

Approaching a Problem

Where do we start?
How do we proceed?

08a-Requirements1 1

Where Do We Start?

- Start with the requirements
 - Capture your goals and possible constraints
 - Environmental assumptions
- Use-case analysis to better understand your requirements
 - Find actors and a first round of use-cases
- Start conceptual modeling
 - Conceptual class diagram
 - Interaction diagrams to clarify use-cases
 - Activity diagrams to understand major processing

08a-Requirements1 2

How Do We Continue?

- Refine use-cases
 - Possibly some "real" use-cases
 - ◆ Using interface mockups
- Refine (or restructure) your class diagram
 - Based on your hardware architecture
 - ◆ For instance, client server
- Refine and expand your dynamic model
 - Until you are comfortable that you understand the required behavior
- Identify most operations and attributes

08a-Requirements1 3

How Do We Wrap Up?

- Refine the class diagram based on platform and language properties
 - Navigability, public, private, etc
 - Class libraries
- Identify most operations
 - Not the trivial get, set, etc.
- Write a contract for each operation
- Define a collection of invariants for each class
- Implement

08a-Requirements1 4

Putting It Together

- **Principles**
 - Rigor and Formality
 - Separation of Concerns
 - Modularity
 - Abstraction
 - Anticipation of Change
 - Generality
 - Incrementality
- **Notion of Process**

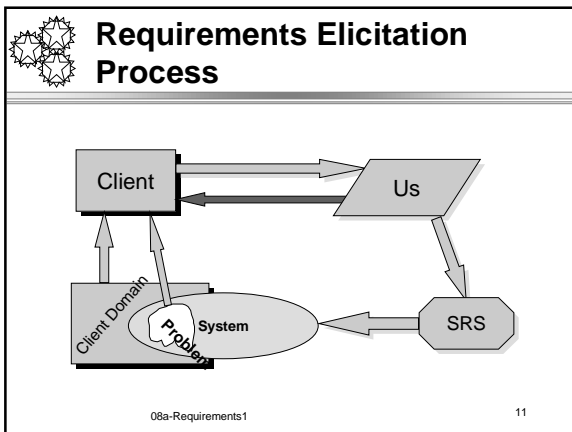
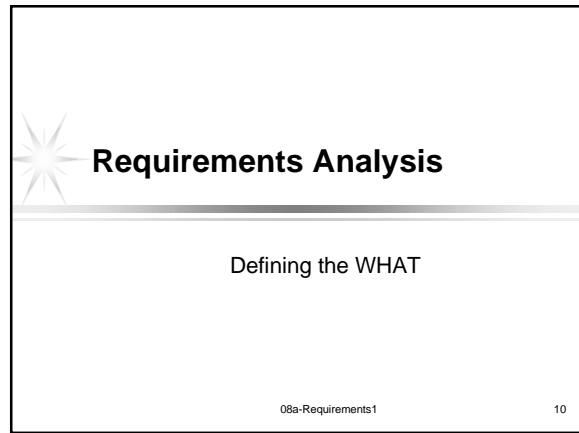
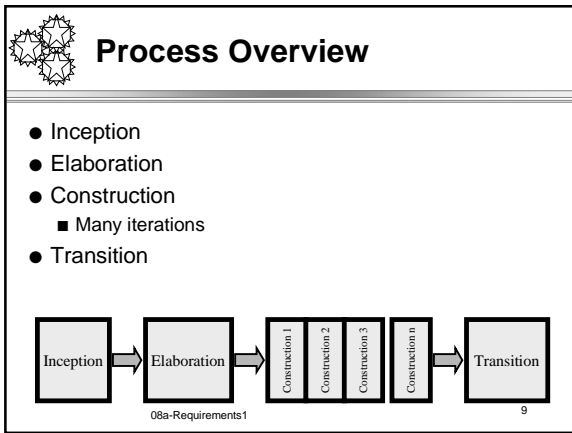
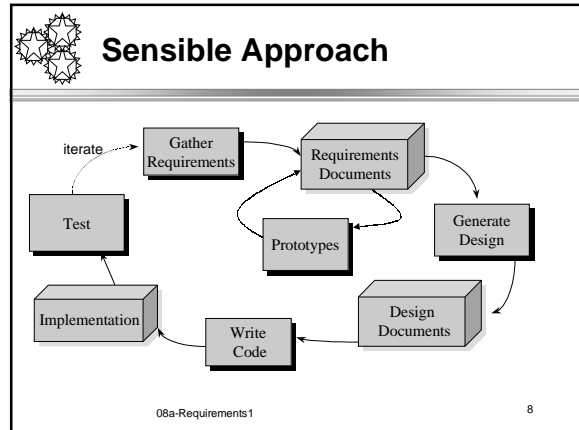
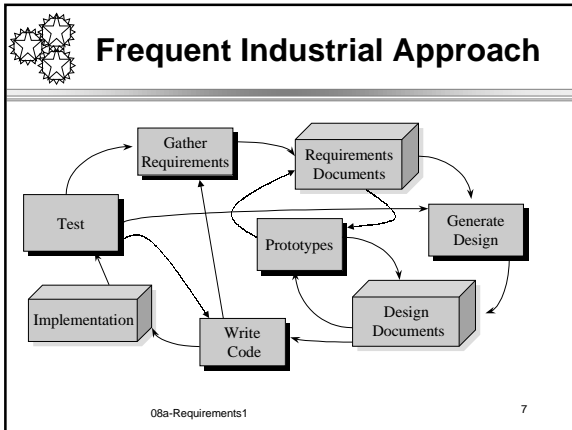
08a-Requirements1 5

