Modeling the
Enterprise Storage Management System

Prepared By:
Dan Glasser, Madeline Hardojo, Anand Surandam, Nate Wells

for

Computer Science 466/866 – Software Design Methodology
Dr. Mohamed Fayad
University of Nebraska – Lincoln

Abstract:

The ever-expanding data storage requirements force research groups to be on their toes in inventing new techniques for storage. The ability of computers to interact with each other over a network necessitates accessing of data stored across network. Proliferation of security attacks force companies to safe guard their data by performing backups regularly. Lincoln Telephone Company required a system that will address all these issues and manage data stored across network effectively. The system is built on a 3-tier client server model. The motivation is to come up with a good model that is enduring and can be used to implement the system. We considered both traditional and stability models in describing the system. We found that traditional model has lot of inefficacies that are being addressed by the stability model. We also found that stability model actually exemplifies the goals and the purpose of the system.

Table of Contents

I. Introduction:

The Enterprise Storage Management System is designed to be used by large companies that operate large internal, distributed networks. It is designed to offer convenient access to electronic data stored by the company as well as providing a means to easily provide several layers of data protection.

Due to the expanse of the network and the business, it is not feasible for all the data required by the company to be stored in one place. Instead of having to look for the data in different locations, the transfer of information is simplified by providing one uniform interface that provides access to all the data stored electronically by the company. Even though this interface will make it appear that all the data lies in one single location, the data can actually be distributed to different locations, which reduces the risk of lost of data due to catastrophes, such as fire.

The second purpose of the Enterprise Storage Management System is to provide safe storage of the company's data. This is mainly achieved by the backup of the data used by the company. Again, the data at one location can be backed up to a physically separate location to guard against local loss of data.

The Enterprise Storage Management System is designed to adapt to physical changes in the network, such as the addition of storage devices. It also is designed to increase the efficiency of the storage of electronic data by using compression techniques and by archiving inactive files. It can be installed with minimal changes to the network infrastructure as well as little or no training of the basic users of the network. Once installed and configured, it will operate on its own with minimal human supervision.

The process of designing this system started with a detailed analysis of the problem statement. This involved the development of a use case diagram, which showed the underlying and crucial aspects of the problem. This helped immensely later in the design of a stability model. A detail description of each use case was written with the aid of this use case diagram. With the core knowledge of the problem from the use case diagram and use case descriptions, a traditional class diagram was developed, along with the complementary CRC cards.

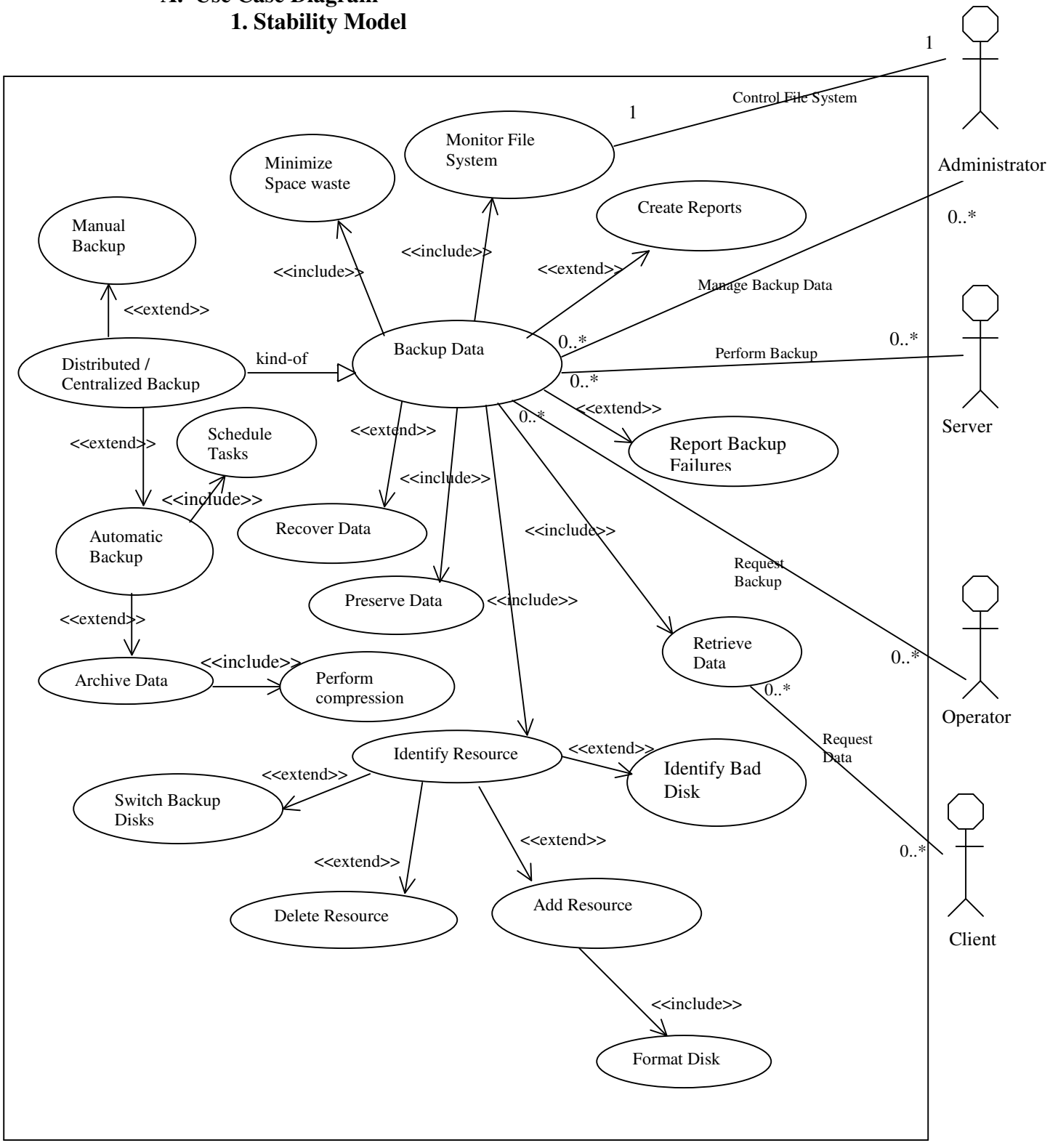
By again using the knowledge of the problem situation, the EBT's and BO's were identified and the whole process was repeated. This included the development of a stability model and the CRC cards that go along with it. This model represented a complete and stable model of the system. In order to fully understand and evaluate the behavior of this design, behavior diagrams, including sequence diagrams and state transitions diagrams, were used. By examining these diagrams, the correctness of the design was confirmed.

As a final step, the whole process was examined to come up with Design and Analysis Patterns that will help speed this process on separate projects. Two EBT's were patterned: Storage and Efficiency. Two BO's were also patterned: Schedule and Compression.

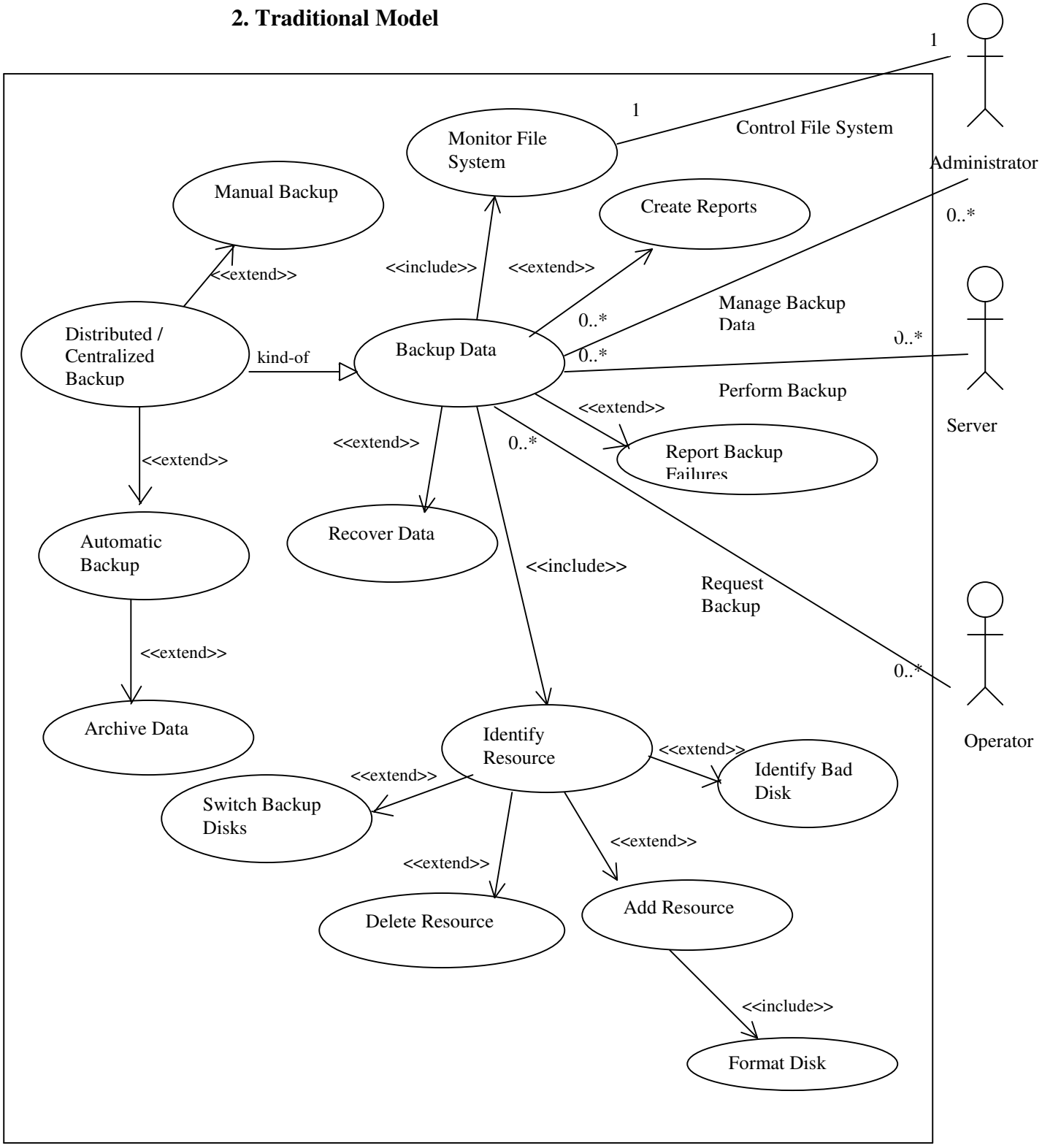
II. Use Case Model:

A. Use Case Diagram

1. Stability Model



2. Traditional Model



B. Use Case Description

Use Case No.: 1
Use Case Title: *Backup Data*
Actors/Roles

Actors	Roles
Administrator	Data Manager
Operator	Data Protector
Server	Backup Performer

Classes:

Class	Attributes	Operations
Data Protection	Data	PreserveData()
Data Backup	Data	CopyDevice()
User	ID, Name, Type, Password, Permission	GetUserType(), SetUserType()
Storage	Location	Deposit(), Withdraw()
Administrator	ID, Name, Type, Password, Permission	BackupData()
Operator	ID, Name, Type, Password, Permission	BackupData()
Allocator	Statistics	AllocateDisk()
Disk System	Disk System Type, Total Size, Total Free Space	GetFreeSpace(), AllocateSpace(), getDiskID()
Backup	Type Name	BackupData()
Regular	Type Name	SendData()
Disk Backup	Log, Compression Type	Backup()
Primary	Device Type	StoreData()
Secondary	Device Type	RestoreData()
Computer	MachineID, numDisk, total Space	TransferData()
Server	ServerID, machineID, processed	RequestedData()

EBTs: Data Protection, Storage
BOs: User, Data Backup, Allocator, Primary, Secondary
IOs: Disk System, Backup, Regular, Disk Backup, Administrator, Operator, Computer, Server
Description:

1. Backup request is generated from the users (Administrator or Operator) through the Allocator.
2. For efficiency, Allocator allocates disk to where the backup data will be backup.
3. Allocator identifies the resource to find out whether there is a space in the Disk backup.
4. If there is a space, the backup request will pass to the Disk System.
5. The Disk system will indicate where is the location of the data to be backup in the regular disk system.
6. The Disk backup will facilitate the request to Data Backup, retrieve the data from the location in regular disk system and copy it the backup disk system.

7. Disk System will record the location of the backup data in the backup disk system.

Alternatives:

4. If there is no space, Allocator will create an error log to the users to let the users know about the problem

Use Case No.: 2

Use Case Title: *Distributed/Centralized Backup*

Actors/Roles

Actors	Roles
Administrator	Data Manager
Operator	Data Protector

Classes:

Class	Attributes	Operations
Data Protection	Data	PreserveData()
Data Backup	Data	CopyDevice()
User	ID, Name, Type, Password, Permission	GetUserType(), SetUserType()
Storage	Location	Deposit(), Withdraw()
Administrator	ID, Name, Type, Password, Permission	BackupData()
Operator	ID, Name, Type, Password, Permission	BackupData()
Allocator	Statistics	AllocateDisk()
Disk System	Disk System Type, Total Size, Total Free Space	GetFreeSpace(), AllocateSpace(), getDiskID()
Backup	Type Name	BackupData()
Regular	Type Name	SendData()
Disk Backup	Log, Compression Type	Backup()
Primary	Device Type	StoreData()
Secondary	Device Type	RestoreData()

EBTs: Data Protection, Storage

BOs: User, Data Backup, Allocator, Primary, Secondary

IOs: Disk System, Backup, Regular, Disk Backup, Administrator, Operator

Description:

1. Backup request is generated from the users (Administrator or Operator) through the Allocator.
2. For efficiency, Allocator allocates disk to where the backup data will be backup.
3. Allocator identifies the resource to find out whether there is a space in the Disk backup.
4. If there is a space, the backup request will pass to the Disk System.
5. If the backup wanted is a Distributed backup, the Administrator will assign more than one backup disk as location of backup data.
6. The Disk system will indicate where is the location of the data to be backup in the regular disk system.

7. The Disk backup will facilitate the request to Data Backup, retrieve the data from the location in regular disk system and copy it the backup disk system.
8. Disk System will record the location(s) of the backup data in the backup disk system.

Alternatives:

4. If there is no space, Allocator will create an error log to the users to let the users know about the problem
5. If the backup wanted is a centralized backup, the Administrator will assign one (and only one) backup disk as location of backup data.

Use Case No.: 3

Use Case Title: *Manual Backup*

Actors/Roles

Actors	Roles
Administrator	Data Manager
Operator	Data Protector

Classes:

Class	Attributes	Operations
Data Protection	Data	PreserveData()
Data Backup	Data	CopyDevice()
User	ID, Name, Type, Password, Permission	GetUserType(), SetUserType()
Storage	Location	Deposit(), Withdraw()
Administrator	ID, Name, Type, Password, Permission	BackupData()
Operator	ID, Name, Type, Password, Permission	BackupData()
Allocator	Statistics	AllocateDisk()
Disk System	Disk System Type, Total Size, Total Free Space	GetFreeSpace(), AllocateSpace(), getDiskID()
Backup	Type Name	BackupData()
Regular	Type Name	SendData()
Disk Backup	Log, Compression Type	Backup()
Primary	Device Type	StoreData()
Secondary	Device Type	RestoreData()

EBTs: Data Protection, Storage

BOs: User, Data Backup, Allocator, Primary, Secondary

IOs: Disk System, Backup, Regular, Disk Backup, Administrator, Operator

Description:

1. Users (Administrator or Operators) will specify which file to be protected and send the request through the server to Allocator.
2. For efficiency, Allocator allocates disk to where the backup data will be backup.
3. Allocator identifies the resource to find out whether there is a space in the Disk backup.
4. If there is a space, the backup request will pass to the Disk System.

5. The Disk system will indicate where is the location of the data to be backup in the regular disk system.
6. The Disk backup will facilitate the request to Data Backup, retrieve the data from the location in regular disk system and copy it the backup disk system.
7. Disk System will record the location of the backup data in the backup disk system.

Alternatives:

4. If there is no space, Allocator will create an error log to the users to let the users know about the problem

Use Case No.: 4

Use Case Title: *Automatic Backup*

Actors/Roles

Actors	Roles
Server	Automatic Backup Performer

Classes:

Class	Attributes	Operations
Data Protection	Data	PreserveData()
Data Backup	Data	CopyDevice()
Computer	MachineID, numDisk, totalSpace	TransferData()
Storage	Location	Deposit(), Withdraw()
Server	ServerID, machineID, processId	ProcesRequest()
Allocator	Statistics	AllocateDisk()
Disk System	Disk System Type, Total Size, Total Free Space	GetFreeSpace(), AllocateSpace(), getDiskID()
Backup	Type Name	BackupData()
Regular	Type Name	SendData()
Disk Backup	Log, Compression Type	Backup()
Primary	Device Type	StoreData()
Secondary	Device Type	RestoreData()

EBTs: Data Protection, Storage

BOs: Data Backup, Allocator, Primary, Secondary

IOs: Disk System, Backup, Regular, Disk Backup, Computer, Server

Description:

1. When an item in the schedule due, the scheduler will process request to the Allocator through the server.
2. For efficiency, Allocator allocates disk to where the backup data will be backup.
3. Allocator identifies the resource to find out whether there is a space in the Disk backup.
4. If there is a space, the backup request will pass to the Disk System.
5. The Disk system will indicate where is the location of the data to be backup in the regular disk system.
6. The Disk backup will facilitate the request to Data Backup, retrieve the data from the location in regular disk system and copy it the backup disk system.

7. Disk System will record the location of the backup data in the backup disk system.
8. If the automatic backup successful, it will add entry to the backup log that contains the time of the backup.

Alternatives:

4. If there is no space, Allocator will create an error log to the users to let the users know about the problem.
5. If the automatic backup not successful (because of the network), it will immediately e-mail an error log that contains the time of the attempted backup to the system administrator.

Use Case No.: 5

Use Case Title: *Schedule Tasks*

Actors/Roles

Actors	Roles
Operator	Task Scheduler

Classes:

Class	Attributes	Operations
User	ID, Name, Type, Password, Permission	GetUserType(), SetUserType
Operator	ID, Name, Type, Password, Permission	ScheduleTask()
Efficiency		AutomateProcess()
Schedule	ScheduledTask, ScheduleOfTask	AddSchedule()

EBTs: Efficiency

BOs: User, Schedule

IOs: Operator

Description:

1. Operator will select “Schedule Tasks” interface on the system
2. Operator will choose a file to be backup automatically
3. If a schedule for that file exists, the current schedule will be displayed.
4. Operator can modify the schedule either by the time or the period of the backup should be repeated.
5. The new schedule will be saved.

Alternatives:

3. If the schedule for the selected file does not exist, the schedule will generate an empty schedule.

Use Case No.: 6

Use Case Title: *Archive Data*

Actors/Roles

Actors	Roles
Server	Data Archiver

Classes:

Class	Attributes	Operations
Efficiency		AutomateProcess()
Schedule	ScheduledTask, ScheduleOfTask	AddSchedule()

Computer	MachineID, numDisk, totalSpace	TransferData()
Server	ServerID, machineID, processId	ProcesRequest()
Primary	Device Type	StoreData()
Regular	Type Name	SendData()
Disk System	Disk System Type, Total Size, Total Free Space	GetFreeSpace(), AllocateSpace(), getDiskID()
Storage	Location	Deposit(), Withdraw()
Data Management	Data	TransferData()
Facilitator	MachineID, DiskID, dataArray	RetrieveData
Secondary	Device Type	RestoreData()
Backup	Type Name	BackupData()
Disk Backup	Log, Compression Type	Archive()

EBTs: Efficiency, Storage
BOs: Primary, Schedule, Data Management, Secondary
IOs: Server, Computer, Regular, Backup, Disk System, Facilitator
Description:

1. When an item "Compress Dormant File" is due on the schedule, scheduler will notify the server
2. The server will call the Facilitator to search through the Regular and Backup disk system in the disk system for files that have not been used for a period of time (specified by Operator in the Schedule task use case).
3. If a file is found, it will take the location, either in the Regular and Backup disk system, and pass the location to the Perform Compress action (Use case called "Perform Compress")

Alternatives:

3. If no files are found, it will create a log that say "Compress 0 files"

Use Case No.: 7
Use Case Title: *Perform Compression*
Actors/Roles

Actors	Roles
Server	Compression Performer

Classes:

Class	Attributes	Operations
Computer	MachineID, numDisk, totalSpace	TransferData()
Server	ServerID, machineID, processed	ProcesRequest()
Efficiency		MinimizeWaste()
Compression	Method	CompressObject()
Disk Backup	Log, Compression Type	Backup(), Compress()
Backup	TypeName	BackupData()
Storage	Location	Deposit()
Disk System	Disk System Type, Total Size, Total Free Space	GetFreeSpace(), AllocateSpace(), getDiskID()

EBTs: Efficiency, Storage,
BOs: Compression
IOs: Disk Backup, Backup

Description:

1. When a file is found (from the Use Case No. 6), an empty space will be identified.
2. If there exist empty space, the file is going to be compress according to the method specified in Compression.
3. The compressed file will be saved to the empty space in the backup disk.
4. The location of the compression will be stored in the Storage
5. The original file will be deleted from the disk system.
6. A log of the files that were compressed and the time of compression will be created.

