Requirements Engineering



CSC 326

© NC State Software Engineering Faculty

1

Upcoming Lectures/Labs/Deadlines

Date	Thing	Notes
M Sep 21	*Lecture*	*Today – Architecture + Requirements Engineering*
W Sep 23	Lecture	Project Workday – no class meeting – work on Milestone 4!
W Sep 23	OBP M4 T	M4 Team Deadline @ 11:45 pm
H Sep 24	Lab	Grade M4 Team & Start M5 (Frontend)
M Sep 28	OBP M4 I	M4 Individual Deadline @ 5:00 pm + Retro @ 11:45 pm
M Sep 28	Lecture	Requirements Modeling (+ User Stories?)
W Sep 30	Lecture	Project Workday
H/F Oct 2/3	Lab	Grade M4 Individual + Continue M5 (Frontend)
M Oct 5	Lecture	Project Workday – no class meeting – work on Milestone 5!
M Oct 5	OBP M5 T	M5 Team Deadline @ 11:45 pm
W Oct 7	Lecture	Presentations + Software Process
W Oct 7	OBP M5 I	M5 Individual Deadline @ 5:00 pm + Retro @ 11:45 pm
H Oct 8	Lab	M5 Team Grading + M5 Individual Grading
F Oct 9	OBP M6 T	M6 Team Deadline @ 11:45 pm
S Oct 10	OBP M6 I	M6 Individual Deadline @ 5:00 pm + Retro @ 11:45 pm
M Oct 12	Lecture	Midterm Exam
T Oct 13	OBP M7	M7 Reflection Deadline @ 11:45 pm

2

2

Plan early for M6: Onboarding Project Deadline!

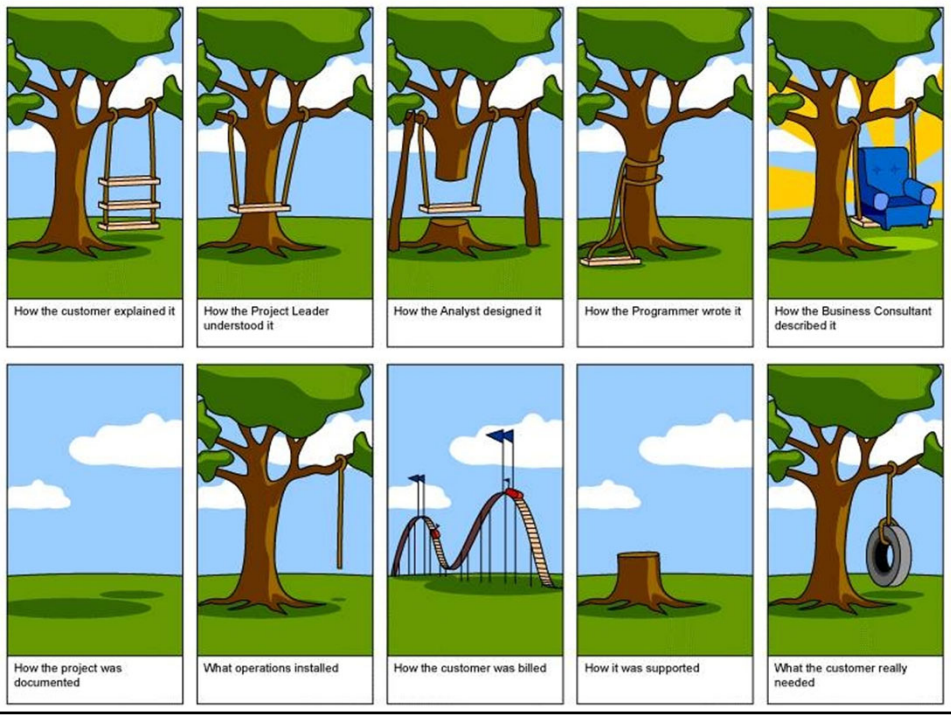
Milestone 4: Backend Implementation + Testing	H Sep 17	W Sep 23 @ 11:45 pm	M Sep 28 @ 5:00 pm
Milestone 5: Frontend Implementation + Testing	H Sep 24	M Oct 5 @ 11:45 pm	W Oct 7 @ 5:00 pm
Milestone 6: Onboarding Project Deadline	H October 8	F Oct 9 @ 11:45 pm	S Oct 10 @ 5:00 pm
Milestone 7: Onboarding Project Reflection	S Oct 10	N/A	T Oct 13 @ 11:40 pm

You should be 99% done with the project by the end of Milestone 5. Milestone 6 is for last-minute bug fixes.

There is a 24-hour late window for Milestone 6 Team and Individual deadlines.
There will be a penalty of no more than 10% of the Milestone 6 grade (90 points).

3

3



© NC Stat

4

4

Understand Requirements is Challenging…

- Need to figure out what the customers and other stakeholders want
  - Not make false/incorrect assumptions
- Need to chunk work into bite-sized pieces
  - So work can be divided up
  - So systems can be built incrementally
  - So estimates are remotely accurate
  - So we can know when we're done

© NC State Software Engineering Faculty

5

Software Requirements

- **Software Requirement**: condition or capability needed by a user to solve a problem or achieve an objective that must be met or possessed by a system or system component. Represented in a contract, standard, specification, or other formally imposed document.
  - Informally: a description of what the customers want
  - May be buried beneath layers of assumptions, misconceptions, and politics

© NC State Software Engineering Faculty

6

Software Requirements

- Requirements may range from a **high-level** abstract statement of a service or of a system constraint to a **detailed** mathematical functional specification

