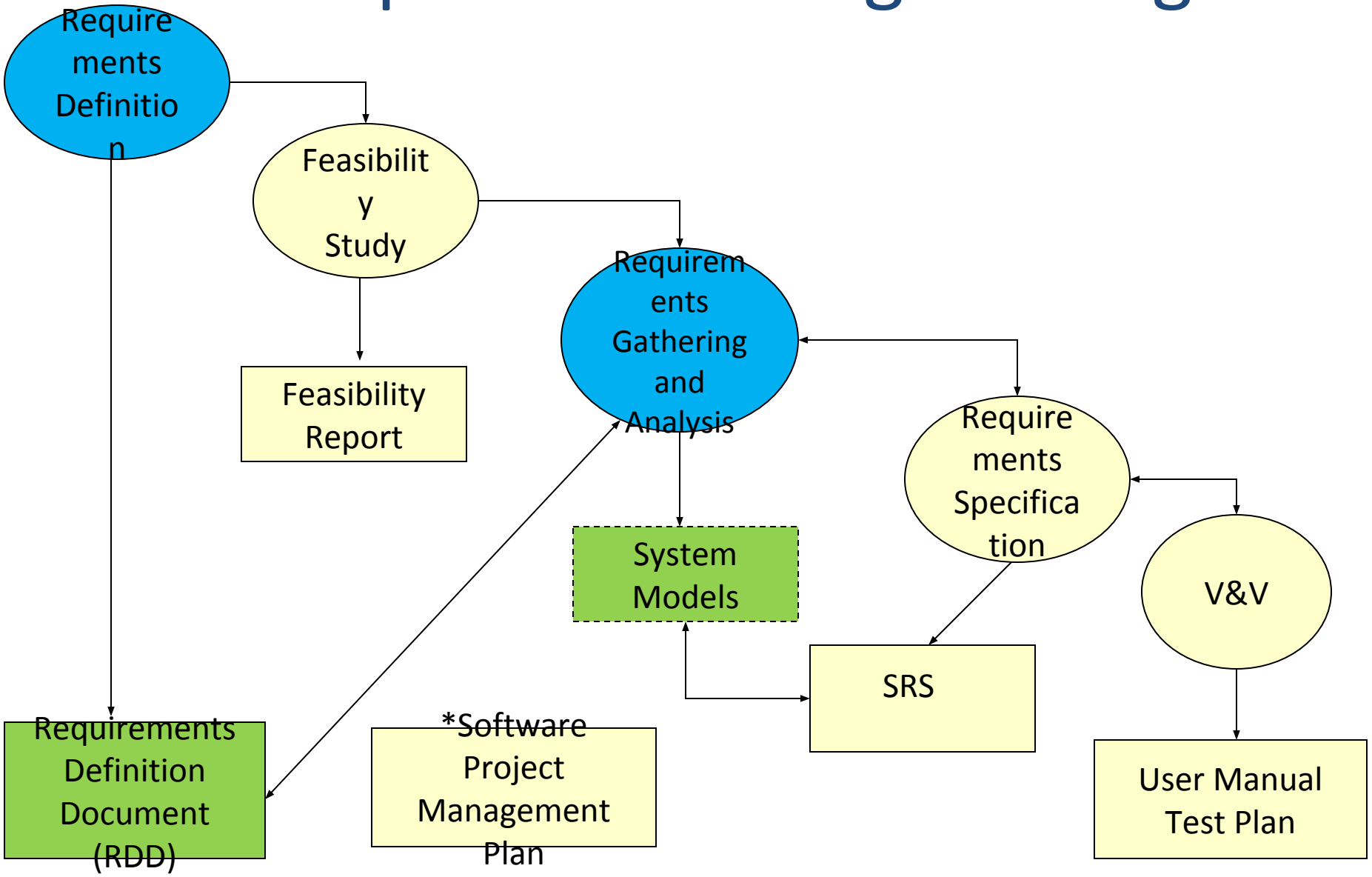
Unit 3- Requirement Phase

Contents

- Requirement Engineering.
  - Requirement Gathering
  - Types of Requirment
  - Comparison
    - Task
    - Use Case
- Feasibility Study
    - Types of Feasibility
- Software Requirement Document
  - Template

Requirements Engineering



Requirement Engineering

- The broad spectrum of tasks and techniques that lead to an understanding of requirements is called *requirements engineering.*
- Requirements engineering provides the appropriate mechanism for understanding **what the customer wants,** analyzing need, assessing feasibility, negotiating a reasonable solution, specifying the solution unambiguously, validating the specification, and managing the requirements as they are transformed into an operational system [Tha97].
- It encompasses seven distinct tasks: **inception, elicitation, elaboration, negotiation, specification, validation, and management.** It is important to note that some of these tasks occur in parallel and all are adapted to the needs of the project

- **Inception:** At project inception,you establish a basic understanding of the problem, the people who want a solution, the nature of the solution that is desired, and the effectiveness of preliminary communication and collaboration between the other stakeholders and the software team.
- **Elicitation.**
  - Problems of scope.
  - Problems of understanding.
  - Problems of volatility.
- **Elaboration:** Elaboration is driven by the creation and refinement of **user scenarios** that describe how the end user (and other actors) will interact with the system. Each user scenario is parsed to extract analysis classes—business domain entities that are visible to the end user.

- **Negotiation.**

- **Specification:** A specification can be a written document, a set of graphical models, a formal mathematical model, a collection of usage scenarios, a prototype, or any combination of these.
- **Validation:**

Why do we need Requirements?

- When 38 IT professionals in the UK were asked about which project stages caused failure, respondents mentioned “requirements definition” more than any other phase.

What are requirements?

- A requirement is a statement about an intended product that specifies what it should do or how it should perform.

