



# Software Requirements Analysis and Specification

---



## Background..

---

- Requirements understanding is hard
  - Visualizing a future system is difficult
  - Capability of the future system not clear, hence needs not clear
  - Requirements change with time
  - ...
- Essential to do a proper analysis and specification of requirements



## Background..

---

- Requirements understanding is hard
  - Visualizing a future system is difficult
  - Capability of the future system not clear, hence needs not clear
  - Requirements change with time
  - ...
- Essential to do a proper analysis and specification of requirements



## Background..

---

- Requirements understanding is hard
  - Visualizing a future system is difficult
  - Capability of the future system not clear, hence needs not clear
  - Requirements change with time
  - ...
- Essential to do a proper analysis and specification of requirements



## Need for SRS...

---

### Example ...

- After req. phase 65% req errs detected in design , 2% in coding, 30% in Acceptance testing, 3% during operation
- If 50 requirement errors are not removed in the req. phase, the total cost  
 $32.5 * 5 + 1 * 15 + 15 * 50 + 1.5 * 150 = 1152$  hrs
- If 100 person-hours invested additionally in req to catch these 50 defects , then development cost could be reduced by 1152 person-hours.
- Net reduction in cost is 1052 person-hours.



## Need for SRS...

---

### Example ...

- After req. phase 65% req errs detected in design , 2% in coding, 30% in Acceptance testing, 3% during operation
- If 50 requirement errors are not removed in the req. phase, the total cost  
 $32.5 * 5 + 1 * 15 + 15 * 50 + 1.5 * 150 = 1152$  hrs
- If 100 person-hours invested additionally in req to catch these 50 defects , then development cost could be reduced by 1152 person-hours.
- Net reduction in cost is 1052 person-hours.





## Need for SRS...

---

### Example ...

- After req. phase 65% req errs detected in design , 2% in coding, 30% in Acceptance testing, 3% during operation
- If 50 requirement errors are not removed in the req. phase, the total cost  
 $32.5 * 5 + 1 * 15 + 15 * 50 + 1.5 * 150 = 1152$  hrs
- If 100 person-hours invested additionally in req to catch these 50 defects , then development cost could be reduced by 1152 person-hours.
- Net reduction in cost is 1052 person-hours.



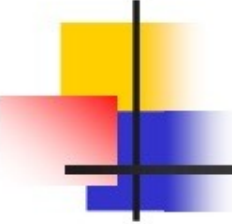
## Need for SRS...

---

### Example ...

- After req. phase 65% req errs detected in design , 2% in coding, 30% in Acceptance testing, 3% during operation
- If 50 requirement errors are not removed in the req. phase, the total cost  
 $32.5 * 5 + 1 * 15 + 15 * 50 + 1.5 * 150 = 1152$  hrs
- If 100 person-hours invested additionally in req to catch these 50 defects , then development cost could be reduced by 1152 person-hours.
- Net reduction in cost is 1052 person-hours.





---

Phase	Cost (person-hours)
Requirements	2
Design	5
Coding	15
Acceptance test	50
Operation and maint.	150

. Cost of fixing requirement errors

- Assume that there are 50 requirement errors, not removed,
- the requirements errors that remain after the requirements phase, are detected as follows:  
about 65% are detected during design, 2% during coding, 30% during testing,  
and 3% during operation and maintenance.
- calculate The total cost of fixing the errors in the later phases

Phase	Cost(p-h) of fixing error	Percenta ge of error detected	No. of errors	Cost per phase
Design				
Coding				
Testing				
Operation				

Total Cost=                      person-hours

Phase	Cost(p-h) of fixing error	Percentage of error detected	No. of errors	Cost per phase
Design	5	65%	$50 * 65\% = 32.5$	$32.5 * 5$
Coding	15	2%	$50 * 2\% = 1$	$1 * 15$
Testing	50	30%	$50 * 30\% = 15$	$15 * 50$
Operation	150	3%	$50 * 3\% = 1.5$	$1.5 * 150$

