A Use Case Model for RAS (Reliability, Availability, and Serviceability) in an MPP (Massively Parallel Processors) Environment

May 18, 2004

Sue Kelly
Sandia National Laboratories
smkelly@sandia.gov, 505-845-9770

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company
for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Outline of Talk

- A brief tutorial on Use Cases
- RAS Features for MPPs Use Case Model

References

Applying Use Cases by Geri Schneider and Jason P. Winters, Addison-Wesley, 1998.

Object-Oriented Software Engineering: A Use Case Driven Approach by Ivar Jacobson, et. al., Addison-Wesley, 1992.

UML Distilled by Martin Fowler with Kendall Scott, Addison-Wesley, 1997.

An investigation into RAS Features for Massively Parallel Processor Systems by Suzanne M. Kelly and Jeffry B. Ogden, SAND2002-3164, 2002.

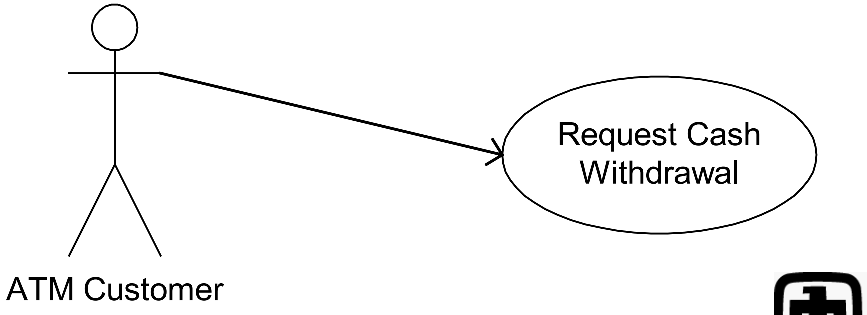
The Unified Modeling Language

- A *Standard** object modeling language
- Unifies the models of Booch, Rumbaugh (OMT) and Jacobson
- Not a method; no notion of process
- Can incorporate some or all of the UML notations and diagrams (e.g. use cases) into your software development process of choice.

* One of the nice things about standards is that there are so many to choose from – ***Andrew S. Tanenbaum***

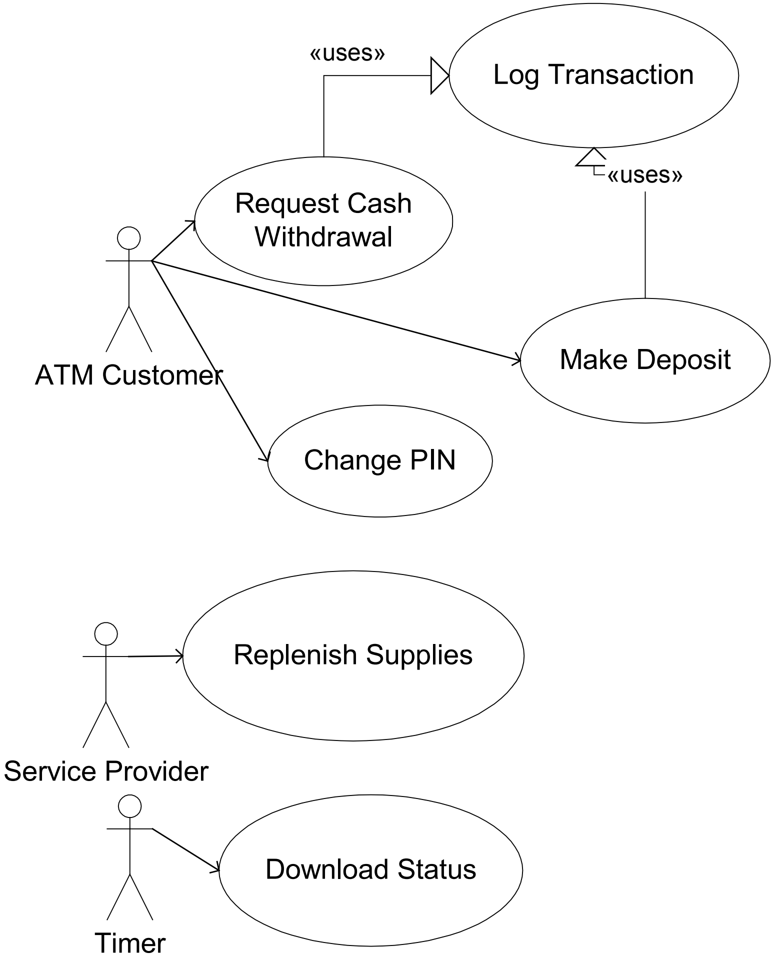
Use Case Concepts

- Use Case – A specific way of using the system by performing some part of the functionality.
- Actor – A representation of what interacts with the system. May be a person, another system, or something else (e.g. cron).
- Use cases are represented by ovals. I use a naming convention of verb followed by object. Subject is implied by the initiating actor.
- An actor is represented by a stick figure.
- An arrow indicates the direction of initiation (not necessarily data flow).



