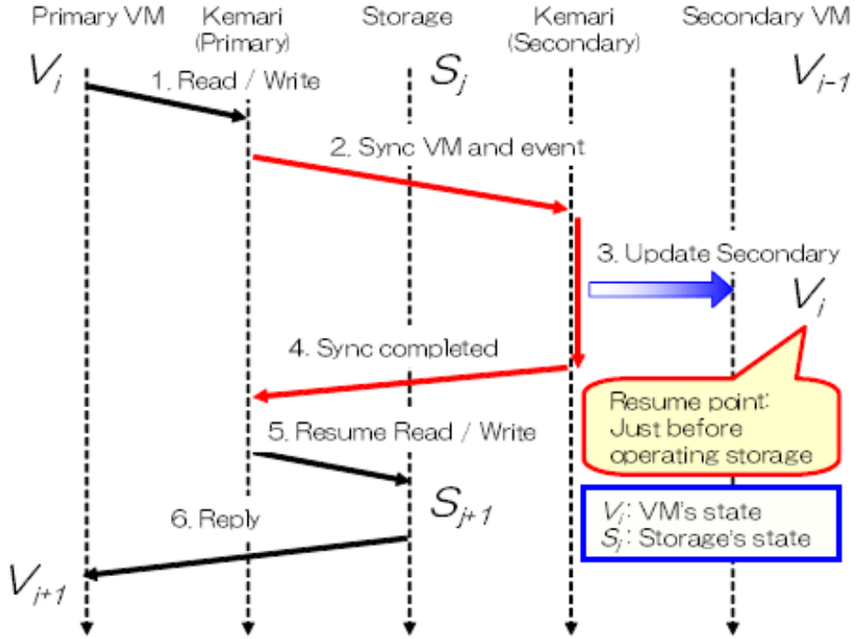
- Once a system fails, an important related concept becomes the time it takes to repair it.
  - Since a system failure is observable by users, the time to repair is the time until the failure is no longer observable.

AVAILABILITY

- The distinction between faults and failures allows discussion of automatic repair strategies.
  - That is, if code containing a fault is executed but the system is able to recover from the fault without it being observable, there is no failure.

AVAILABILITY

- For example：
  - Kemari
  - http://www.osrg.net/kemari/
  - Kemari: Virtual Machine Synchronization for Fault Tolerance
  - http://wiki.xensource.com/xenwiki/Open_Topics_For_Discussion?action=AttachFile&do=get&target=Kemari_08.pdf



AVAILABILITY

- Source of stimulus.
  - We differentiate between internal and external indications of faults or failure since the desired system response may be different.

不同的fault不同的解决方案

- Stimulus. A fault of one of the following classes occurs.
  - omission. A component fails to respond to an input.
  - crash. The component repeatedly suffers omission faults.
  - timing. A component responds but the response is early or late.
  - response. A component responds with an incorrect value.

- Artifact.
  - This specifies the resource that is required to be highly available, such as a processor, communication channel, process, or storage.

- Environment.
  - The state of the system when the fault or failure occurs may also affect the desired system response. For example, if the system has already seen some faults and is operating in other than normal mode, it may be desirable to shut it down totally. However, if this is the first fault observed, some degradation of response time or function may be preferred.