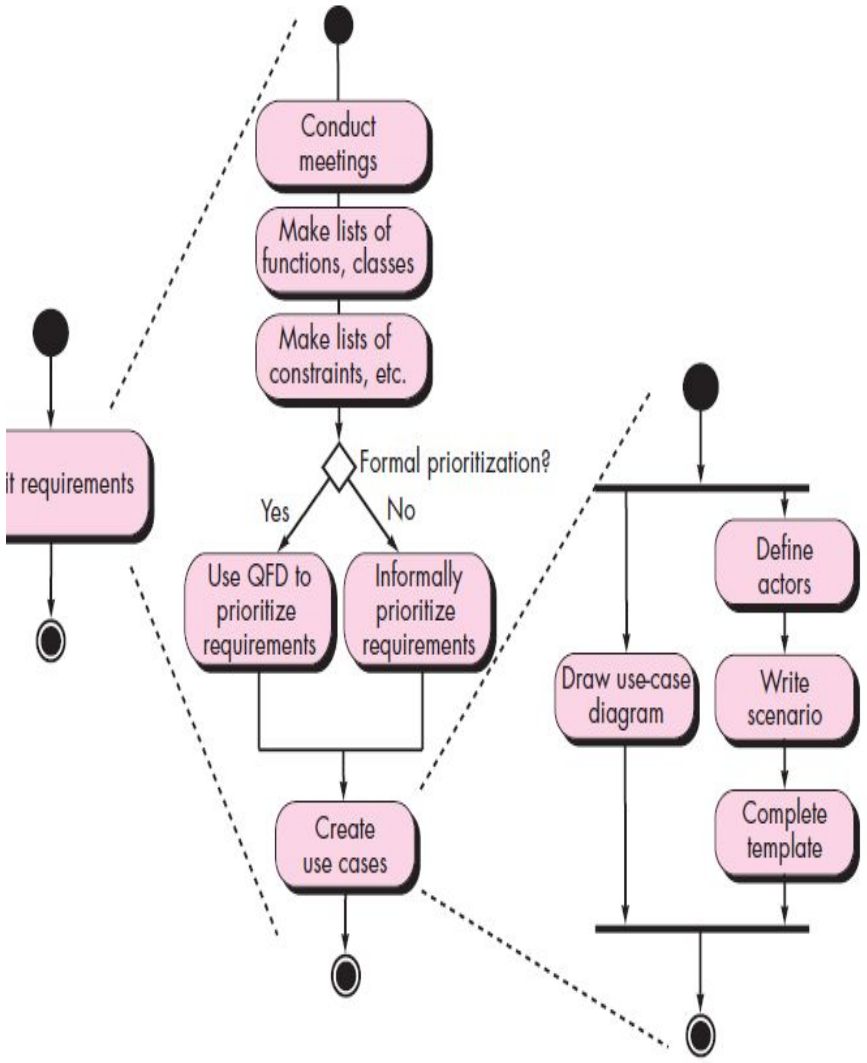
- Goal: To make as specific, unambigous, and clear as possible.

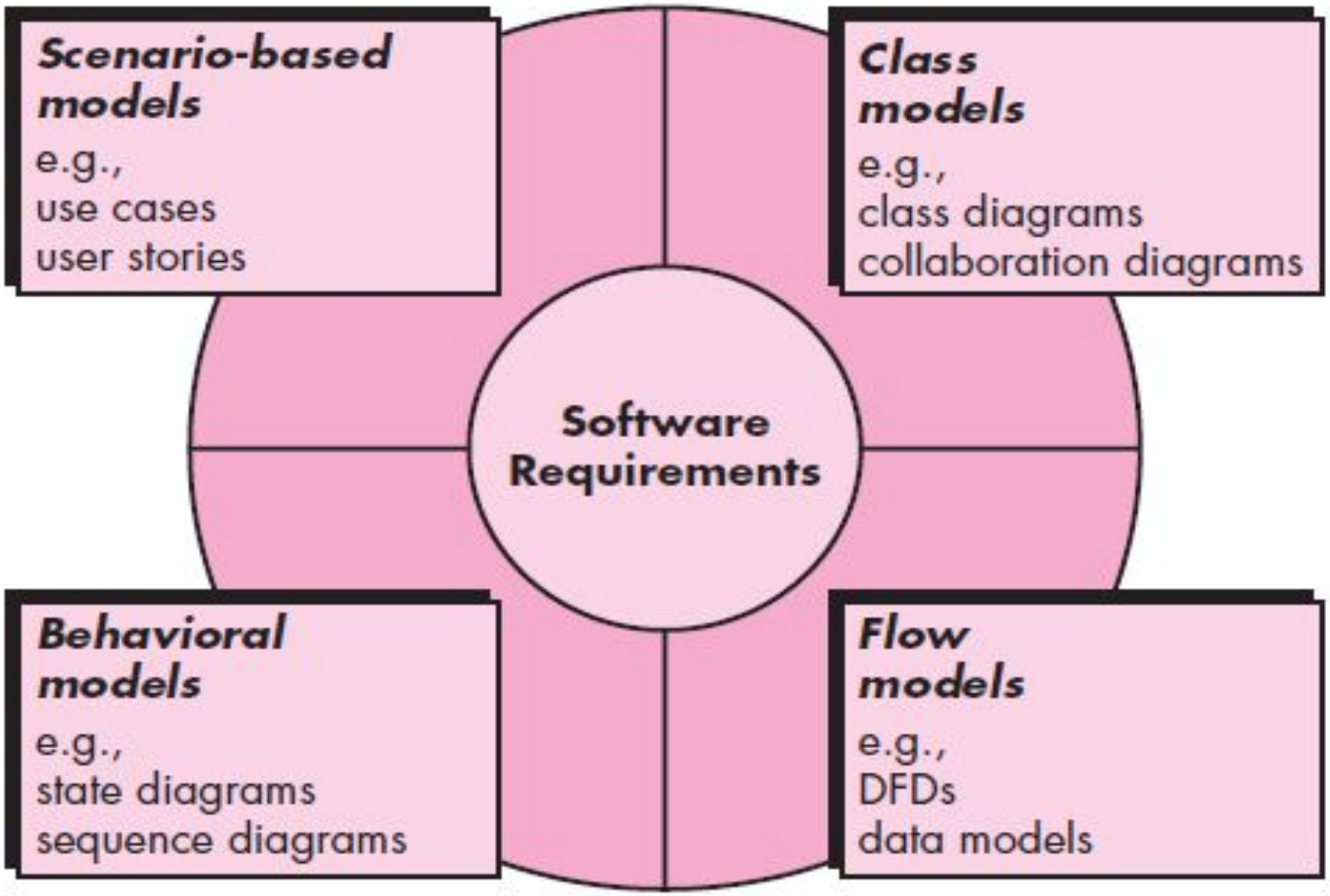
What requirements should be gathered?