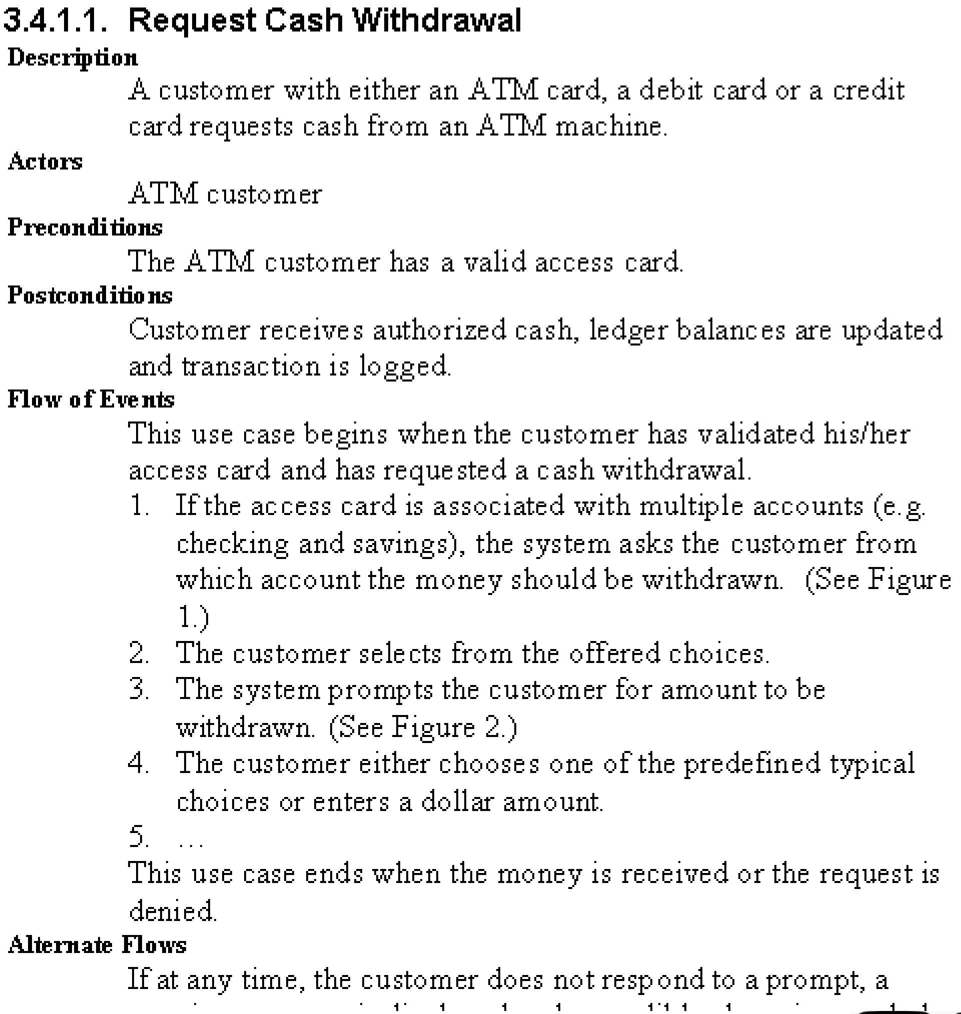
Use Case Concepts (cont.)

- **Each use case constitutes a complete course of events initiated by an actor and specifies the interactive between the actor and the system**
- **Use Case Diagram – a graphical representation of the entire set of actors and use cases.**
- **Use Case Model – the use case diagram plus the descriptive text for each use case.**



Use Case Documentation

- My preferred template for each use case:
  - Description - one or two lines
  - Actors - list
  - Pre & Post conditions
  - Detailed Flow of Events
  - Alternate Flows
  - User Interface
  - Data Requirements



3.4.1.1. Request Cash Withdrawal

Description

A customer with either an ATM card, a debit card or a credit card requests cash from an ATM machine.

Actors

ATM customer

Preconditions

The ATM customer has a valid access card.

Postconditions

Customer receives authorized cash, ledger balances are updated and transaction is logged.

Flow of Events

This use case begins when the customer has validated his/her access card and has requested a cash withdrawal.

1. If the access card is associated with multiple accounts (e.g. checking and savings), the system asks the customer from which account the money should be withdrawn. (See Figure 1.)
2. The customer selects from the offered choices.
3. The system prompts the customer for amount to be withdrawn. (See Figure 2.)
4. The customer either chooses one of the predefined typical choices or enters a dollar amount.
5. …

This use case ends when the money is received or the request is denied.

Alternate Flows

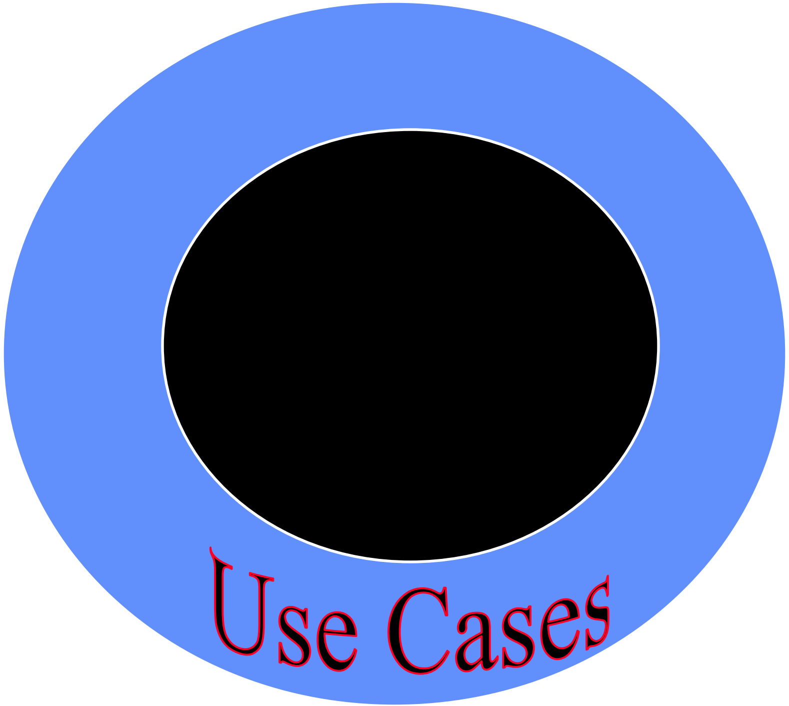
If at any time, the customer does not respond to a prompt, a

The Value of Use Cases

- A customer-friendly way of describing functional and performance requirements
- A good basis for developing test cases
- An excellent basis for developing the user guide
- Can be applied even if not using object-oriented development (OOAD)
- A great place to rough-out the GUI
- A great place to start finding your data requirements

What Use Cases Do Not DO

- They only define the customer visible portion of the system.
- They provide minimal information for system architecture design.