- This is inevitable as requirements may serve a dual function
  - May be the **basis for a bid for a contract** – therefore must be **open for interpretation**
  - May be the **basis for the contract** itself – therefore must be **defined in detail**
  - Both of these statements may be called requirements

© NC State Software Engineering Faculty

7

Requirements Engineering



Illustration: The overall requirements engineering process

img

8

Requirements Engineering

- **Analysis**: studying user needs to generate system definition that **users understand**
  - Fact-finding
  - Communication
  - Fact-validation



- **Modeling**: translating requirements the user understands to a form that **software engineers understand**
  - Representation
  - Organization



© NC State Software Engineering Faculty

Challenges of Requirements Engineering

- Many software developers argue that
  - Things will become clear as we build the software
  - Project stakeholders will be able to better understand what they need *only after examining early iterations of the software*
  - Things change so rapidly that requirements engineering is a waste of time
  - Bottom line is producing a working program and that all else is secondary

What do you think about requirements?

© NC State Software Engineering Faculty

Challenges of Requirements Analysis

- Trouble understanding the requirements or needs of the customer
- Recording requirements in a disorganized manner
- Spending far too little time verifying
- Allowing change to control us, rather than establishing mechanisms to control change
- Failure to establish a solid foundation for the system or software that the user wants built

© NC State Software Engineering Faculty

Challenges of Requirements Analysis

- Stakeholders
  - Don't know what they really want
  - Express their needs in their own terms
  - Have conflicting requirements
- Organizational and political factors may influence
- New stakeholders may emerge and the business environment may change.

What's an example of conflicting requirements?

© NC State Software Engineering Faculty

Examples of Requirements Conflicts

- **Data**: example: The UI team wants to immediately display all results (which could be over 1,000) from a search when the backend team knows they can only get 100 at a time.
- **Privacy/COI**: example: Customer service wants to see all customer information when on a call, but privacy rules stipulate that only necessary information should be shared
- **Values/Importance**: example: one stakeholder might find it acceptable to make a question about cultural heritage mandatory, where another considers this a sensitive question that should be optional.

13

13

Requirements Analysis Tasks

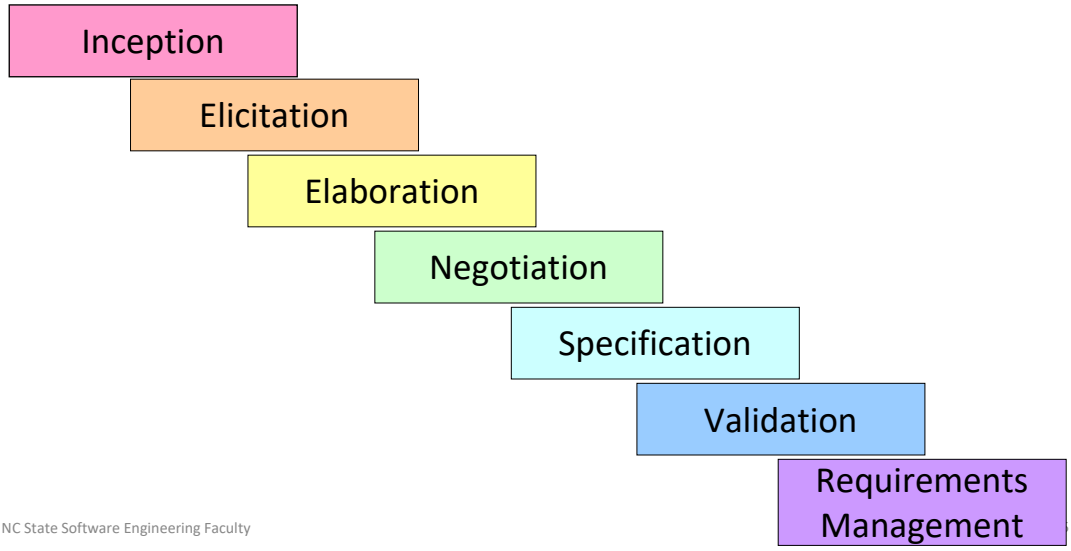
- 7 Distinct Tasks
  - Inception, Elicitation, Elaboration, Negotiation, Specification, Validation, and Requirements Management
- Some of these tasks may occur in parallel and all are adapted to the needs of the project
- All strive to define what the customer wants
- All serve to establish a solid foundation for the design and construction of the software system