Waterfall Approach

```

    graph LR
      GR[Gather Requirements] --> RD[Requirements Documents]
      RD --> GD[Generate Design]
      GD --> ID[Implementation]
      ID --> T[Test]
      DL[Domain language] --> RD
      MTL[Models, technical language] --> DD[Design Documents]
      DD --> ID
  
```

08a-Requirements1 6



- ### Requirements
- Specify functionality
 - model objects and resources
 - model behavior
 - Specify data interfaces
 - type, quantity, frequency, reliability
 - providers, receivers
 - operational profile (expected scenarios)
 - stress profile (worst case scenarios)
- 08a-Requirements1 12



Requirements

- Specify interfaces
 - Control interfaces (APIs)
 - User interfaces - functionality and style
 - Hardware interfaces
- Specify error handling
- Identify potential modifications

08a-Requirements1

13



Requirements

- Identify necessary constraints
 - performance, security, reliability
- Identify areas of risk
 - alternatives to be explored
- Specify validation plans
- Specify documentation to be provided

08a-Requirements1

14



Analysis Principles

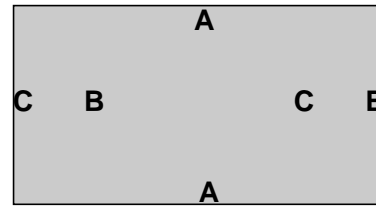
- Document reason for specific requirements
- Prioritize requirements
 - High, medium, low
- Ignore implementation details
 - Need to know feasible solutions can be developed
 - If feasibility is a concern, then propose alternatives to be explored
- Be prepared to change

08a-Requirements1

15



Perspective and Early Binding of Constraints



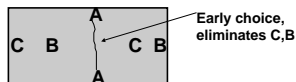
Connect like letters without crossing lines or leaving box.

08a-Requirements1

16



Early Focus on Constraints



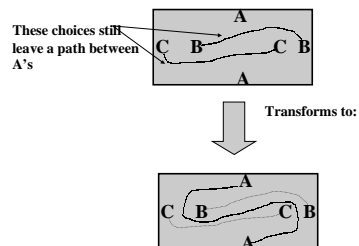
- A-A line seems to be only solution
 - But, is it really?
 - Need to examine domain and constraints more?
- What transforms or shifts would yield easier problem?

08a-Requirements1

17



Focus Change



08a-Requirements1

18



Reviewing a requirements document

- Is it ambiguous?
 - Carefully define terms and use these terms
- Is it consistent?
- Is it complete?
 - Vague requirements
 - Omitted requirements
- Is it verifiable?
- Is it realistic?
- Does it plan for change?
- Does it not overly constrain the problem?
- Have alternatives been considered and explored?
- Is it clearly presented?
 - Precise, concise, clear
 - diagram complex objects and behaviors
- Is it what the customer wants?

08a-Requirements1

19



Why is requirements analysis difficult?

- Communication: misunderstandings between the customer and the analyst
 - Analyst doesn't understand the domain
 - Customer doesn't understand alternatives and trade-offs
- Problem complexity
 - Inconsistencies in problem statement
 - Omissions/incompleteness in problem statement
 - Inappropriate detail in problem statement

08a-Requirements1

20



Escalator System Requirements

Two Signs on Escalator:



08a-Requirements1

21



Why is requirements analysis difficult?

- Need to accommodate change
 - Hard to predict change
 - Hard to plan for change
 - Hard to foresee the impact of change

08a-Requirements1

22



First Law of Software Engineering

“No matter where you are in the system lifecycle, the system will change, and the desire to change it will persist throughout the lifecycle.”

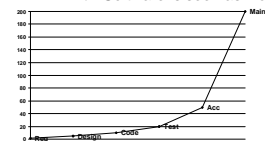
08a-Requirements1

23



Reasons for changing requirements

- Poor communication
- Inaccurate requirements analysis
- Failure to consider alternatives
- New users
- New customer goals
- New customer environment
- New technology
- Competition
- Software is seen as malleable



Changes made after the requirements are approved increase cost and schedule

08a-Requirements1

24



Requirements Products

- Specification document
 - Agreement between customer and developer
 - Validation criteria for software
 - Problem statement in domain language
 - ◆ *external* behavior
 - ◆ constraints on system
- Preliminary users manual
- Prototype (*do not deliver the prototype!*)
 - If user interaction is important
 - If resources are available
- Review by customer and developer
 - Iteration is almost always required

08a-Requirements1

25



Overview: Steps to Follow

- Map out environment as-is
- Map out environment as required
- Decide on systems boundaries / goals
- List actions with types
- Define terms
- Construct model
- Challenge model
- Modify as required