Use Case Model of a RAS system for MPPs

Definition of RAS

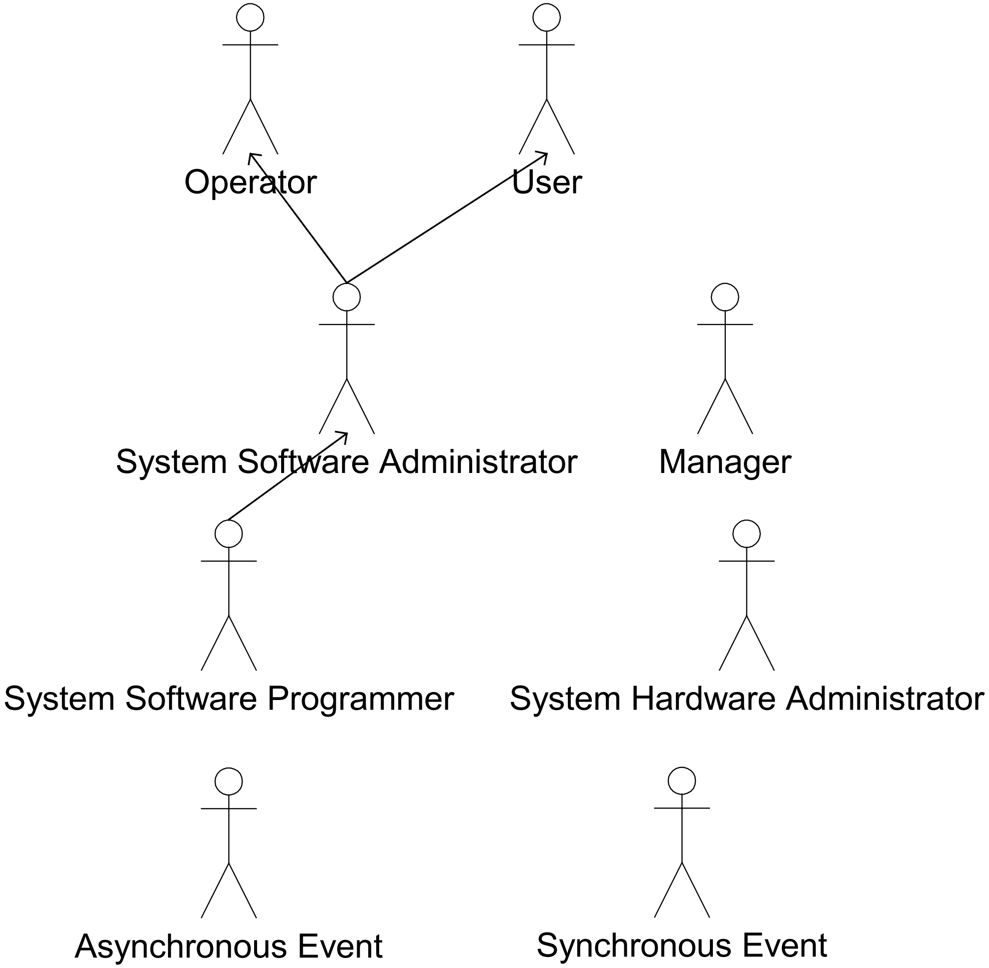
- ***Reliability*** **- fault avoidance**
  - **the likelihood a system or component will sustain full functional operation over its lifetime.**
  - **Measured in MTBF (mean time between failures).**
- ***Availability*** **- fault tolerance**
  - **the likelihood a system is operational at any given time.**
  - **Measured in up time percentage.**
- ***Serviceability*** **- fault identification and repair**
  - **measure of a system’s ability to sustain repairs to faulty components.**
  - **Measured in MTTR (mean time to repair) and $$$s.**