© NC State Software Engineering Faculty

14

14

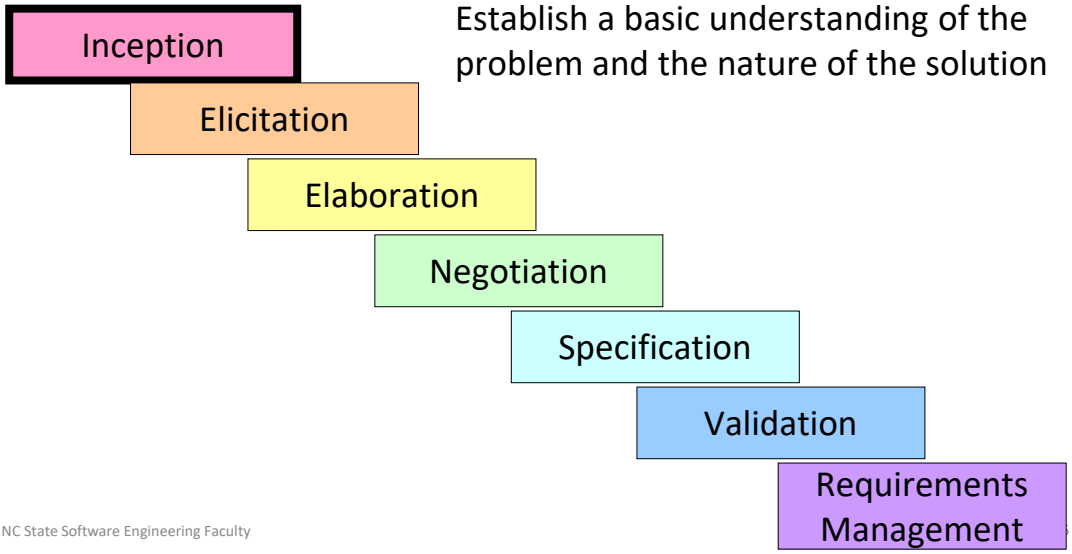
Requirements Engineering Tasks



© NC State Software Engineering Faculty

15

Requirements Engineering Tasks

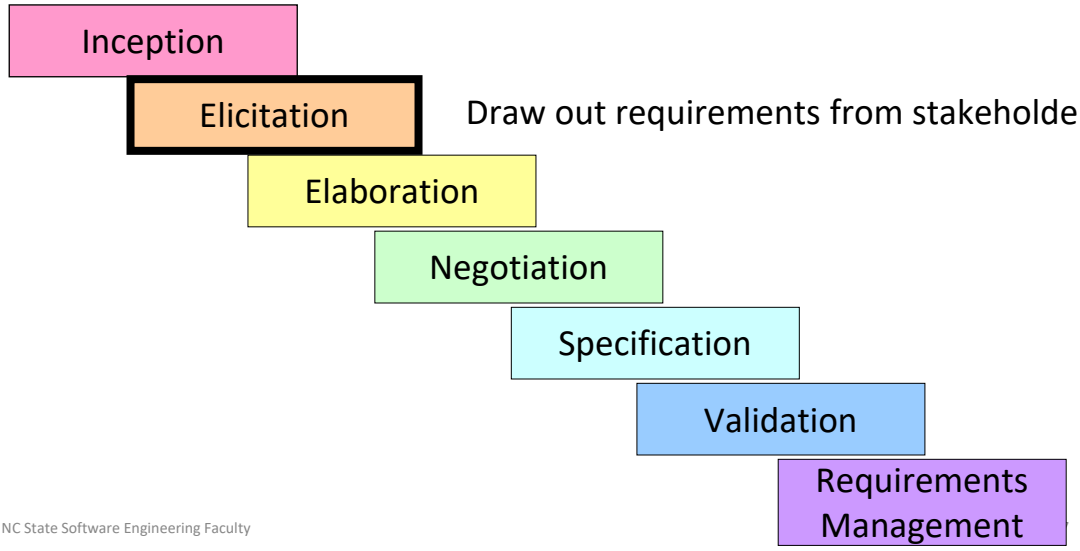


Establish a basic understanding of the problem and the nature of the solution

© NC State Software Engineering Faculty

16

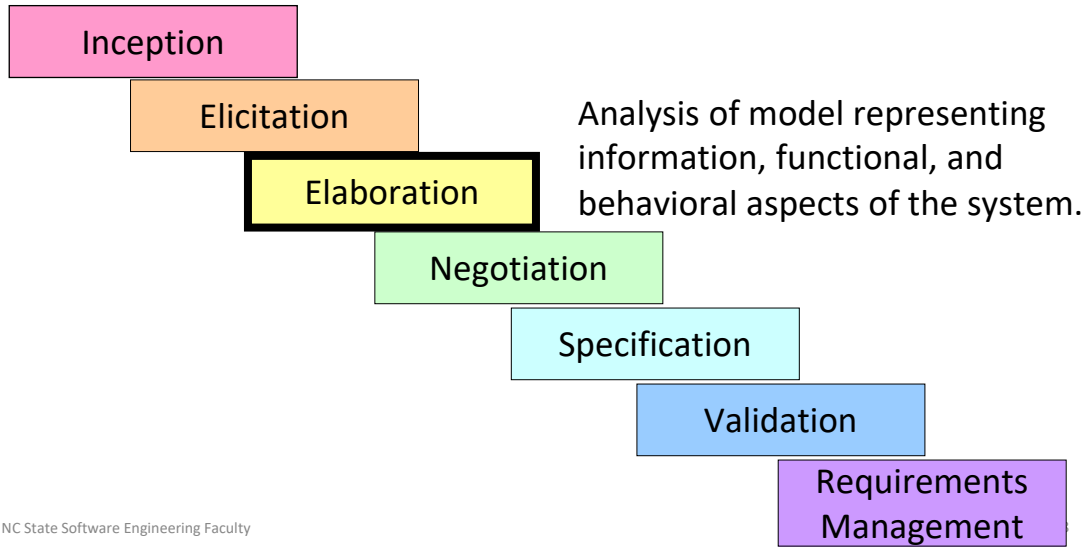
Requirements Engineering Tasks