Alternatives:

2. If there is not enough space for compressed file, an error log will be generated instead of the compression

Use Case No.: 8

Use Case Title: *Identify Resource*

Actors/Roles

Actors	Roles
Server	Resource Identifier

Classes:

Class	Attributes	Operations
Storage	Location	Deposit(), Withdraw()
Efficiency		AutomateProcess()
Allocator	Statistics	AllocateDisk()
Secondary	Device Type	RestoreData()
Disk System	Disk System Type, Total Size, Total Free Space	GetFreeSpace(), AllocateSpace(), getDiskID()
Backup	Type Name	BackupData()

EBTs: Storage, Efficiency

BOs: Allocator, Secondary

IOs: Disk System, Backup

Description:

1. The device that is being used for backup is full or a device becomes corrupt that needs to be removed from the network.
2. The Administrator chooses a device from the available sets of devices.
3. Once the device is chosen, two options are available. Administrator can either add the device as a new backup device to the network or delete the device from the network.
4. If “Add device” is chosen, then the device is formatted with the appropriate file system by class Disk System. Now the device is ready to replace the device that is full.
5. The administrator adds the device to the network.
6. If “Delete Device” is chosen, then the data in the device can be backup if needed.
7. The device is deleted from the network.

Alternatives:

5. If there is a network problem, the “add device” fails. The administrator is sent an appropriate message.

7. If there is a network problem, the “delete device” fails. The administrator is sent an appropriate message.

Alternatives:

Use Case No.: 9
Use Case Title: *Switch Backup Disk*
Actors/Roles

Actors	Roles
Server	Backup Disk Switcher

Classes:

Class	Attributes	Operations
Storage	Location	Deposit(), Withdraw()
Secondary	Device Type	RestoreData()
Disk System	Disk System Type, Total Size, Total Free Space	GetFreeSpace(), AllocateSpace(), getDiskID()
Backup	Type Name	BackupData()
Primary	Device Type	StoreData()
Regular	Type Name	SendData()
Computer	MachineID, numDisk, totalSpace	TransferData()
Server	ServerID, machineID, processed	ProcesRequest()
Allocator	Statistics	AllocateDisk()

EBTs: Storage
BOs: Primary, Secondary, Allocator
IOs: Disk System, Computer, Server, Regular, Backup
Description:

1. When the allocator notices that the current backup disk or regular disk is full (From “Identify Resource” use case), the allocator will mark the current backup or regular disk so that it will not be used anymore.
2. If another disk in the Disk system is found, the allocator will write to the new disk instead.
3. The Disk System will save the modification to reflect in the file system.

Alternatives:

1. If there is no more disk in the Disk system is found, the allocator will notify the Administrator and suggest another disk to be added and create an error for the request.

Use Case No.: 10
Use Case Title: *Delete Resource*
Actors/Roles

Actors	Roles
Server	Resource Removal Performer

Classes:

Class	Attributes	Operations
Storage	Location	Deposit(), Withdraw()

Disk System	Disk System Type, Total Size, Total Free Space	GetFreeSpace(), AllocateSpace(), getDiskID()
Disk	DiskID, DiskSpace, SpaceUsed	FormatDisk()
Computer	MachineID, numDisk, totalSpace	TransferData()
Server	ServerID, machineID, processed	ProcesRequest()
Allocator	Statistics	AllocateDisk()

EBTs: Storage
BOs: Allocator
IOs: Disk System, File, Computer, Server
Description:

1. A disk is physically removed from the network.
2. Allocator detects whether the changes is in Backup Disk System or in Regular Disk System.
3. If the current Backup or Regular disk is the one being removed, the allocator will call Identify Resource to allocate a new disk to 'replace' the disk being removed.
4. Allocator copies the Disk ID that being removed and pass the information to the Disk System.
5. The Disk system will modify the information to reflect the current network condition.

Alternatives:

2. If the disk being removed is not being used currently, continue to the 4th step of description.

Use Case No.: 11
Use Case Title: *Add Resource*
Actors/Roles

Actors	Roles
Server	Resource Attacher

Classes:

Class	Attributes	Operations
Storage	Location	Deposit(), Withdraw()
Disk System	Disk System Type, Total Size, Total Free Space	GetFreeSpace(), AllocateSpace(), getDiskID()
Disk	DiskID, DiskSpace, SpaceUsed	FormatDisk()
Computer	MachineID, numDisk, totalSpace	TransferData()
Server	ServerID, machineID, processed	ProcesRequest()
Allocator	Statistics	AllocateDisk()

EBTs: Storage
BOs: Allocator
IOs: Disk System File, Computer, Server
Description:

1. A disk is physically added to the network.
2. Allocator detects whether the changes is in Backup Disk System or in Regular Disk System.

3. If the disk that is added has not been formatted, the allocator will format the disk (Use case called "Format Disk")
4. Allocator copies the Disk ID that being added and pass the information to the Disk System.
5. The Disk system will modify the information to reflect the current network condition.

Alternatives:

3. If the disk that is added has been formatted, go to the 4th step of the description.

Use Case No.: 12

Use Case Title: *Format Disk*

Actors/Roles

Actors	Roles
Server	Disk Formatter

Classes:

Class	Attributes	Operations
Storage	Location	Deposit(), Withdraw()
Disk System	Disk System Type, Total Size, Total Free Space	GetFreeSpace(), AllocateSpace(), getDiskID()
Disk	DiskID, DiskSpace, SpaceUsed	FormatDisk()
Computer	MachineID, numDisk, totalSpace	TransferData()
Server	ServerID, machineID, processed	ProcesRequest()
Allocator	Statistics	AllocateDisk()

:EBTs: Storage

BOs: Allocator

IOs: Disk System File, Computer, Server

Description:

1. A new disk that has been added has not been formatted.
2. The Allocator will get the ID of the disk to be formatted and pass the information to the Disk System.
3. The Disk system will get the type of disk and format the disk according to its type.
4. If the formatting successful, an email to the administrator will be sent.

Alternative:

3. If the formatting not successful, an error log will be generated.

Use Case No.: 13

Use Case Title: *Identify Bad Disk*

Actors/Roles

Actors	Roles
Operator	Bad Disk Seeker

Classes:

Class	Attributes	Operations
Storage	Location	Deposit(), Withdraw()
Efficiency		AutomateProcess()
Allocator	Statistics	AllocateDisk()

User	ID, Name, Type, Password, Permission	GetUserType(), SetUserType()
Operator	ID, Name, Type, Password, Permission	ScheduleTask()
Disk System	Disk System Type, Total Size, Total Free Space	GetFreeSpace(), AllocateSpace(), getDiskID()
Disk	DiskID, DiskSpace, SpaceUsed	FormatDisk()
Schedule	ScheduledTask, ScheduleOfTask	PerformTask()

EBTs: Storage, Efficiency
BOs: Allocator, User, Schedule
IOs: Operator, Disk System, Disk
Description:

1. Either when Operator request for scan disk or a schedule for scan disk is due, Allocator will run a scan disk program and inspect the whole disk system for bad disk
2. If a bad disk is found, Allocator will:
    a. switch disk if the bad disk is found is being used currently and 'delete' the bad disk from the network (file system will mark the disk is being removed, even if it is not physically removed)
    b. 'delete' disk if the bad disk is not being used currently (file system will mark the disk is being removed, even if it is not physically removed)
3. Allocator will pass the Disk ID of the bad disk to the System Disk.
4. System Disk will save the modification on the structure of the disks.
5. A report will be generated for the operator/administrator to alert them of the inoperative disk to be removed physically from the network.

Alternatives:

2. If there is no bad disk found, a report will be generated that say "No Bad Disk" to mark a successful scan disk.

Use Case No.: 14
Use Case Title: *Monitor File System*
Actors/Roles

Actors	Roles
Administrator	File System Controller

Classes:

Class	Attributes	Operations
Storage	Location	Deposit(), withdraw()
Data Management	Data	TransferData()
Facilitator	DataArray, DiskID, MachineID	RetrieveData(), StoreData()
Disk system	Type, TotalSize, TotalSpace	GetDiskID
Disk	DiskID, Space, SpaceUsed	FormatDisk
File	ID, Name, Type, Size	GetFileName, getFileSize, open, close, append
User	ID, Name, Type, Passwd, Permission	getUserType
Administrator	ID, Name, Type, Passwd,	monitorFiles()

	Permission	
Allocator	Statistics	fixBadDisk

EBTs: Storage
BOs: Data Management, User, Allocator
IOs: Facilitator, Disk System, Disk, File, Administrator
Description:

1. Administrator routinely performs check on the file systems and the storage devices for faults by running monitorFiles.
2. The check on the file systems is logged into log files.
3. If the check returns errors, Allocator runs fixBadDisk, which performs a check on all the files on all the file systems for faults and fixes all of them
4. A report is generated and is duly sent to both administrator and operator. This will help them either to decide to format the disk or replace the disk if the number of faults is high

Alternatives:

3. If the check doesn't return errors, a report is sent to administrator and operator. The result of the check is stored in log files.

Use Case No.: 15
Use Case Title: *Minimize Space Waste*
Actors/Roles

Actors	Roles
Administrator	File System Controller

Classes:

Class	Attributes	Operations
Efficiency		minimizeWaste()
Compression	Method	compressObject
DiskBackup	log, compressionType	backup(),compress(),archive()
Disk	diskId,diskSpace,spaceUsed	formatDisk()
Backup	TypeName	BackupData()
User	ID, Type, Name, Passwd, Permission	GetUserType()
Administrator		backupData()

EBTs:Efficiency
BOs:Compression, User
IOs:DiskBackup, Disk, Backup, Administrator
Description:

1. Efficiency performs continual searching for space usage on storage disks. Backup returns the total space required for the data storing.
2. Before storing the data for backup, Efficiency scans the disk for minimizing space storage.
3. Compression performs de-fragmentation on the disk.
4. After performing step 3, Compression runs compressObject to perform compression on the data to be stored as backup.

Alternatives:

3. If de-fragmentation is not needed, step 3 is not performed and step 4 is carried out after step 2.

Use Case No.: 16

Use Case Title: *Create Reports*

Actors/Roles

Actors	Roles
Administrator	File System Controller

Classes:

Class	Attributes	Operations
Storage	Location	deposit(),withdraw()
Primary	DeviceType	storeData()
Secondary	DeviceType	restoreData()
Regular	TypeName	storeData()
Backup	TypeName	BackupData()
DiskBackup	log, compressionType	Backup(),archive(),compress(),report_stats()
Administrator		BackupData(),monitorFiles(),createReports()
User	ID, Type, Name, Passwd, Permission	GetUserType()

EBTs:Storage

BOs:Primary, Secondary,User

IOs:Regular, Backup, DiskBackup, Administrator

Description:

1. Administrator runs createReports on a weekly, monthly or even daily basis.
2. createReports gathers data from the log files.
3. The data gathered from the data files are used to perform statistical operations
4. Administrator can get reports on backup performed during a time-period, bad disks identified, compression techniques applied, new storage devices added, etc.

Alternatives:

1. If there are no activities in the time-period selected, an empty report is sent to administrator

Use Case No.: 17

Use Case Title: *Report Backup Failures*

Actors/Roles

Actors	Roles
Server	Backup Failure Reporter

Classes:

Class	Attributes	Operations
Storage	Location	deposit(), withdraw()
Compression	Method	compressObject()
DiskBackup	log, compressionType	backup(), compress(), archive()
Server	serverId, machineId, processed	processRequest(), performBackup()

EBTs: Storage
BOs:Compression
IOs: DiskBackup, Server
Description:

1. Operator selects one of the following backup options. Automatic, Manual, Distributed and Centralized backups.
2. After selecting the option, the DiskBackup runs backup function that will perform backup of data.
3. If the backup process is not successful, the reason for the failure is logged into the error log file.
4. A failure report is being sent to operator and administrator

Alternatives:

3. If the backup process is successful, "Backup Successful" is displayed and the corresponding message is logged into log file

Use Case No.: 18
Use Case Title: *Recover Data*
Actors/Roles

Actors	Roles
Operator	Data recovery requester
Server	Request Processor

Classes:

Class	Attributes	Operations
Storage	Location	Withdraw()
Data Protection	Data	PreserveData()
Secondary	DeviceType	RestoreData()
User	ID, Type, Name, Passwd, Permission	GetUserType()
Operator	ID, Type, Name, Passwd, Permission	BackupData()
Disk Backup	Log, CompressionType	Restore()
Backup	TypeName	BackupData()
Primary	Device Type	StoreData()
Regular	TypeName	StoreData()
Facilitator	MachineId, DiskID, dataArray	StoreData()
Disk System	DiskSystemType, TotalSize, TotalSpace	getDiskID

EBTs: Storage, Data Protection
BOs: User, Primary, Secondary
IOs: Operator, Server, Regular, Backup
Description:

1. Operator receives a request of restoring a file. Operator will choose the data to be restored and send the request to the server.
2. Server sends the request to Facilitator to locate the backup data.
3. Facilitator sends the requested data information to the disk system.
4. The disk system will seek for the data in it collection of disk backup.
5. If file is found in the backup and,
    a. if it is compressed, uncompress it and pass the location to disk system.

b. If it is not compressed, pass the location to disk system

6. Disk system copies the data to the regular disk, overwriting the corrupted data in the regular disk.
7. Send a message to operator that the data has been restored.

Alternatives:

4. If no file of the requested data is found, it will send an error message to operator and state that restoration data has failed.

Use Case No.: 19

Use Case Title: *Preserve Data*

Actors/Roles

Actors	Roles
Operator	Data Preserver

Classes:

Class	Attributes	Operations
Storage	Location	deposit(), withdraw()
Compression	Method	compressObject()
DiskBackup	Log,compressionType	backup(),compress()
DiskSystem	disksystemtype,totalsize,totalspace	allocateSpace()
Operator		backupData()
User	ID, Type, Name, Passwd, Permission	GetUserType()

EBTs:Storage

BOs:Compression,User

IOs:DiskBackup, DiskSystem, Operator

Description:

1. Operator performs backup on a regularly basis. One of the following options is selected – Manual, Automatic, Distributed and Centralized.
2. DiskBackup runs backup function to backup the data in one of the available storage devices.
3. If the backup is successful, “backup successful” message is displayed and message is logged into a log file.
4. The backup function, in turn, incorporates integrity constraints that preserve data.
5. If there is an integrity violation, an error message is displayed and the message is logged into log file

Alternatives:

3. If the backup is not successful, “backup failed” message is displayed and the message is logged into a log file.

Use Case No.: 20

Use Case Title: *Retrieve Data*

Actors/Roles

Actors	Roles
Client	Data Requester

Server	Request Performer

Classes:

Class	Attributes	Operations
Storage	Location	Deposit(), Withdraw()
Data Management	Data	TransferData()
Facilitator	MachineID, DiskID, DataArray	retrieveData
User	ID, Name, Type, Password, Permission	GetUserType()
Server	ServerID, machineID, requestedData, ProcessID	ProcessRequest()
Client	ClientID, machineID, requestedData,, disked	SendRequest()
Computer	MachineID, numDisk, TotalSpace	GetData, transferData
Disk System	DiskSystemType, TotalSize, TotalSpace	getDiskID

EBTs: Storage
BOs: User, Data Management
IOs: Facilitator, Disk System, Computer, Server, Client
Description:

1. A client would like to use some data and send request to server to fetch the requested data.
2. Server processes the request and through Computer, it request for the data from facilitator.
3. Facilitator sends the requested data information to the disk system.
4. The disk system will seek for the data in it collection of disk.
5. If data is found and is a:
    a. Compressed file, the file will be uncompressed, then it will be passed to Facilitator
    b. Not a compress file, it will be passed to the facilitator.
6. Facilitator send the requested data to the computer to satisfy the client's request

Alternatives:

5. If data is not found anywhere in the system, an error message will be generated to say that the data is not exist.

III. CRC Cards

Storage (Secure Storage)		
Responsibility	Collaboration	
To allow files to be saved and used upon request	Client	Service
	Data Management Primary Secondary	Deposit Data Withdraw Data

Efficiency (System expeditor)		
Responsibility	Collaboration	
To expedite the processes of storing and managing data	Client	Service
	Allocator Compression Schedule	Organize Data

Data Protection (Security Insurer)		
Responsibility	Collaboration	
To provide security to the data management system	Client	Service
	Data Backup	Preserve Data

Data Management (Storage Manager)		
Responsibility	Collaboration	
To provide organization to the backup system	Client	Service
	Storage Facilitator	Transfer Data

Primary (Primary Storage)		
Responsibility	Collaboration	
To provide for regular storage for the system	Client	Service
	Storage Regular	Store Data

Secondary (Secondary Storage)		
Responsibility	Collaboration	
To provide for secondary storage for the system	Client	Service
	Storage Backup	Store Data

Compression (Disk Space Maximizer)		
Responsibility	Collaboration	
To provide efficiency by maximizing disk space in the system	Client	Service
	Disk Backup Efficiency	Compress Object

Schedule (Automator)		
Responsibility	Collaboration	
To automatically manage the system by scheduling and performing tasks	Client	Service
	Disk Backup Efficiency	Schedule Task Perform Task

Data Backup (Data Protector)		
Responsibility	Collaboration	
To provide data protection by backing-up the data	Client	Service
	Disk Backup Data Protection	Copy Data

Disk Backup ()		
Responsibility	Collaboration	
Backup the data in the disk system by copying the data from regular disks to backup disks.	Client	Service
	Regular Backup Data Backup Schedule Compression	Backup Compress Report Stats Archive Results Restore Data

User (System user)		
Responsibility	Collaboration	
To provide a system user with access to the system	Client	Service
	Administrator Operator	Login to the System

Allocator (Allocator)		
Responsibility	Collaboration	
Assigns location for backup data efficiently	Client	Service
	Facilitator Disk System	Back Up Data Generates Report Archives Data Compresses Data

Disk System (Disk Container)		
Responsibility	Collaboration	
Allow an interface to a collection of disks	Client	Service
	Facilitator Disk Backup Disk Regular Disk	Allocate Free Space Set Disk Id Retrieve Disk Id

Backup Disk (Secondary Storage)		
Responsibility	Collaboration	
Provide an interface to backup data	Client	Service
	Disk System Regular Disk	Back up Data Restore Data

Regular Disk (Primary Storage)		
Responsibility	Collaboration	
Provide an interface to available information	Client	Service
	Backup Disk Disk System	Store Data to Backup

Disk (Storage Media)		
Responsibility	Collaboration	
Provide physical space for storage of data	Client	Service
	Disk System File	Format Disk Provide disk information

Facilitator (Facilitator)		
Responsibility	Collaboration	
Smooth the progress of data retrieval	Client	Service
	Disk System Computer	Retrieving Data Storing Data

Computer (Data Requestor)		
Responsibility	Collaboration	
To handle the requests for information by the user	Client	Service
	Facilitator Server Client	Send Requests for Data Transfer Data

Server (Request Processor)		
Responsibility	Collaboration	
Process the clients' requests for data	Client	Service
	Computer Client	Process Requests for Data

Client (Data Accessor)		
Responsibility	Collaboration	
To provide the user with access to the desired data	Client	Service
	User Server Computer	Sends the Users' Data Request