Features of the Model

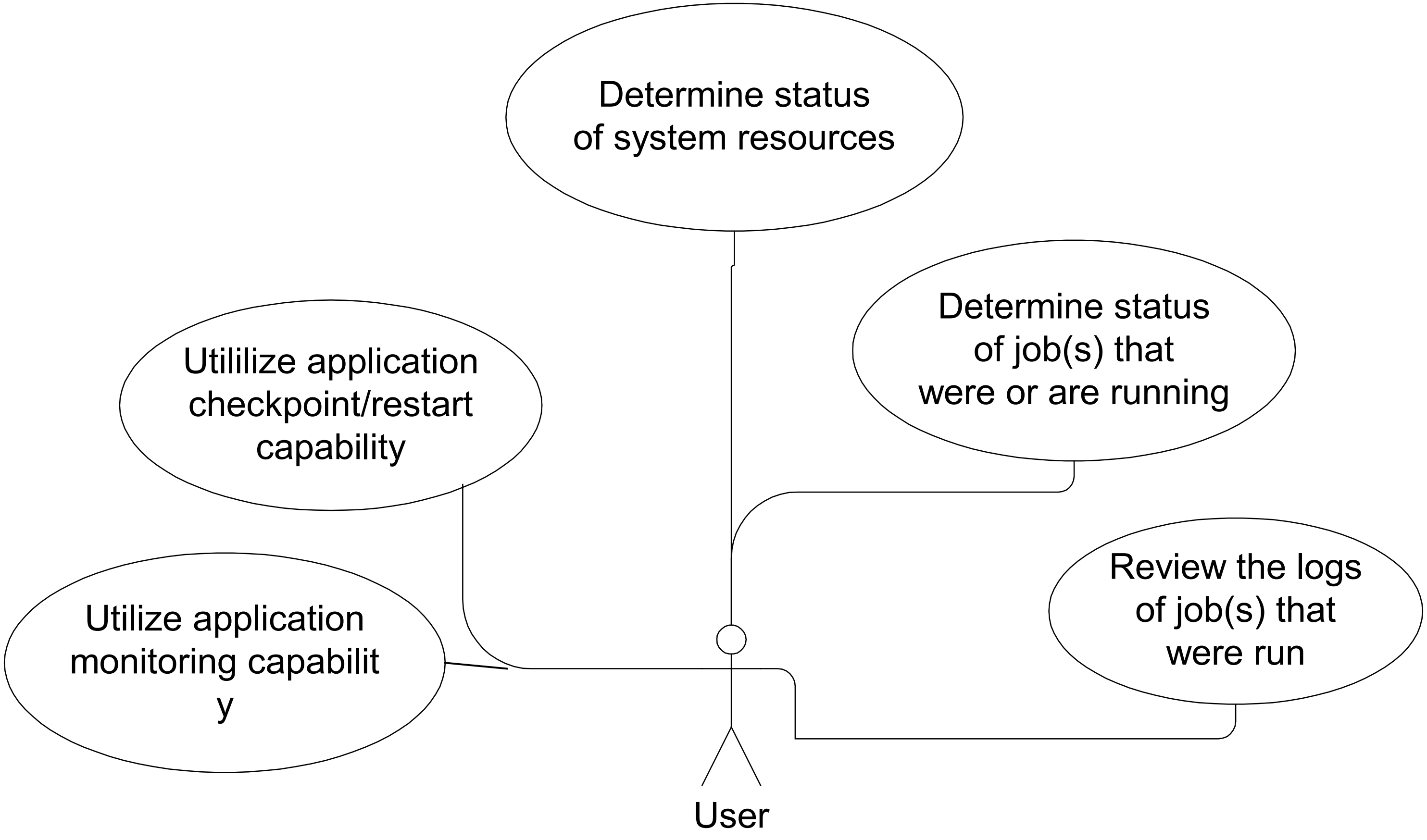
- Integrates hardware and software RAS
- Comprehensive model - I.e. includes RAS features found on the most humble PC all the way to unique MPP-unique RAS features
- Generally applicable to clusters and embarrassingly parallel systems

The Actors

- Asynchronous Event
- Manager
- Operator
- Synchronous Event
- System Hardware Administrator
- System Software Administrator
- System Software Programmer
- User


Operator
User
System Software Administrator
Manager
System Software Programmer
System Hardware Administrator
Asynchronous Event
Synchronous Event

Use Case Diagram for User



Determine status
of system resources

Determine status
of job(s) that
were or are running

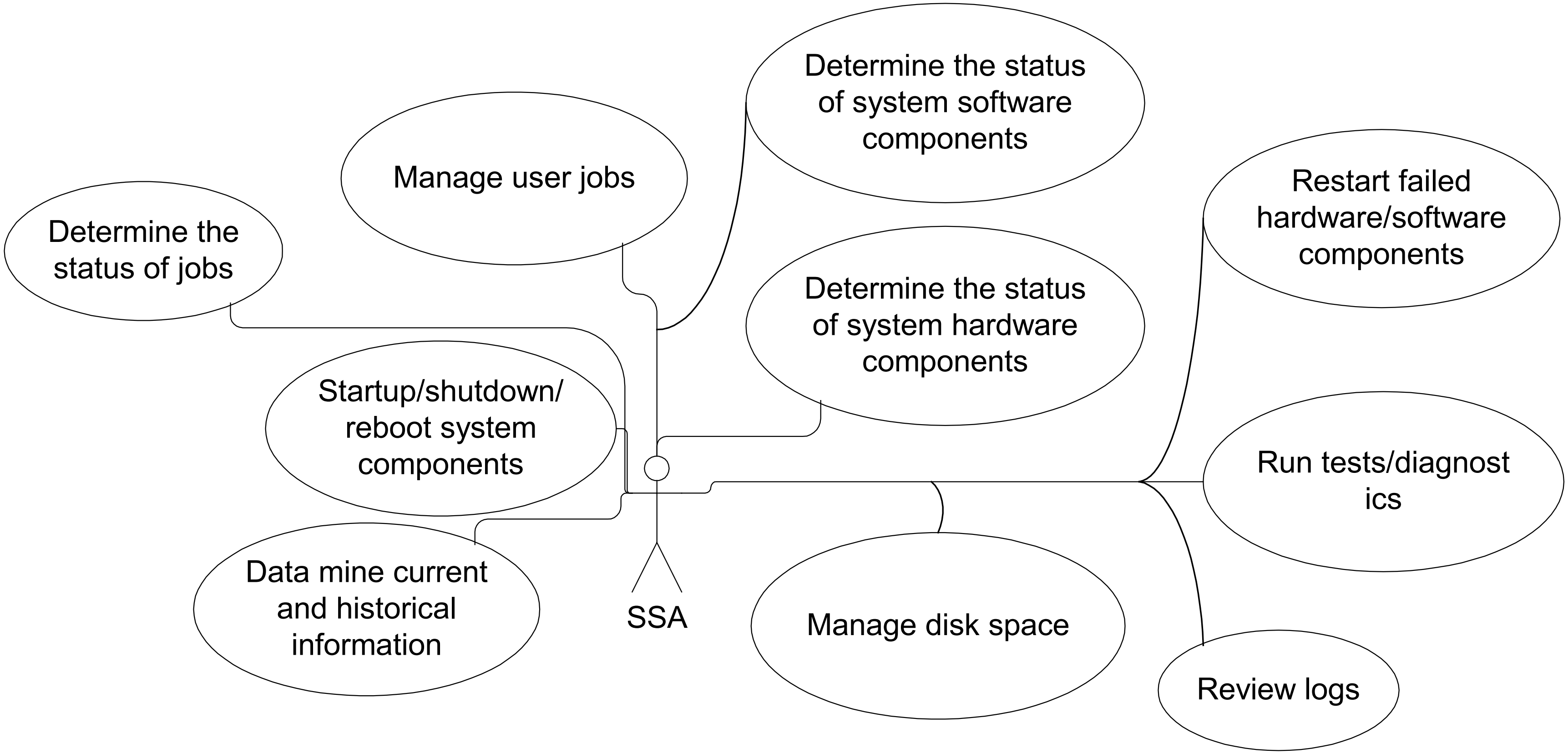
Utililize application
checkpoint/restart
capability