AVAILABILITY

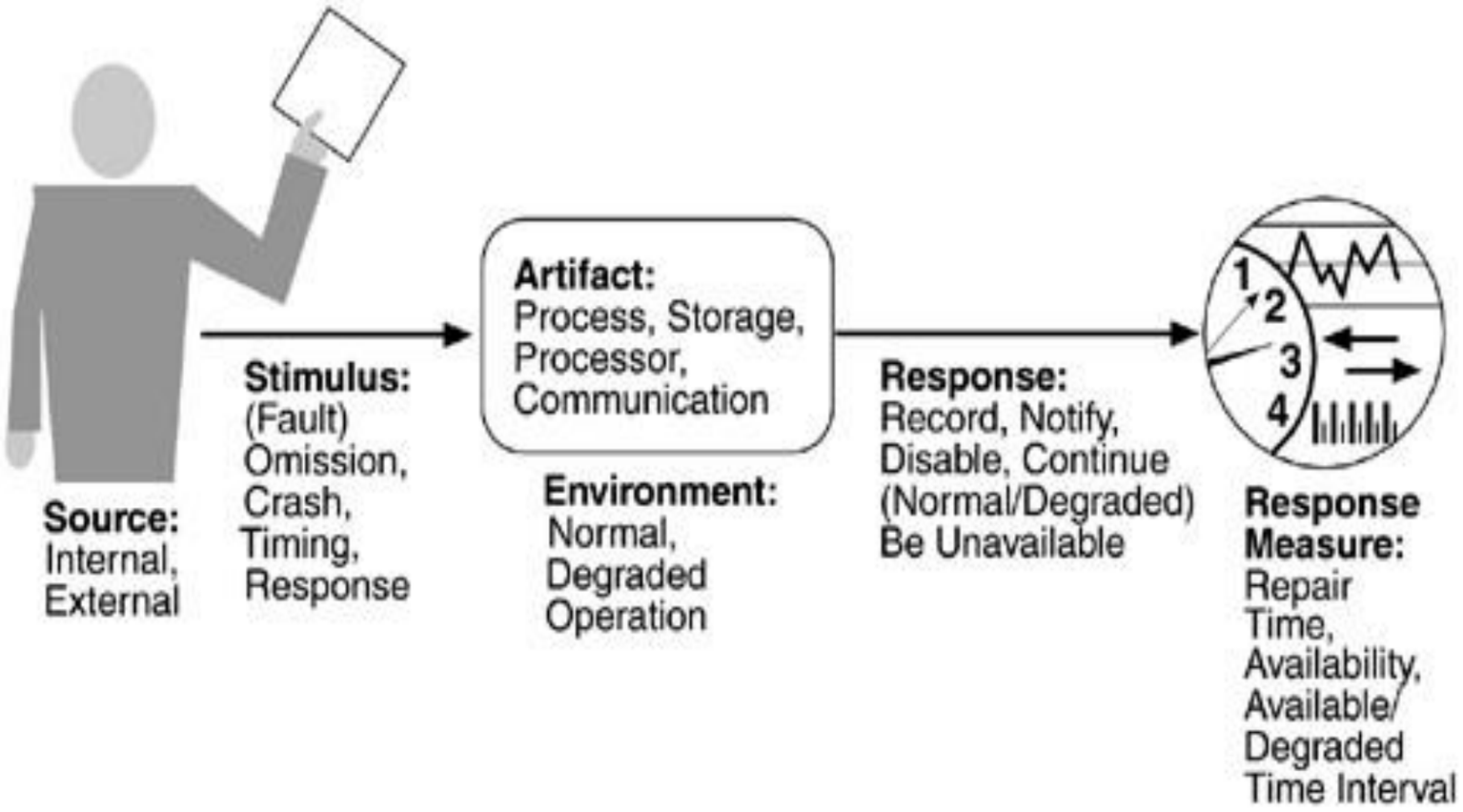
- Response.
  - There are a number of possible reactions to a system failure. These include
    - logging the failure
    - notifying selected users or other systems
    - switching to a degraded mode with either less capacity or less function
    - shutting down external systems
    - becoming unavailable during repair.

- Response measure.
  - The response measure can specify an availability percentage, or it can specify a time to repair, times during which the system must be available, or the duration for which the system must be available.

AVAILABILITY

Portion of	Possible Values
Scenario Source	Internal to the system; external to the system Fault: omission,
Stimulus Artifact	crash, timing, response
	System's processors, communication channels, persistent storage, processes
Environment	Normal operation; degraded mode (i.e., fewer features, a fall back solution)
Response	System should detect event and do one or more of the following: record it notify appropriate parties, including the user and other systems disable sources of events that cause fault or failure according to defined rules be unavailable for a pre-specified interval, where interval depends on criticality of system continue to operate in normal or degraded mode
Response Measure	Time interval when the system must be available Availability time Time interval in which system can be in degraded mode Repair time