- Functional: What the product should do.
- Data requirements: Capture the type, volatility, size/amount, persistence, accuracy and the amounts of the required data.
- Environmental requirements: a) context of use b) Social environment (eg. Collaboration and coordination) c) how good is user support likely to be d) what technologies will it run on
- User Requirements: Capture the characteristics of the intended user group.
- Usability Requirement: Usability goals associated measures for a particular product (More info on Chapter 6).

Requirement Modeling

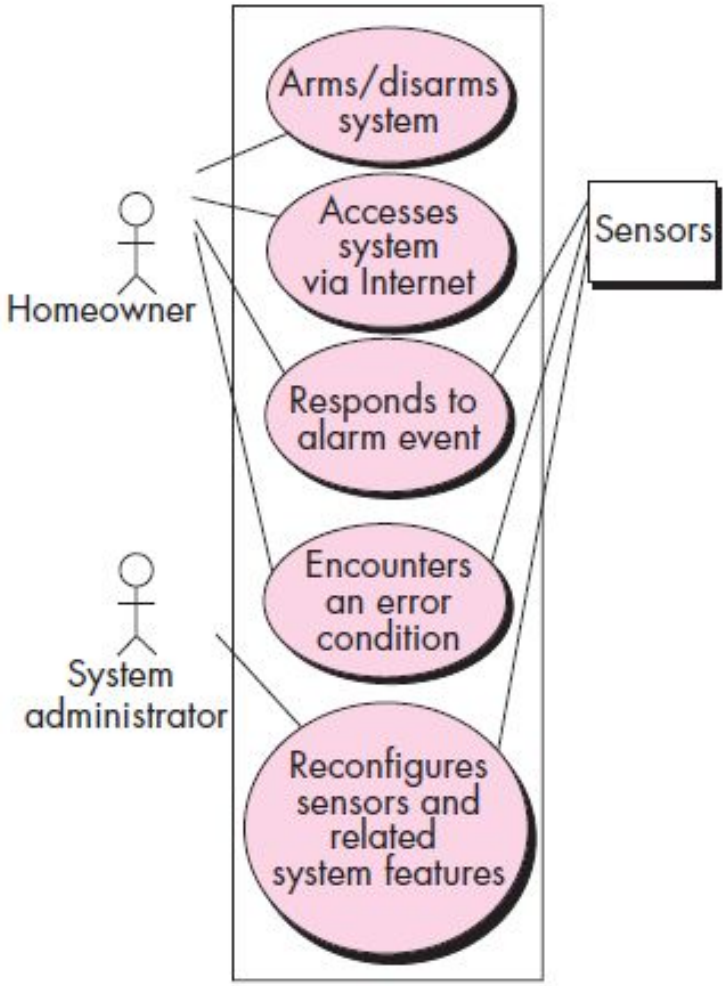
- *Scenario-based models* of requirements from the point of view of various system "actors"
- *Data models* that depict the information domain for the problem
- *Class-oriented models* that represent object-oriented classes (attributes and operations) and the manner in which classes collaborate to achieve system requirements
- *Flow-oriented models* that represent the functional elements of the system and how they transform data as it moves through the system
- *Behavioral models* that depict how the software behaves as a consequence of external "events"




Scenario-based models
e.g.,
use cases
user stories
Class models
e.g.,
class diagrams
collaboration diagrams
Software Requirements
Behavioral models
e.g.,
state diagrams
sequence diagrams
Flow models
e.g.,
DFDs
data models

Requirement Modeling

- Who is the primary actor, the secondary actor(s)?
- What are the actor's goals?
- What preconditions should exist before the story begins?
- What main tasks or functions are performed by the actor?
- What exceptions might be considered as the story is described?
- What variations in the actor's interaction are possible?
- What system information will the actor acquire, produce, or change?
- Will the actor have to inform the system about changes in the external environment?
- What information does the actor desire from the system?
- Does the actor wish to be informed about unexpected changes?



Sensor

Name
Type
Location
Area
Characteristics

Identify()
Enable()
Disable()
Reconfigure()

Reading commands

System status = "Ready"
Display msg = "enter cmd"
Display status = steady

Entry/subsystems ready
Do: poll user input panel
Do: read user input
Do: interpret user input

State name
State variables
State activities

Illustration

Use case: Access camera surveillance via the Internet—display camera views (ACS-DCV)

