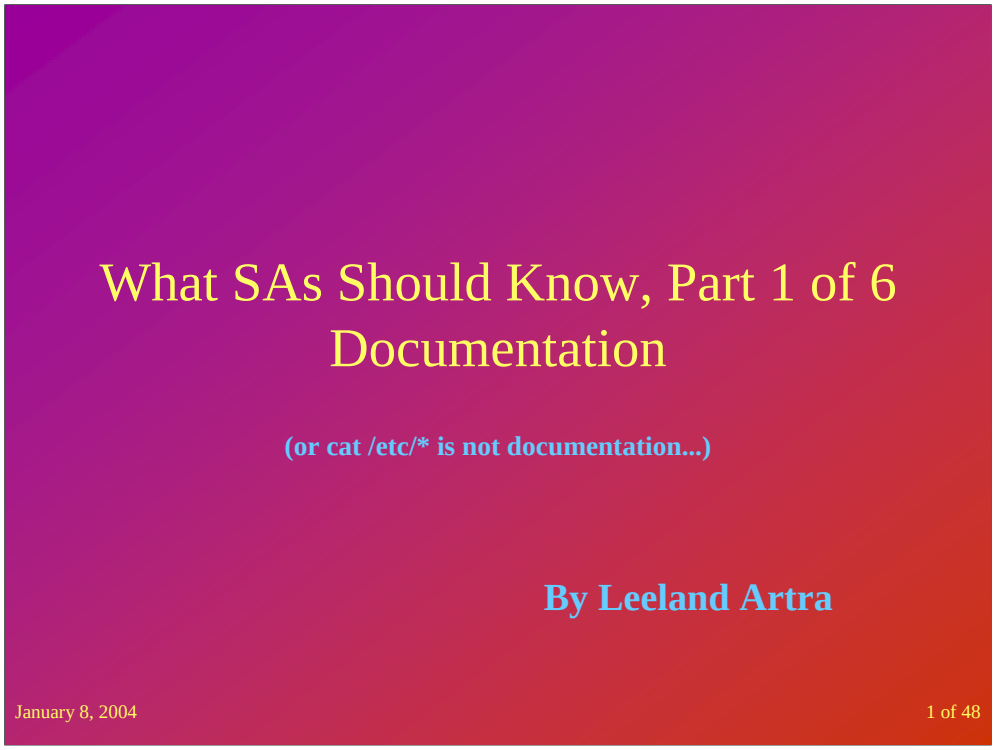


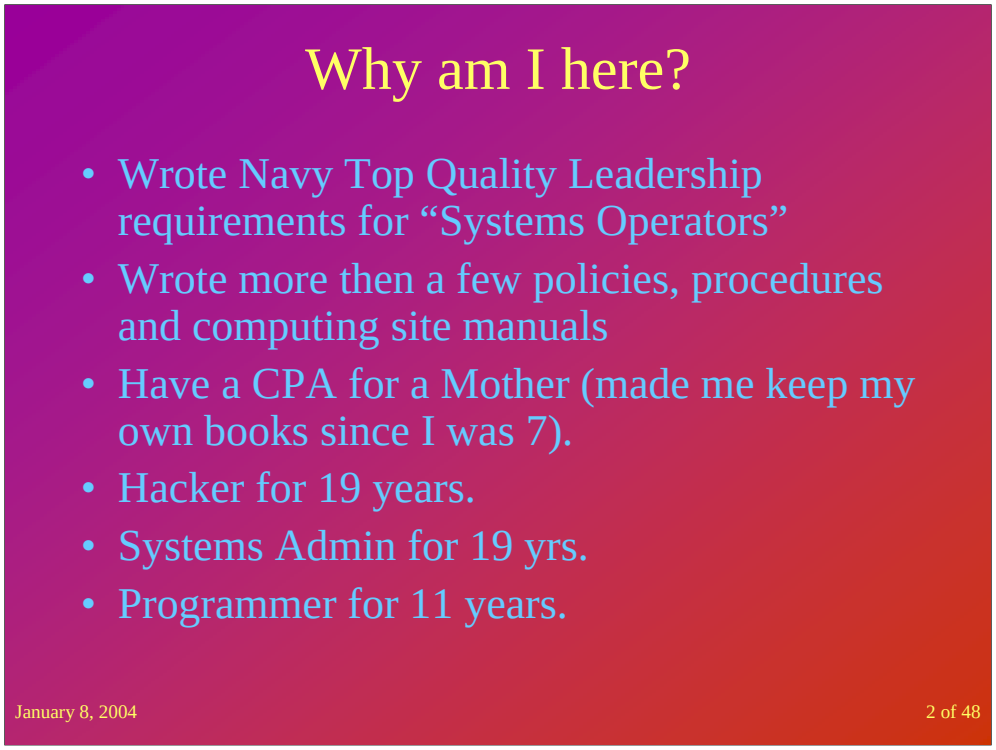
The talk will start with a discussion of the various types of documentation methods and technical charting with a brief how to draw, read and analyze each one. The discussion for each type of document will include suggestions for when, where and how it is appropriate to use. The talk will then provide a list of 'site documentation' and show how they interrelate.

This talk has spawned a technical white paper that describes in more detail each of the documentation methods discussed. Plus the paper includes examples and bibliographic references.