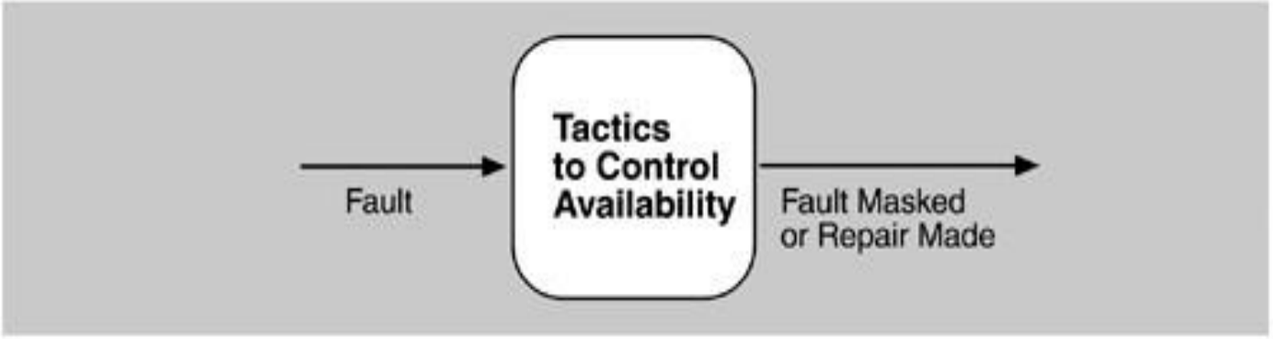
AVAILABILITY



Availability Tactics solution

- A failure occurs when the system no longer delivers a service that is consistent with its specification; this failure is observable by the system's users.
- A fault (or combination of faults) has the potential to cause a failure.
- Recovery or repair is an important aspect of availability.
- The tactics we discuss in this section will keep faults from becoming failures or at least bound the effects of the fault and make repair possible.

Goal of availabilitv tactics



Availability Tactics

- Many of the tactics we discuss are available within standard execution environments such as operating systems, application servers, and database management systems.

- It is still important to understand the tactics used so that the effects of using a particular one can be considered during design and evaluation.

所有维护可用性的方法都包括某种类型的冗余、某种类型的运行状况监视(用于检测故障)以及在检测到故障时某种类型的恢复

- All approaches to maintaining availability involve some type of redundancy, some type of health monitoring to detect a failure, and some type of recovery when a failure is detected.
  - In some cases, the monitoring or recovery is automatic and in others it is manual.

- We first consider fault detection. We then consider fault recovery and finally, briefly, fault prevention.

Availability Tactics-fault detection

- Three widely used tactics for recognizing faults are ping/echo, heartbeat, and exceptions.
  - Ping/echo.
    - One component issues a ping and expects to receive back an echo, within a predefined time, from the component under scrutiny.
    - This can be used within a group of components mutually responsible for one task.
    - It can also used be used by clients to ensure that a server object and the communication path to the server are operating within the expected performance bounds.
    - "Ping/echo" fault detectors can be organized in a hierarchy, in which a lowest---level detector pings the software processes with which it shares a processor, and the higher---level fault detectors pinglower---- level ones.
    - This uses less communications bandwidth than a remote fault detector that pings all processes.

Availability Tactics-fault detection

- Heartbeat (dead man timer).
  - In this case one component emits a heartbeat message periodically and another component listens for it.
  - If the heartbeat fails, the originating component is assumed to have failed and a fault correction component is notified.
  - The heartbeat can also carry data. For example, an automated teller machine can periodically send the log of the last transaction to a server. This message not only acts as a heartbeat but also carries data to be processed.

-在这种情况下，一个组件定期发出心跳消息，另一个组件侦听它。
-如果心跳失败，则假定发起组件失败，并通知错误纠正组件。
-心跳也可以携带数据。例如，自动柜员机可以定期将最后一个事务的日志发送到服务器。这个消息不仅起到心跳的作用，而且还携带要处理的数据

Availability Tactics-fault detection

- Exceptions.
  - One method for recognizing faults is to encounter an exception, which is raised when one of the fault classes is recognized.
  - The exception handler typically executes in the same process that introduced the exception.

- The ping/echo and heartbeat tactics operate among distinct processes, and the exception tactic operates within a single process.
- The exception handler will usually perform a semantic transformation of the fault into a form that can be processed.

Availability Tactics-fault detection

- Suppose we want to add fault detection of DBMS into SNS in order to detect the connectivity error of DBMS
  - Now that the workload of DB server is high, the influence of added fault detection on performance shall be as light as possible.
  - Meanwhile, since connectivity error is a serious error, we hope we can detect it as soon as possible.

假设我们想在SNS中加入数据库管理系统的故障检测，以检测数据库的连接错误
-由于DB服务器的工作负载较高，增加故障检测对性能的影响应尽可能轻。
-同时，由于连通性错误是一种严重的错误，我们希望能够尽快检测到它。

- Consequently, heartbeat is chosen as the tactic for fault detection.
  - To add a service in the system which creates connection to DBMS periodically and sends the result of operation to other components.