Total Cost=  $32.5 * 5 + 1 * 15 + 15 * 50 + 1.5 * 150 = 1152$  person-hours



## Need for SRS...

---

### Example ...

- After req. phase 65% req errs detected in design , 2% in coding, 30% in Acceptance testing, 3% during operation
- If 50 requirement errors are not removed in the req. phase, the total cost  
 $32.5 * 5 + 1 * 15 + 15 * 50 + 1.5 * 150 = 1152$  hrs
- If 100 person-hours invested additionally in req to catch these 50 defects , then development cost could be reduced by 1152 person-hours.
- Net reduction in cost is 1052 person-hours.



# Requirements Process..

- Transition from analysis to specs is hard
  - in specs, external behavior specified
  - during analysis, structure and domain are understood
  - analysis structures helps in specification, but the transition is not final
  - methods of analysis are similar to that of design, but objective and scope different
  - analysis deals with the problem domain, whereas design deals with solution domain





# Requirements Process..

Transition from analysis to specs is hard

- in specs, external behavior specified
- during analysis, structure and domain are understood
- analysis structures helps in specification, but the transition is not final
- methods of analysis are similar to that of design, but objective and scope different
- analysis deals with the problem domain, whereas design deals with solution domain





# 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



# Requirements Process..

- Transition from analysis to specs is hard
  - in specs, external behavior specified
  - during analysis, structure and domain are understood
  - analysis structures helps in specification, but the transition is not final
  - methods of analysis are similar to that of design, but objective and scope different
  - analysis deals with the problem domain, whereas design deals with solution domain



# Requirements Process..

- Transition from analysis to specs is hard
  - in specs, external behavior specified
  - during analysis, structure and domain are understood
  - analysis structures helps in specification, but the transition is not final
  - methods of analysis are similar to that of design, but objective and scope different
  - analysis deals with the problem domain, whereas design deals with solution domain



# Problem Analysis...

---

- Interpersonal issues are important
- Communication skills are very important
- Basic principle: problem partition
- Projection - get different views
- Will discuss few different analysis techniques



# Problem Analysis...

---

- Interpersonal issues are important
- Communication skills are very important
- Basic principle: problem partition
- Projection - get different views
- Will discuss few different analysis techniques



# Problem Analysis...

---

- Interpersonal issues are important
- Communication skills are very important
- Basic principle: problem partition
- Projection - get different views
- Will discuss few different analysis techniques





# Characteristics of an SRS

---

- What should be the characteristics of a good SRS? Some key ones are
  - Correct.
  - Complete.
  - Unambiguous.
  - Consistent.
  - Verifiable.
  - Ranked for importance and/or stability.



# Characteristics...

---

- Ranked for importance/stability
  - Needed for prioritizing in construction
  - To reduce risks due to changing requirements
  - Stability of a requirement reflects the chances of it changing in the future.



# Characteristics...

---


- Ranked for importance/stability
  - Needed for prioritizing in construction
  - To reduce risks due to changing requirements
  - Stability of a requirement reflects the chances of it changing in the future.



# Characteristics...

---

- Ranked for importance/stability
  - Needed for prioritizing in construction
  - To reduce risks due to changing requirements
  - Stability of a requirement reflects the chances of it changing in the future.

- 
- 
- Completeness, is the most important and difficult property to establish:
    - Missing requirements necessitate additions and modifications to the requirements later in the development cycle.
      - Will increase cost.
    - Incompleteness is also a major source of disagreement between the client and the supplier.



# Components of an SRS

---

- What should an SRS contain ?
  - Having guidelines helps Clarifying this will help ensure completeness
- An SRS must specify requirements on
  - Functionality
  - Performance
  - Design constraints
  - External interfaces





# Functional Requirements

---

- Heart of the SRS document; this forms the bulk of the specs.
- Specifies all the functionality that the system should support (expected behavior of the system).
- Outputs for the given inputs and the relationship between them.
- All operations the system is to do.
- Must specify behavior for invalid inputs :
  - range of valid inputs must be specified.