© NC State Software Engineering Faculty

17

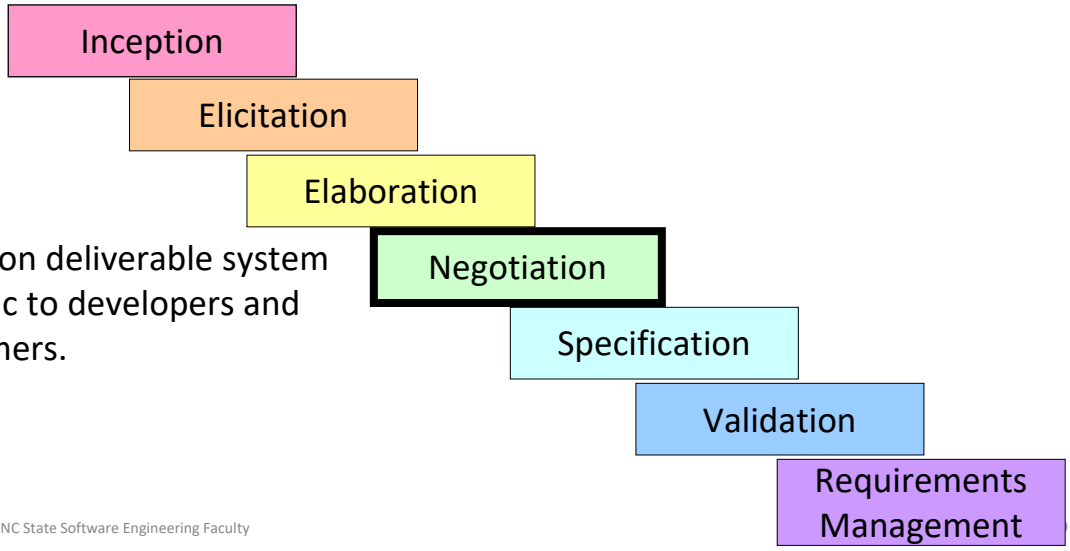
Requirements Engineering Tasks



© NC State Software Engineering Faculty

18

Requirements Engineering Tasks

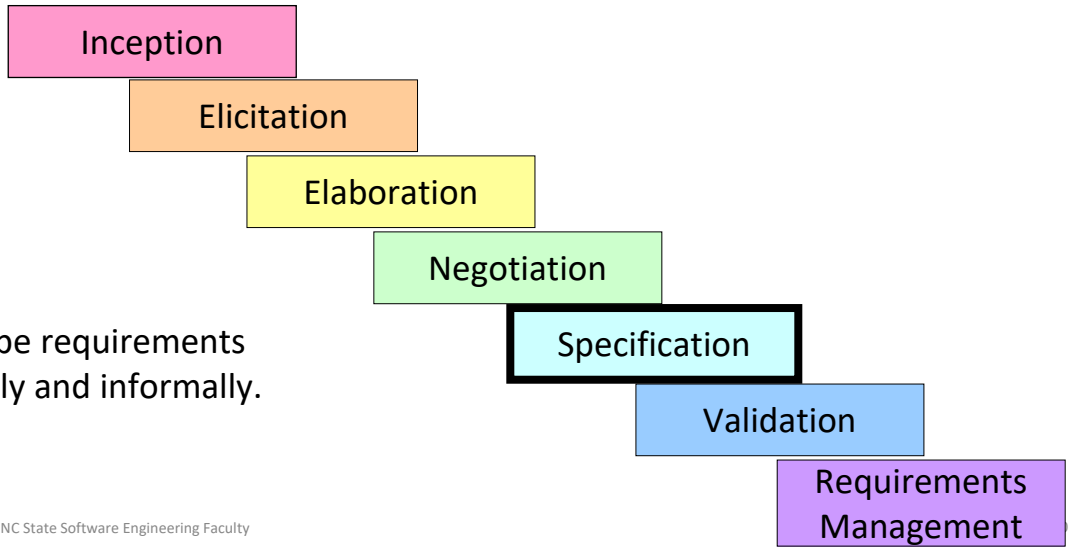


Agree on deliverable system realistic to developers and customers.

© NC State Software Engineering Faculty

19

Requirements Engineering Tasks

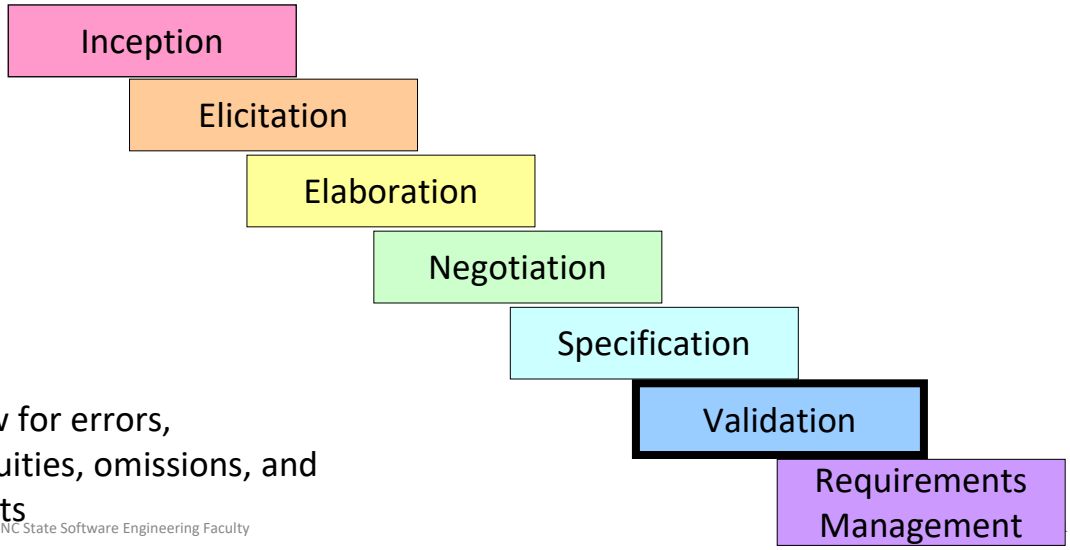


Describe requirements formally and informally.