因此，选择心跳作为故障检测策略。
-在系统中添加一个服务，定期创建与DBMS的连接，并将操作结果发送到其他组件

Availability Tactics-fault detection

- In an embedded safety---critical system: aircraft

- Self check
  - Power---on self check
  - Periodical self check
  - Manual self check
  - Command triggered self check

  - To display the error code with LED

Availability Tactics-fault recovery

- Fault recovery consists of preparing for recovery and making the system repair. Some preparation and repair tactics follow.
- Active redundancy (hot restart).
  - All redundant components respond to events in parallel. Consequently, they are all in the same state.
  - The response from only one component is used (usually the first to respond), and the rest are discarded.
  - When a fault occurs, the downtime of systems using this tactic is usually milliseconds since the backup is current and the only time to recover is the switching time.
  - Active redundancy is often used in a client/server configuration, such as database management systems, where quick responses are necessary even when a fault occurs. In a highly available distributed system, the redundancy may be in the communication paths. For example, it may be desirable to use a LAN with a number of parallel paths and place each redundant component in a separate path. In this case, a single bridge or path failure will not make all of the system's components unavailable.

Availability Tactics-fault recovery

- Synchronization is performed by ensuring that all messages to any redundant component are sent to all redundant components.
- If communication has a possibility of being lost (because of noisy or overloaded communication lines), a reliable transmission protocol can be used to recover.
- A reliable transmission protocol requires all recipients to acknowledge receipt together with some integrity indication such as a checksum.
- If the sender cannot verify that all recipients have received the message, it will resend the message to those components not acknowledging receipt.
- The resending of unreceived messages (possibly over different communication paths) continues until the sender marks the recipient as out of service.

Availability Tactics-fault recovery

- Passive redundancy (warm restart/dual redundancy/triple redundancy).
  - One component (the primary) responds to events and informs the other components (the standbys) of state updates they must make. When a fault occurs, the system must first ensure that the backup state is sufficiently fresh before resuming services.
  - This approach is also used in control systems, often when the inputs come over communication channels or from sensors and have to be switched from the primary to the backup on failure.
  - Describing an air traffic control example, shows a system using it. In the air traffic control system, the secondary decides when to take over from the primary, but in other systems this decision can be done in other components.

Availability Tactics-fault recovery

- This tactic depends on the standby components taking over reliably. Forcing switchovers periodically—for example, once a day or once a week—increases the availability of the system.
- Some database systems force a switch with storage of every new data item. The new data item is stored in a shadow page and the old page becomes a backup for recovery. In this case, the downtime can usually be limited to seconds.
- Synchronization is the responsibility of the primary component, which may use atomic broadcasts to the secondaries to guarantee synchronization.

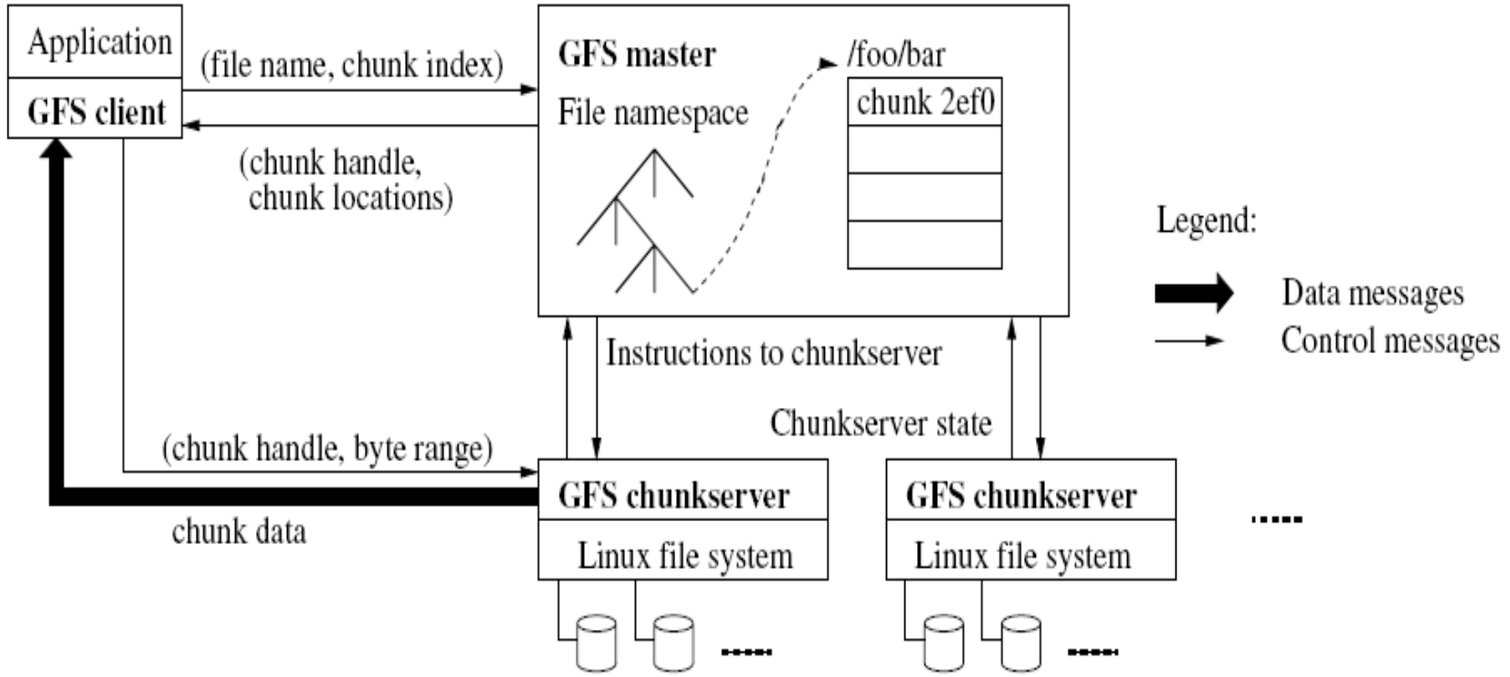
-这种策略依赖于备用组件的可靠接管。强制定期切换，例如一天一次或一周一次，可以增加系统的可用性。
-一些数据库系统强制切换每个新数据项的存储。新数据项存储在一个影子页中，旧页作为恢复的备份。在这种情况下，停机时间通常可以限制为几秒。
-同步是主组件的责任，它可以使用原子广播到次要组件来保证同步