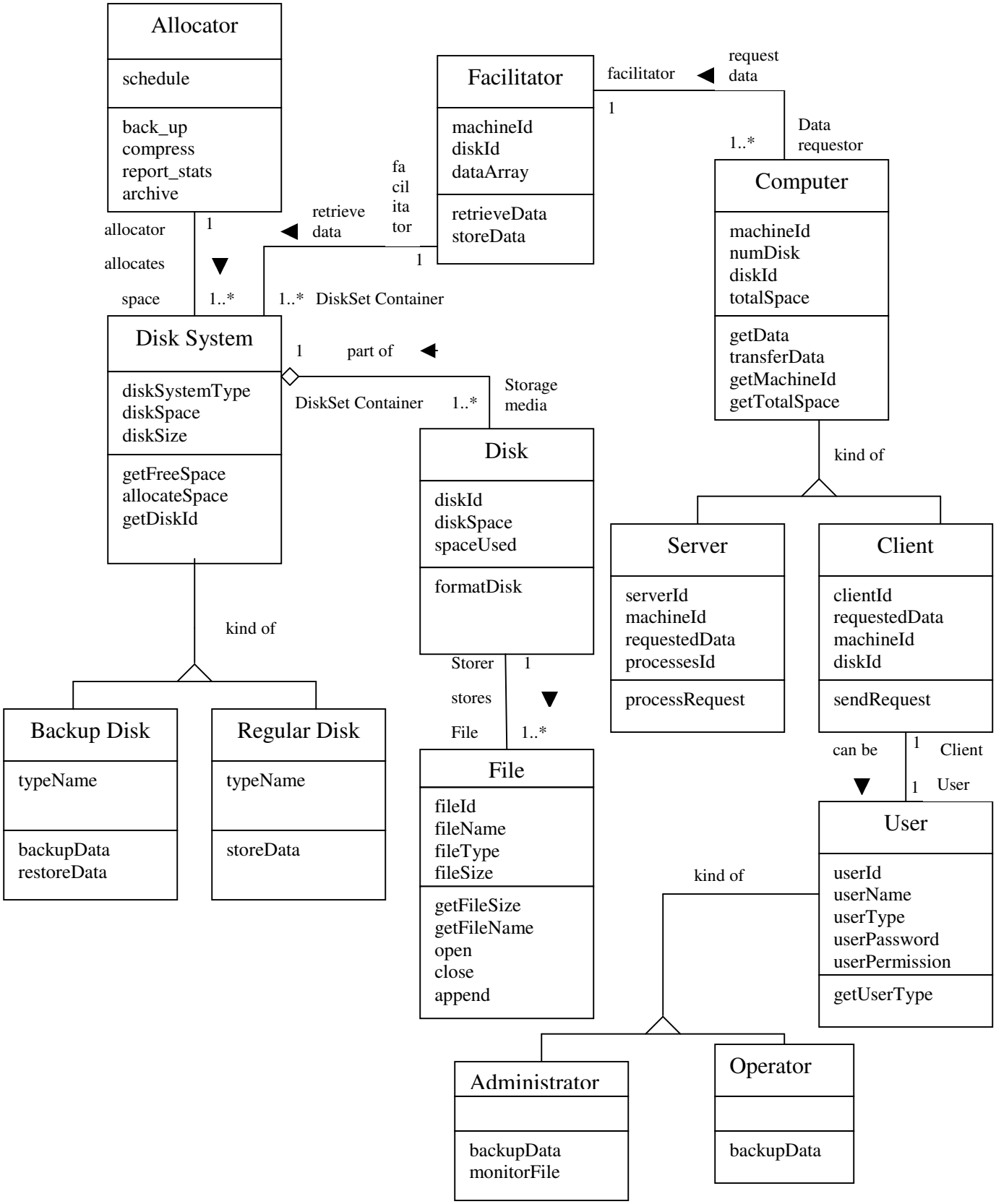
User (Data Utilizer)		
Responsibility	Collaboration	
To provide the human users access to the system and its data	Client	Service
	Administrator Operator Client	Identifies the User and Permissions

Administrator (Data Manager)		
Responsibility	Collaboration	
To provide an administrator access to the system for backing up files and for monitoring purposes	Client	Service
	User	Back up Data Monitor Files

Operator (Data Protector)		
Responsibility	Collaboration	
To give an operator access to the system to backup files	Client	Service
	User	Back up Data

File (Data Holder)		
Responsibility	Collaboration	
To store data securely and efficiently	Client	Service
	Disk	Saves Data Allows Efficient Access to Data

IV. Traditional Class Diagram



A. Description of the Traditional Class Diagram

The most important class in the diagram is the **Disk System** class. The whole system is designed to provide safe data storage on the disk system. The responsibility of every other class in the diagram involves accessing or administering the Disk System. The disk system class simply consists of a collection of **Disks**. This does not refer to a bunch of floppy disks, but rather a collection of disks that are physically attached together to form a single, coherent system. These disks store information in **Files**..

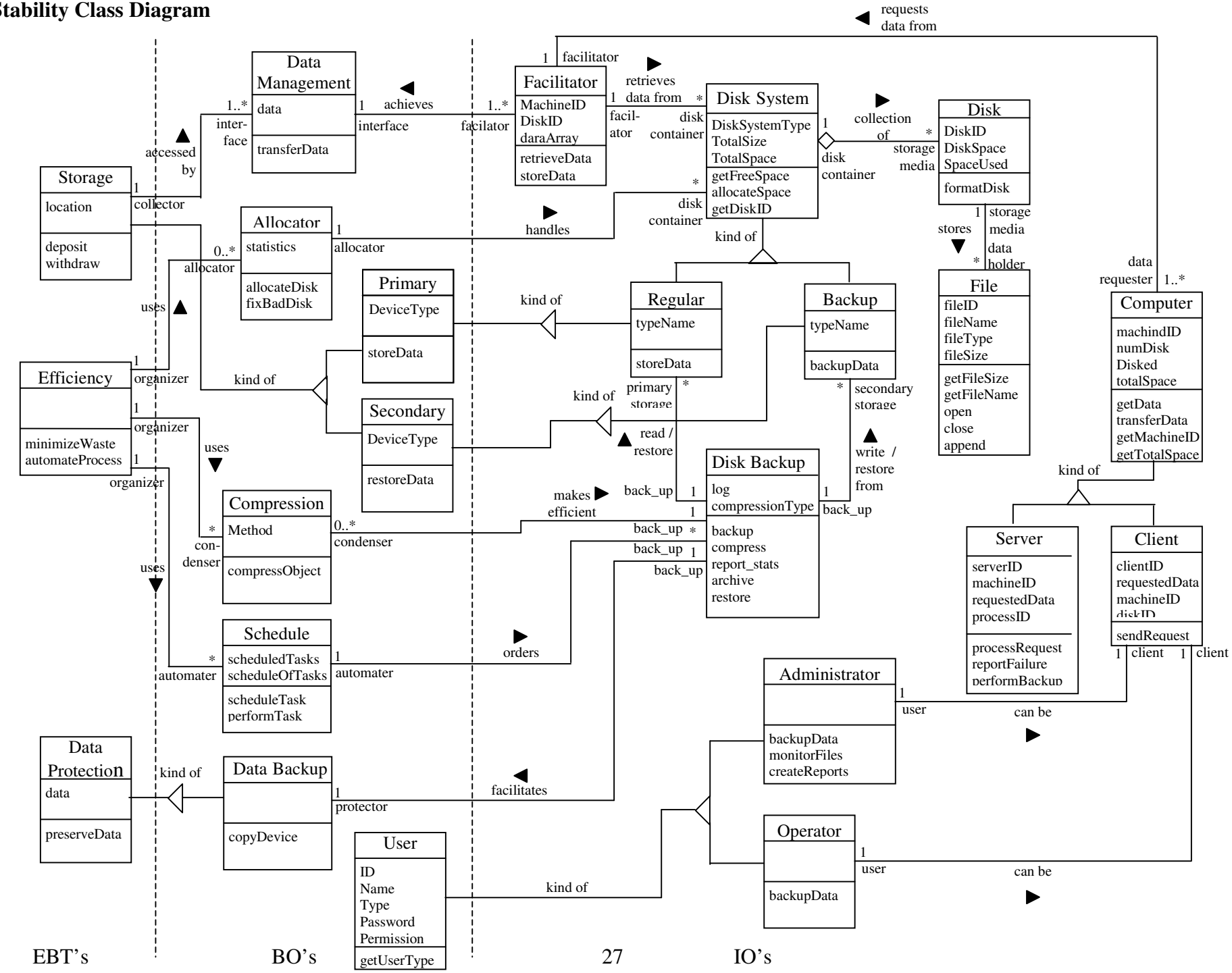
There are two kinds of Disk Systems, **Backup Disk Systems** and **Regular Disk Systems**. Files are stored directly to the Regular Disk. Information is stored to a Backup Disk only through backup procedures. The information on a Backup Disk can only be used to restore lost or damaged data on Regular Disks.

There are two classes, **Allocator** and **Facilitator,** which administer the Disk Systems. They do just what their names suggest. The Allocator class determined what portion of the disk system should be used in what manner. It is in charge of choosing which disks to use as backup disks and which to use as regular disks. It also handles the automatic and manually requested backup of data. The Facilitator class interfaces with the disk systems by providing a channel through which data can be read and written.

Everything that accesses the Disk Systems through the Facilitator is thought of as a **Computer**. This simplification is possible because even human users of the system will actually be accessing it through a Computer. Instead of considering the actions of a human, it is better to consider the corresponding actions of the Computer which the human is using as an interface.

There are two basic kinds of Computers, **Servers** and **Clients**. Servers are computers that store data or programs for use by many Clients. They both need access to the Facilitator and the Disk Systems. Clients can be other Computers, or they can in fact be human users of the system. The human users can be split into to categories: **Operators** and **Administrators**. Operators are just the basic users of the system who require normal access to the Disk System. Administrators must have access to the entire operating parameters of the system. They must be able to control the methods of backup, times of automatic backup, and access to the disk systems. They must also monitor the file system in order to be able to react promptly to any incident.

V. Stability Class Diagram



A. Description of Stability Diagram

There are three EBT's in our model: **Data Protection**, **Storage**, and **Efficiency**. These three things exemplify the purpose of the Enterprise Storage Management System. The system is designed to safely store data efficiently.

Data Protection is achieved specifically in this case by backing up the data. There are many methods of protecting data, encryption being one example, but **Data Backup** is the only method used in this system.

Storage can be classified as **Primary** storage or **Secondary** storage. Primary storage is the part of storage that is directly accessed by any entity that requests data to be read or stored. The secondary storage is used solely to provide a location where copies of primary data can be stored securely.

In order to provide easy access to the data stored in the system, there must be a system of **Data Management**. This describes a specific method, or a set of rules, that is followed to determine where to store data, how to store it, where to find requested data, and how to transfer data back and forth between the storage location and the client.

Efficiency of the system can be measured in two critical areas: time and space. Both are key qualities of computer systems. To ensure efficient use of storage space, the data can be **Compressed**, which will store the data is the smallest possible space. To make sure system operations are done on a timely basis, a **Schedule** is used. The schedule will initiate system operations at regular intervals. Operations can also be initiated whenever certain environmental variables require action by the system. For example, the system data can be backed up automatically everyday at midnight or whenever a process is terminated prematurely. An **Allocator** is used to achieve efficient use of both time and space. The allocator decides which disk in the system should be active, meaning data is kept together more, decreasing fragmented data and search time due to the locality of the data.

The only remaining BO is the **User**. The reason the user is a BO is because in the context of this system, a user is a human being. A human being is not replaceable by something different that would perform the same task. There is no EBT that corresponds to a user because a human being is purely an actor in this system. The main role of a user would be to initialize the system and define its operating parameters and conditions. Once the system is running, a user can initiate specific system operations and other things. However, the design of the system (which is what the stability model aims to capture) is to run independently of human supervision, or equivalently, without a user. The User is included in the diagram purely for the sake of completeness, because users do interact with the system, even though the offer no functionality to the system.

The IO portion of the diagram is almost the same as the traditional diagram. As mentioned before, User and Allocator were moved to the BO section. The only addition to the IO portion is the class **Disk Backup**. This class performs all backup duties of the systems. Previously, in the traditional diagram, the Allocator class handled this. However, through the development of the EBT's, it was discovered that the Allocator class had, in fact, two completely separate responsibilities. Thus, it could not be mapped to a single EBT. Notice that the Disk Backup class operates independently from the Facilitator class, which conducts regular data transfers. This compartmentally of the system increases the data security of the system.

VI. Comparison Between Traditional and Stability Class Diagrams

As discussed in the description of the Stability Class Diagram, the Stability Class Diagram includes that additional IO class called Disk Backup. This class is an important addition to the model of the system. Without it, the Allocator class was in charge of two unrelated activities. By incorporating this class, it is possible to implement the system is a way that is more Object Oriented in nature.

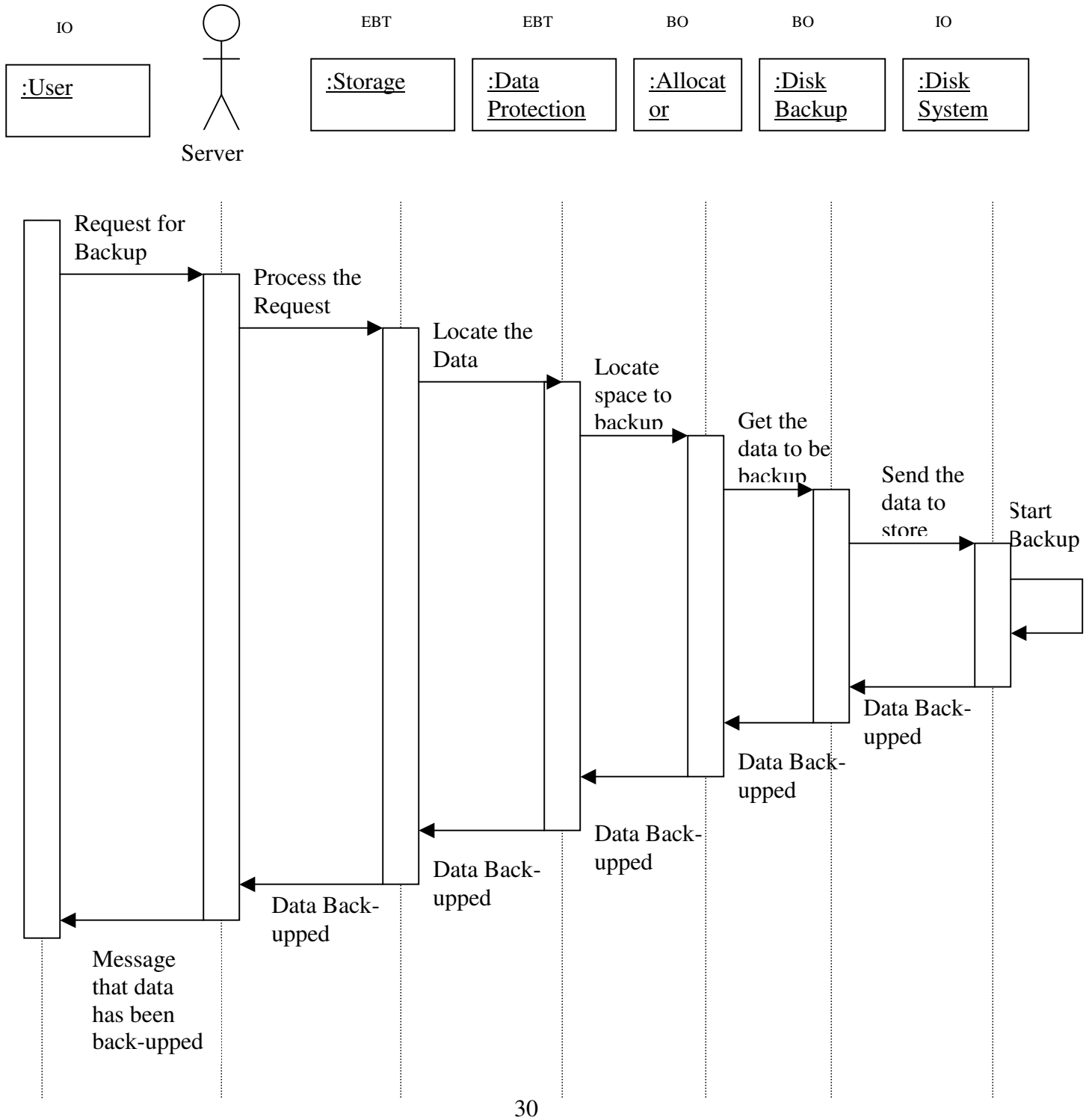
With this exception, the functionality of the two models is the same. In fact, when actually building this system, the either model could be followed with the same end result, again with the exception of the Disk Backup class. This is because when actually coding a program, it generally is not productive to make classes out of EBT's or even most BO's. This is simply because the coding of a program is based on a given programming language and compiler. These certainly are industrial objects. Therefore, there is no point to implement an EBT using a programming language. After a finite amount of time, the entire program, including that EBT, will need to be implemented using a newer, better programming language. This replacement is exactly what an EBT is designed to protect against. So it is best to not even implement EBT's or most BO's.

Again, with the exception of the Disk Backup class, either class diagram would have worked to implement this design. This certainly begs the question of why do the stability diagram at all. There are two answers to this. First of all, the stability model did in fact lead to the addition of the Disk Backup class, which will lead to a better implementation. Secondly, with a stability model in hand, a programmer has a much better idea at the underlying issues of designing the system. This will lead to better coding and a more efficient implementation.

VII. Sequence Diagram

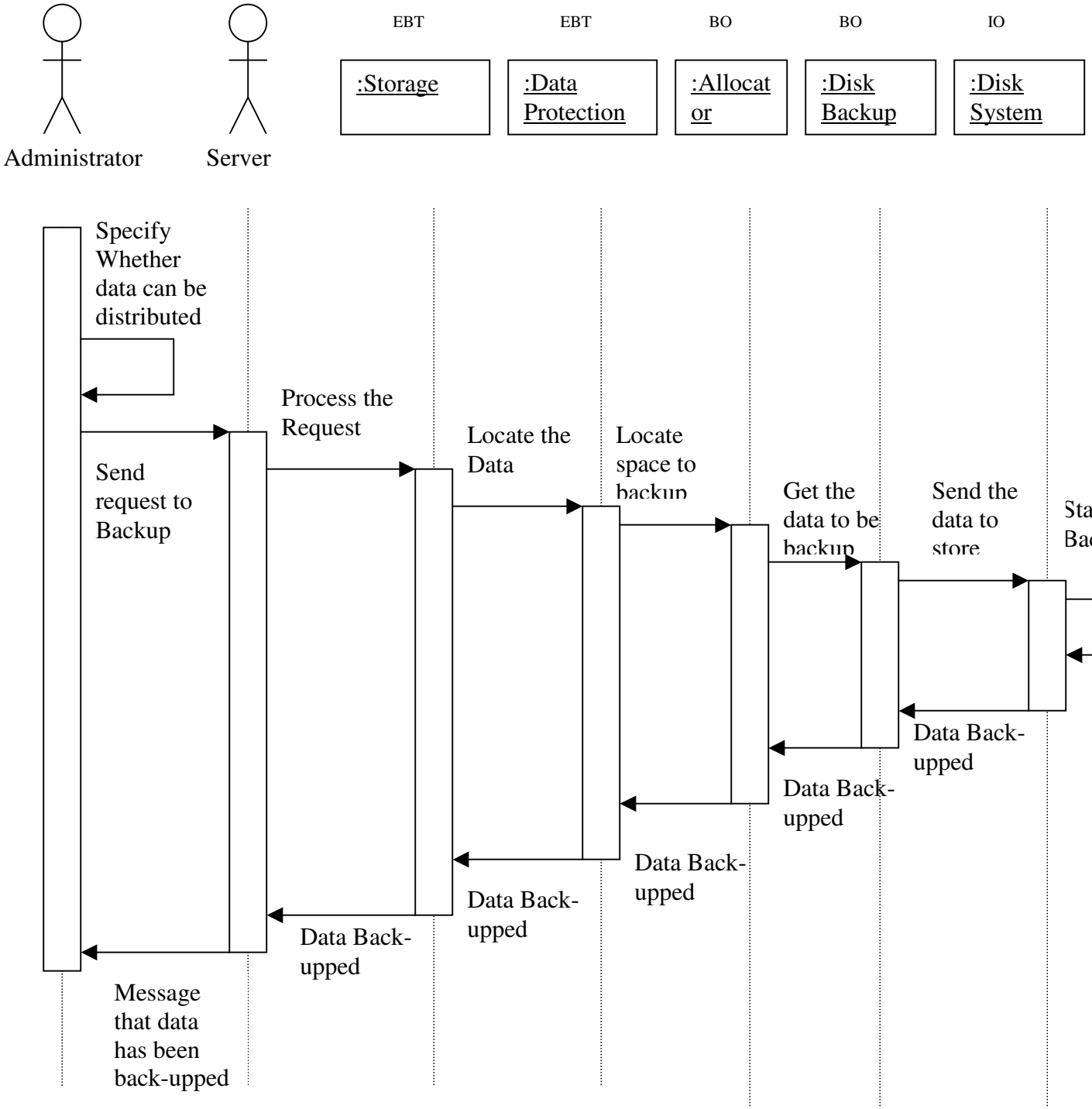
Use Case 1: Backup Data

Description: There is a request to server to backup a certain file. Server passes the data information to storage to be located whereabouts in the Disk system. When the location of the data is found, the Disk Backup sends the data to be stored in the Disk System. The backup can occur now. When the backup has finished, the information is passed all the way to the user.