© NC State Software Engineering Faculty

20

Requirements Engineering Tasks

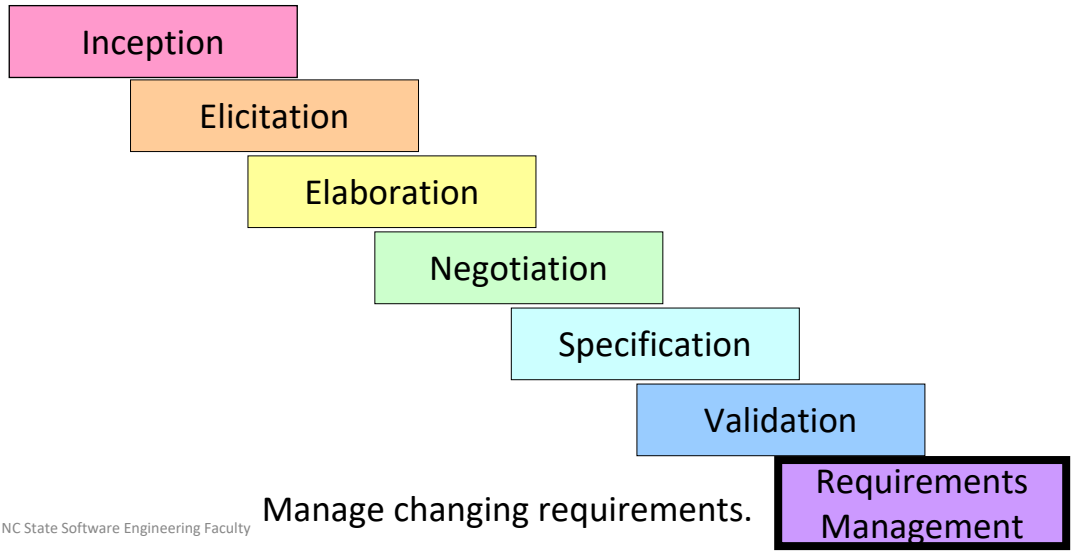


Review for errors, ambiguities, omissions, and conflicts

© NC State Software Engineering Faculty

21

Requirements Engineering Tasks



© NC State Software Engineering Faculty

Manage changing requirements.

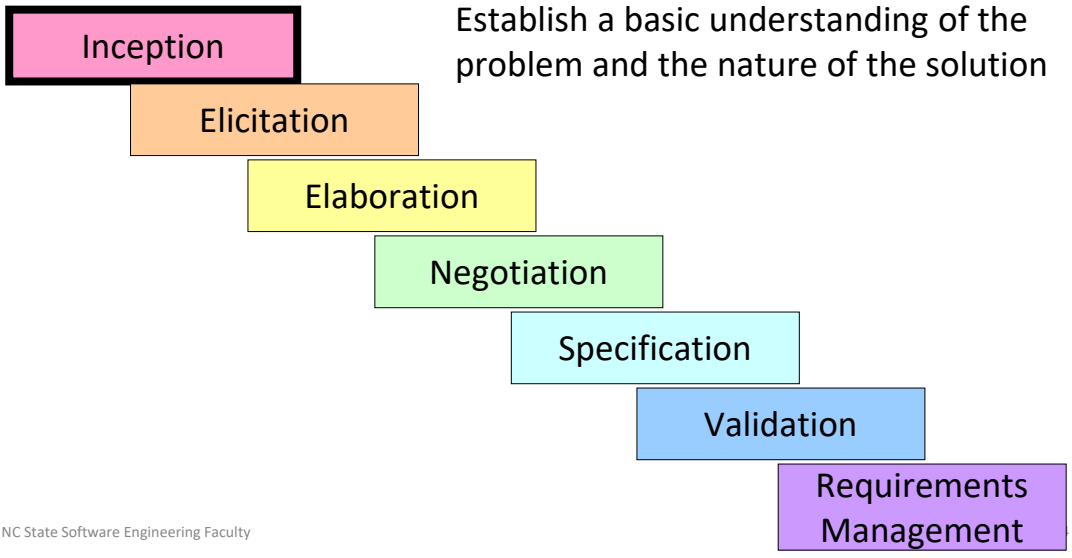
22

Let's dive in

© NC State Software Engineering Faculty

23

Task #1: Inception



Establish a basic understanding of the problem and the nature of the solution

© NC State Software Engineering Faculty

24

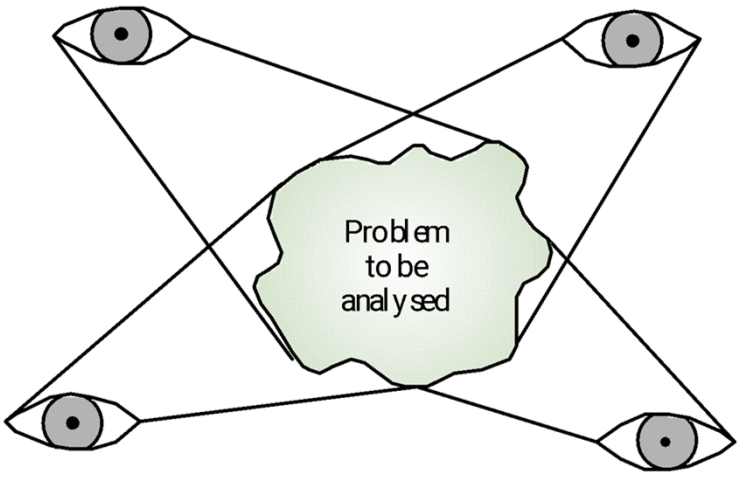
Inception Task

- During inception, the requirements engineer asks a set of questions to establish…
  - A basic understanding of the **problem**
  - The **people** who want a solution
  - The nature of the **solution** that is desired
  - The effectiveness of preliminary **communication and collaboration** between the customer and the developer
- Through these questions, the requirements engineer needs to…
  - Identify the stakeholders
  - Break the ice and initiate the communication
  - Work toward collaboration
  - Recognize multiple viewpoints