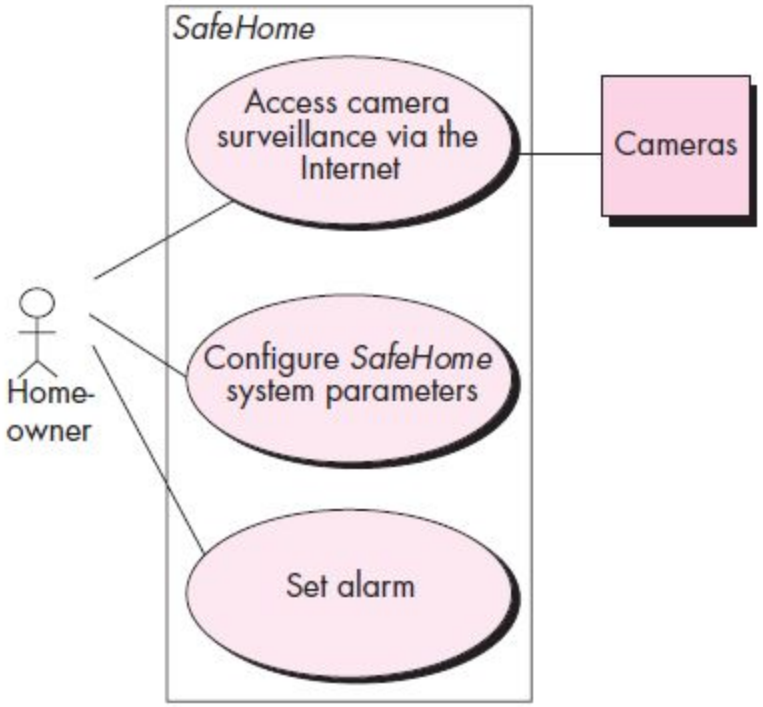
Actor: homeowner

If I'm at a remote location, I can use any PC with appropriate browser software to log on to the *SafeHome Products* website. I enter my user ID and two levels of passwords and once I'm validated, I have access to all functionality for my installed *SafeHome* system. To access a specific camera view, I select "surveillance" from the major function buttons displayed. I then select "pick a camera" and the floor plan of the house is displayed. I then select the camera that I'm interested in. Alternatively, I can look at thumbnail snapshots from all cameras simultaneously by selecting "all cameras" as my viewing choice. Once I choose a camera, I select "view" and a one-frame-per-second view appears in a viewing window that is identified by the camera ID. If I want to switch cameras, I select "pick a camera" and the original viewing window disappears and the floor plan of the house is displayed again. I then select the camera that I'm interested in. A new viewing window appears.

Use case: Access camera surveillance via the Internet—display camera views (ACS-DCV)

Actor: homeowner

1. The homeowner logs onto the *SafeHome Products* website.
2. The homeowner enters his or her user ID.
3. The homeowner enters two passwords (each at least eight characters in length).
4. The system displays all major function buttons.
5. The homeowner selects the "surveillance" from the major function buttons.
6. The homeowner selects "pick a camera."
7. The system displays the floor plan of the house.
8. The homeowner selects a camera icon from the floor plan.
9. The homeowner selects the "view" button.
10. The system displays a viewing window that is identified by the camera ID.
11. The system displays video output within the viewing window at one frame per second.


SafeHome
Access camera surveillance via the Internet
Cameras
Configure SafeHome system parameters
Set alarm
Home-owner