Availability Tactics-fault recovery

- Spare.
  - A standby spare computing platform is configured to replace many different failed components.
  - It must be rebooted to the appropriate software configuration and have its state initialized when a failure occurs.
  - Making a checkpoint of the system state to a persistent device periodically and logging all state changes to a persistent device allows for the spare to be set to the appropriate state.
  - This is often used as the standby client workstation, where the user can move when a failure occurs.
  - The downtime for this tactic is usually minutes.

Availability Tactics-fault recovery

- GFS



Availability Tactics-fault recovery

- Suppose we want to add backup mechanism to the DB in order to replace main server with backup server when the former has some faults.
  - Now that SNS is not a critical system, its availability is not necessary to be very high, we allow some sessions of clients to be lost.
  - Meanwhile, once a user generates new data, it should be persistently stored into DB

- Consequently, we use passive redundancy as the backup mechanism of SNS

Availability Tactics-fault recovery

- In an embedded safety---critical system: aircraft

- Redundancy
  - Storage
  - Binary code
  - Communication linkage

Availability Tactics-fault repair

- There are tactics for repair that rely on component reintroduction.
  - When a redundant component fails, it may be reintroduced after it has been corrected. Such tactics are shadow operation, state resynchronization, and rollback.

- Shadow operation.
  - A previously failed component may be run in "shadow mode" for a short time to make sure that it mimics the behavior of the working components before restoring it to service. 以前失败的组件可能会以“影子模式”运行一段时间，以确保它在恢复服务之前模仿工作组件的行为
- Checkpoint/rollback.
  - A checkpoint is a recording of a consistent state created either periodically or in response to specific events.
  - Sometimes a system fails in an unusual manner, with a detectably inconsistent state. In this case, the system should be restored using a previous checkpoint of a consistent state and a log of the transactions that occurred since the snapshot was taken.