25

25

Multiple problem viewpoints



26

26

The First Set of Questions

These questions focus on the customer, other stakeholders, the overall goals, and the benefits

- Who is behind the request for this work?
- Who will use the solution?
- What will be the economic benefit of a successful solution?
- Is there another source for the solution that you need?

27

27

The Next Set of Questions

These questions enable a better understanding of the problem and allow the customer to voice his or her perceptions about a solution

- How would you characterize "good" output that would be generated by a successful solution?
- What problem(s) will this solution address?
- Can you show me (or describe) the business environment in which the solution will be used?
- Will special performance issues or constraints affect the way the solution is approached?

28

28

The Final Set of Questions

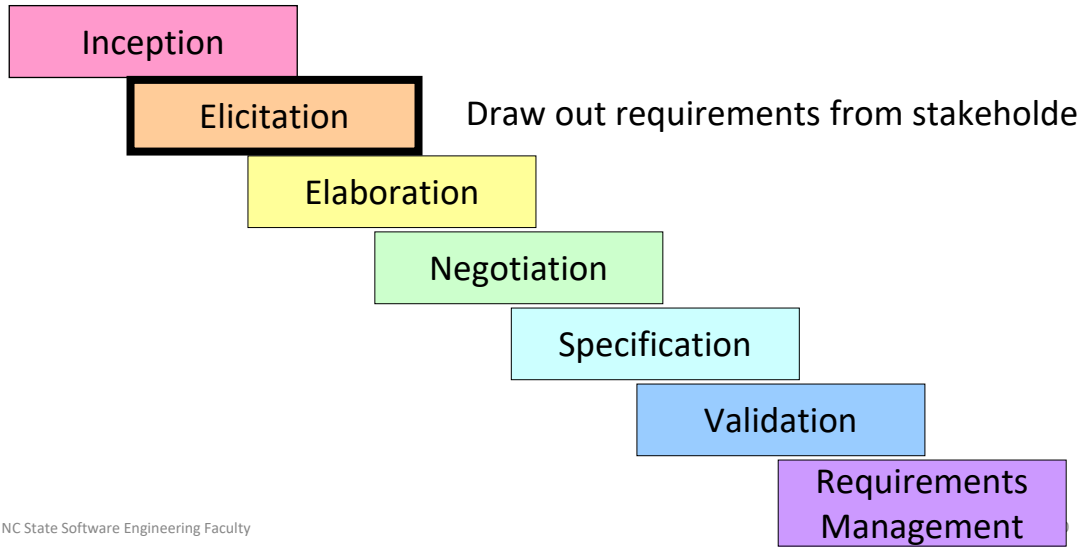
These questions focus on the effectiveness of the communication activity itself

- Are you the right person to answer these questions? Are your answers "official"?
- Are my questions relevant to the problem that you have?
- Am I asking too many questions?
- Can anyone else provide additional information?
- Should I be asking you anything else?

29

29

Task #2: Elicitation



Draw out requirements from stakeholder

© NC State Software Engineering Faculty

30

Where do requirements come from?

Richard Branson: "*I would like a spaceship.*"

PM/SA/Dev:



Why do you need a spaceship?

31

31

Where do requirements come from?

Richard Branson: "*I would like a spaceship.*"

PM/SA/Dev :



Why you need it and how much it will cost?

How much load it will carry?

Where do you want to send it in space?

32

32

Elicitation Task

- Eliciting requirements is difficult because of
  - Problems of scope
    - boundaries of the system is ill-defined
  - Problems of understanding
    - customers/users are not sure what they want
  - Problems of volatility
    - the requirements change over time

33

Requirements Elicitation

- Need to understand why not just what

The *why* is the business case!

- Requirements elicitation is a process with the goal to
  - Identify the problem(s)
  - Propose elements of the solution
  - Negotiate different approaches
  - Specify a preliminary set of solution requirements

© NC State Software Engineering Faculty

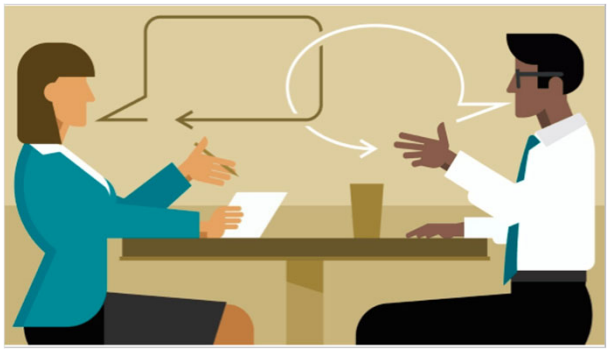
34

Elicitation Process Guideline

- Meetings are conducted and attended by both software engineers, customers, and other interested stakeholders
- Rules for preparation and participation are established
- An agenda is suggested
- A "facilitator" (customer, developer, or outsider) controls the meeting
- A "definition mechanism" is used such as worksheets, flip charts, wall stickers, electronic bulletin board, chat room, or some other virtual forum