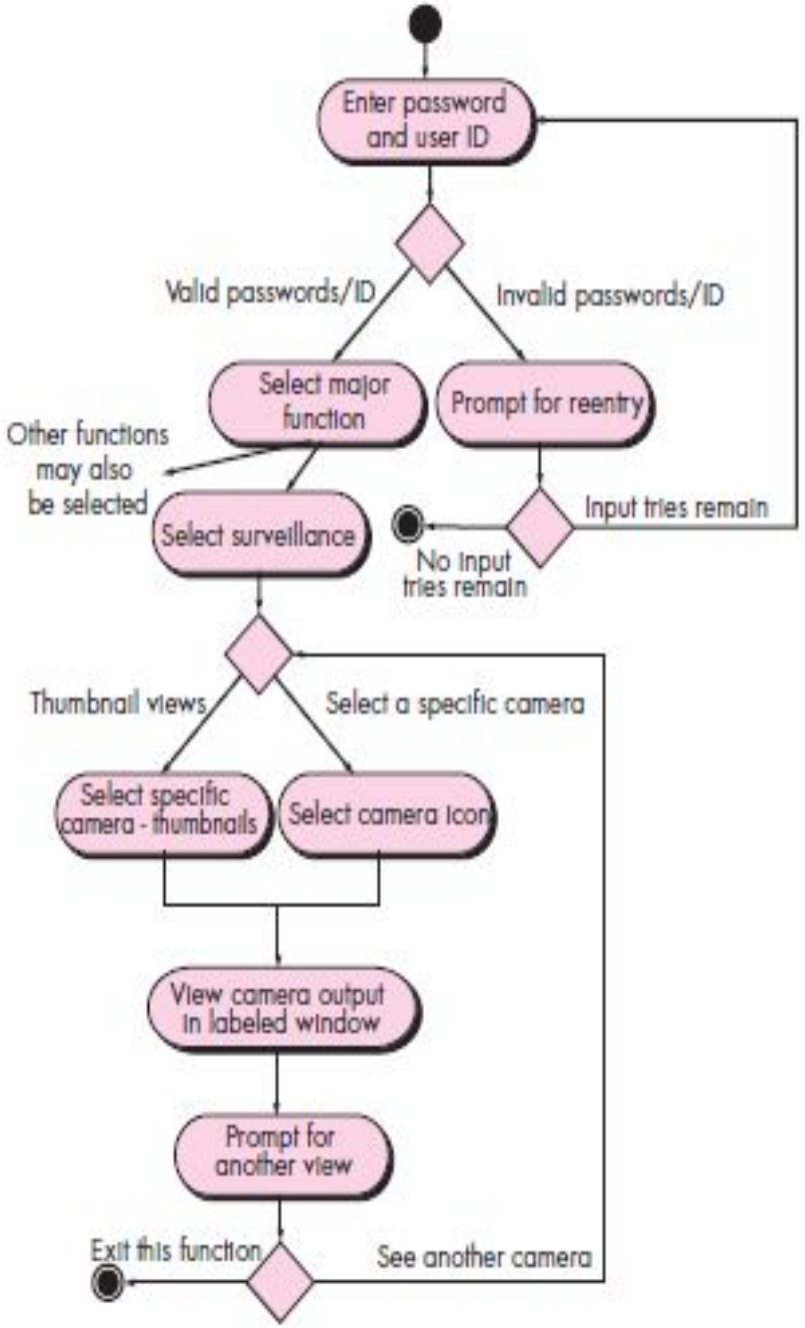
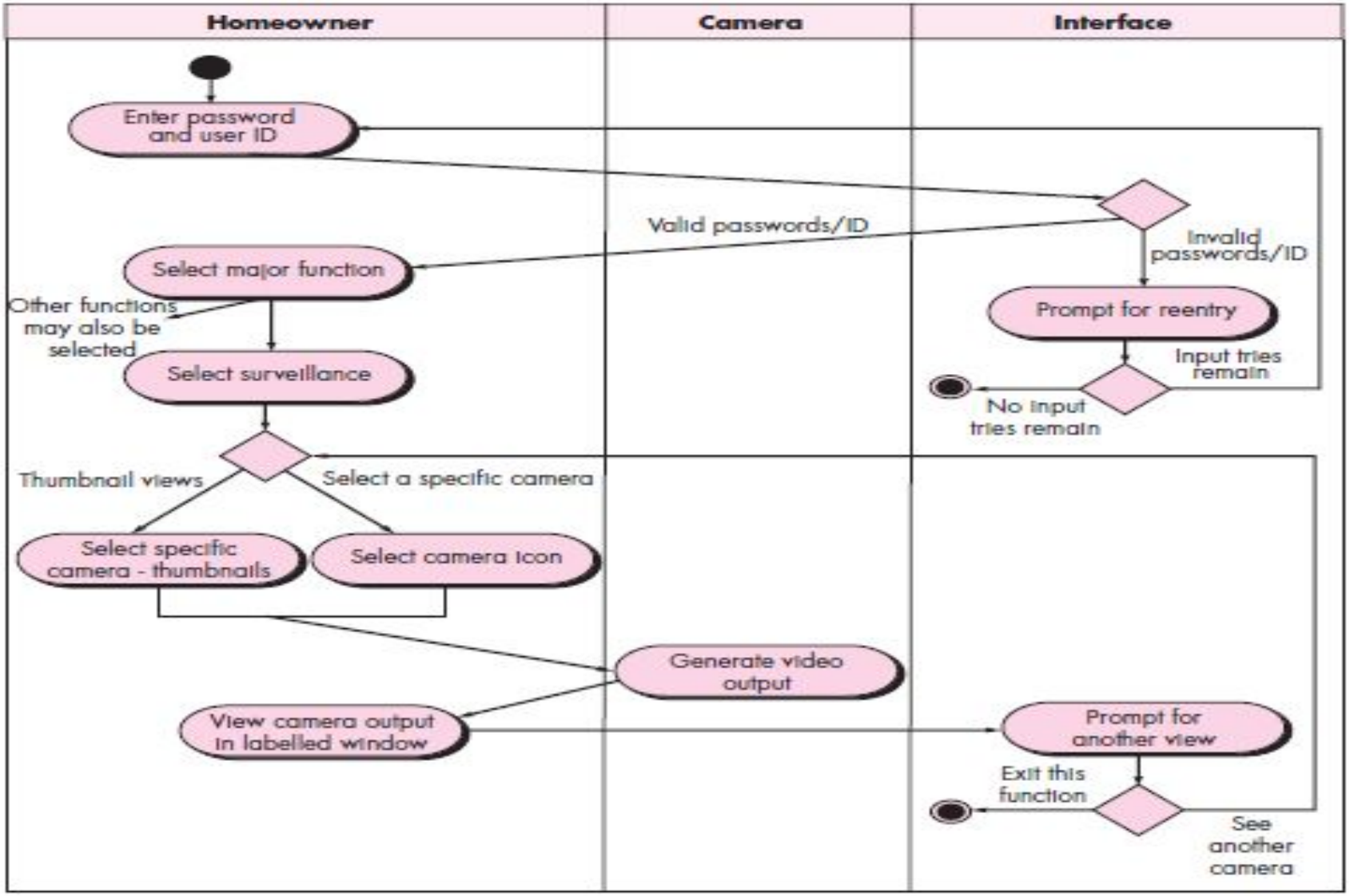

Enter password and user ID
Valid passwords/ID
Invalid passwords/ID
Select major function
Prompt for reentry
Other functions may also be selected
Select surveillance
Input tries remain
No input tries remain
Thumbnail views
Select a specific camera
Select specific camera - thumbnails
Select camera icon
View camera output in labeled window
Prompt for another view
Exit this function
See another camera

FIGURE 6.6 Swimlane diagram for Access camera surveillance via the Internet—display camera views function



Purpose of SRS document?

- SRS establishes basis of agreement between the user and the supplier.
  - Users needs have to be satisfied, but user may not understand software
  - Developers will develop the system, but may not know about problem domain
- SRS is
  - the medium to bridge the communications gap, and
  - specifies user needs in a manner both can understand

Need for SRS…

- Helps user understand his needs.
  - users do not always know their needs
  - must analyze and understand the potential
  - The requirement process helps clarify needs

- SRS provides a reference for validation of the final product
  - Clear understanding about what is expected.
  - Validation - “ SW satisfies the SRS “

Need for SRS…

- High quality SRS essential for high Quality SW
  - Requirement errors get manifested in final sw
  - To satisfy the quality objective, must begin with high quality SRS
  - Requirements defects cause later problems
    - 25% of all defects in one study; 54% of all defects found after user testing
    - defects often found in previously approved SRS.