Review the logs
of job(s) that
were run

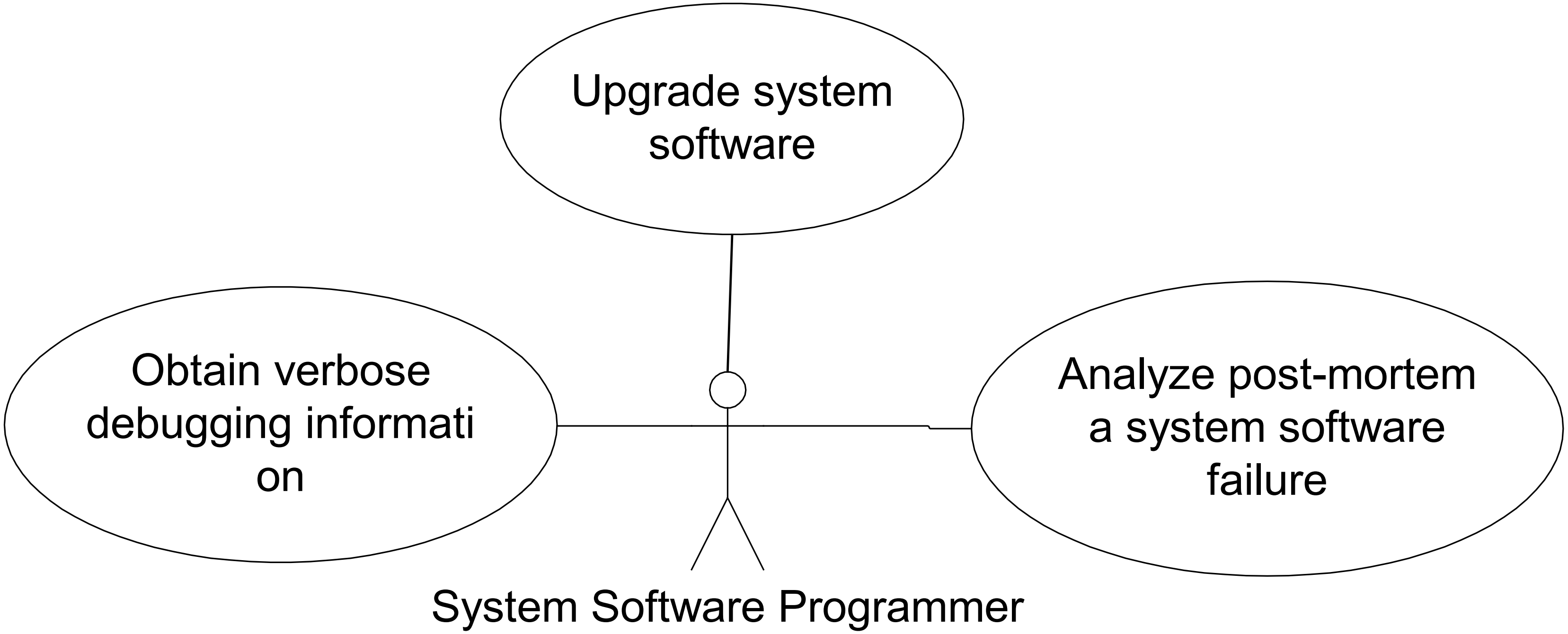
Utilize application
monitoring capabilit
y