Version 1.0 June 2001

Version 1.1 August 2001

Version 2.0 January 2004

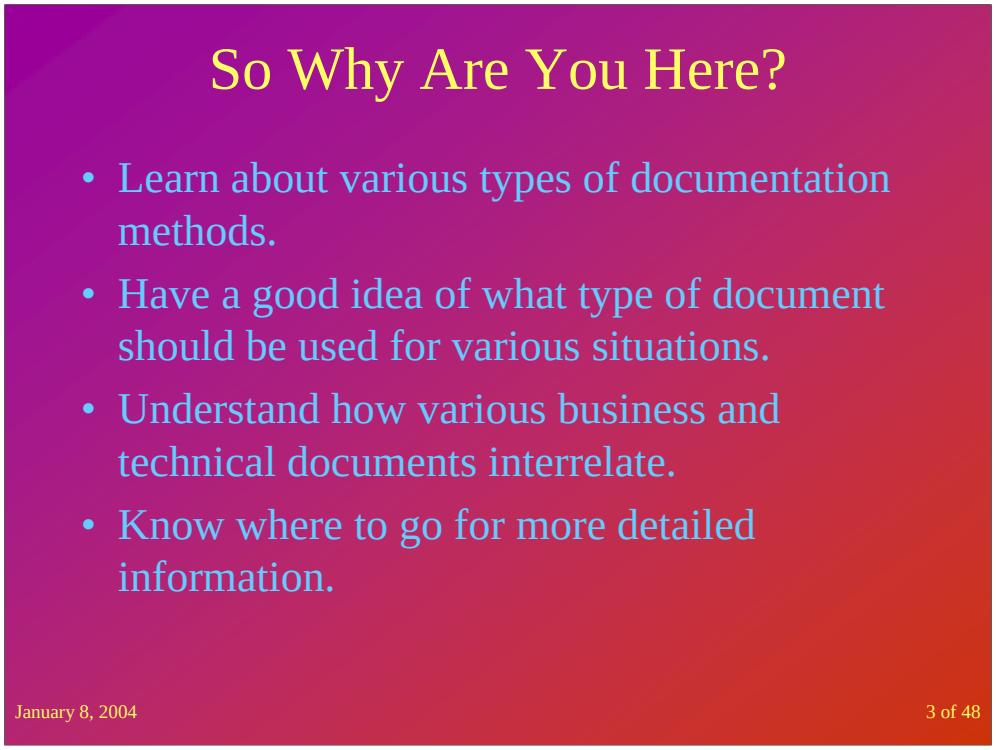


Leeland Artra is the President of Hero Network. He is currently trying to deny that he has long since moved into management roles and therefore remains very active in all the technology and systems development efforts. He is actively working to keep Hero Network's systems and technologies at peak performance.

Leeland sometimes bills himself as a information systems researcher. He has personally designed and implemented a number of new computer technologies. Currently Leeland's efforts are on creating a new real time cluster file system synchronization system for multiple OSs and a Java based Object Oriented database architecture.

Previously he has been: Director of Computer Systems Technology for CSI at the University of Washington; Senior Systems Researcher for the Cellworks Project at the University of Washington; Oracle futures development partner; Senior Systems Administrator for MBT at the University of Washington; Senior Systems Administrator for ITER and SAIC in San Diego Network; Senior Systems Training Officer for TTGP in the US Navy; and, of course, computer systems and programming consultant since 1984.

He is also actively involved with the International Systems Administrators Guild (SAGE) and jumps at most opportunities to promote that organization.



The talk will start with a discussion of the various types of documentation methods and technical charting with a brief how to draw, read and analyze each one. The discussion for each type of document will include suggestions for when, where and how it is appropriate to use. The talk will then provide a list of 'site documentation' and show how they interrelate.

By the end of the talk attendees should:

•Recognize various types of documentation methods;

•Have a good idea of what type of document should be used for various situations;

•Understand how various business and technical documents interrelate; and

•Know where to go for more detailed information.

Do You Wonder

- Why programs and systems are now not really worth using until the third or forth major release?
- Why you and your colleagues always seem to be 20 hours or more behind while working so many extra hours?
- Why fire control management of time and resources is reaching epidemic proportions?

Its Simple

You wish the industry would

"Do what I want, not what I do."

What do you Mean by that!

Things are just not getting done effectively.

This is because:

- Time to completion is given unrealistically high priority (because)
- Time for “delivery of profits” is set unreasonably soon

This is creating a ‘Just get it done.’ Environment.

So What?

My point exactly.

Lets Get Back to work.

OK, But What Can Be Done?

- Fix the attitude, get a “release is important, but doing it correctly is more important.”
- Recognize that deadlines are usually just random guesses that can be changed.
- **Work better**.

Work Better? How?

By doing something that is very hard:

- Become **self disciplined** to:
  - think things through.
  - plan things out well (technical specifications, flowcharts, project descriptions, procedural manuals)

Good planning and using technical charts has never been easy. But, it has historically been worth the effort.

So Why Are You Here Really?

Because, Grandpa always said

Prior Proper Planning
Prevents Poor Performance

Ouch!