Need for SRS…

- Good SRS reduces the development cost
  - SRS errors are expensive to fix later
  - Req. changes can cost a lot (up to 40%)
  - Good SRS can minimize changes and errors
  - Substantial savings; extra effort spent during req. saves multiple times that effort
- An Example
  - Cost of fixing errors in req. , design , coding , acceptance testing and operation are 2 , 5 , 15 , 50 , 150 person-months

Requirements Process

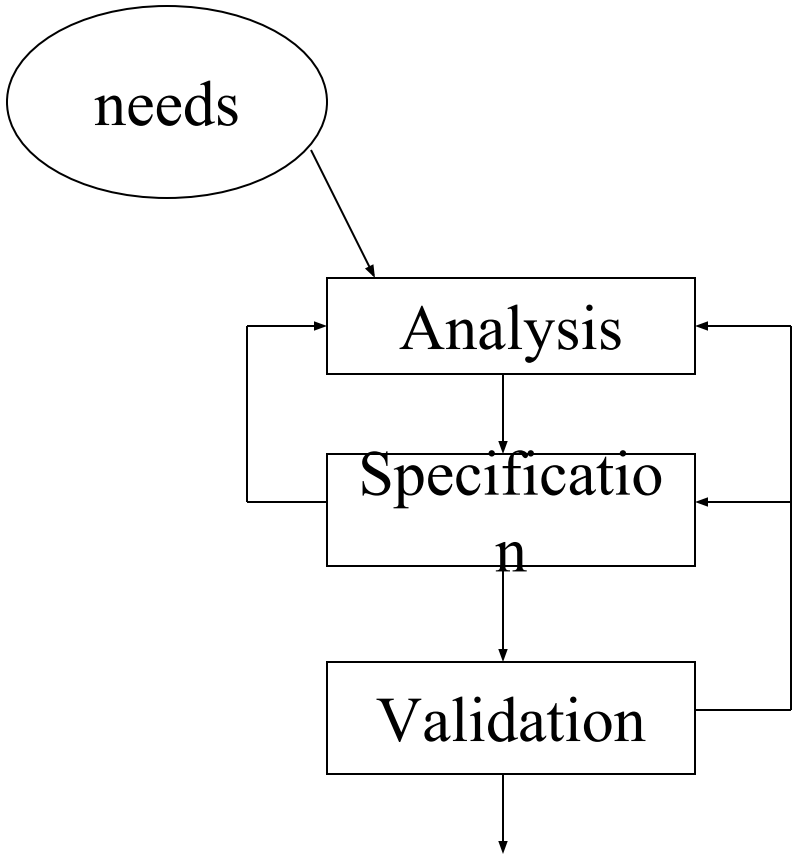
- Basic activities:
  - Problem or requirement analysis
  - Requirement specification
  - Validation

- Analysis involves elicitation and is the hardest

Need for SRS…

- Good SRS reduces the development cost
  - SRS errors are expensive to fix later
  - Req. changes can cost a lot (up to 40%)
  - Good SRS can minimize changes and errors
  - Substantial savings; extra effort spent during req. saves multiple times that effort
- An Example
  - Cost of fixing errors in req. , design , coding , acceptance testing and operation are 2 , 5 , 15 , 50 , 150 person-months

Requirement process..



- Process is not linear, it is iterative and parallel
- Overlap between phases - some parts may be analyzed and specified
- Specification itself may help analysis
- Validation can show gaps that can lead to further analysis and spec

INFO

Software Requirements Specification Template

A *software requirements specification* (SRS) is a document that is created when a detailed description of all aspects of the software to be built must be specified before the project is to commence. It is important to note that a formal SRS is not always written. In fact, there are many instances in which effort expended on an SRS might be better spent in other software engineering activities. However, when software is to be developed by a third party, when a lack of specification would create severe business issues, or when a system is extremely complex or business critical, an SRS may be justified.

Karl Wiegers [Wie03] of Process Impact Inc. has developed a worthwhile template (available at **www.processimpact.com/process_assets/srs_template.doc**) that can serve as a guideline for those who must create a complete SRS. A topic outline follows:

Table of Contents
Revision History

1. Introduction
1.1 Purpose
1.2 Document Conventions
1.3 Intended Audience and Reading Suggestions
1.4 Project Scope
1.5 References

2. Overall Description
2.1 Product Perspective
2.2 Product Features
2.3 User Classes and Characteristics
2.4 Operating Environment
2.5 Design and Implementation Constraints
2.6 User Documentation
2.7 Assumptions and Dependencies

3. System Features
3.1 System Feature 1
3.2 System Feature 2 (and so on)

4. External Interface Requirements
4.1 User Interfaces
4.2 Hardware Interfaces
4.3 Software Interfaces
4.4 Communications Interfaces

5. Other Nonfunctional Requirements
5.1 Performance Requirements
5.2 Safety Requirements
5.3 Security Requirements
5.4 Software Quality Attributes

6. Other Requirements

Appendix A: Glossary
Appendix B: Analysis Models
Appendix C: Issues List

A detailed description of each SRS topic can be obtained by downloading the SRS template at the URL noted earlier in this sidebar.

INFO

Requirements Validation Checklist

It is often useful to examine each requirement against a set of checklist questions. Here is a small subset of those that might be asked:

- Are requirements stated clearly? Can they be misinterpreted?
- Is the source (e.g., a person, a regulation, a document) of the requirement identified? Has the final statement of the requirement been examined by or against the original source?
- Is the requirement bounded in quantitative terms?
- What other requirements relate to this requirement? Are they clearly noted via a cross-reference matrix or other mechanism?
- Does the requirement violate any system domain constraints?
- Is the requirement testable? If so, can we specify tests (sometimes called validation criteria) to exercise the requirement?
- Is the requirement traceable to any system model that has been created?
- Is the requirement traceable to overall system/product objectives?
- Is the specification structured in a way that leads to easy understanding, easy reference, and easy translation into more technical work products?
- Has an index for the specification been created?
- Have requirements associated with performance, behavior, and operational characteristics been clearly stated? What requirements appear to be implicit?