# Performance Requirements

---

- All the performance constraints on the software system
- Two types of performance requirements: static and dynamic.
- Static: do not impose constraint on the execution.
  - number of terminals, number of simultaneous users, number of files..
  - called capacity requirements.
- Dynamic: specify constraints on the execution behavior.
  - Generally on response time , throughput etc.
  - Must be in measurable terms (verifiability):
  - "response time should be good" not acceptable,
  - response time for x should be less than one second 90% of the time.



# Design Constraints

---

- Factors in the client environment that restrict the choices
- standards that must be followed, resource limits, operating environment, reliability, security requirements, and policies
- Some such restrictions
  - **Standard compliance** and compatibility with other systems: (report format, accounting procedures)
  - **Hardware Limitations:** (the type of machines, operating system, languages supported, limits on primary/secondary storage.)
  - **Reliability,** fault tolerance, backup req. they make the system more complex and expensive.
  - **Security** place restrictions on the use of certain commands, control access...



# External Interface

---

- All interactions of the software with people, hardware, and sw
- User interface most important.
- A preliminary user manual should be created:(all user commands, ...)
- General requirements of: "the system should be user friendly" should be avoided and use : "commands should be no longer than six characters"
- These should also be verifiable





# Specification Language

---

- Language should support desired char of the SRS
- Formal languages are precise and unambiguous but hard
- Natural languages mostly used, with some structure for the document
- Formal languages used for special features or in highly critical systems



## Structure of an SRS...

---

- Specific requirements
  - External interfaces
  - Functional requirements
  - Performance requirements
  - Design constraints
- Acceptable criteria
  - desirable to specify this up front.
- This standardization of the SRS was done by IEEE.





## Structure of an SRS...

---

- Specific requirements
  - External interfaces
  - Functional requirements
  - Performance requirements
  - Design constraints
- Acceptable criteria
  - desirable to specify this up front.
- This standardization of the SRS was done by IEEE.



# Other Approaches to Analysis

---

# The Data Flow Model

**Every computer-based system is an information transform ....**





# Data Flow Modeling

---

- Widely used; focuses on functions performed in the system
- Views a system as a network of data transforms through which the data flows
- Uses data flow diagrams (DFDs) and functional decomposition in modeling
- The SSAD methodology uses DFD to organize information, and guide analysis



# Data flow diagrams...

---

- Focus on what transforms happen , how they are done is **not important**
- Usually major inputs/outputs shown, minor are ignored in this modeling
- **No loops , conditional thinking , ...**
- DFD is NOT a control chart(flow chart), no algorithmic design/thinking
- Sink/Source , external files



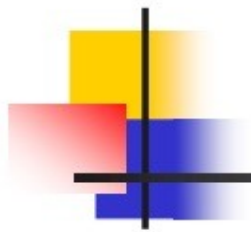
# Data flow diagrams...

---

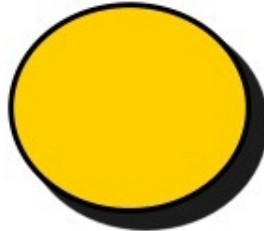
- Focus on what transforms happen , how they are done is **not important**
- Usually major inputs/outputs shown, minor are ignored in this modeling
- **No loops , conditional thinking , ...**
- DFD is NOT a control chart(flow chart), no algorithmic design/thinking
- Sink/Source , external files