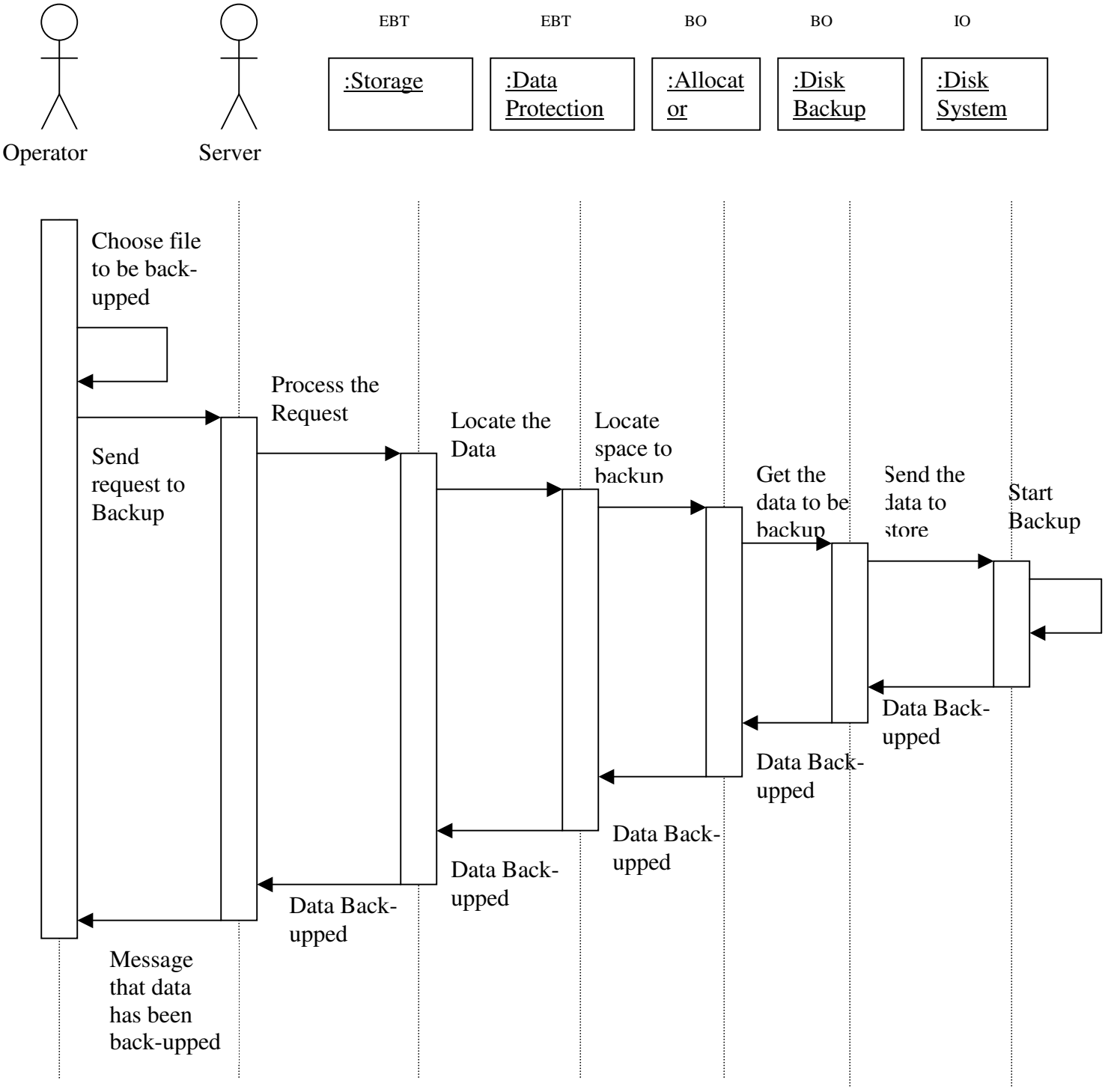
Use Case 2: Distributed/Centralized Backup

Description: There is a request from Administrator to server to backup a certain file. Previously, the administrator has decided whether the data will be backup distributed or centralized. Server passes the data information to storage to be located whereabouts in the Disk system. When the location of the data is found, the Disk Backup sends the data to be stored in the Disk System. The backup can occur now. When the backup has finished, the information is passed all the way to the user.



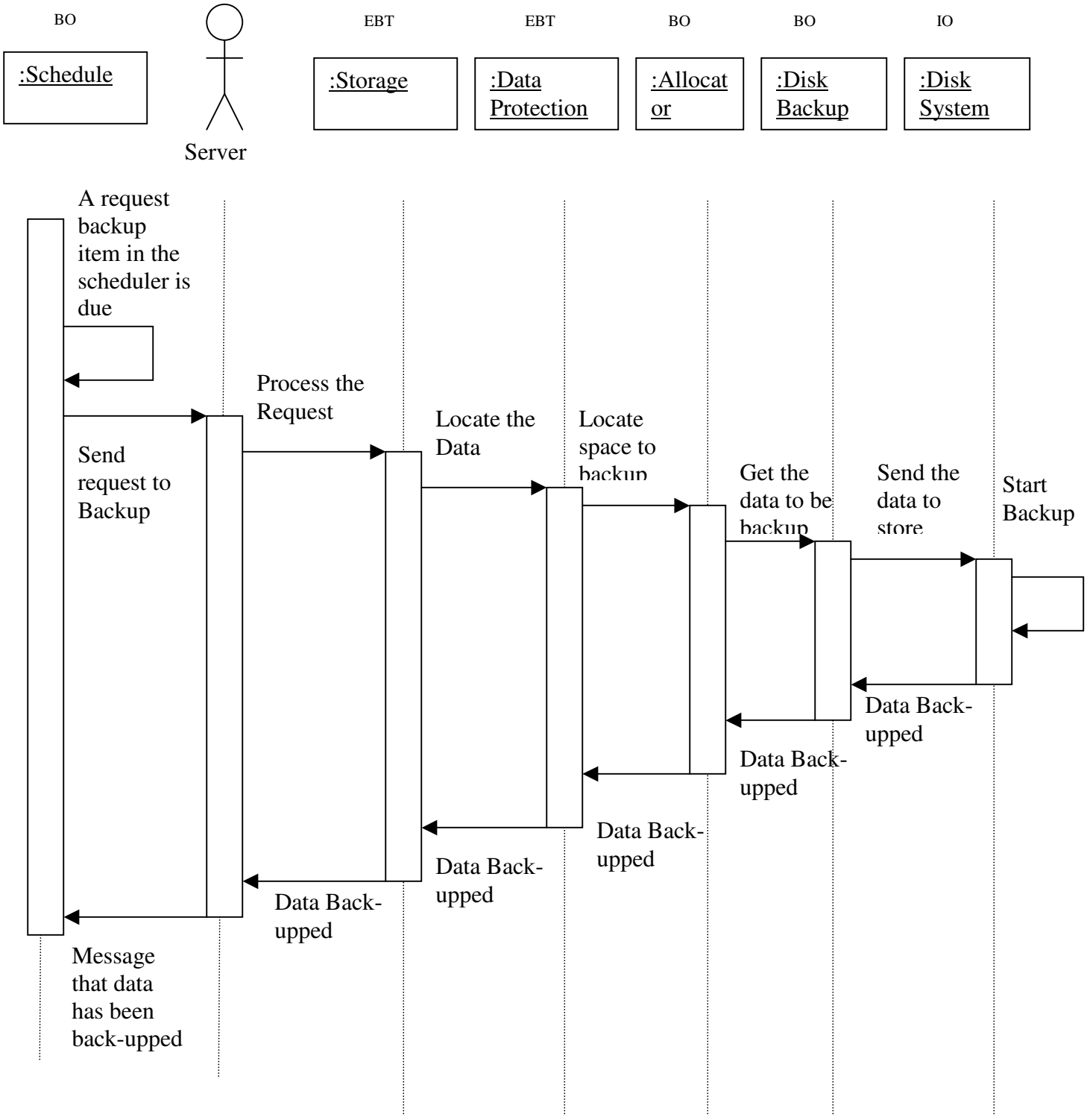
Use Case 3: Manual Backup

Description: Operator would like to backup a certain file in the system. This request is passed to the Server along with the information of the file to be back-upped. Server passes the data information to storage to be located whereabouts in the Disk system. When the location of the data is found, the Disk Backup sends the data to be stored in the Disk System. The backup can occur now. When the backup has finished, the information is passed all the way to the user.



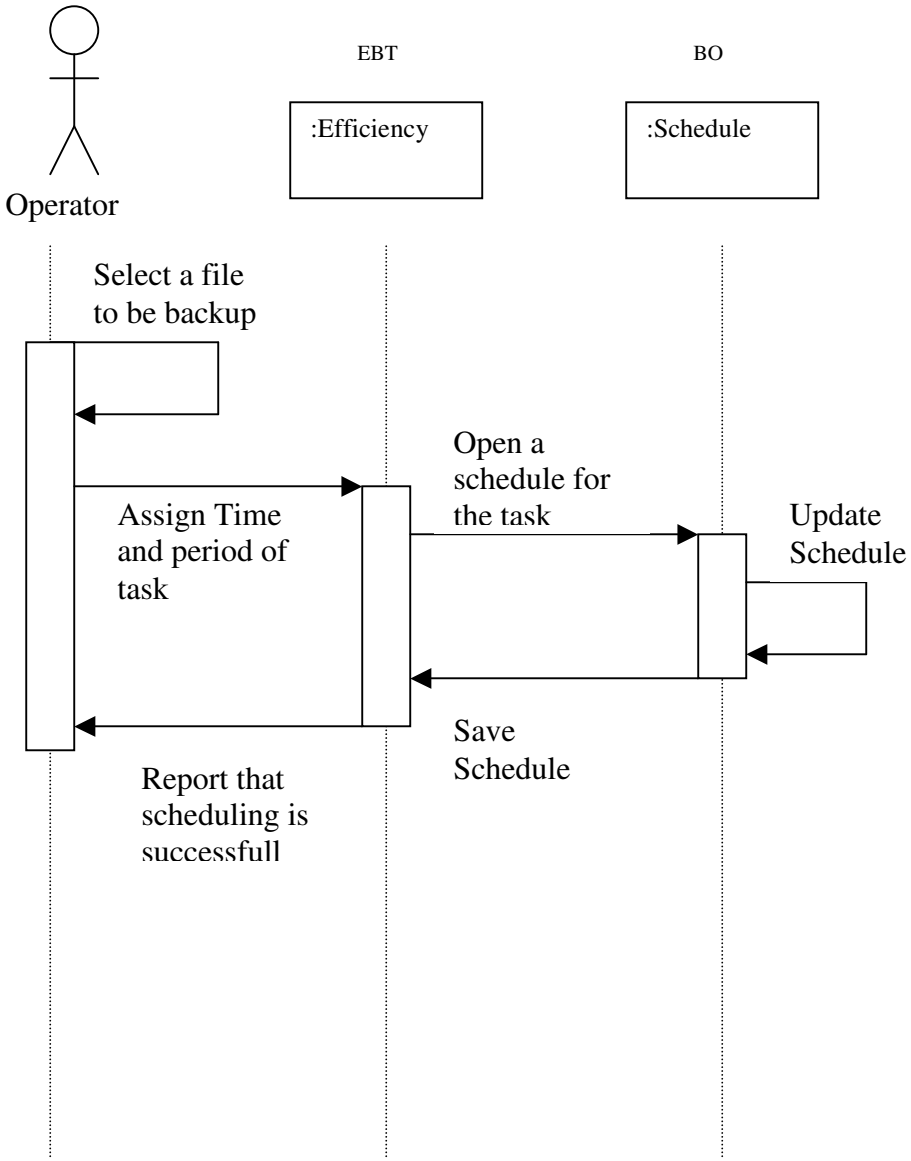
Use Case 4: Automatic Backup

Description: In automatic Backup, there is a schedule that manages when a certain file has to be backup. When there is a file to be backup, it will send request to the server. Server passes the data information to storage to be located whereabouts in the Disk system. When the location of the data is found, the Disk Backup sends the data to be stored in the Disk System. The backup can occur now. When the backup has finished, the information is passed all the way to the user.



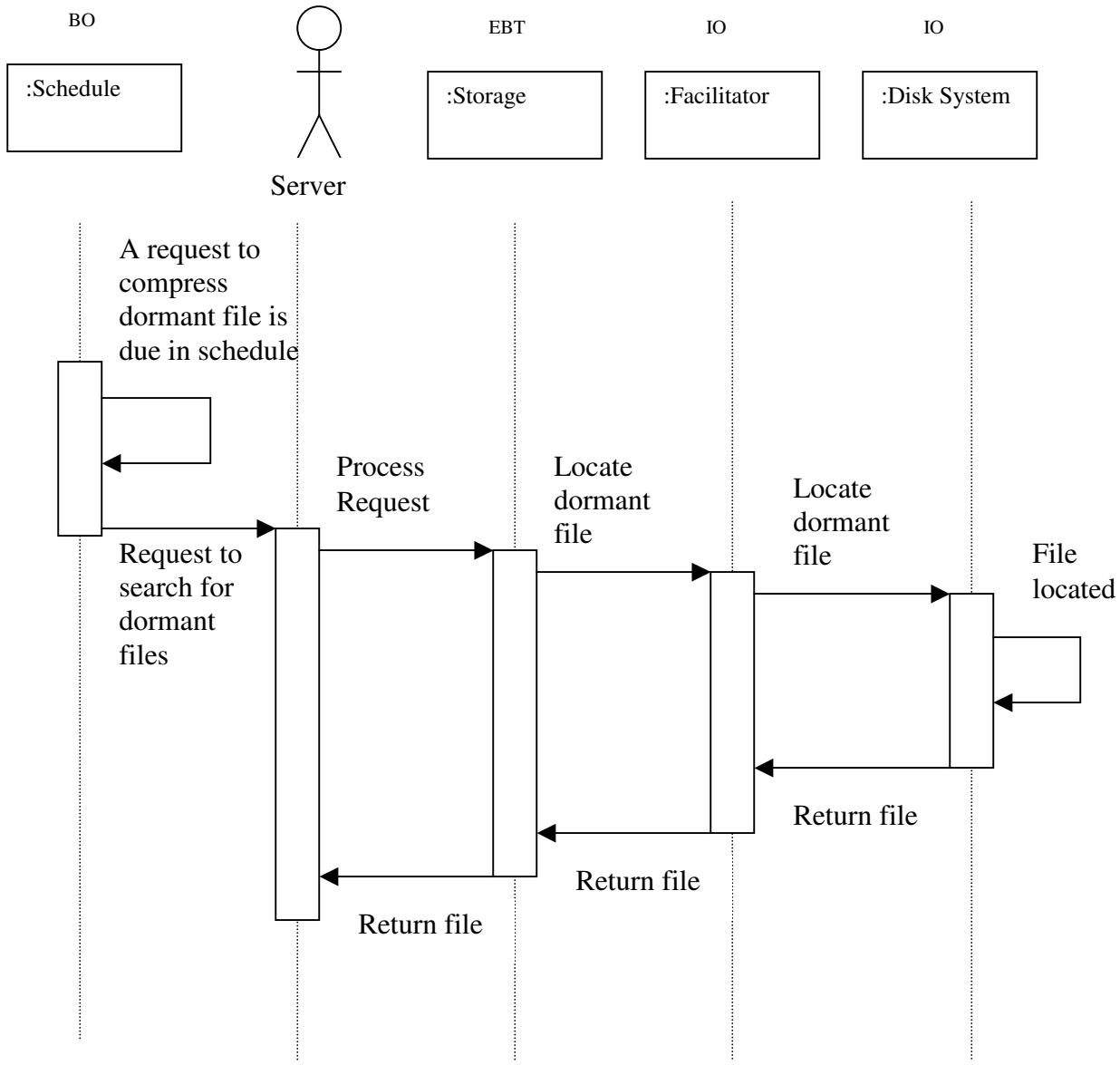
Use Case 5: Schedule Tasks

Description: To create a schedule, operator simply needs to choose what file want to be scheduled. The Efficiency will bring up the schedule, if exist, or create a schedule, if it did not exist. The Operator will then assign a time and interval of the file to be backup. The schedule will then be updated and stored.



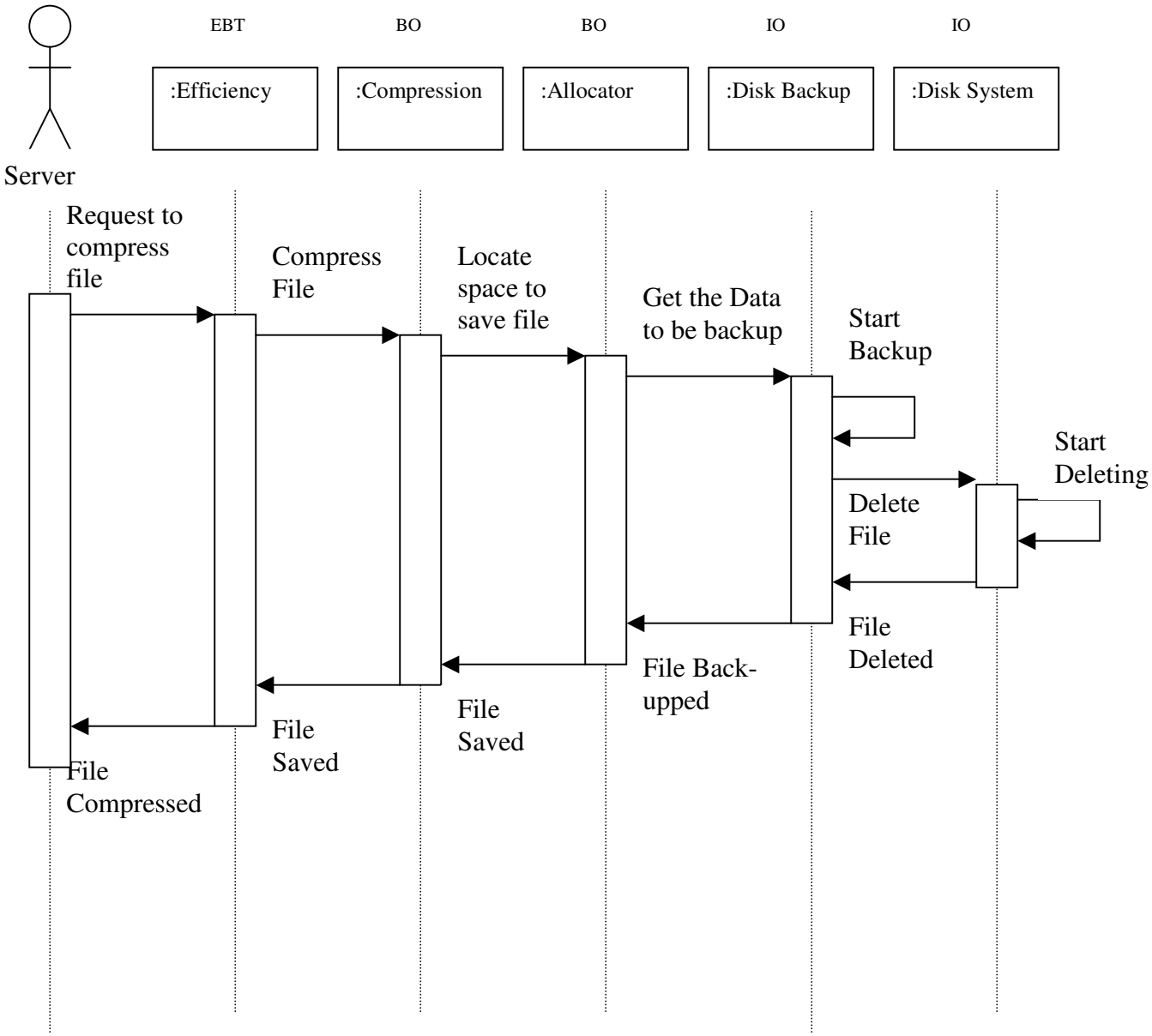
Use Case 6: Archive Data

Description: After a certain time, there is a need to archive the dormant file. This period of time is stored in the schedule. When the time comes, schedule will send a request to the server. Server will then request to look for all the dormant files in the Disk system. When found, the information of the file will be return to the server for later to be compressed.