User

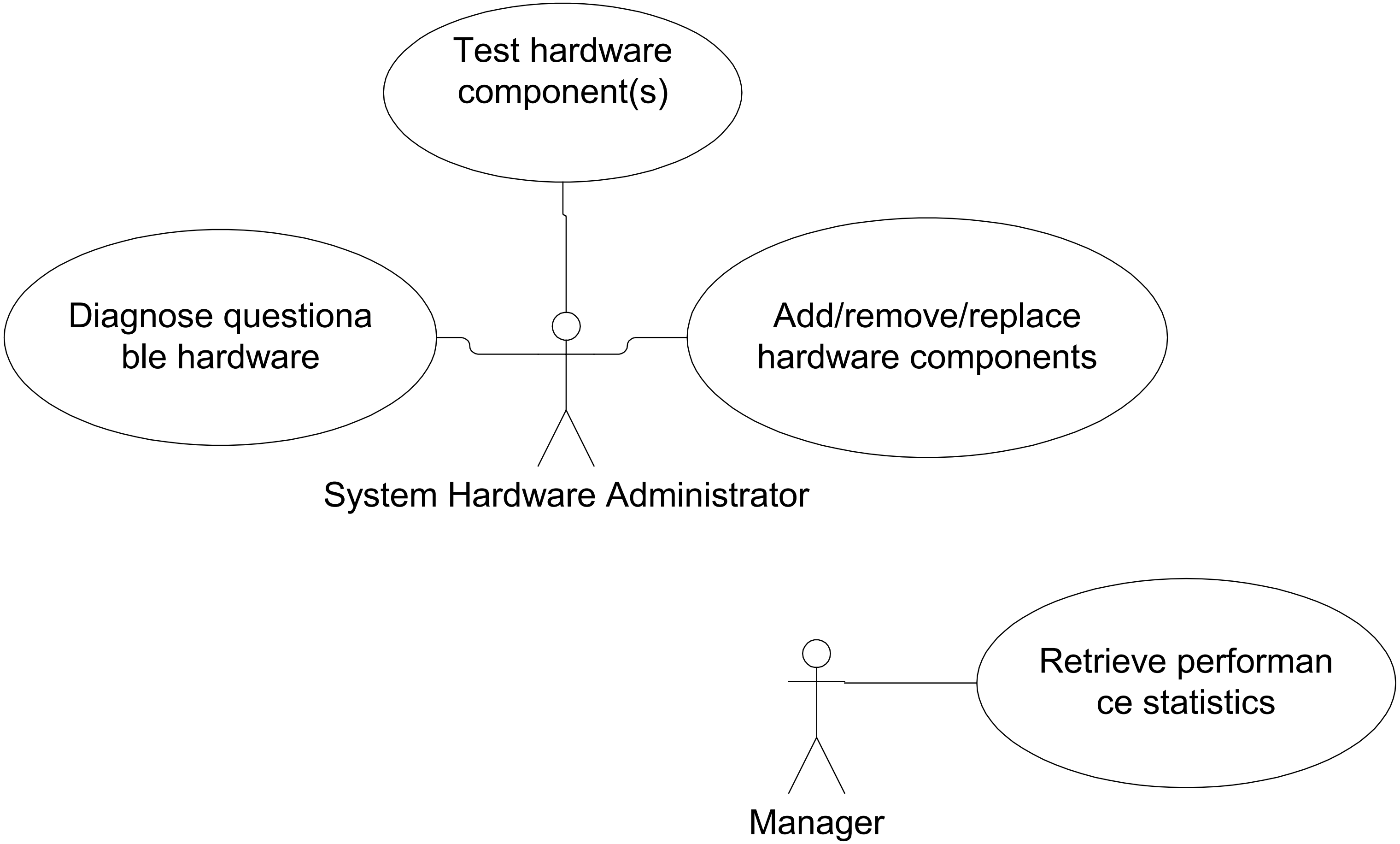
Use Case Diagram for System Software Administrator


Determine the status of system software components
Manage user jobs
Restart failed hardware/software components
Determine the status of jobs
Determine the status of system hardware components
Startup/shutdown/ reboot system components
Run tests/diagnost ics
Data mine current and historical information
SSA
Manage disk space
Review logs

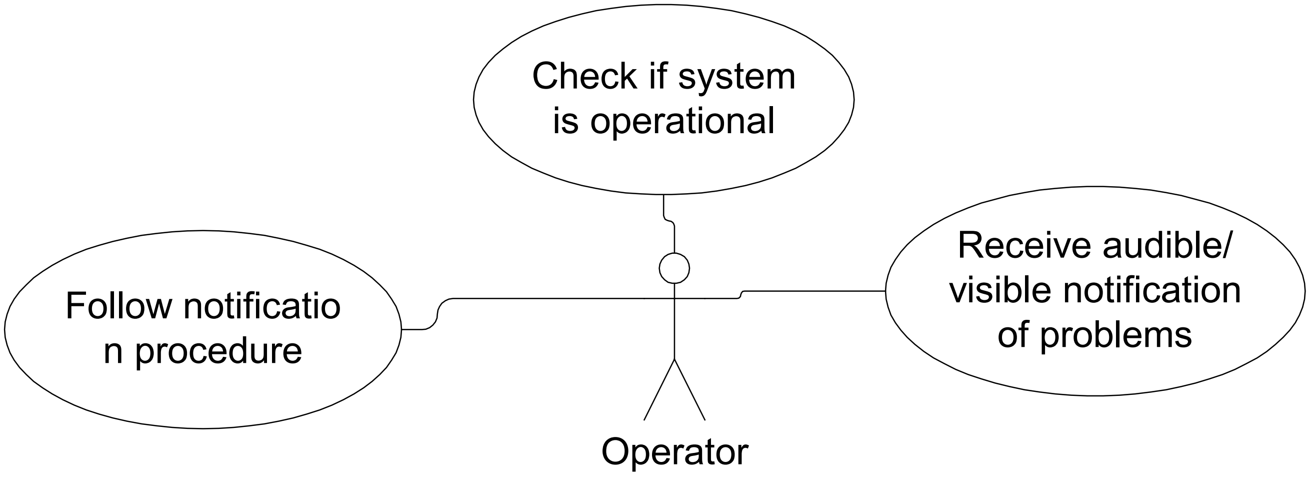
Use Case Diagram for System Software Programmer