Elicitation Techniques

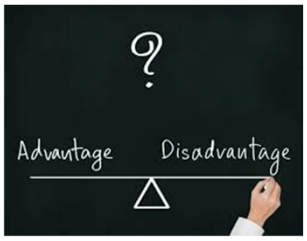
- Background reading
- Interviews
- Survey/questionnaires
- Brainstorming
- Meetings
- Task observation
- Use case approach
- …



img

Background Reading

- Sources of information
  - Company documents, organization charts, docs on existing systems, technical papers, etc.
- Advantages
  - Understanding of the organization before meeting people who work there
  - May find detailed requirements for current system
- Disadvantages
  - Written documents often do not match reality



37

37

Brainstorming

- More natural interaction between people than formal interviews
- Can increase the number of ideas
- Relative short and intensive sessions seem to be most productive
- Selection of personnel is critical
- Do not allow criticism or debate
- Idea reduction
  - Prune ideas that are not worthy of further discussion
  - Group similar ideas into super topics
- Prioritize the remaining ideas

38

38

Interviews

- The requirements engineer or analyst discusses the system with different stakeholders and builds up an understanding of their requirements.
- This is the predominant elicitation technique
- Types:
  - Structured: The requirements engineer looks for answers to a pre-defined set of questions
  - Open-ended: There is no predefined agenda and the engineer discusses, in an open-ended way, what stakeholders want

39

39

Interview Advantages/Disadvantages

- Advantages
  - Rich collection of information
  - Can probe in depth
- Disadvantages
  - Hard to analyze large amounts of qualitative data
  - Hard to compare different respondents



40

40

Interview Tips

- Be prepared with questions
- Take notes
- Don't make interviewees think you're thinking:
  - "My time is more valuable than yours"
  - "I don't know what I'm doing"
  - "You don't know what you're talking about"
- Be intelligible
- Be receptive

41

41

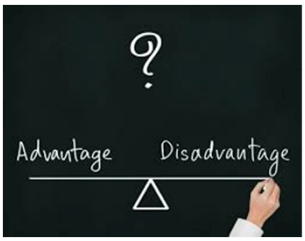
Questions to Ask

- The problems to be addressed
- The process of developing a solution
  - Who is paying for the development?
  - What are the underlying reasons for wanting a system?
  - What is the trade-off between delivery date and cost?
- The elicitation itself
  - Do my questions seem relevant?
  - Are your answers official?
  - Is there anything else I should be asking?
  - Is there anything you would like to ask me?

42

42

Surveys/Questionnaires



- Construct and administer a survey
- **Advantages**
  - Can quickly collect info from large numbers of people
  - Can be administered remotely
- **Disadvantages**
  - Simplistic questions can provide very little context
- Things to be careful of:
  - Sample size
  - Sample selection method (random, voluntarily)
  - Ambiguous questions
- A pilot run of a questionnaire can help eliminate bugs

43

43

Task Observation

Type	Description	Advantage	Disadvantage
Passive observation	watching users without interfering or influencing their actions	capturing authentic and unbiased data	may raise ethical and privacy issues
Active observation	engaging with users and asking questions, giving feedback or providing assistance.	useful to clarify and validate your findings	may introduce bias and affect user behavior
Participant observation	joining users and performing tasks as they do	gaining empathy and understanding a user's perspective	may require more time and effort

44

Ethnography

- Particular variation of observation (over extended period)
- People often find it hard to describe what they do because it is so natural to them. Sometimes, the best way to understand it is to observe them at work
- Ethnography is a technique from social sciences, proven valuable in understanding actual work processes
- Actual work processes often differ from formal, prescribed processes
- An ethnographer spends some time observing people at work and building up a picture of how work is done
- Expensive, so this may be cost effective only for complex and critical systems

45

45

Exercise

Exercise 13.01: Elicitation

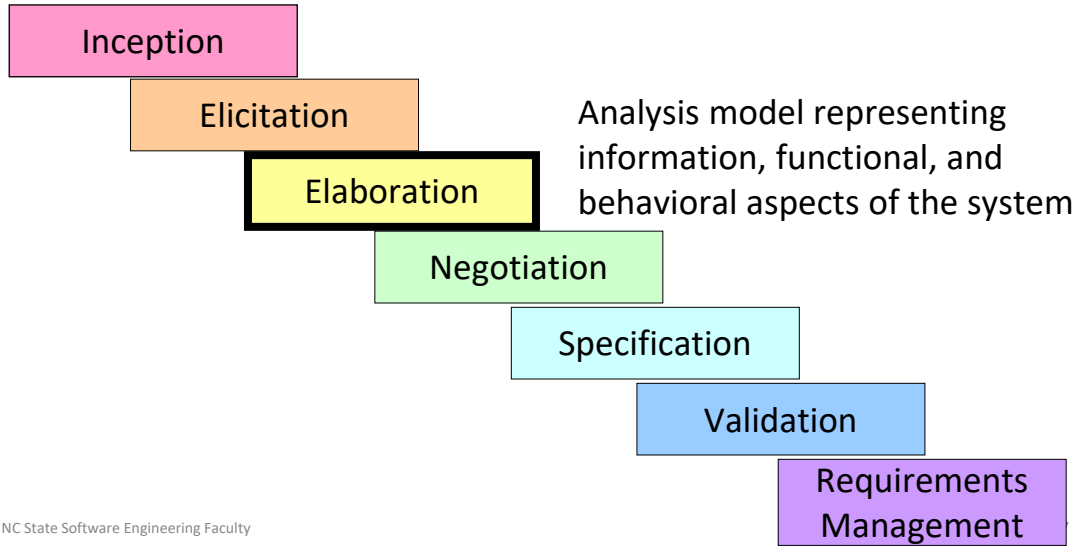
- Form groups of 2-3 students (this must be done in a group!)
  - Domain Expert
  - Interviewer
  - Observer
- Domain Expert – think about your favorite app/website/game in terms of the functional requirements (don't share the app/website/game)
- Interviewer – elicit the requirements from the domain expert
- [Optional] Observer – take notes and fill out the Google form for the group
  - If there is no Observer, then the Interviewer fills out the form