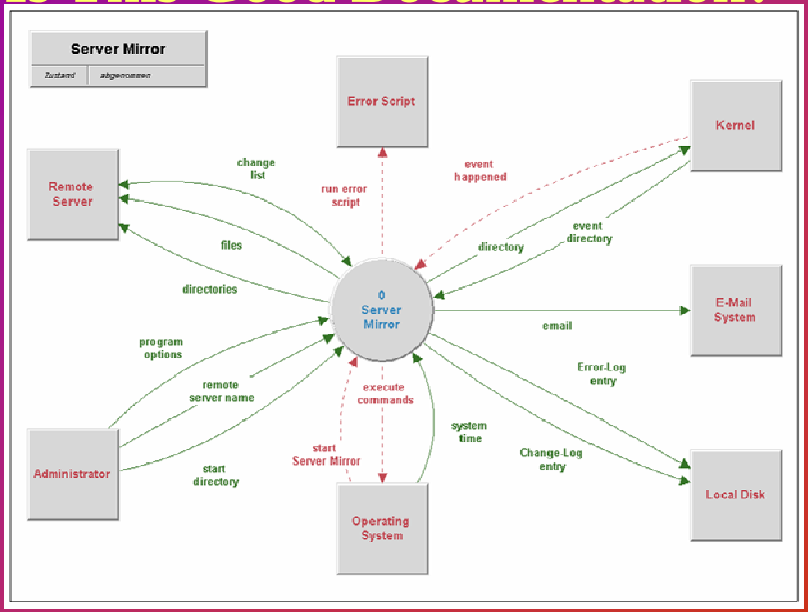
- I hate it when things I already know are the answer to problems I am having.

Documenting Is Not Easy

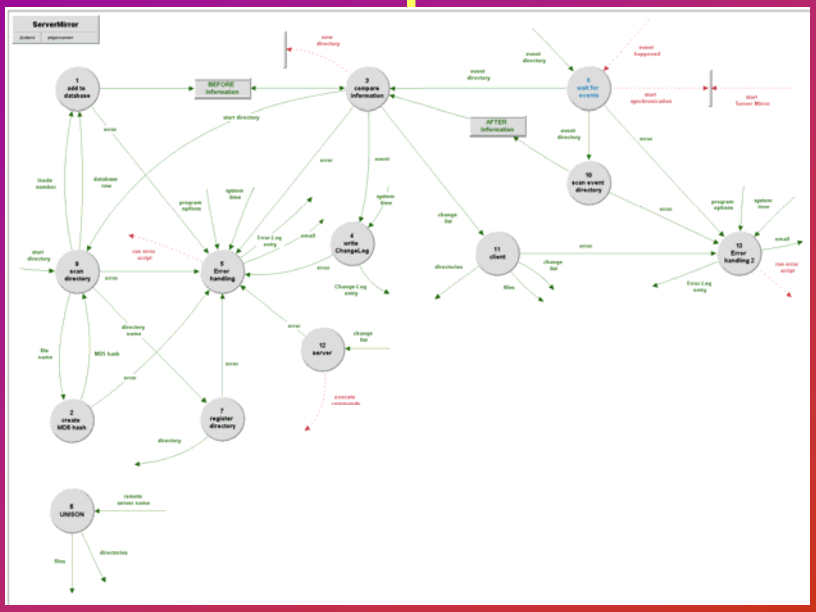
Your documents Must:

- Communicate your intent clearly
- Come together to create a better world

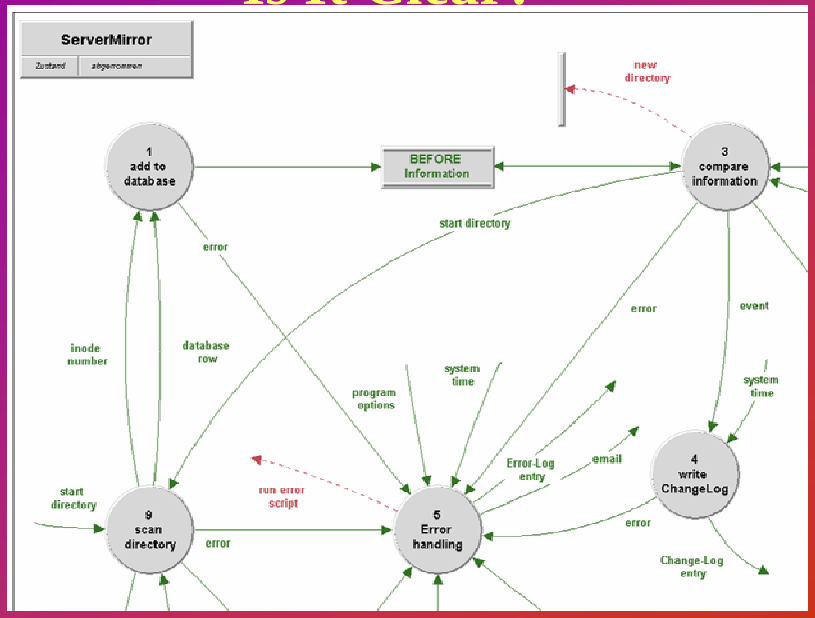
Is This Good Documentation?