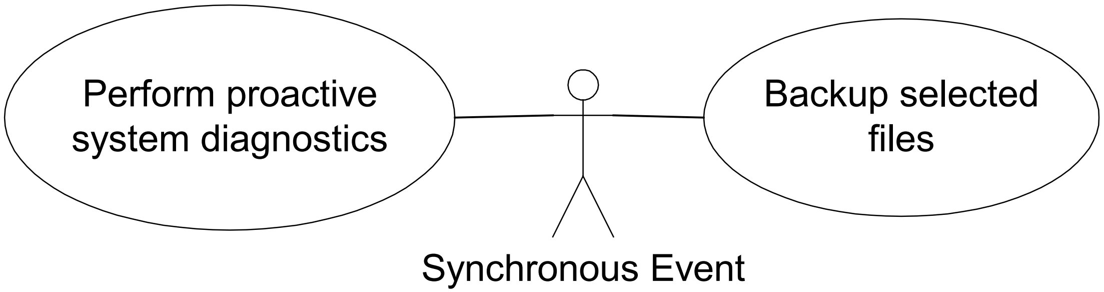

Upgrade system software
Obtain verbose debugging informati on
Analyze post-mortem a system software failure
System Software Programmer

Use Case Diagrams for System Hardware Administrator and Manager

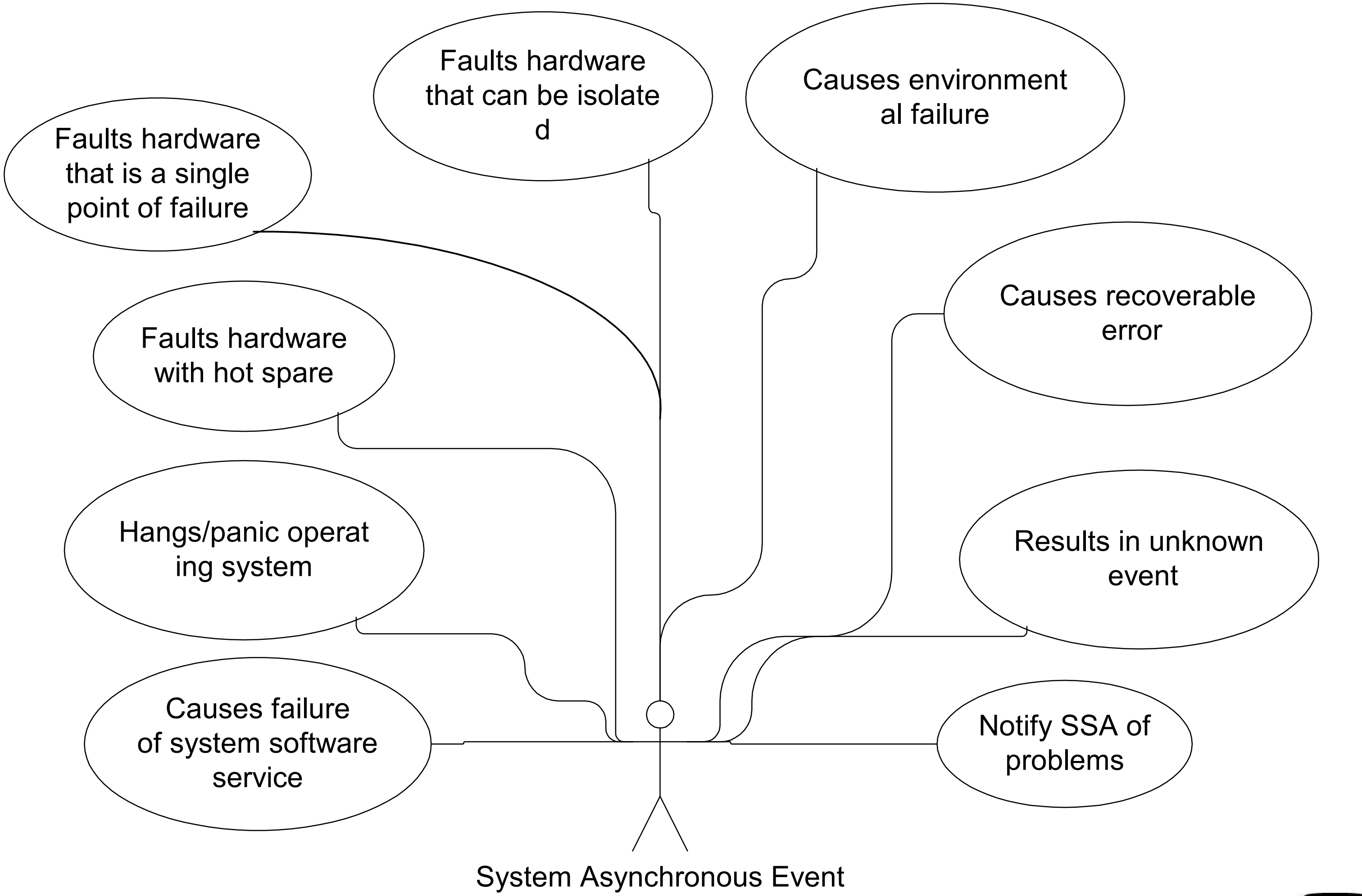

Test hardware component(s)
Diagnose questiona ble hardware
Add/remove/replace hardware components
System Hardware Administrator
Retrieve performan ce statistics
Manager