Availability Tactics-fault repair

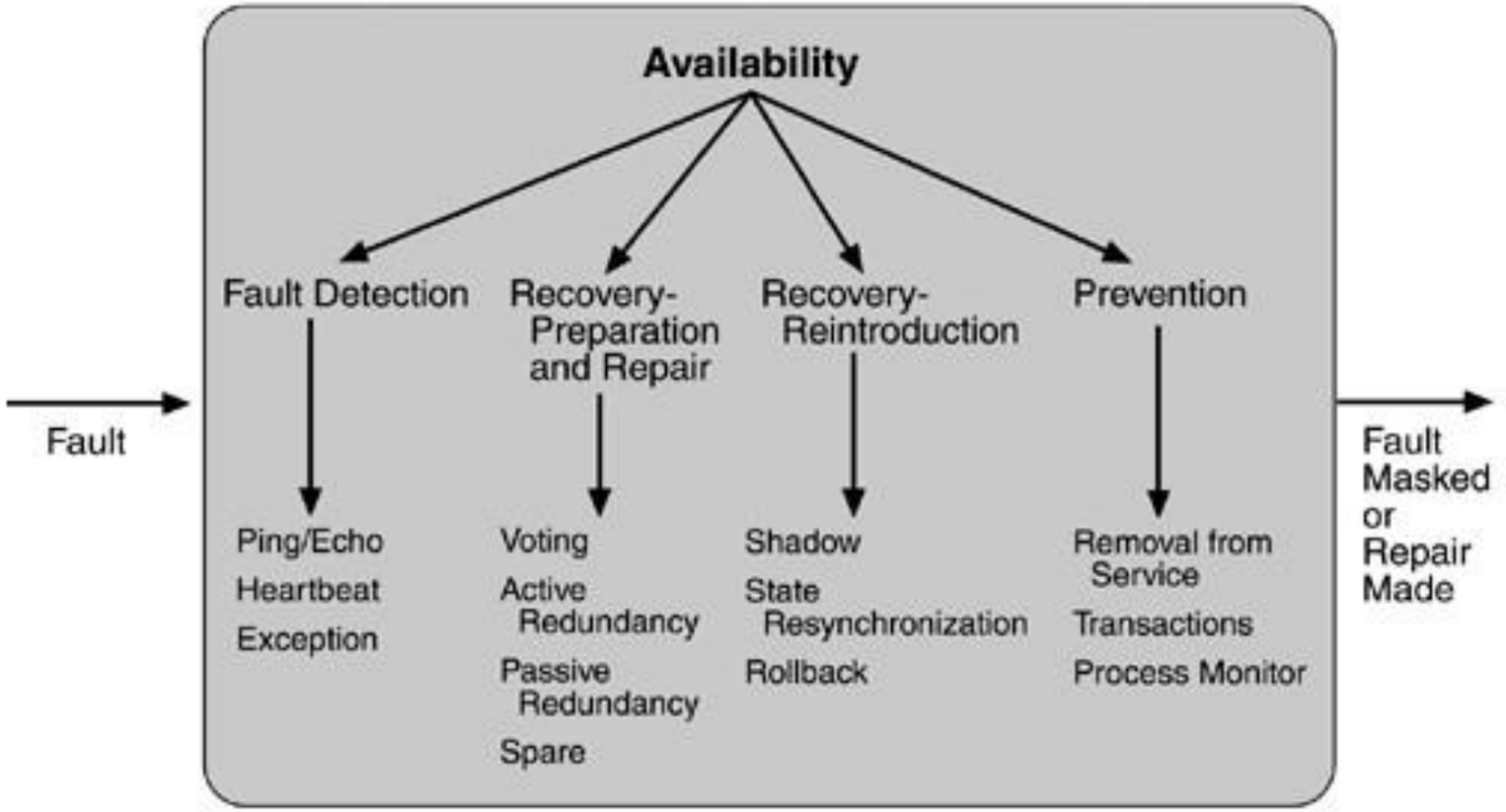
- For SNS,
  - we adopt checkpoints to do fault repair in order to keep the performance of system.

- For instruments of aircraft
  - Watchdog
  - Software interrupts
  - Pluggable components

Availability Tactics-fault prevention

- The following are some fault prevention tactics.
- Removal from service.
  - This tactic removes a component of the system from operation to undergo some activities to prevent anticipated failures.
  - One example is rebooting a component to prevent memory leaks from causing a failure.
  - If this removal from service is automatic, an architectural strategy can be designed to support it. If it is manual, the system must be designed to support it.
- Transactions.
  - A transaction is the bundling of several sequential steps such that the entire bundle can be undone at once.
  - Transactions are used to prevent any data from being affected if one step in a process fails and also to prevent collisions among several simultaneous threads accessing the same data.
- Process monitor.
  - Once a fault in a process has been detected, a monitoring process can delete the nonperforming process and create a new instance of it, initialized to some appropriate state as in the spare tactic.

Availability Tactics-Summary



Project

- Suppose you needs to improve the availability of your SNS website. Please describe your design from the following aspects:
  - Since your SNS website has a huge number of users, you deployed it into a cluster of multiple servers. Please give the details about how your design detects the failure of servers with heartbeats, includingthe information structure of heartbeat, protocol for heartbeat communication.
  - If you want to establish replicas of SNS data, which kind of redundancy do you think is suitable for your SNS website. Please give the detail reason(s).
  - What tactics would you want to apply onto your SNS website. Please give the detail reason(s).