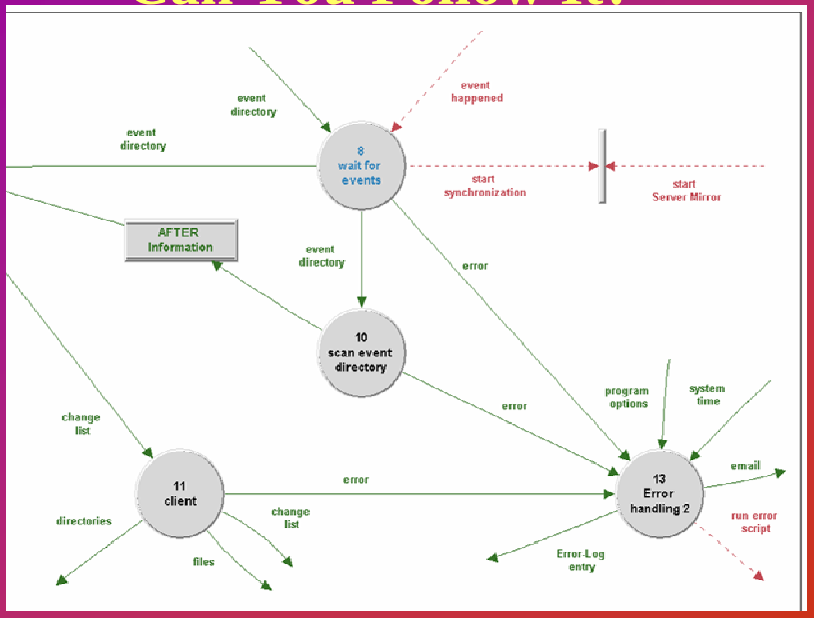
Or Perhaps This?



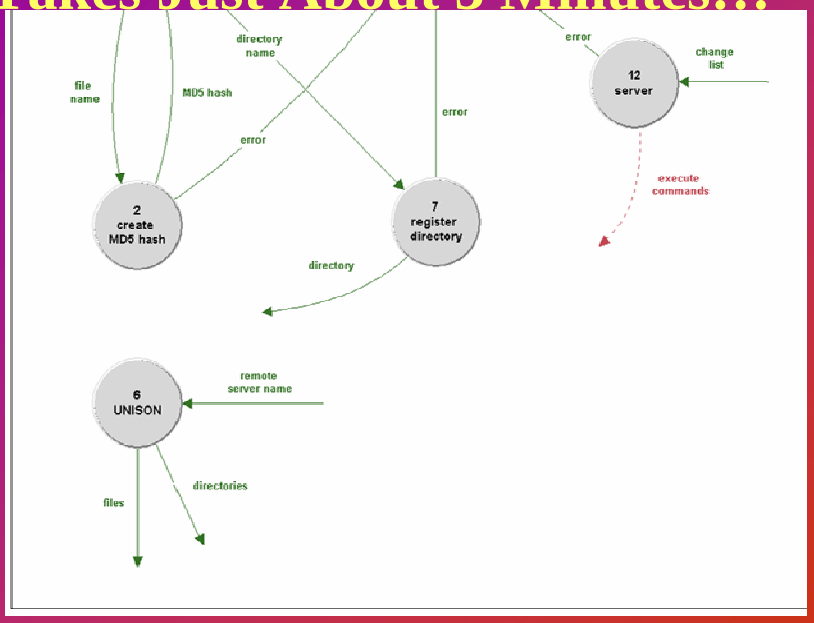
Is It Clear?


ServerMirror
new directory
1 add to database
BEFORE Information
3 compare information
start directory
error
error
event
inode number
database row
system time
program options
system time
Error-Log entry
email
4 write ChangeLog
start directory
run error script
9 scan directory
error
5 Error handling
error
Change-Log entry

Can You Follow It?



Takes Just About 3 Minutes…



Basic Guidelines

- Use Descriptive Titles
- Know your chart types and symbols well
- Keep document focused on one idea or goal
- Keep documents simple
- Use the simplest method when charting
- Provide good cross-references
- Navigation lines should not intersect
- Keep documents as small as possible

Ink to info ratio.

Technical Charts

Main Flavors:

Outline

Matrix

Block

Object

Project

Matrixes

- Organizes information systematically
- Allows for comparison and grouping
- Have been used for as long as we know
- Are easily understood
- Tables or charts come in a few flavors:
  L, Y, T, X
- There are others

Simple Matrix

	Build New BSD Server	Write New Project Plan	Draw Network Diagram	Write Program From Project Plan	Perform Project QA	Give Presentation on Project
Reading Core Material	0	+	0	+	+	+
Doing Research	0	+	0	+	+	+
Diagram Work	+	+	+	0	0	0
Speaking in Front of Peers	Δ	Δ	Δ	Δ	+	+
Programming Skills	+	Δ	0	+	0	Δ
Hardware Supplies	+	Δ	Δ	Δ	0	Δ
Team Work	0	0	Δ	0	+	Δ
Writing	Δ	+	Δ	0	+	+

Legend

+ = strong relationship
0 = some relationship
Δ = no relationship

What Project is Best For You?

KWHL Chart

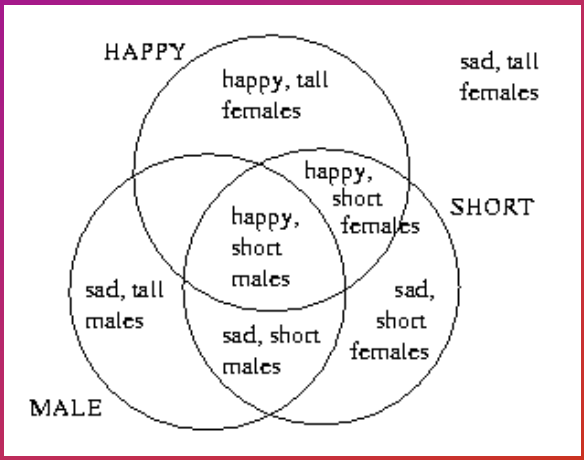
- "Cool"
- Know, Want, How, Learn
- Created in 1986 as a teaching tool by Donna Ogle
- Captures known information well

K	W	H	L
What do we know?	What do we want to find out?	How can we find out what we want to learn?	What did we learn?