# Data Flow Modeling Notation



source or sink



**process**

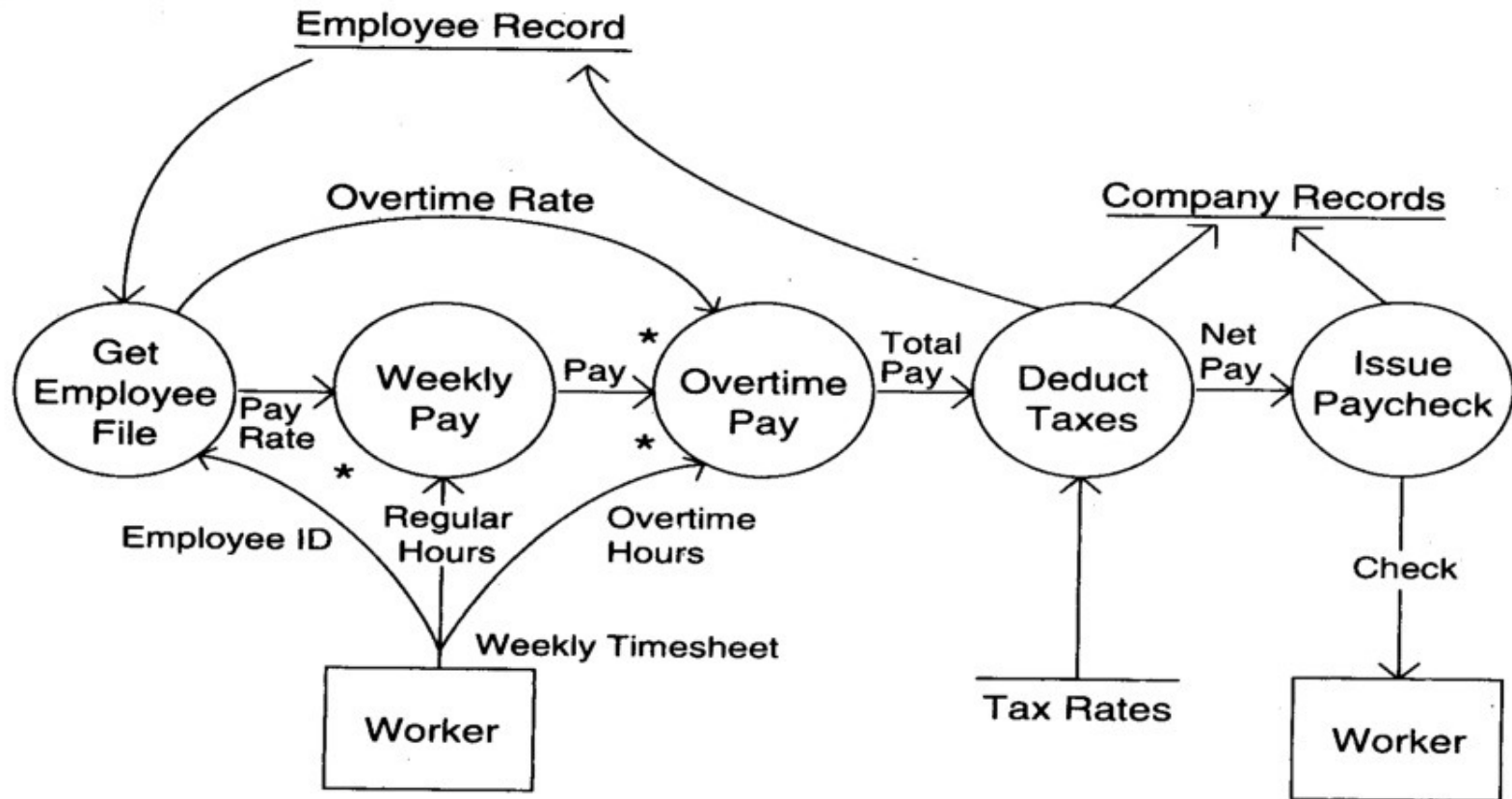


**data flow**



**data store**

# DFD Example





# DFD Conventions

---

- External files shown as labeled straight lines
- Need for multiple data flows by a process represented by \* (means and)
- OR relationship represented by +
- All processes and arrows should be named
- Processes should represent transforms, arrows should represent some data