Use Case Diagrams for Operator and Synchronous Event


Check if system is operational
Follow notificatio n procedure
Receive audible/ visible notification of problems
Operator

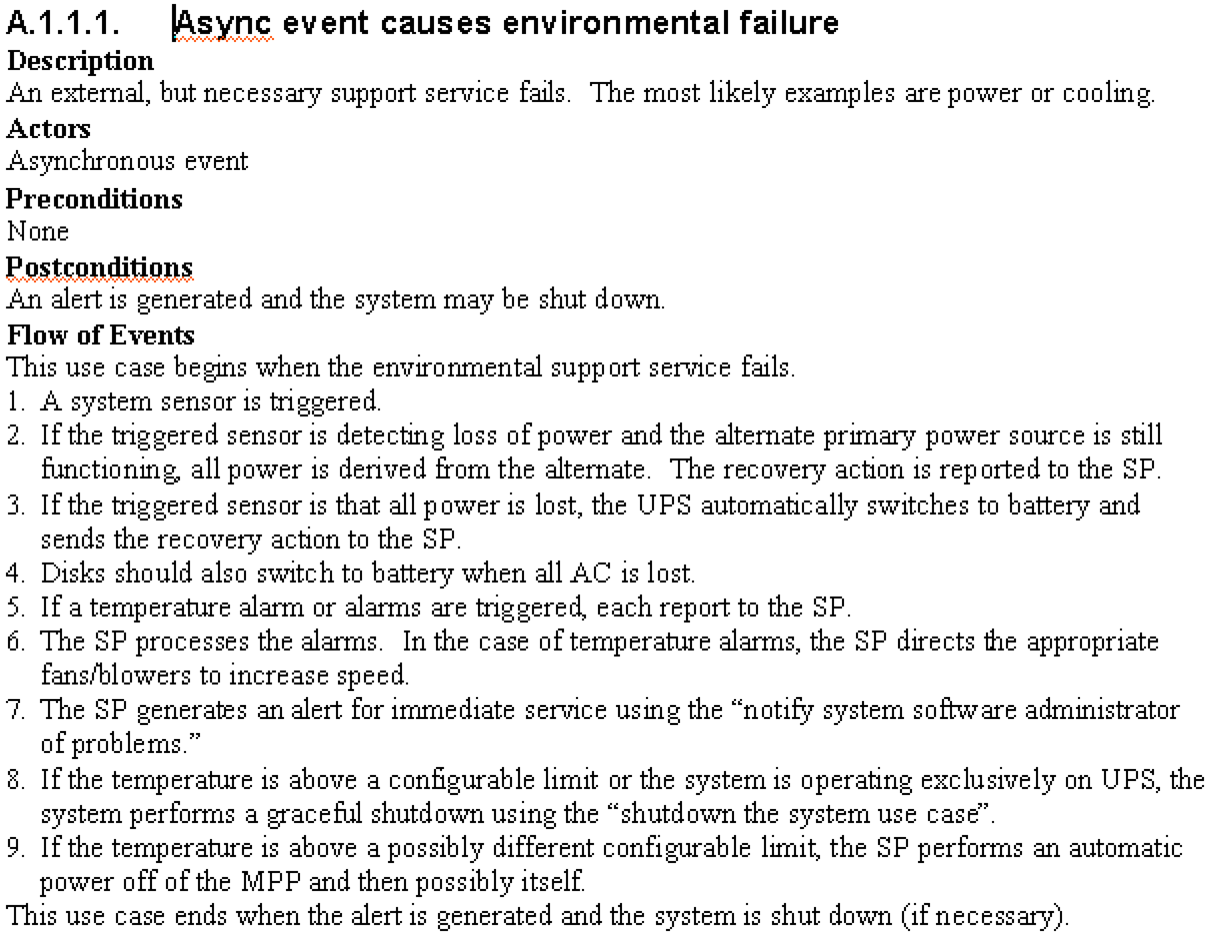


Use Case Diagram for Asynchronous Event



Faults hardware that can be isolate d
Causes environment al failure
Faults hardware that is a single point of failure
Causes recoverable error
Faults hardware with hot spare
Hangs/panic operat ing system
Results in unknown event
Causes failure of system software service
Notify SSA of problems
System Asynchronous Event

Example Use Case Description



A.1.1.1. Async event causes environmental failure

Description

An external, but necessary support service fails. The most likely examples are power or cooling.

Actors

Asynchronous event

Preconditions

None

Postconditions

An alert is generated and the system may be shut down.

Flow of Events

This use case begins when the environmental support service fails.

1. A system sensor is triggered.
2. If the triggered sensor is detecting loss of power and the alternate primary power source is still functioning, all power is derived from the alternate. The recovery action is reported to the SP.
3. If the triggered sensor is that all power is lost, the UPS automatically switches to battery and sends the recovery action to the SP.
4. Disks should also switch to battery when all AC is lost.
5. If a temperature alarm or alarms are triggered, each report to the SP.
6. The SP processes the alarms. In the case of temperature alarms, the SP directs the appropriate fans/blowers to increase speed.
7. The SP generates an alert for immediate service using the "notify system software administrator of problems."
8. If the temperature is above a configurable limit or the system is operating exclusively on UPS, the system performs a graceful shutdown using the "shutdown the system use case".
9. If the temperature is above a possibly different configurable limit, the SP performs an automatic power off of the MPP and then possibly itself.

This use case ends when the alert is generated and the system is shut down (if necessary).

Conclusions

- Use cases are an effective communication tool.
- This model is the basis for the Red Storm system.