Attributes or Characteristics we expect to use:

Venn Diagrams

- Pretty Basic
- No one else in the history of math has been known so well for so little



But, Wait

- A Century before John Venn in Leonhard Euler's *Opera Omnia*

III. Or si la notion C étoit toute entiere hors de la notion B, elle seroit aussi tout entiere hors de la notion A, comme on voit par cette figure

d'où nait cette forme de syllogisme:

Tout A est B:
Or Nul C n'est pas B, ou Nul B n'est pas C:
Donc Nul C n'est pas A.

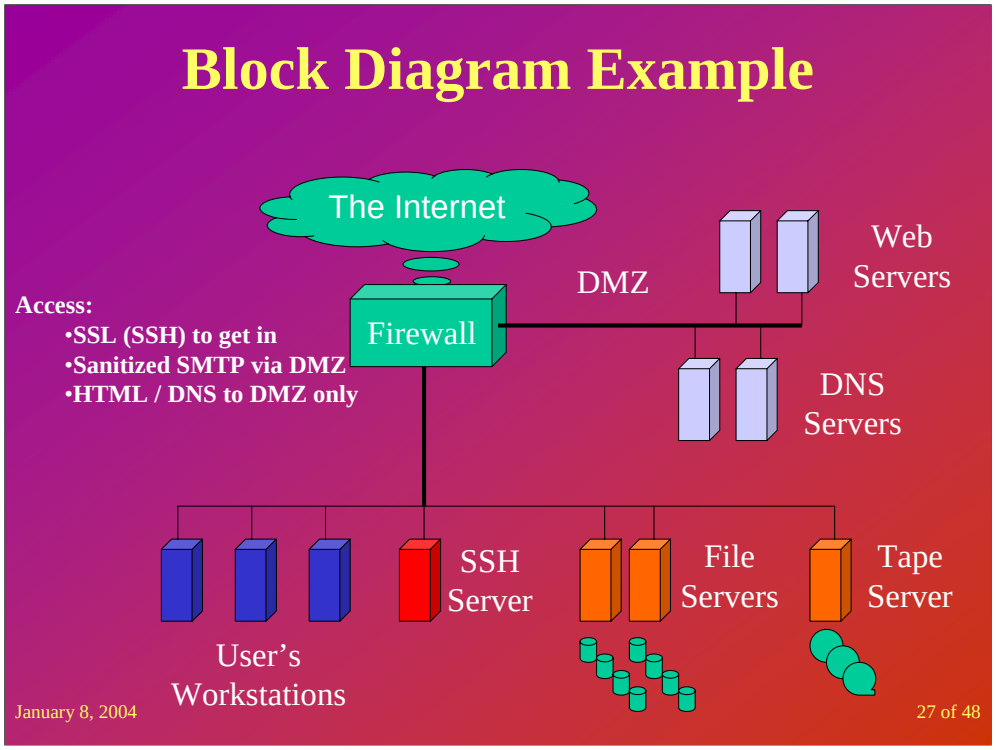
A "Euler diagram"

(Courtesy of Lehigh University Library)

Block Diagrams

Block diagram are used to:

- Represent entire processes
- Person / Component through a specific process
- Combinations of people and machines
- Transactions following forms or other documents
- etc.



Block diagrams, as a tool for clarifying situations and thus improving knowledge and understanding, is particularly useful when used by a group or team. This is because by drawing a block diagrams together, the team:

•develops a common understanding of the situation;

•contributes a larger pool of knowledge than an individual can (assuming team members are well chosen for their knowledge and experience); and

•can agree a common approach to solving problems, resolving ambiguities and making improvements.

Flowchart

- Is block diagram that ***follows*** a standard

Used to:

- Document process and interrelationship of process steps;
- Identify actual and ideal paths that any product or process moves or flows through;
- Flowcharting to help communicate what actually happens or needs to happen;
- Identify problems and potential improvements in a process; and
- Describe:
  - An entire processes and all its components,
  - One person or component through a process
  - Combinations of people and machines
  - Transactions following forms or other documents,
  - Labor intensive processes, and
  - Organizational procedures and cycles.

Flowchart Types

- Data Flowchart
- Program Flowchart
- System Flowchart
- Program Network flowchart
- System Resource Flowchart

January 8, 2004 29 of 48

1. **Data Flowchart**: The data flowchart describes the path data moves or flows through during a solving of a problem. The data flow chart is made using data symbols (to show existence of data or media usage), process symbols (to show the process to be executed on the data or a machine function), line symbols and special symbols. This flowchart has specialized symbols to show the reading and writing of data during processing. It has the requirement that process symbols be preceded and followed by data symbols. The data flowchart begins and ends with a data symbol or a special symbol.
2. **Program Flowchart**: The program flowchart describes the sequence of operations in a program or procedure. It consists of the process symbols including logic symbols, line symbols and the special symbols. The program flowchart begins and ends with a special symbol.
3. **System Flowchart**: System flowcharts describe the control operations and the data flow for a given system. It consists of the data symbols (to show existence of data or media usage), process symbols (to show the process to be executed on the data or the logical path to be followed), line symbols and special symbols.
4. **Program Network flowchart**: The program network flowchart describes the path of process, program or procedure activities and the interactions to related data. Each object (program, procedure or process) in a program network chart is shown only once. This chart is made using data symbols (to show existence of data), process symbols (to show the process to be executed on the data), line symbols and special symbols
5. **System Resource Flowchart**: The system resource flowchart shows the configuration of data and process objects for the solving of a problem or set of problems. It consists of data symbols (to show media usage), process symbols (to represent processors or CPUs or channels), line symbols, and special symbols.