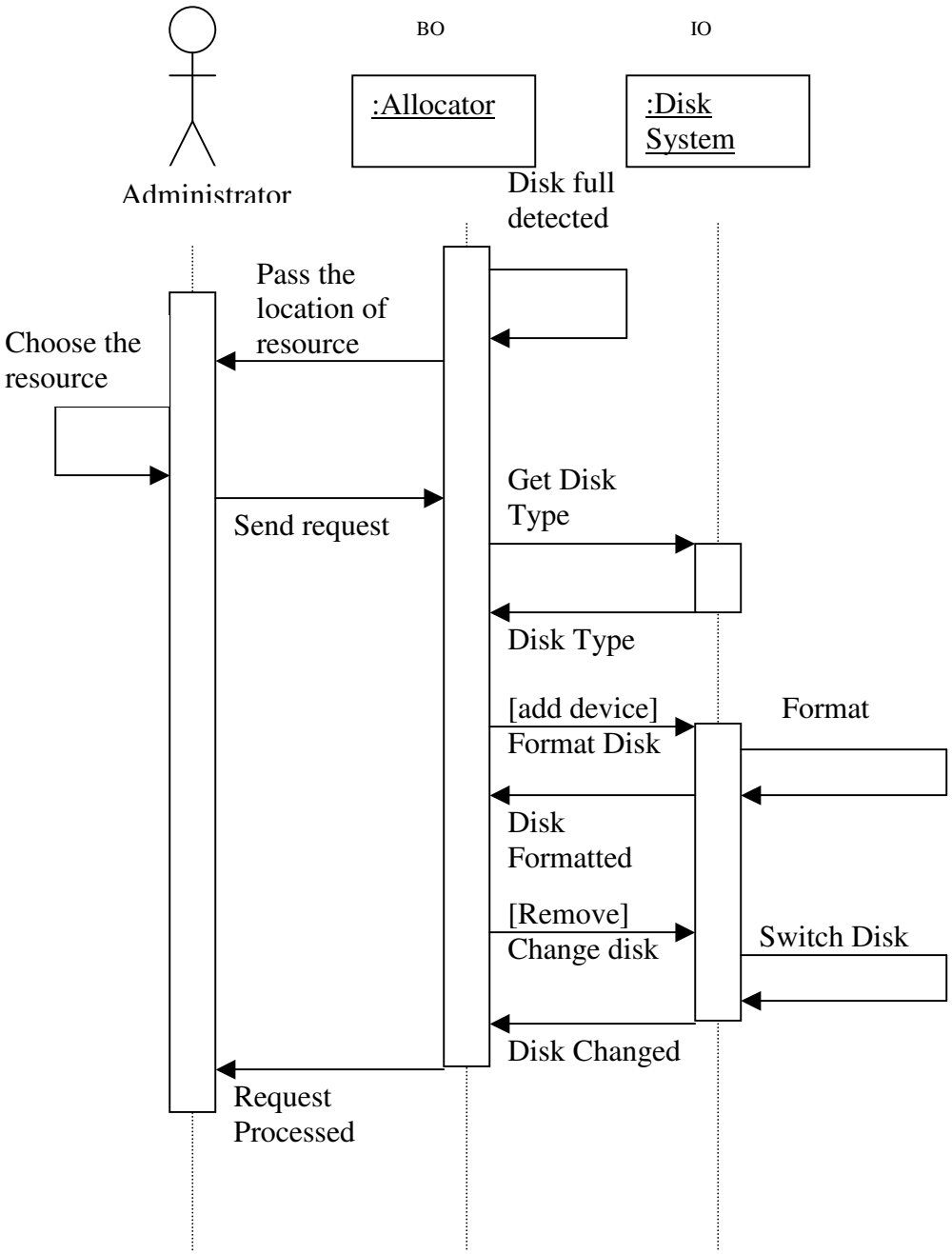
Use Case 7: Perform Compression

Description: When a dormant file found, the server will request that the file should be compressed and then stored. The compression will then compress the file according to the compression method specified. It will then ask the Allocator to allocate a space in disk backup. The Disk backup will backup the file and pass the location of the original file to the disk system. The disk system will delete the file and let the Disk backup know when it has finished. The server will be notified that the files have been successfully compressed.



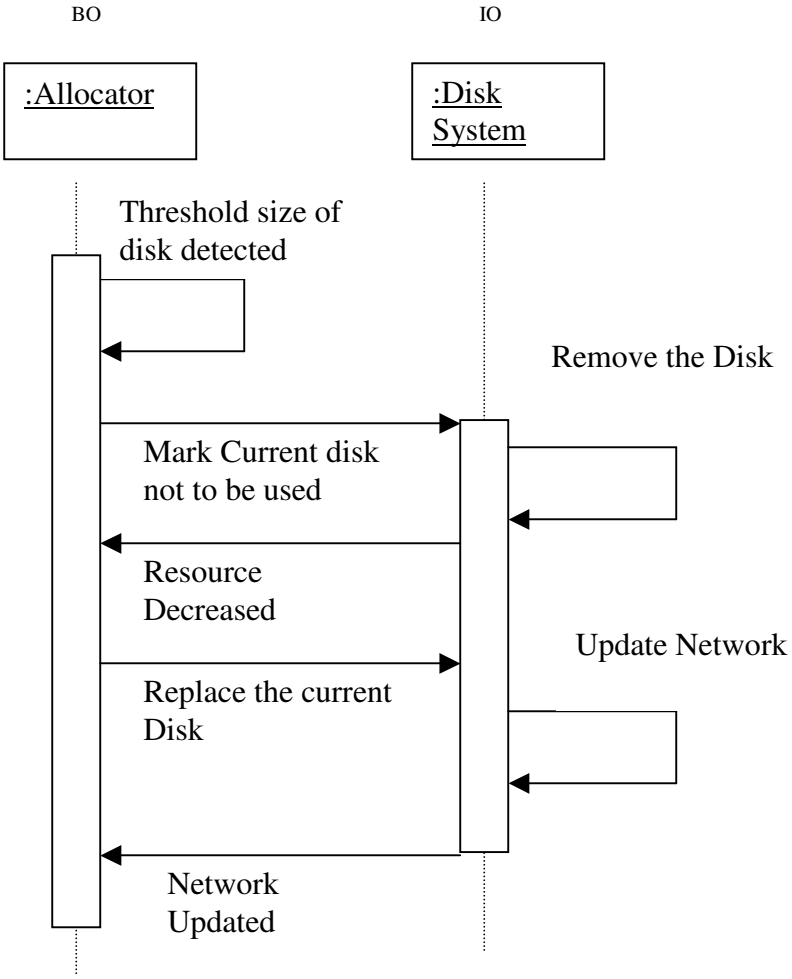
Use Case 8: Identify Resources

Description: When allocator detected that the disk is almost full, it will warn the administrator. The administrator then choose the device to be tend and send request to allocator. If the request is to add device, then a device is to be added (Use Case 11). If the request is to remove device, then a device is to be removed (Use Case 10). The allocator then let the administrator know that the request has been processed.



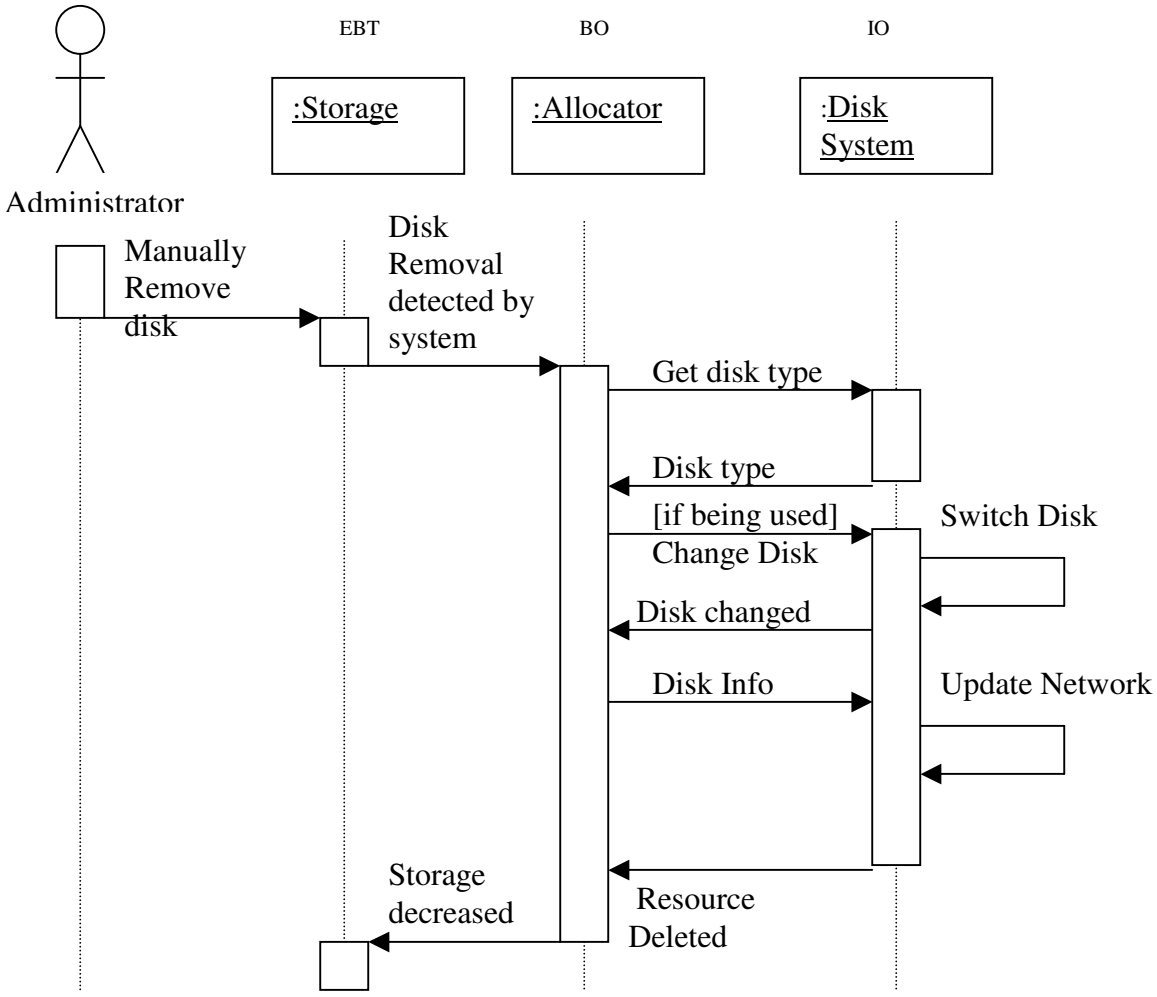
Use Case 9: Switch Backup Disk

Description: When allocator detected that the current disk that is being used is almost full, it will tell the disk system to not using the current disk. The disk system will remove the disk from the system (not physically). When allocator detected that the resource has decreased, it will tell the Disk system to replace the current disk. The disk system will then update the network and pass the information to allocator.



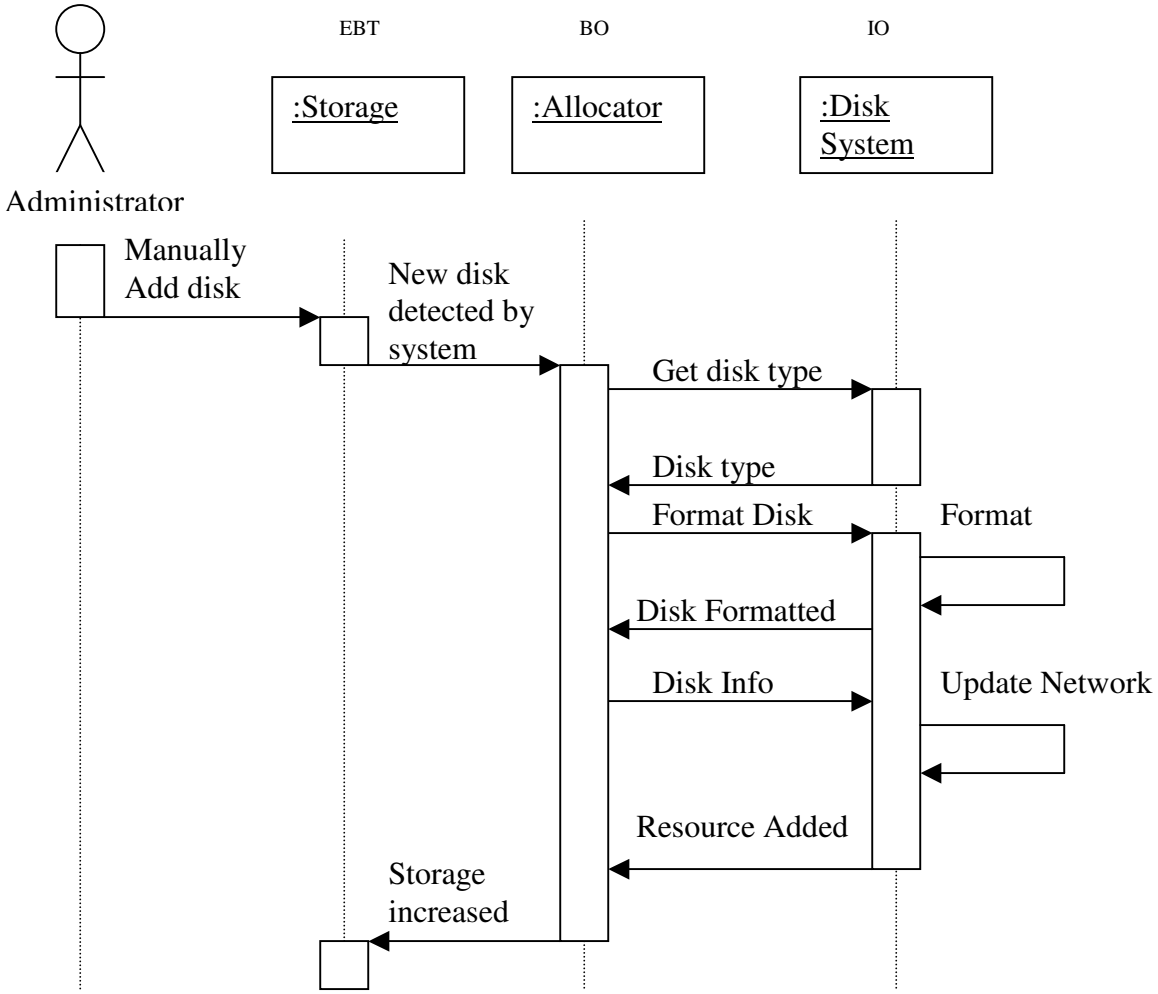
Use Case 10: Delete Resource

Description: Upon the manual removal of resources to the network, the disk must be configured and the network updated. The allocator class gets the disk type (backup or regular disk) from the disk system information. Then, if it is necessary, the switch disk will be initiated. Then, Allocator copies the Disk ID information and passes it to the disk system. Finally, the network is configured and the system recognizes the added resource.

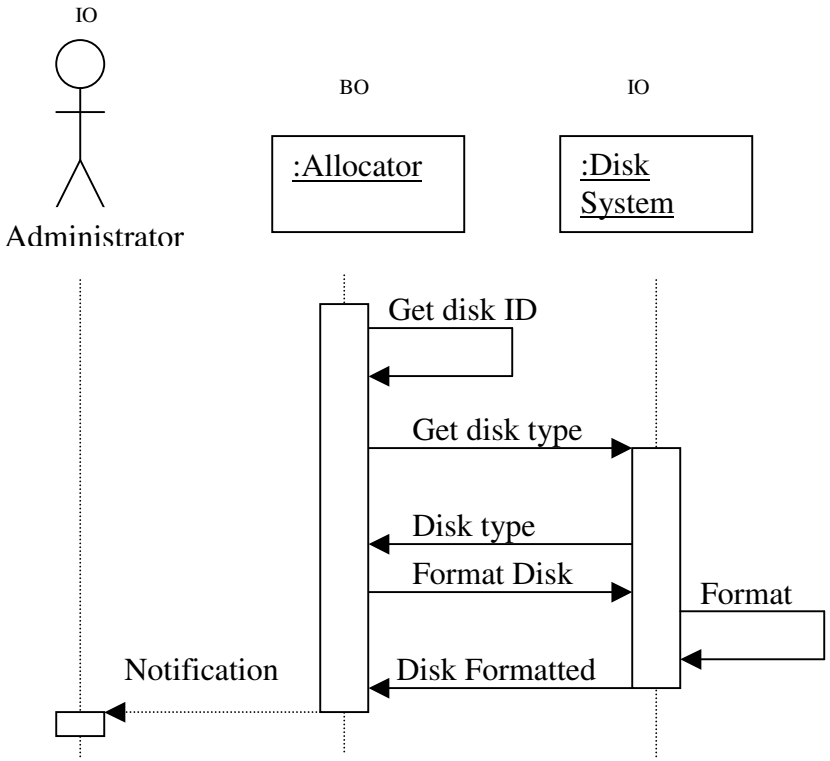


Use Case 11: Add Resource

Description: Upon the manual addition of resources to the network, the disk must be configured and the network updated. The allocator class gets the disk type (backup or regular disk) from the disk system information. Then, if it is necessary, the new disk is formatted. Then, Allocator copies the Disk ID information and passes it to the disk system. Finally, the network is configured and the system recognizes the added resource.

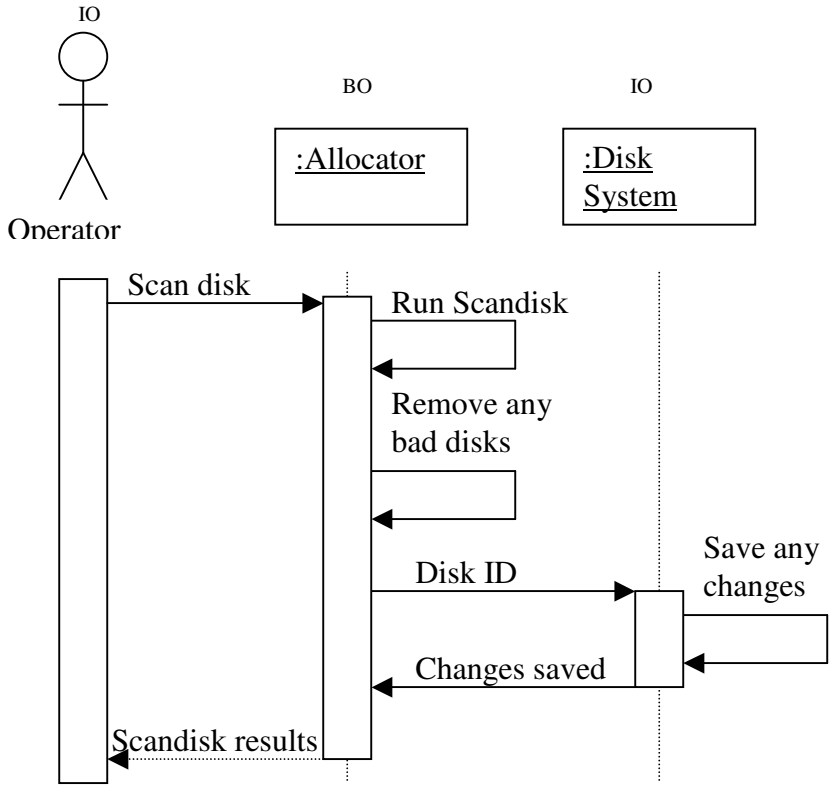


Use Case 12: Format Disk:
Description: First the Allocoter gets the disk ID. Based on this information, it asks for the disk type from the disk system. Then, the disk can be formatted. The format is done by the disk system and the allocator will in turn send a message to the administrator of the successful operation.



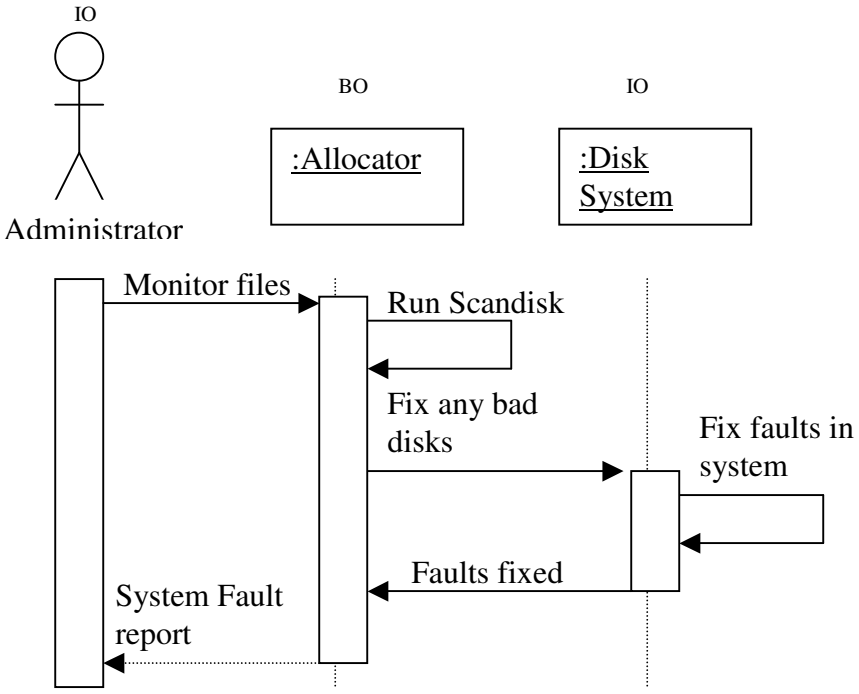
Use Case 13: Identify Bad Disk

Description: The operator will request the ScanDisk operation from the system. The Allocator will run the scandisk program in search of any bad disks. Any bad disks found will then be removed from the system. To do this, the allocator sends the disk ID to the disk system which will modify the system to show that the bad disk is no longer available. The results of the scandisk are then passed back to the Operator.