08a-Requirements1

26



Analysis: Steps to follow

- Obtain a problem statement
- Develop use cases (depict scenarios of use)
- Build an object model and data dictionary
- Develop a dynamic model
 - state and sequence diagrams
- Verify, iterate, and refine the models
- Produce analysis document

08a-Requirements1

27



Use Cases

- High-level overview of system use
- Identify scenarios of usage
- Identify actors of the system:
 - External entities (e.g., users, systems, etc.)
- Identify system activities
- Draw connections between actors and activities
- Identify dependencies between activities (i.e., extends, uses)

08a-Requirements1

28



Analysis: Object Model

- Organization of system into classes connected by associations
 - Shows the static structure
 - Organizes and decomposes system into more manageable subsystems
 - Describes real world classes and relationships

08a-Requirements1

29



Analysis: Object Model

- Object model precedes the dynamic model because
 - static structure is usually better defined
 - less dependent on details
 - more stable as the system evolves

08a-Requirements1

30



Analysis: Object Model

- Information comes from
 - The problem statement and use cases
 - Expert knowledge of the application domain
 - ◆ Interviews with customer
 - ◆ Consultation with experts
 - ◆ Outside research performed by analyst
 - General knowledge of the real world

08a-Requirements1

31



Client View of Domain

- Clients can't be expected to have rigorous or formal view of domain
- Hence, can't be expected to be completely aware of domain-problem relationship
- Some knowledge is *explicit*
 - Easier to get at
- Some knowledge is *implicit* ("everybody knows...")
 - Many constraints are implicit
 - Hard to get at

08a-Requirements1

32



Object Model: Steps to follow

- Identify classes and associations
 - nouns and verbs in a problem description
- Create data dictionary entry for each
- Add attributes
- Combine and organize classes using inheritance

08a-Requirements1

33



Analysis: Dynamic model

- Shows the time dependent behavior of the system and the objects in it
- Expressed in terms of
 - states of objects and activities in states
 - events and actions
- State diagram summarizes permissible event sequences for objects with important dynamic behavior

08a-Requirements1

34



Dynamic Model: Steps to follow

- Use cases provide scenarios of typical interaction sequences
- Identify events between objects (Sequence Diagram)
- Prepare an event trace for each scenario
- Build state diagrams
- **Match events between objects to verify consistency**



08a-Requirements1

35



Analysis: Iteration

- Analysis model will require multiple passes to complete
- Look for inconsistencies and revise
- Look for omissions/vagueness and revise
- Validate the final model with the customer
 - Pose scenarios from various perspectives
 - ◆ Look for consistency in responses

08a-Requirements1

36



Object Model: Four main system objects or classes

- Controller object
 - might be made up of several controllers
 - is the brains of the system.
 - Takes input from the sensors and gives instructions to the actuators.
- Sensor object
 - environmental objects that gives information to controller.
 - Can be passive (thermometer) or active (button).

08a-Requirements1

37



Meeting Purposes

- Disseminate information (including stating a problem)
- Gathering opinions
- Confirming consensus
- Social requirements
 - team building
 - approval

08a-Requirements1

38



Meeting Requirements

- Agenda
- Leader
- Action list
 - With assignments so we know who is doing what.
 - Timelines so we know when it's to get done.
- Summary
 - Something happened or there would not have been a meeting. Record it briefly.

08a-Requirements1

39



Project Issue List

- Every issue goes on the list
 - Date and brief description
- Make assignment to get it resolved
- Put resolution on list.
 - "Close" issue.
- 1st version usually generated on 1st read of problem statement.
 - And then, back to the customer...

08a-Requirements1

40



Interviewing

- Have a list of things you want to know.
- Listen.
- Listen.
- Ask open-ended questions.
- Don't express (show, say) opinions on answers. Just record, and think.
- Listen.
- Ask questions more than one way.

08a-Requirements1

41



Close-ended questions

Q: When a vehicle cuts in front of the car, you have to slow down quickly and not hit it, right?

A: Yes.



You learned absolutely nothing.

08a-Requirements1

42



Open-ended questions

Q: Tell me what happens when a car cuts in front of you.

A: Well, if the lead car is too close, the driver has to intervene or else a crash results. I guess we need a warning light in this case. If the car is moving faster, you don't have to do anything. He's pulling away. I guess the only time brakes are used is when the closing speed is too high for the distance and yet within the capabilities of the system to slow down. But I guess if a collision is imminent, we should max out the braking...



Now, we learned something...

Ah ha!, new requirement!

And a clarification

08a-Requirements1

43



Responses

Q: Tell me what should happen if a car cuts in front of our car too close to avoid a collision?

Much better

A: I guess since there is nothing the system can do. Turn off the controller and hope the driver brakes in time.

Not good

Q: What? Are you nuts? We should at least try to stop. Shouldn't we?

Q: We have quite a bit of braking power in the system. What would happen if we used it here?

A: Perhaps...

A: Well, I guess it could avoid a collision and at least get the car slowed down but the attorneys tell me we don't want the system active when a collision occurs.

You are done at at this point, and still unresolved.

Ah ha! Non-technical constraint

08a-Requirements1

44