Flowchart Data Symbols


Data


Stored Data


Internal Storage


Sequential Access Storage


Direct Access Storage


Document


Manual Input


Card


Punched Tape

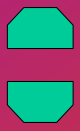

Display

Flowchart Process Symbols


Basic Process Symbol


Parallel Mode


Specific Process Symbol


Loop Limit


Manual Operation


Preparation


Decision

Flowchart Line Symbols

Line (Logic Flow)

Control Transfer

Communication Link

Dashed Line (Alternative Relationship)

Flowchart Special Symbols

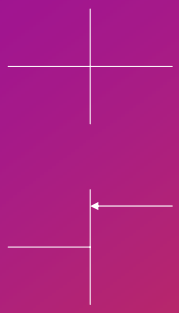
Connector

Terminator

Annotation

Ellipsis (three dots, omission)

Flowchart Crossing Lines

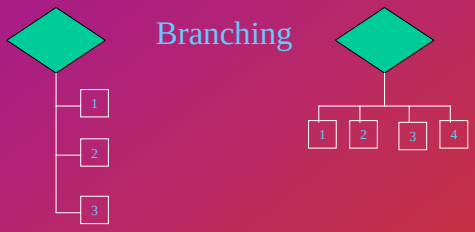


No connection

Join lines of logic

Flowchart Extras

Multiple Symbols



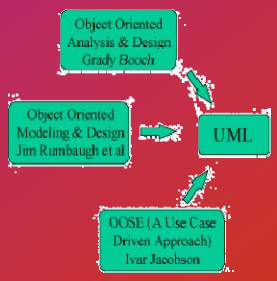
Flowchart Recommended Policies

- Drawn on white, unlined 8 1/2" x 11" paper on one side only.
- Place name, and the title at the top of each page, along with the page number
- Use only standard flowcharting symbols
- If possible draw using a template or program
- Print the contents of each symbol legibly
- Flowcharts start on the top of the page and flow down and to the right
- Comments are in English, not programming languages
- Each subroutine is flowcharted on a separate page
- Each subroutine begins with a terminal symbol labeled with its name and a terminal symbol labeled return at the end
- Flow lines between symbols use arrowheads to indicate the direction of the logic flow

Unified Modeling Language (UML)

- Graphs of object interactions and relationships
- "Modeling Language" (not a method)
  - Expresses design
  - Defines interactions

"It can be used to model anything. It is designed to be user extended to fill any modeling requirement."



January 8, 2004 37 of 48

Most UML diagrams and some complex symbols are graphs containing nodes connected by paths. The information is mostly in the topology, not in the size or placement of the symbols (there are some exceptions, such as a sequence diagram with a metric time axis). There are three kinds of visual relationships that are important: connection (usually of lines to 2-d shapes), containment (of symbols by 2-d shapes with boundaries), and visual attachment (one symbol being "near" another one on a diagram). These visual relationships map into connections of nodes in a graph, the parsed form of the notation.

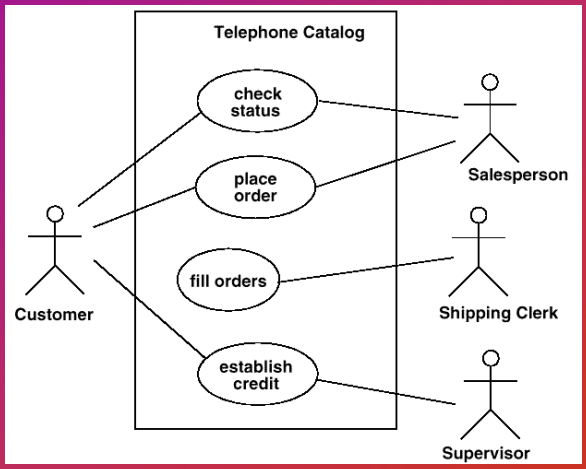
UML notation is intended to be drawn on 2-dimensional surfaces. Some shapes are 2-dimensional projections of 3-d shapes (such as cubes) but they are still rendered as icons on a 2-dimensional surface. In the near future true 3-dimensional layout and navigation may be possible on desktop machines but it is not currently practical.

UML came about when James Rumbaugh joined Grady Booch at Rational Software. They both had object oriented syntaxes and needed to combine them. Semantically they were very similar, it was mainly the symbols that needed to be unified. The result was UML 1.0

Then Ivar Jaconson joined them. He brought with him the syntax for use cases which was added in UML 1.1. The Object Management Group adopted the UML1.1 specification in November 1997 making it an independent industry standard. Some small changes were made in in versions 1.3 and 1.4. Version 2.0 is currently being researched.