Characteristics of an SRS

- Correct
- Complete
- Unambiguous
- Consistent
- Verifiable
- Traceable
- Modifiable
- Ranked for importance and/or stability

Characteristics…

- Correctness
  - Each requirement accurately represents some desired feature in the final system
- Completeness
  - All desired features/characteristics specified
  - Hardest to satisfy
  - Completeness and correctness strongly related
- Unambiguous
  - Each req has exactly one meaning
  - Without this errors will creep in
  - Important as natural languages often used

Characteristics…

- Verifiability
  - There must exist a cost effective way of checking if sw satisfies requirements
- Consistent
  - two requirements don't contradict each other
- Traceable
  - The origin of the req, and how the req relates to software elements can be determined
- Ranked for importance/stability
  - Needed for prioritizing in construction
  - To reduce risks due to changing requirements

the Feasibility Study

- What is a feasibility study?
  - What to study and conclude?
- Types of feasibility
  - Technical
  - Economic
  - Schedule
  - Operational
- Quantifying benefits and costs
  - Payback analysis
  - Net Present Value Analysis
  - Return on Investment Analysis
- Comparing alternatives

Exploring Feasibility

- The "PIECES" framework
  - Useful for identifying operational problems to be solved, and their urgency
  - Performance
    - Is current throughput and response time adequate?
  - Information
    - Do end users and managers get timely, pertinent, accurate and usefully formatted information?
  - Economy
    - Are services provided by the current system cost-effective?
    - Could there be a reduction in costs and/or an increase in benefits?
  - Control
    - Are there effective controls to protect against fraud and to guarantee information accuracy and security?
  - Efficiency
    - Does current system make good use of resources: people, time, flow of forms,…?
  - Services
    - Are current services reliable? Are they flexible and expandable?

Four Types of feasibility

Technical feasibility

- Is the project possible with current technology?
- What technical risk is there?
- Availability of the technology:
  - Is it available locally?
  - Can it be obtained?
  - Will it be compatible with other systems?

Economic feasibility

- Is the project possible, given resource constraints?
- What are the benefits?
  - Both tangible and intangible
  - Quantify them!
- What are the development and operational costs?
- Are the benefits worth the costs?

Schedule feasibility

- Is it possible to build a solution in time to be useful?
  - What are the consequences of delay?
  - Any constraints on the schedule?
  - Can these constraints be met?

Operational feasibility

- If the system is developed, will it be used?
- Human and social issues…
  - Potential labour objections?
  - Manager resistance?
  - Organizational conflicts and policies?
  - Social acceptability?
  - legal aspects and government regulations?

Economic Feasibility

- Can the bottom line be quantified yet?
  - Very early in the project…
    - a judgement of whether solving the problem is worthwhile.
  - Once specific requirements and solutions have been identified…
    - …the costs and benefits of each alternative can be calculated
- Cost-benefit analysis
  - Purpose - answer questions such as:
    - Is the project justified (I.e. will benefits outweigh costs)?
    - What is the minimal cost to attain a certain system?
    - How soon will the benefits accrue?
    - Which alternative offers the best return on investment?
  - Examples of things to consider:
    - Hardware/software selection
    - Selection among alternative financing arrangements (rent/lease/purchase)
  - Difficulties
    - benefits and costs can both be intangible, hidden and/or hard to estimate
    - ranking multi-criteria alternatives

Benefits Costs

- Tangible Benefits
  - Readily quantified as $ values
  - Examples:
    - increased sales
    - cost/error reductions
    - increased throughput/efficiency
    - increased margin on sales
    - more effective use of staff time
- Intangible benefits
  - Difficult to quantify
    - But maybe more important!
    - business analysts help estimate $ values
  - Examples:
    - increased flexibility of operation
    - higher quality products/services
    - better customer relations
    - improved staff morale
- How will the benefits accrue?
  - When - over what timescale?
  - Where in the organization?

- Development costs (OTO)
  - Development and purchasing costs:
    - Cost of development team
    - Consultant fees
    - software used (buy or build)?
    - hardware (what to buy, buy/lease)?
    - facilities (site, communications, power,...)
  - Installation and conversion costs:
    - installing the system,
    - training personnel,
    - file conversion,....
- Operational costs (on-going)
  - System Maintenance:
    - hardware (repairs, lease, supplies,...),
    - software (licenses and contracts),
    - facilities
  - Personnel:
    - For operation (data entry, backups,…)
    - For support (user support, hardware and software maintenance, supplies,…)
    - On-going training costs

Example: costs for small Client-Server project

Personnel:

2	System Analysts (400 hours/ea $35.00/hr)	$28,000
4	Programmer/Analysts (250 hours/ea $25.00/hr)	$25,000
1	GUI Designer (200 hours/ea $35.00/hr)	$7,000
1	Telecommunications Specialist (50 hours/ea $45.00/hr)	$2,250
1	System Architect (100 hours/ea $45.00/hr)	$4,500
1	Database Specialist (15 hours/ea $40.00/hr)	$600
1	System Librarian (250 hours/ea $10.00/hr)	$2,500

Expenses:

4	Smalltalk training registration ($3500.00/student)	$14,000

New Hardware & Software:

1	Development Server (Pentium Pro class)	$18,700
1	Server Software (operating system, misc.)	$1,500
1	DBMS server software	$7,500
7	DBMS Client software ($950.00 per client)	$6,650

Total Development Costs:	**$118,200**

PROJECTED ANNUAL OPERATING COSTS

Personnel:

2	Programmer/Analysts (125 hours/ea $25.00/hr)	$6,250
1	System Librarian (20 hours/ea $10.00/hr)	$200

Expenses:

1	Maintenance Agreement for Pentium Pro Server	$995
1	Maintenance Agreement for Server DBMS software	$525
	Preprinted forms (15,000/year @ .22/form)	$3,300

Total Projected Annual Costs:	**$11,270**

Analyzing Costs vs. Benefits

- **Identify costs and benefits**
  - Tangible and intangible, one-time and recurring
  - Assign values to costs and benefits
- **Determine Cash Flow**
  - Project the costs and benefits over time, e.g. 3-5 years
  - Calculate Net Present Value for all future costs/benefits
    - determines future costs/benefits of the project in terms of today's dollar values
    - A dollar earned today is worth more than a potential dollar earned next year
- **Do cost/benefit analysis**
  - Calculate Return on Investment:
    - Allows comparison of lifetime profitability of alternative solutions.

$$ROI = \frac{\text{Total Profit}}{\text{Total Cost}} = \frac{\text{Lifetime benefits - Lifetime costs}}{\text{Lifetime costs}}$$

  - Calculate Break-Even point:
    - how long will it take (in years) to pay back the accrued costs:

@T(Accrued Benefit > Accrued Cost)

Calculating Present Value

- A dollar today is worth more than a dollar tomorrow…
  - Your analysis should be normalized to "current year" dollar values.
- The discount rate
  - measures opportunity cost:
    - Money invested in this project means money not available for other things
    - Benefits expected in future years are more prone to risk
  - This number is company- and industry-specific.
    - "what is the average annual return for investments in this industry?"
- Present Value:
  - The "current year" dollar value for costs/benefits n years into the future
    - … for a given discount rate i

$$\text{Present\_Value}(n) = \frac{1}{(1+i)^n}$$

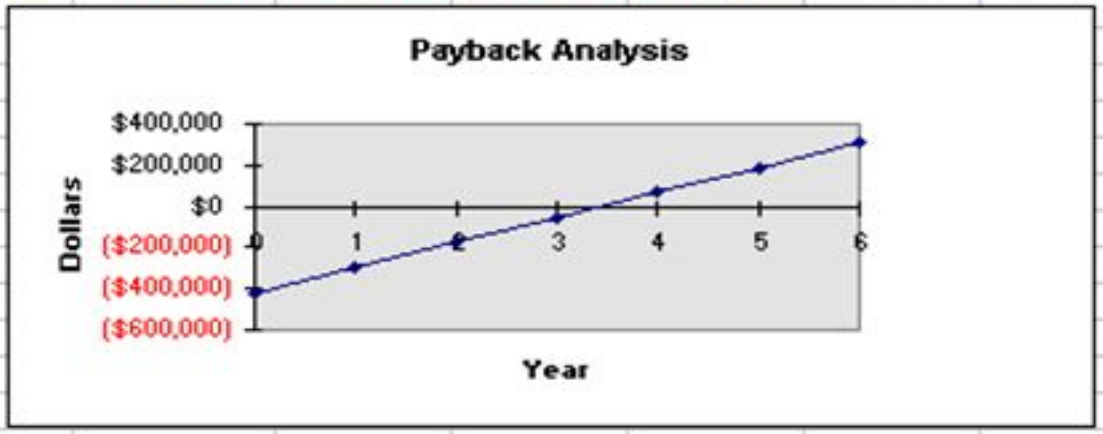
  - E.g. if the discount rate is 12%, then
    - $\text{Present\_Value}(1) = 1/(1 + 0.12)^1 = 0.893$
    - $\text{Present\_Value}(2) = 1/(1 + 0.12)^2 = 0.797$

© 2004-5 Steve Easterbrook. This presentation is available free for non-commercial use with attribution under a creative commons license.

Payback Analysis for Client-Server System Alternative

(Numbers rounded to nearest $1)

Cash flow description	Year 0	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6
Development cost:	($418,040)						
Operation & maintenance cost:		($15,045)	($16,000)	($17,000)	($18,000)	($19,000)	($20,000)
Discount factors for 12%:	1.000	0.893	0.797	0.712	0.636	0.567	0.507
Time-adjusted costs (adjusted to present	($418,040)	($13,435)	($12,752)	($12,104)	($11,448)	($10,773)	($10,140)
Cumulative time-adjusted costs over	($418,040)	($431,475)	($444,227)	($456,331)	($467,779)	($478,552)	($488,692)
Benefits derived from operation of new	$0	$150,000	$170,000	$190,000	$210,000	$230,000	$250,000
Discount factors for 12%:	1.000	$0.89	$0.80	$0.71	$0.64	$0.57	$0.51
Time-adjusted benefits (current of present	$0	$133,950	$135,490	$135,280	$133,560	$130,410	$126,750
Cumulative time-adjusted benefits over	$0	$133,950	$269,440	$404,720	$538,280	$668,690	$795,440
	0	1	2	3	4	5	6
Cumulative lifetime time-adjusted costs +	($418,040)	($297,525)	($174,787)	($51,611)	$70,501	$190,138	$306,748



© 2004-5 Steve Easterbrook. This presentation is available free for non-commercial use with attribution under a creative commons license.

Schedule Feasibility

- How long will it take to get the technical expertise?
  - We may have the technology, but that doesn't mean we have the skills required to properly apply that technology.
    - May need to hire new people
    - Or re-train existing systems staff
    - Whether hiring or training, it will impact the schedule.
- Assess the schedule risk:
  - Given our technical expertise, are the project deadlines reasonable?
  - If there are specific deadlines, are they mandatory or desirable?
    - If the deadlines are not mandatory, the analyst can propose several alternative schedules.
- What are the real constraints on project deadlines?
  - If the project overruns, what are the consequences?
    - Deliver a properly functioning information system two months late…
    - …or deliver an error-prone, useless information system on time?
  - Missed schedules are bad, but inadequate systems are worse!

- Software Requirements Specification document with example - Krazytech