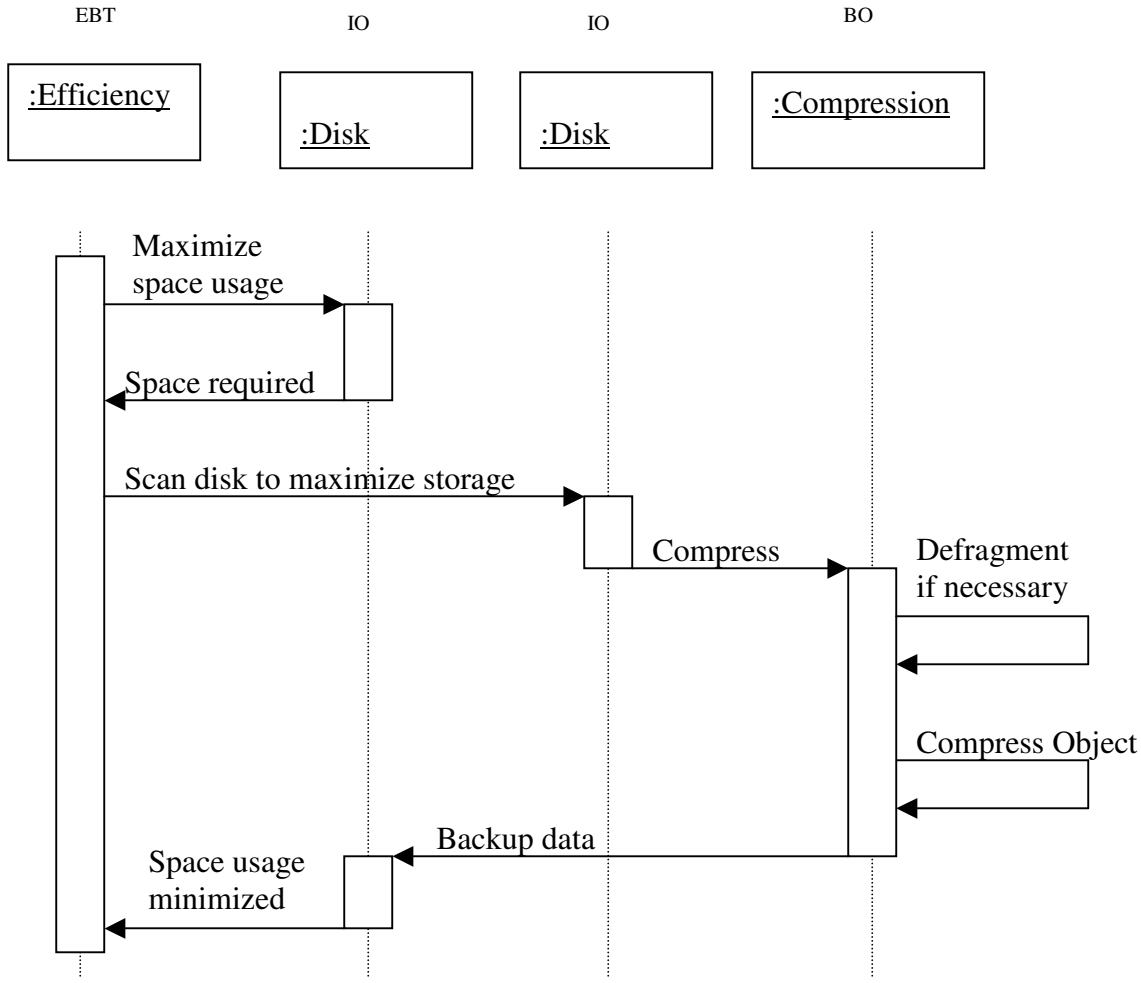
Use Case 14: Monitor File System

Description: The administrator is in charge of monitoring the file system. That person performs the monitor files function of the system to start the process. The allocator will first run a scandisk to find any bad disks. From there it will try fix the bad disks which will be carried out by the Disk System. Once the faults are fixed, a full report of system faults is created for the administrator.



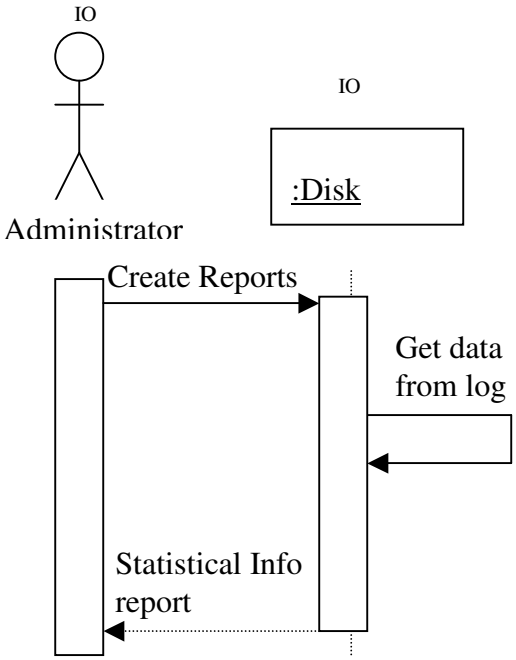
Use Case 15: Minimize Space Waste

Description: To maintain efficiency in the system, it is important to maximize the space usage when storing any files. Once the amount of space required is determined, it scans the disk system for a location that will maximize storage capacity. Compression and defragmentation are performed by "Compression" and the data can then be backed up. By maximizing disk organization and minimizing storage requirements efficiency is maintained in the system.



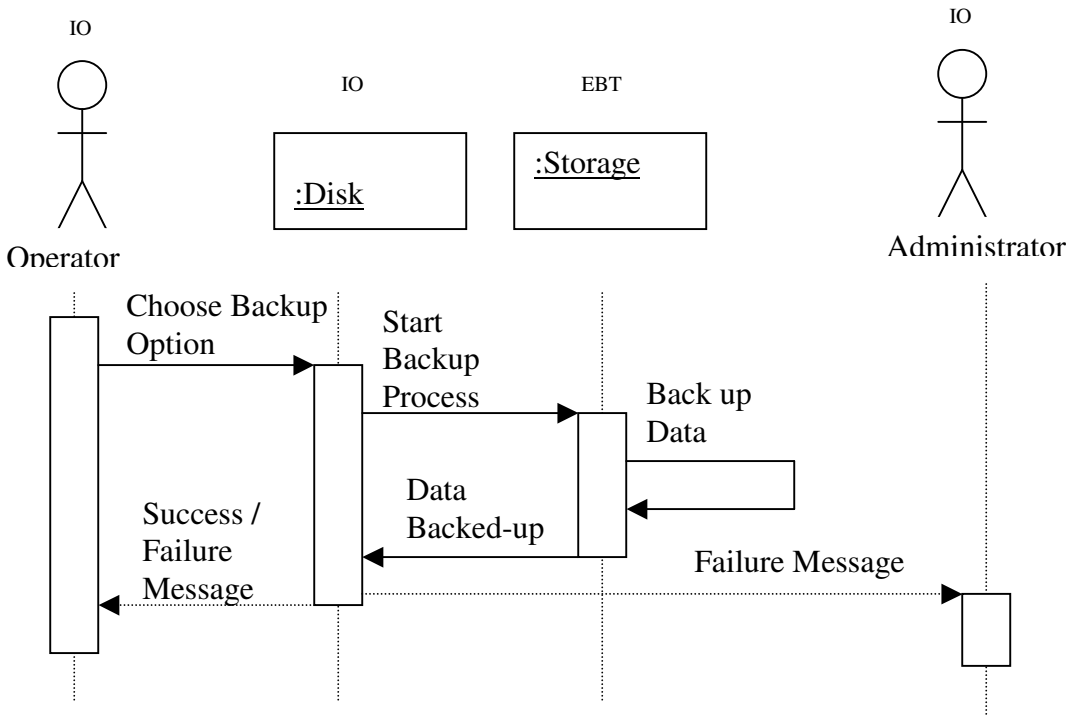
Use Case 16: Create Reports

Description: Create Reports is a simple operation performed by the administrator to keep up to date with the system's performance. The reports are gathered from the system's data log stored by the disk backup. The statistical information is then passed back to the administrator.



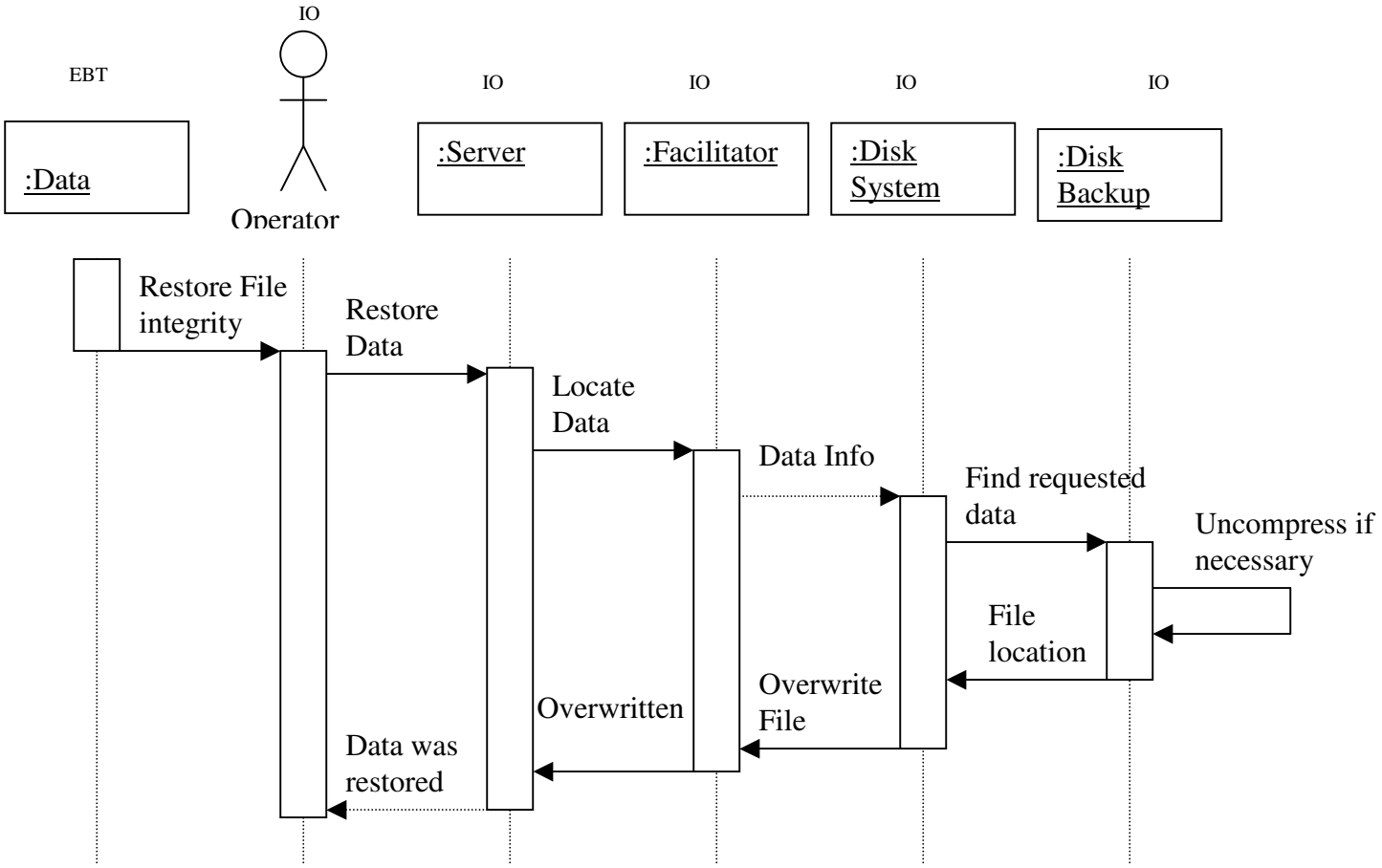
Use Case 17: Report Backup Failures

Description: Reporting backup failures is done when an operator attempts to backup data, but is not successful. The operator first attempts to backup some data. The system starts back up and storage stores the data (if all goes well). However, if the data could not be backed up for any reason, that message is sent out to both the administrator and the operator.



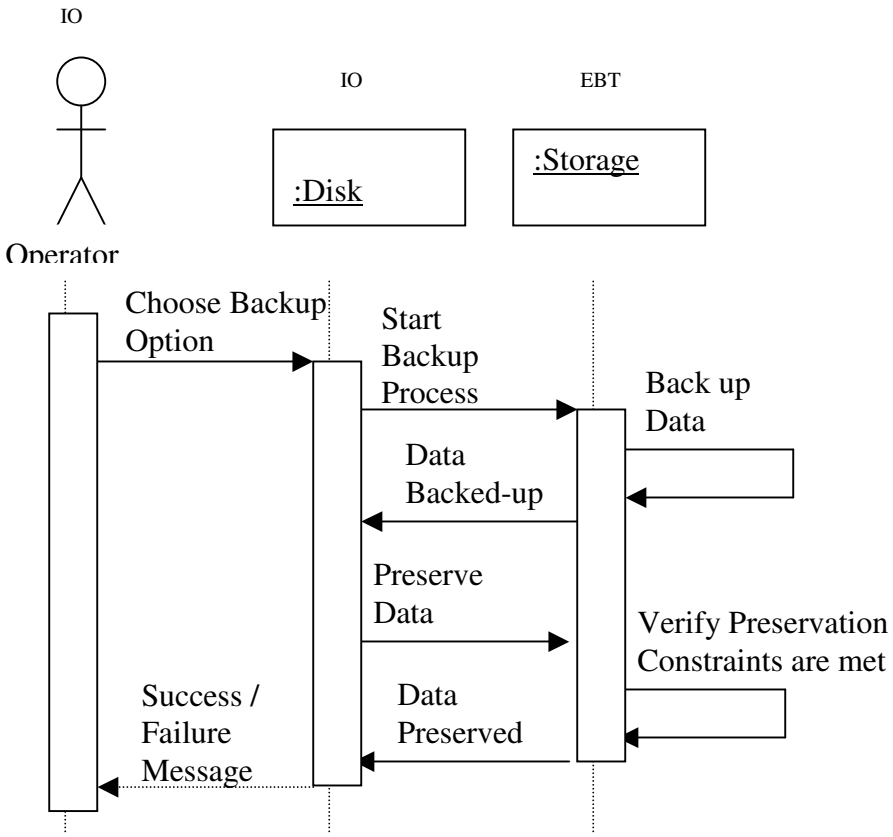
Use Case 18: Recover Data

Description: It is crucial to the system that the stored data can be protected, so when files are corrupted it needs to restore the file integrity. The operator receives the request and forwards it to the server, which then accesses the facilitator. The facilitator locates the data information and passes it to the disk system, which will locate the actual stored data. That data is then retrieved from the disk backup (uncompressed if necessary) and passes the location back to the disk system. Disk system overwrites the corrupted data and a message is then sent back to the operator that the data has been restored.



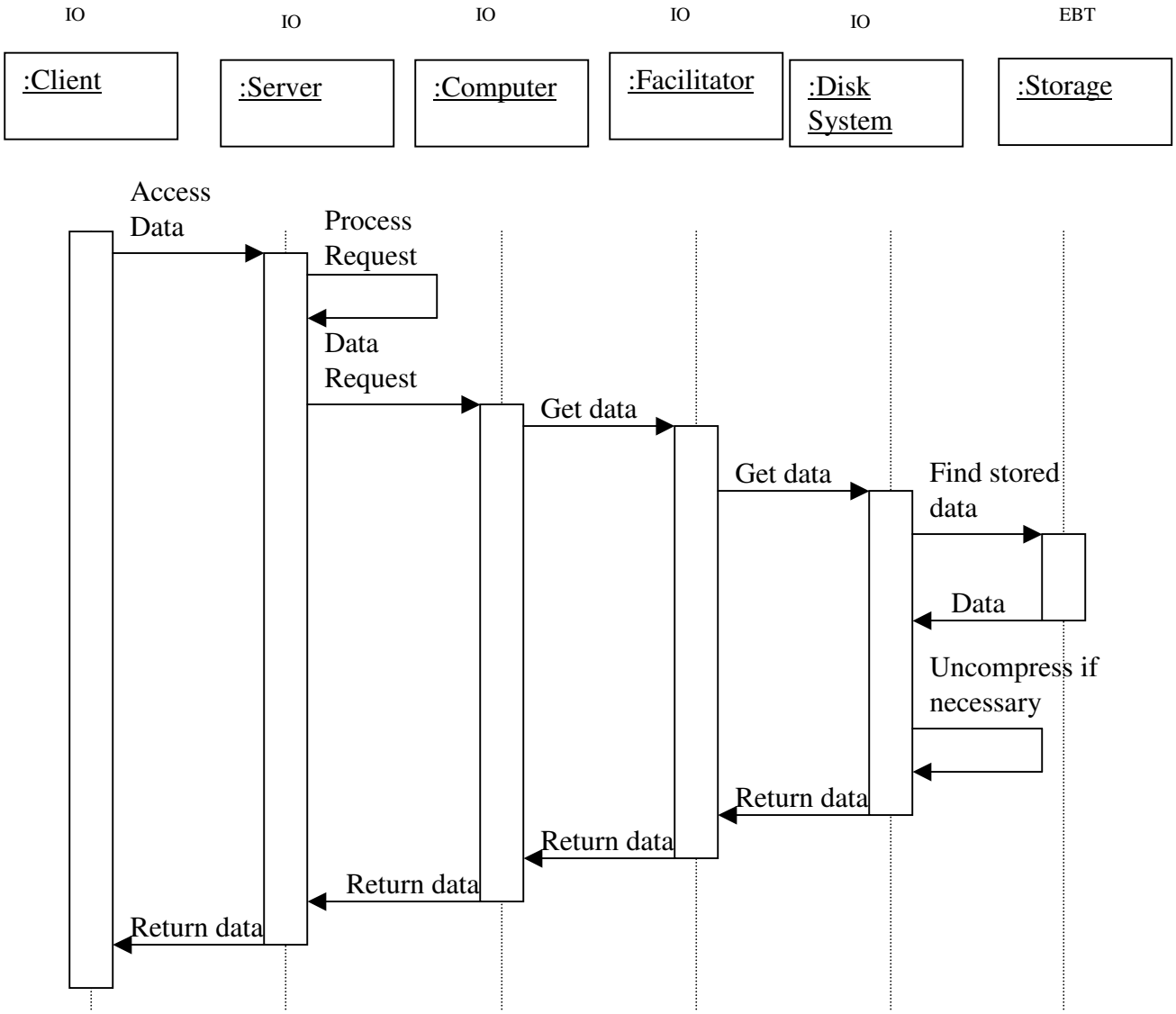
Use Case 19: Preserve Data:

Description: The Operator performs a routine backup and chooses which backup option to perform. Disk Backup starts the backup operation and the data is then stored. The data is then preserved verifying the constraints are met. The operator is also informed of the success of the backup.



Use Case 20: Retrieve Data

Description: First, a client makes a request for some data. The server handles the request and passes it to the facilitator through a computer. The facilitator then tries to retrieve the data from the disk system. The disk system looks in storage for the requested data. It is then uncompressed (if necessary) and sent back to the Facilitator. The data is passed back to the server through the computer and onto the client.



VIII. State Transition Diagrams

State transition diagrams do not need to be made for each class in the stability model. Some of the classes clearly are passive classes, laying idle until their services are requested. The behavior of other classes can be modeled by the state transition diagram of a related class. The following tables divide the classes appearing in the stability model accordingly.