Use Case

- Shows the relationship among actors and use cases within a system.
- Actors



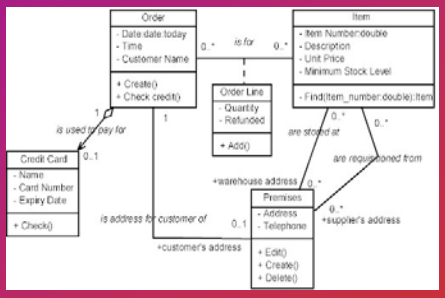
Shows an outside-in view of the procedures available in the use of the system. These are summary diagrams and between them should contain all use cases available in the system and so all the available functionality of the system, represented at a high level.

An actor is a role of object or objects outside of a system that interacts directly with it as part of a coherent work unit (a use case). An Actor element characterizes the role played by an outside object; one physical object may play several roles and therefore be modeled by several actors.

An actor may be shown as a class rectangle with the stereotype "actor". The standard stereotype icon for an actor is the "stick man" figure with the name of the actor below the figure.

Class / Object Diagram

- Shows the static structure of the system components.
- Define the properties of objects.



Class diagrams show the static structure of the systems. Classes define the properties of the objects which belong to them.

A class diagram is a graph of Classifier elements connected by their various static relationships. (Note that a "class" diagram may also contain interfaces, packages, relationships, and even instances, such as objects and links. Perhaps a better name would be "static structural diagram" but "class diagram" is shorter and well established.)

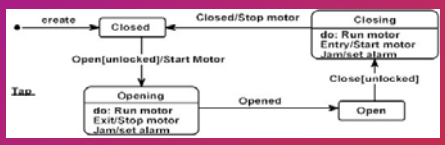
A class diagram is a graphic view of the static structural model. The individual class diagrams do not represent divisions in the underlying model.

These include:

Attributes - (second container) the data properties of the classes including type, default value and constraints

State Charts

- Often used in real time embedded systems
- For a class they show:
  - Order of operations
  - Conditions for operations responces
  - The response.
- Class-centric view of system functionality



State charts, often used more in real time embedded systems than in information systems, show, for a class, the order in which incoming calls to operations normally occur, the conditions under which the operations respond and the response. They are a class-centric view of system functionality as opposed to sequence and collaboration diagrams which are a use case-centric view of functionality.

They include:

•States - oblong boxes which indicate the stable states of the object between events.

•Transitions - the solid arrows which show possible changes of state.

•Events - the text on the transitions before the '/' showing the incoming call to the object interface which causes the change of state.

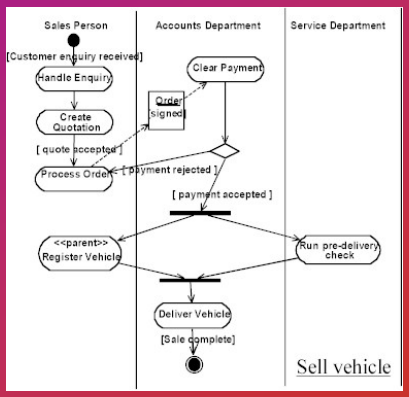
•Conditions - a Boolean statement in square brackets which qualifies the event.

•Actions - the text after the '/' which defines the objects response to the transition between states.

•Extra syntax which defines state centric functionality

Activity Diagram

- A general purpose flowchart with a few extras. It can be used to detail a business process or to help define complex iteration and selection in a use case description.



It includes:

•Active states - oblongs with rounded corners which describe what is done.

•Transitions - which show the order in which the active states occur and represent a thread of activity.

•Conditions - (in square brackets) which qualify the transitions.

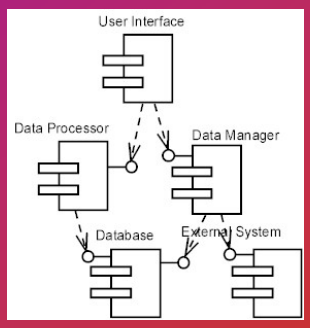
•Decisions - (nodes in the transitions) which cause the thread to select one of multiple paths.

•Swimlanes - (vertical lines the length of the diagram) which allow activities to be assigned to objects.

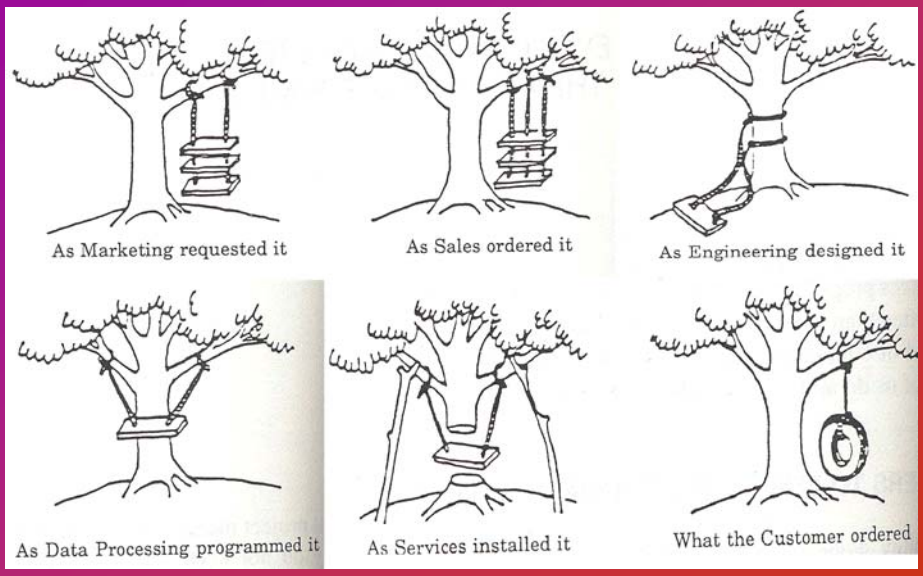
•Synch States - horizontal or vertical solid lines which split or merge threads (transitions)

Component Diagrams

- Shows the types of software components in the system, their interfaces and dependencies.