# Data flow diagrams...

---

- Focus on what transforms happen , how they are done is **not important**
- Usually major inputs/outputs shown, minor are ignored in this modeling
- **No loops , conditional thinking , ...**
- DFD is NOT a control chart(flow chart), no algorithmic design/thinking
- Sink/Source , external files



# Drawing a DFD

---

- If get stuck , reverse direction
- If control logic comes in , stop and restart
- Label each arrows and bubbles
- Make use of + & \*
- Try drawing alternate DFDs

## **Leveled DFDs :**

- DFD of a system may be very large
- Can organize it hierarchically
- Start with a top level DFD with a few bubbles
- then draw DFD for each bubble
- Preserve I/O when "exploding".



# Drawing a DFD for a system..

---

- Never show control logic; if thinking in terms of loops/decisions, stop & restart
- Label each arrows and bubbles; carefully identify inputs and outputs of each transform
- Make use of + & \*
- Try drawing alternate DFDs





# Drawing a DFD for a system..

---

- Never show control logic; if thinking in terms of loops/decisions, stop & restart
- Label each arrows and bubbles; carefully identify inputs and outputs of each transform
- Make use of + & \*
- Try drawing alternate DFDs



# Leveled DFDs

---

- DFD of a system may be very large
- Can organize it hierarchically
- Start with a top level DFD with a few bubbles
- then draw DFD for each bubble
- Preserve I/O when “exploding” a bubble so consistency preserved
- Makes drawing the leveled DFD a top-down refinement process, and allows modeling of large and complex systems



# DFD drawing – common errors

---

- Unlabeled data flows
- Missing data flows
- Extraneous data flows
- Consistency not maintained during refinement
- Missing processes
- Too detailed or too abstract
- Contains some control information
- Black Hole.
- Miracle.



# Requirements Validation

---

- The basic objective of the requirements validation activity is to ensure that the SRS reflects the actual requirements accurately and clearly. and to check that SRS document is of good quality.
- Lot of room for misunderstanding
- Errors possible
- Expensive to fix req. defects later
- Must try to remove most errors in SRS



# common errors

---

- Most common errors
  - Omission - 30%
  - Inconsistency - 10-30%
  - Incorrect fact - 10-30%
  - Ambiguity - 5 -20%
  - clerical errors



# Omission

---

- some user requirement is simply not included in the SRS;
- omitted requirement may be related to the behavior of the system, its performance, constraints, or any other factor.
- Omission directly affects the external completeness of the SRS.





# Inconsistency

---

- contradictions within the requirements themselves
- incompatibility of the stated requirements with the actual requirements of the client
- incompatibility with the environment in which the system will operate.



## Incorrect fact

---

- some fact recorded in the SRS is not correct.



# Ambiguity

---

- when there are some requirements that have multiple meanings, that is, their interpretation is not unique.

# Validation Method: Requirements Review



---

- requirements are generally textual documents that cannot be executed.
- Validation Must include client and a user.
- SRS reviewed by a group of people
- Group:
  - author of the requirements document,
  - someone who understands the needs of the client
  - a person of the design team,
  - person(s) responsible for maintaining the requirements document.



# Review process

---

- Process – standard inspection process
- Goal:
  - reveal any errors in the requirements,
  - consider factors affecting quality, such as testability and readability.
- Process Effectiveness - can catch 40-80% of req. errors
- requirements reviews are needed even if the requirements are specified through a tool or are in a formal notation.



# Summary..

---

- Specification
  - must contain functionality , performance , interfaces and design constraints
  - Mostly natural languages used
- Use Cases is a method to specify the functionality; also useful for analysis
- Validation - through reviews





# Summary..

---

- Specification
  - must contain functionality , performance , interfaces and design constraints
  - Mostly natural languages used
- Use Cases is a method to specify the functionality; also useful for analysis
- Validation - through reviews