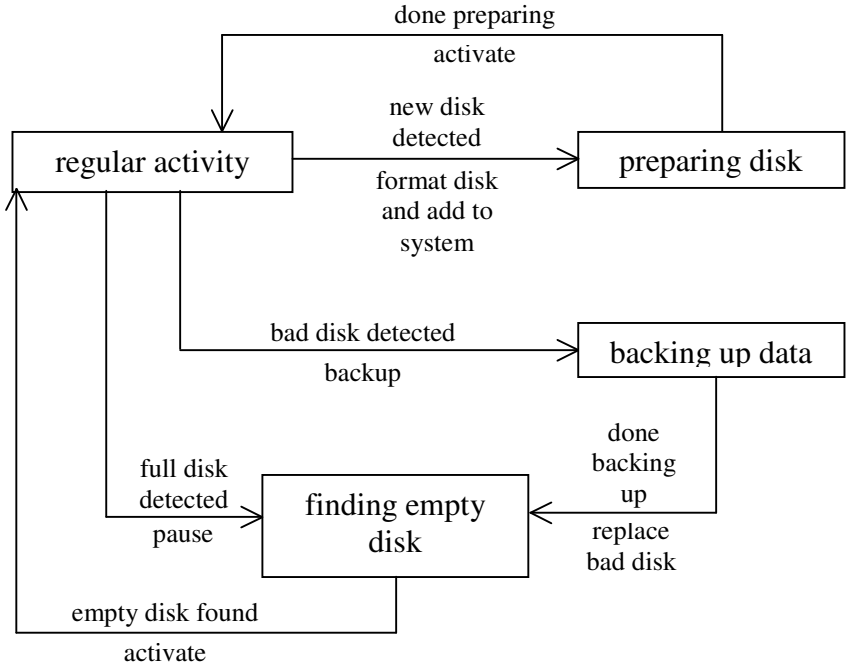
Active Classes	Passive Classes
Allocator Facilitator Efficiency	Disk File Compression Data Management Disk System Schedule Storage

General Class	Related Classes
Computer	Server Client User Administrator Operator
Data Protection	Data Backup Disk Backup
Storage	Primary Storage Secondary Storage Regular Backup

Using these tables, the behavior of each relevant aspect of the system can be modeled with state transition diagrams of the following classes:

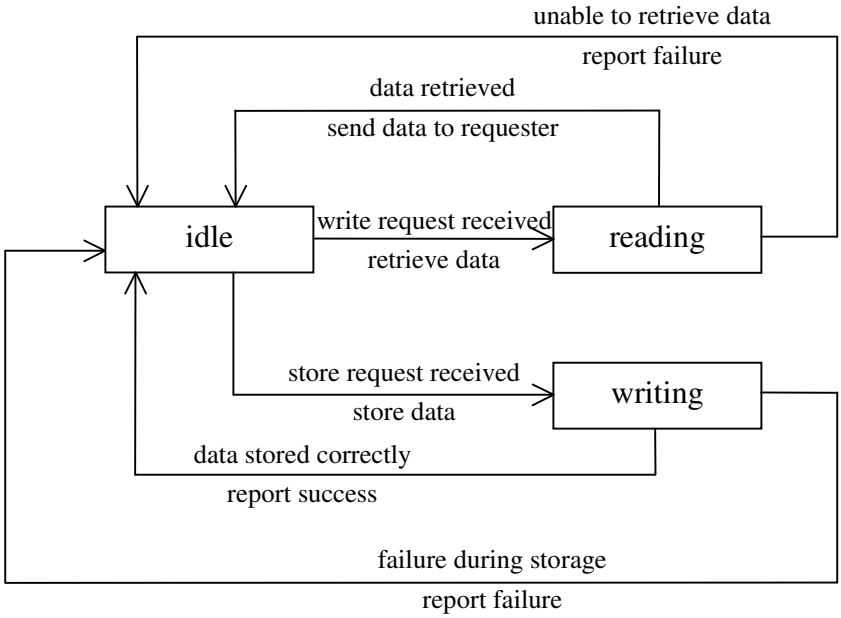
1. Allocator
2. Facilitator
3. Efficiency
4. Computer
5. Data Protection

Allocator



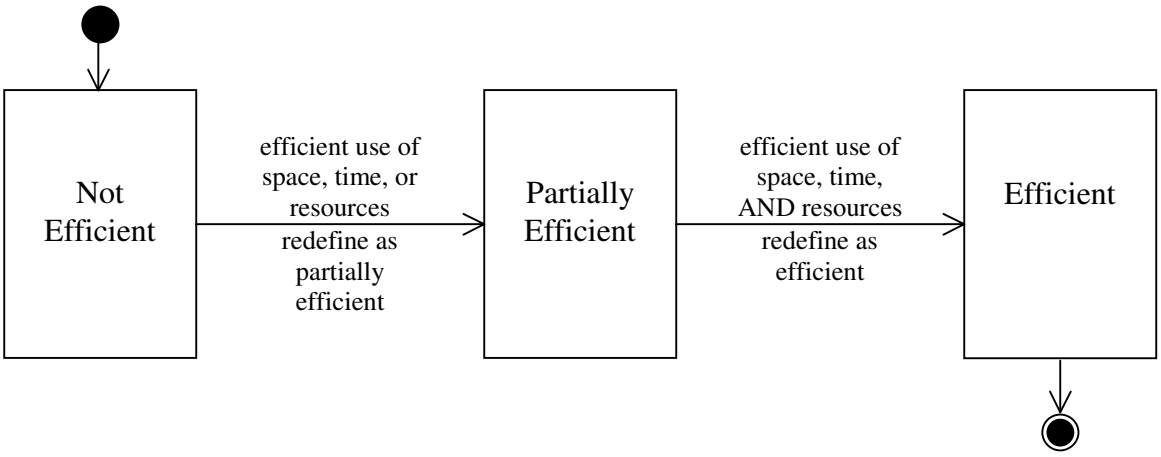
The Allocator class is in charge of determining where in the disk system new data is to be stored. Aside from this, it performs three specialized tasks for the system. When a new disk is physically added to the system, it will automatically detect the new disk, and initiate all of the tasks needed to fully integrate the new disk into the system. When a full disk is detected, it interrupts data being written to that disk and finds an empty disk to activate and write to. When a defective disk is detected, it backs up the data on that disk and replaces that disk with an empty disk.

Facilitator



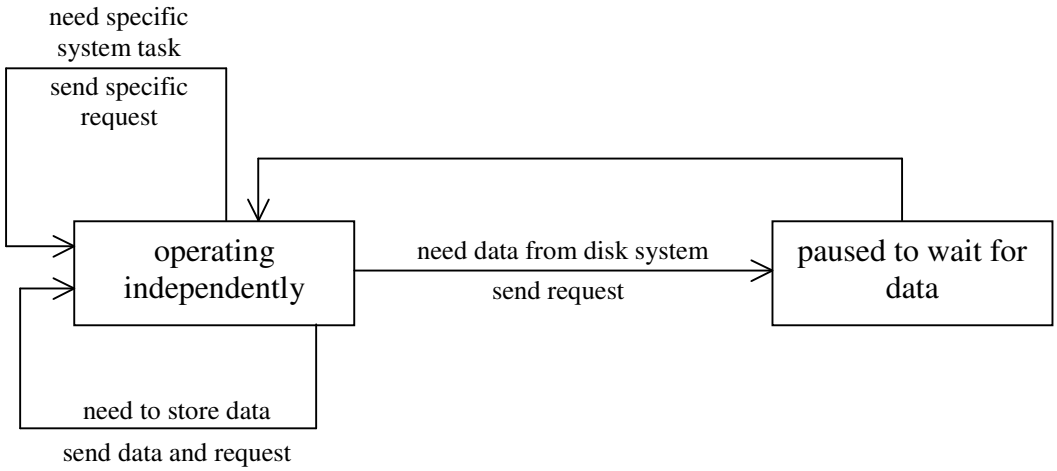
It may appear that the Facilitator class is not an active class, and indeed a good argument could be made that statement is true. However, the behavior of the class when errors are encountered needs to be documented, so the STD for it is included. The Facilitator class operates as an interface to the disk system. When data is requested from the disk system, the Facilitator class searches for the data and retrieves it if possible. When data needs to be stored in the disk system, it writes the data to the appropriate location, as determined by Allocator class. If an error occurs in either case, the error is reported to the entity that requested the retrieval or storage.

Efficiency



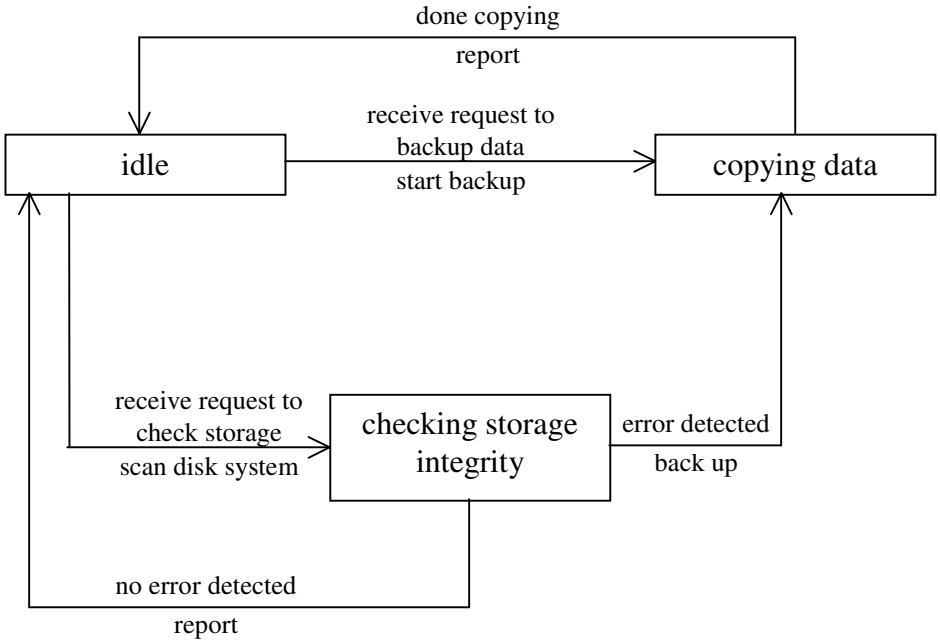
Efficiency is measured in three different areas, use of space, use of time, and use of resources. There are different definitions of what "efficient" use of these three things actually mean, but for something to be truly, and completely efficient, it must use space, time and resources in a efficient manner. In the domain of this particular system, efficiency is most likely to be measured in the use of disk space, network traffic, and the time needed to access the disk system.

Computer



The Computer class is used to model the behavior of all users of the system, including the classes of Server, Client, User, Administrator, and Operator. Most of the time, the computer is operating independently of the disk system, because all the necessary data is stored in more local memory. When data from outside local memory is needed, the computer must send a request to the system and wait for the results. When the computer needs to store data, it can send the data and the request to the system and return to its normal operation. There are also times when the computer (or user) will need the system to perform a specific task, such as format or backup a disk. Once again, when the request is sent to the system, the computer can return to normal operation.

Data Protection



The Data Protection class performs two major tasks to insure the data is being stored securely. The first task is to copy data to reduce the chance of losing data. The second is performing inspection of the storage objects to make sure there is not a error or inconsistency that potentially may cause lost of data. These tasks can be initiated internally, based on a set schedule, or externally by direct requests from users or other classes, such as an independent Schedule class.

IX. Analysis and Design Patterns

A. Analysis Patterns

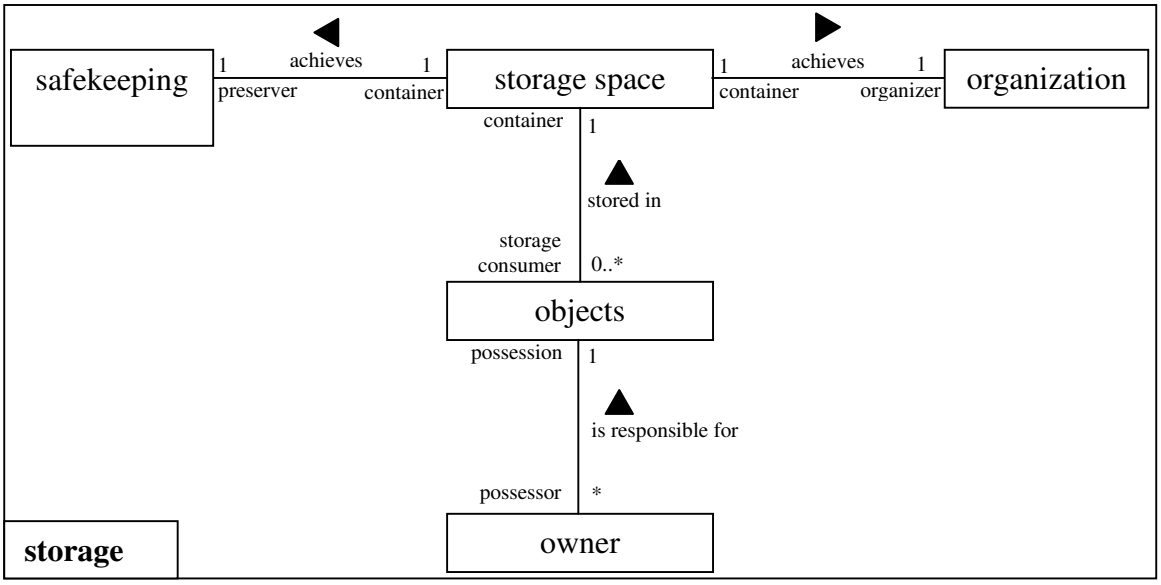
Storage

Description:

The Storage EBT is a very simple concept of human nature. Storage is a direct consequence of having possessions, which is a core concept to human beings. In order to store an object, there must be an available storage location the meets certain storage requirements of the object. The most obvious requirement is size. The storage location must be big enough to contain the object. In addition to this, some objects may require different storage conditions than other objects. For example, milk should be kept in a cool place and a plant should be stored in a sunny place. Some objects only require storage for a finite period of time. Most bank statements should only be kept for one year.

A storage location has important attributes. Again, the most obvious is the size of storage space. The environmental conditions of the location are also important because they determine whether an object can be stored at that location safely. Another attribute that is important is the schedule of storage. Some objects may request space at a storage location at specific times. Before agreeing to store an object, the possibility that the same space was not already promised to another object must be ruled out, hence the need for a schedule.

Meta Model:



CRC cards:

Owner(possessor)

Responsibility	Collaboration	
Care for possessions by storing them properly.	Client	Service
	objects	put in storage(Object) take out of storage(Object)

Object(consumer)		
Responsibility	Collaboration	
Consume storage space in an efficient manner	Client	Service
	storage space owner	store() remove()

Safekeeping(preserver)		
Responsibility	Collaboration	
Provide a location that preserves the condition of a stored object	Client	Service
	storage space	preserve()

Storage Space(container)		
Responsibility	Collaboration	
Provide an acceptable location where objects can be stored.	Client	Service
	Objects safekeeping organizer	get available size () is full ()

Organization(organizer)		
Responsibility	Collaboration	
Provide an easy and efficient method for locating stored objects	Client	Service
	storage space	organize()

Efficiency

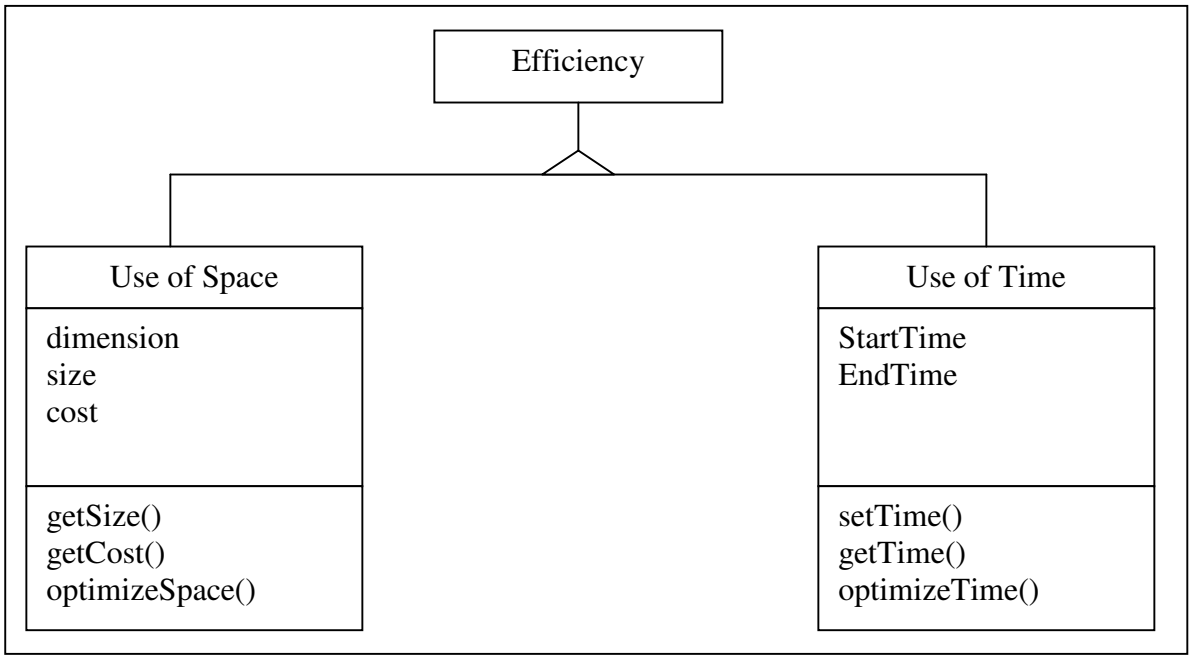
Description:

Efficiency is achieved by the efficient use of resources. Resources include time as well as spatial resources. Therefore, Efficiency can be split into two separate subclasses, Efficient Use of Space and Efficient User of Time.

Space has several attributes relevant to efficiency, namely dimension, size, and cost. The use of this space can be optimized, or made efficient, by an optimization routine based on the relative importance of these attributes. If cost is most important, then a cheap space would be more efficient than a large one.

In the context of efficiency, time is constant. In order to make the use of time efficient, the time between the start time and the end time should be minimized.

Meta Model:



CRC cards:

Note: This is such an abstract concept that anyone (or anything) could be a client of the Efficiency class.

Use of Space (optimizer of space)		
Responsibility	Collaboration	
Provide an efficient use of space.	Client	Service
	any	getSize() getCost() optimizeSpace()

Use of Time (optimizer of time)		
Responsibility	Collaboration	
Provide for an efficient use of time.	Client	Service
	any	setTime() getTime() optimizeTime()

B. Design Patterns

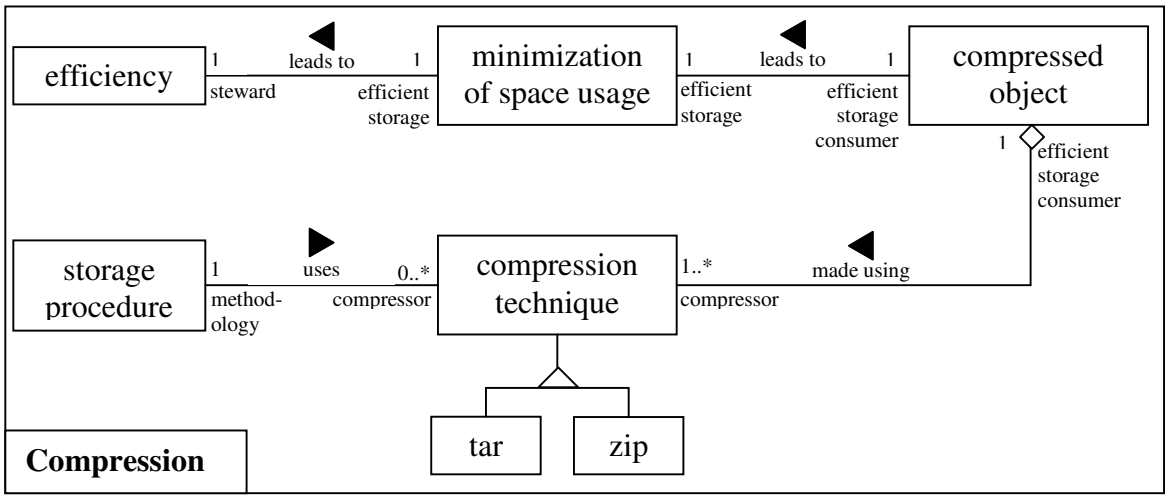
Compression

Description:

In order to save space when storing an object, the object is often compressed before it is stored. This is used extensively in the storage of electronic data, but also in other applications where space is a premium, such as designing camping gear. Compression is al regularly used to transport large amount of industrial gasses by cooling them until they liquefy and take up a much smaller volume.

The whole point of storing objects in a compressed form is to minimize the use of storage space. Saving storage space actually achieves efficient use of the storage space, which is a important concept in all types of storage. Compressed objects are created by compressing a regular object using a compression technique. There are countless techniques, many of the very specific to the type of object being compressed. Some compression techniques can be applied to a compressed object, in which case the object would be compressed by more than one compression technique. The compression techniques are used according to the storage procedure of the storage location.