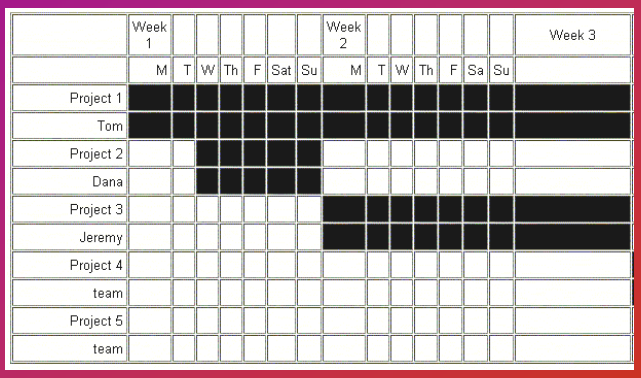


Project Management



Gantt Charts

- Standard format for displaying a schedule graphically.
  - **Resources**
  - **Time line**
  - **Work calendar**
  - **Jobs (operations)**
  - **Production lots**



January 8, 2004 44 of 48

The Gantt chart is the standard format for displaying a schedule graphically. It consists of a horizontal bar chart with time as the horizontal axis and either resources, jobs, or orders as the vertical axis. Individual operations are displayed as horizontal bars in the chart, indicating the time at which the job begins and ends. Many variations on the Gantt chart exist to display additional kinds of information.

Gantt charts can be drawn physically on paper, but nowadays are usually implemented through computer software.

•**Resources** Resources are displayed on the left side of the Gantt chart as colored bars along marked with their resource codes (Mixer1, Packer1, etc.).

•**Time line** At the top of the Gantt chart, you can see the continuous time line, with the days and days of the week marked for clarity.

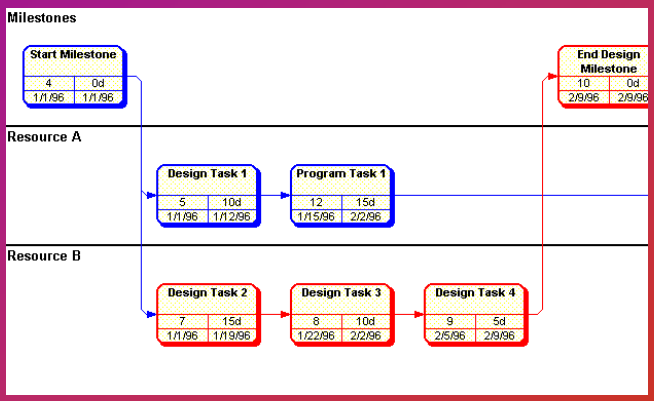
•**Work calendar** The gray shaded areas in the Gantt chart represent the time periods in which each resource is available to do work. Unavailable time (e.g., vacations, lunch breaks, maintenance breaks, etc.) are shown with a plain white background. For the "InspectionCenter1" resource, the height of the gray areas represents the number or quantity of resources that are available during that time period.

•**Jobs (operations)** The colored bars in the body of the Gantt chart represent the individual jobs or operations that have been scheduled. From the position of the job on the Gantt chart, you can see which resources the job has been assigned to, when the job is scheduled to start and end, as well as the job's setup time and whether the job will be suspended over any unavailable time periods. The text on the colored bars conveys various additional information, such as the job code, the item being produced, and the quantity being produced.

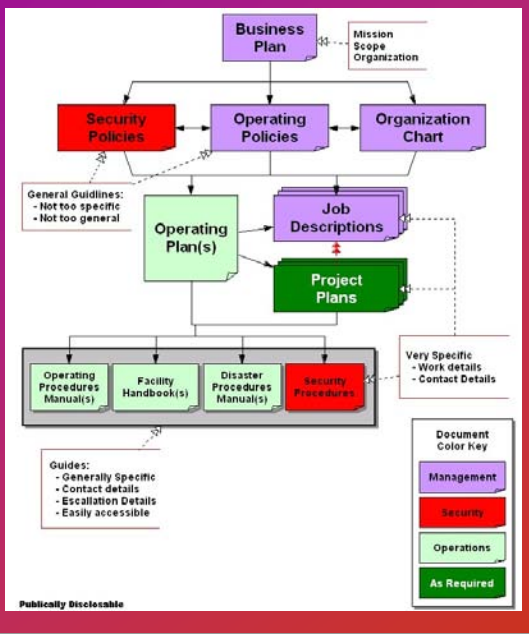
•**Production lots (orders)** In this example, jobs belonging to the same production lot or order are displayed in the same color. For example, the 6 jobs displayed in yellow correspond to the various processes in lot "04". When the mouse cursor is placed over one of these jobs, lines appear showing the precedence constraints between the jobs in the production lot.

PERT Charts

- Effective method of presenting a project's timetable visually
- Can include things like project deadlines and group meeting times as well as individual roles and responsibilities



Why Document this?



See the next talk for to answer this questions.