Meta Model:



CRC cards:

Storage Procedure (methodology)		
Responsibility	Collaboration	
Define methods of compression in order to store objects in an efficient manner	Client	Service
	compression technique	add_compress_technique()

Compression Technique (compressor)		
Responsibility	Collaboration	
When given an object, create a copy of that object that will take up less storage space	Client	Service
	compressed object	compress()

Compressed Object (efficient storage consumer)		
Responsibility	Collaboration	
When stored, to take up less space than the original object would have	Client	Service
	decompress	store() remore()

Minimization of space usage (efficient storer)		
Responsibility	Collaboration	
Manage the use of space in such a way that the overall use is minimized	Client	Service
	Efficiency	minimize space()

Efficiency (steward)		
Responsibility	Collaboration	
Be a good steward by using the resources in an efficient manner.	Client	Service
	minimization of space usage	make efficient()

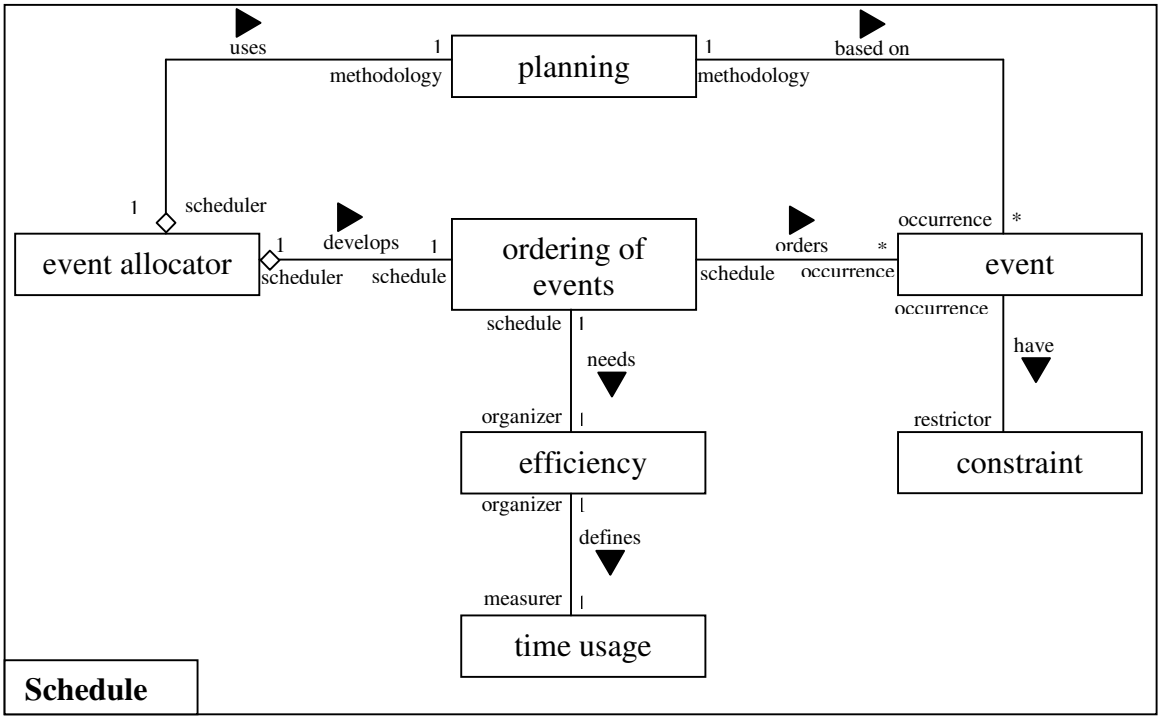
Schedule

Description:

A schedule is used to create an ordering of event that is usually disjoint, that is no two events overlap in time. A good schedule orders all events in such a way that they are all given adequate time to complete and there is a maximal amount of free, or idle, time in the schedule.

A schedule is created by first ordering a group of events by allocating a time slot to each event, according to the methods defined by the planning that is based on the events themselves as well as their constraints. This ordering must be efficient. The efficiency of the ordering is measured by the time usage of the ordering, and in particular, the amount of free time in the ordering.

Meta Model:



CRC Cards:

time usage (measurer)		
Responsibility	Collaboration	
Allow for the measurement of the efficiency of a ordering of events	Client	Service
	efficiency	show free time()

constraint (restrictor)		
Responsibility	Collaboration	
Define the restrictions on the possible time slots during which an event can take place	Client	Service
	event	show_constraint()

Event (scheduler)		
Responsibility	Collaboration	
Perform a task in a finite amount of time	Client	Service
	ordering of events planning constraints	initiate() show constraints()

planning (methodology)		
Responsibility	Collaboration	
Define a methodology for allocating time slots to events which is based on the characteristics of the events.	Client	Service
	event allocator events	set plan()

Event Allocator (scheduler)		
Responsibility	Collaboration	
Give each event a specific time slot during which it can take place.	Client	Service
	Ordering of events	allocate()

Efficiency (organizer)		
Responsibility	Collaboration	
Organize the ordering of events in such a way as to use time efficiently.	Client	Service
	maximize free time	make efficient()

X. Object Oriented Heuristics

Heuristic No. 2.4: Implement a minimal public interface that all classes understand.
The interfaces that are provided by the classes perform a unique operation that is used by the other classes that collaborate with this class. The interface is also designed to make sure that each interface represents a functional objective. In our system we have Disk System class of which the class Facilitator and Allocator make use of.

Heuristic No. 2.8: A class should capture one and only one key abstraction.
Each of the classes in the system, EBTs, BOs, and IOs, has exactly only one key abstraction. This is made sure with the use of CRC card in which each class can have only one role which essentially captures a key abstraction. We use this heuristic when we are trying to come out with a class that to allocate and retrieve data and we realized that we had more than one abstraction for the class when we are trying to build the CRC card for that class.

Heuristic No. 3.2: Do not create god classes/objects in your system. Be very suspicious of a class whose name contains Driver, Manager, System, and Subsystem.
When trying to come out with classes to be modeled for our system, especially for the traditional and the IOs of Stability model, we were considering a class called Data Manager. We realized that the class that we were considering will act as a god class that do most of the work. Then we use Heuristic 2.8 and came out with 2 different classes instead, such as Facilitator and Allocator.

Heuristic No. 5.10: If two or more classes have common data and behavior (i.e. methods), then those classes should each inherit from a common base class which captures those data and methods.
We have many classes, such as Administrator and Operator, that have common data. Because of this, we created a class called Users as a base class and has the common data as its attributes. Administrator and Operator class then inherit their attributes from the Users class.

XI. Lesson Learned

The most difficult part of this project, in our opinion, is to come out with class of which the problem statement can be modeled accurately, especially using the traditional model. We spent a couple of hours just to 'extract' classes from the problem statement only to found that they did not adequately represents the system that we are trying to build. We ended up adding and deleting class to adjust to the system that we want. Another problem with classes are in naming them. We had some difficulty in coming out name without contradicting with some of the heuristic (avoid having manager, director as the name of file). Some of the class' names were confusing and after consulting with Dr. Fayad, we were able to come out with better names.

Another difficult part of the project is in coming out with a good use case description. We tried to make our use case description is detailed and accurate as possible because from our previous experience, a good description is beneficial in aiding construction sequential diagram. The process of creating sequential diagram is much simpler because we just basically convert each step in the use case description to the arrow coming from and to the appropriate object or actors. Creating a use case diagram helped us in understanding what the system was suppose to do and therefore helpful when creating the class diagram, especially the stability model.

We also spent a lot of hours in the design pattern and analysis pattern. We think that this part is quite difficult because we did not have prior experience (in the assignments) to do this. But after we regained further understanding, we learn that it is a lot easier to think about this as modeling the concept in stability model.

As far as stability and traditional model involved, we did not find a lot of problem. We have to admit that last assignment (assignment 3) had resulted us in thinking about the system as stability model instead of traditional model. Therefore, when we tried to model the system in term of stability model, it was more an automatic process. The three questions were very helpful in giving a head start and keeping the system in focus. On the other hand, we had more discussion when doing the traditional model.

XII. References

1. Tivoli® Storage Products by IBM. http://www.tivoli.com

2. Fayad, M., *Lectures and Notes*, Software Design Methodology Spring 2002, University of Nebraska - Lincoln

3. Wu, S., *Presentation*, Software Design Methodology Spring 2002, University of Nebraska - Lincoln

XIII. Appendix A: The Problem Statement

Problem Statement:
Enterprise Storage Management System

Prepared by:
Dan Glasser, Madeline Hardojo, Anand Sundaram, Nate Wells

Abstract:

Enterprise Storage Management System is an interactive and user-friendly program that will enable Lincoln Telephone Company to efficiently manage their storage system. With this system, the Lincoln Telephone Company will be able to optimally back-up and recover

Background:

In a world where information is the key to success, every company must take adequate measures to protect its data. In a typical corporate world, tons of information flows across the network that needs to be processed. These data that are being used for processing need to be stored in a safe and secure medium that is also easily manageable. With the advent of new storage technology and falling price of the storage, IT managers buy more storage devices for their rapidly increasing corporate data. This process in turn makes managing the growing storage very complex.

Lincoln Telephone Company wants to have a solution for the growth of storage in the company. The architecture of Lincoln Telephone Company is illustrated in Fig. 1. The company has a number of servers to accommodate its applications and data. When a client wants to pull out a certain data, the server will check whether the data wanted is inside the server or not. If not, it will locate the data from the various backup disks. They would like to have this process automated with a user-friendly system.

Description of Program that is Wanted:

The storage management system should address the following issues:

- An efficient mechanism to backup and recover the data – this may include a distributed backup or centralized backup
- Monitor the file system to automatically assign and extend additional available resources
- Replace the backup with an available resource if the backup reaches a threshold value.
- Mechanism to format the available disk with the required File System in a multi platform network
- Add/Delete resources to/from the storage network in a multi platform scenario
- Report backup failures due to network failures. Reschedule the incomplete backup tasks as soon as the network is up.

- Identify bad disks in the network and remove them. Backup/recover the disk before doing so.
- Automatically identify the resources that are added or deleted to the network.
- Create an Analysis report of the backups that will help users schedule the backups efficiently.
- Identifying dormant files and automatically compress them to be archived in a backup disk

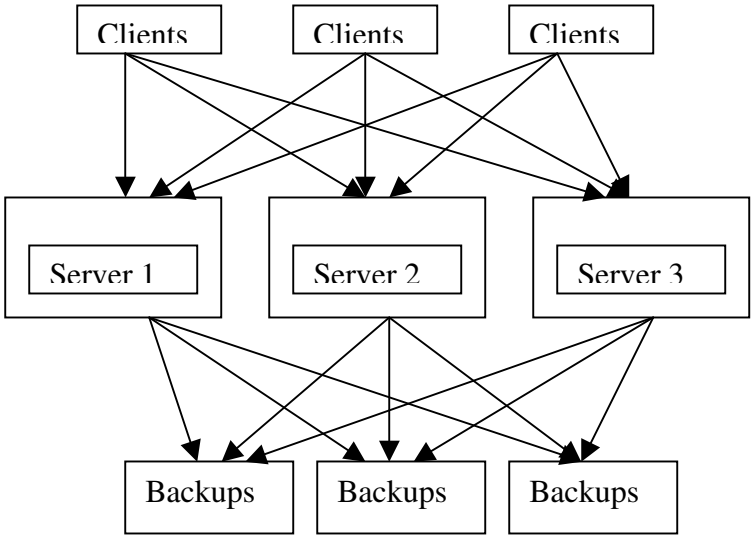


Fig. 1 Architecture

Detailed Requirements:

Lincoln Telephone Company, for protection against destruction, either accidentally or intentionally, feels the vital need to back up their valuable data of customer and billing information on a daily basis sometimes even hourly basis. Lincoln Telephone Company's backup servers are kept in buildings, which are geographically distributed. The Enterprise Storage Management System should be able to manually and automatically backup the data and be able to recover the data if needed. An operator of the system should be able to setup how often the data needed to be backup. Also, the system should be able to recognize if the data is corrupted and should be able to pull out the latest backup and recover the loss of data.

As we can see in Fig. 1, the servers of Lincoln Telephone Company are accommodating applications and data. The storage of data in the servers can be full. Therefore, the Storage Management System should be able to detect if the data storage in the server is nearly full. If such condition is detected, the system should automatically allocate a place in the backup disk for the data and store it. Then there would be free storage space in the server for the application or data.

Lincoln Telephone Company has purchased many disks for backup when the price for storage dropped. All the backup disks, both non-empty and empty, are

connected through a network. The company would like the system to be able to notice if the backup disk currently used to store the data from the server is almost full and to automatically redirect the data to the empty backup disk.

When accessing a disk for backup, the user should be able to specify what file format to store the data in. If the user does not care what format the data is stored in, the Store Management System should choose the optimal and convenient format to store the data. In all cases, the system should be able to accommodate a wide range of storage formats. Additionally, the system should be able to provide data in a given format to a user in a format that they request.

In a multi platform network, the requirements for the storage media are very high. The system should be able to easily detect any new resources that have been added to the network. The system should also be able to remove any resource from the network that is no longer needed or needs to be relocated. This will help the storage management system to manage the growing requirements of the storage disks by allocating the free available disk for backup purposes. Essentially, whenever a backup disk is full, a new free available storage disk is identified for replacement. This is done by checking whether a free disk is available in the network by Handshake mechanism. Once a free disk is found, it can be formatted with the appropriate file system and be marked as a backup disk in use.

In a network where all the servers, storage resources and other systems are connected, there might be failures due to network problems. In order to address this issue, the storage management system should be able to detect the storage related problems and fix it automatically. If the backup fails because of network failures, the system should be able to detect it. The problem can be reported to the administrator. As soon as the network problem is resolved, the backup process that has failed should be automatically rescheduled.

The data that are stored in the disks (either backup or normal) should be protected. If there are problems like data corruption or hardware problem, the system should be able to resolve it. The system should run a tool that will perform data integrity check on each of the storage disks in the network on a regular basis. If the system finds data corruption, it should be fixed. If there is any hardware issue, the disk should be replaced with a new disk after restoring the data from the corrupt disk.

Over the network, to have an efficient backup mechanism, the system should be able to do the backup without adding much to the network traffic. In order to achieve this, the system should be able to schedule the backup processes when the network traffic is less. The traffic in the network system can be studied and a report can be produced that will help the administrator schedule the backup process efficiently. This process can also be automated.

The files that are back-upped by the Lincoln Telephone Company sometimes are only used once or twice and never being used again. This is an inefficient use of storage space. The Lincoln Telephone company would like a system that will be able to periodically check the whole backup disk and automatically compress these inactive files in order to save some storage space and create a log of the files that being compress for the Administration purposes.

Use Cases:

Case 1: Backing Up Vital Company Data Manually

The operator for the Enterprise Storage Management System has received a request from the Billing Department to do a backup of all the billing transactions for the current month for safety measure since the Department is going to implement a new program to manage billing transactions. The operator will then run the Enterprise Storage Management System and put on his/her operator id and from whom the request was received for security and future reference. He/she will then go to the backup interface and choose which directory or file to be back-upped. The system will then automatically allocate a space in the backup disk and name the directory/file appropriately so that it will reflect necessary information such as the date of backup and the original directory/file name. Aside from that, it will generate a "readme" file that will contain information of which operator did the backup and other related information.

Case 2: Backing Up Vital Company Data Automatically

Lincoln Telephone Company is a fast growing company. Therefore, it is necessary to backup their new customer information daily. Of course to do this manually will be such a tedious job, not to mention, it will require an operator to be responsible and to spend his/her valuable time every morning to do backup. An operator will then go to the backup interface of the system and specify which directory or file to be back up. Then, he/she will put the needed information to generate backup automatically. This information includes the time of day of which the backup should happen and how often this should be repeated. This information should be able to modified anytime. On the schedule given, the System will automatically backup the directory/file specified. In addition, it will generate a report log of the necessary information, such as which directory/file is back-upped, etc. In case of the server failure or any case of which the System fail to back up, the system will automatically e-mail the System Administrator Department to warn that automatic backup fail and therefore manual backup and other adjustment is needed.

Case 3: Recovering Data

The System Administrator received a call from Financial Department. They are reporting a crash in the system frantically and would like to have their data back since tomorrow is payday. The System Administrator later called the operator for the Enterprise Storage Management

Case 4: Distributed / Centralized Backup

Lincoln Telephone Company has a multi platform network in which the storage can be made in a centralized backup disk or distributed backup disks. The system has the

functionality that will help the Administrator decide to either distribute the data storage or store them in a single disk based on the requirements of the application. For a centralized backup option, the administrator has to specify the available storage disk. If the disk becomes full, the storage management system detects the free available disk in the network that replaces the disk that is full. In a distributed storage option, the administrator gets to choose the disks from a set of available disks. The administrator can also automate the process by letting the storage system do the data storage in a distributed environment.

Case 5: Monitoring the File System/Server Storage

With a high storage requirement for Lincoln Telephone Company, it is critical for the company to monitor the growth of the data. The rate of the space utilization by the applications determines the need for additional storage disks. A smooth transition must be made when a disk in production becomes full. The company can do this by defining a threshold value. If the utilization of disk space crosses this value, the storage management system will alert the administrator accordingly. The system will also identify the free available storage disk for extending the storage. Once identified, the disk is formatted with the required file system and is marked as live. Now, the new disk runs in parallel with the disk that has just become full.

Case 6: Switching Backup Devices

While moving data to a backup device, the device starts to become full. The Enterprise Storage Management System will discontinue using that device and start using a different device that has space available. The first device will be marked in such a way that it is no longer available to any additional processes as a place to write data. Additionally, the transition to a new backup storage device will be performed in a way that both devices are still accessible by any process that requests any of the data.

Case 7: Formatting the Disk

Since the company data is highly critical, it brings forth the need for data backup. Essentially, whenever a disk in use becomes full (crosses a threshold value set by the administrator), it must be replaced with the free available disk in the network. If no free disk is available in the network, the administrator must be notified well in advance and the administrator will be able to arrange to have free disks available in the network. Before replacing the disk that is full with a free available disk, the new free disk must be formatted. The system allows the administrator to format the disk with any of the supported file systems. The system lets the administrator choose the file system that the disk is to be formatted with, from a list of supported file systems. The system also gives a table consisting of information on the file systems supported by the various operating systems. This information should help the administrator in making the decision to choose the right file system. Once chosen, the disk is formatted with the specified file system and marked “disk in use”

Case 8: Add/Delete Resources *and* Identify Newly Added/Deleted Devices

Whenever changes are made to the physical makeup of the network, the Enterprise Storage Management System will detect the changes and update the configuration of the System automatically. This means that there is no need for an administrator to go through the steps of allocating the new device and reconfiguring the System. For example, if new storage devices are added to the network, the System will detect those devices, format them as needed, and allocate the devices as primary storage devices or backup storage devices. This "plug and play" characteristic of the System gives administrators an easy way to increase their storage capacity, which is very important to the Lincoln Telephone Company; as it is growing quickly.

Case 9: Reporting the Backup failures

Lincoln Telephone Company realizes the significance of the data backup process. The storage management system is able to provide just that. The system allows the administrator to schedule the backup either manually or automatically. The system is very robust. It makes 100 percent sure that the data is not lost during the backup. The system is also capable of handling situations like network failures, data corruption, hardware failures, etc. In the event of network failure, the system alerts the administrator that the network is down. During the data backup, the system logs the state of the backup process in a system file. If there is any backup process that is running when the network failure occurs, the backup starts from the point where it failed last as soon as the network is up. This is also called incremental backup. This can be done automatically.

Case 10: Identifying Bad Disks

There are some occasions in Lincoln Telephone Company when a disk can be damaged or corrupted. The operators need a mechanism that will allow them to avoid these corruptions. This is done with a Scandisk system. The operator can set up a scheduler that will periodically run the Scandisk. The Scandisk will search the backup disks for bad sectors. When found, the operator will be alerted that the disk must be replaced. Meanwhile, the system will redirect the data from the corrupt disk to the new replacement disk. A status report will be generated indicating what portion of the data was salvageable and which files could not be saved.

Case 12: Archiving Dormant Files

Lincoln Telephone Company creates many files purely for record keeping, such as financial records and summaries. Once a new fiscal year is started, those financial records, while important, are no longer relevant to the company's current activities. Instead of letting these files take up precious storage space, the Enterprise Storage Management System will compress these files after they have not been accessed for given length of time, determined by a system administrator. Thus, the use of the storage space containing those files will be more optimal. More generally, the System will regularly examine the files it has stored for any files that have not been accessed for that

length of time. If it finds any dormant files, it will compress these files, noting it in a log file. The files will still be available upon request, however they would be stored in a compressed format, allowing for more efficient use of the storage space.

Interfaces:

The Enterprise Management Storage System will be required to interface with all other applications that Lincoln Telephone Company may be using in the sense that it will need to protect and manage the data used by all other applications. However, there is no need for a direct interface between the System and other applications because the System does not directly change the data, but rather just back it up or possibly move it to a different storage device. In that sense, it does not matter what applications Lincoln Telephone Company use, what format those applications store their data, or even what operating system those applications are run in. The System will protect any data created by any application in an efficient manner.

References:

Tivoli® Storage Products by IBM. http://www.tivoli.com