© NC State Software Engineering Faculty

46

46

Task #3: Elaboration



Analysis model representing information, functional, and behavioral aspects of the system.

© NC State Software Engineering Faculty

47

Elaboration Task

- During elaboration, the software engineer takes the information obtained during inception and elicitation and **begins to expand and refine it**
- Elaboration focuses on developing a refined technical model of software functions, features, and constraints
- It is an analysis modeling task
  - Use cases are developed
  - Domain classes are identified along with their attributes and relationships
  - State machine diagrams are used to capture the life of an object
- The end result is an analysis model that **defines the functional, informational, and behavioral domains of the problem**

48

Developing Use Cases

- Step #1 – Define the set of actors that will be involved in the story
  - Actors are people, devices, or other systems that use the system or product within the context of the function and behavior that is to be described
  - Actors are anything that communicate with the system or product and that are external to the system itself
- Step #2 – Develop use cases, where each one answers a set of questions

49

Questions Commonly Answered by a Use Case

- Who is the primary actor(s), the secondary actor(s)?
- What are the actor's goals?
- What preconditions should exist before the scenario begins?
- What main tasks or functions are performed by the actor?
- What exceptions might be considered as the scenario 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?

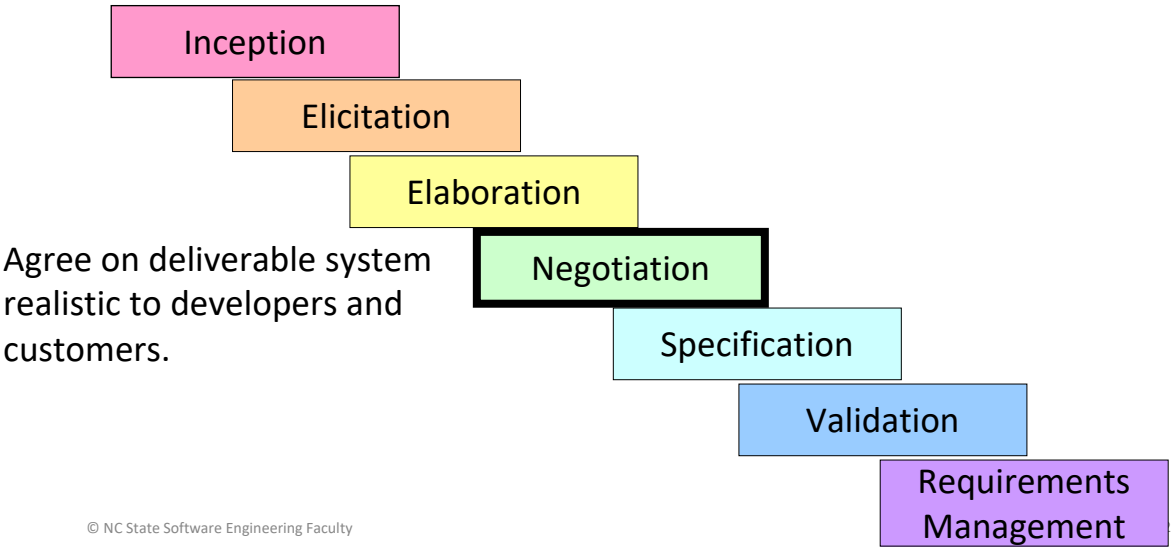
50

Elements of the Analysis Model

- Scenario-based elements
  - Describe the system from the user's point of view using scenarios that are depicted in use cases and activity diagrams
- Class-based elements
  - Identify the domain classes for the objects manipulated by the actors, the attributes of these classes, and how they interact with one another; they utilize class diagrams to do this
- Behavioral elements
  - Use state diagrams to represent the state of the system, the events that cause the system to change state, and the actions that are taken as a result of a particular event; can also be applied to each class in the system
- Flow-oriented elements
  - Use data flow diagrams to show the input data that comes into a system, what functions are applied to that data to do transformations, and what resulting output data are produced

51

Task #4: Negotiation



Agree on deliverable system realistic to developers and customers.

© NC State Software Engineering Faculty

52

Negotiation Task

- Agree on deliverable system that is realistic for developers and customers
- SWE team & other project stakeholders negotiate the priority, availability, and cost of each requirement
- Process:
  - Identify the key stakeholders
    - These are the people who will be involved in the negotiation
  - Determine each of the stakeholders "win conditions"
    - Win conditions are not always obvious
  - Negotiate
    - Work toward a set of requirements that lead to "win-win"

The Art of Negotiation

- Recognize that it is not competition
- Map out a strategy
- Listen actively
- Focus on the other party's interests
- Don't let it get personal
- Be creative
- Be ready to commit



img

Requirements Analysis

- To discover problems, incompleteness and inconsistencies in the elicited requirements.
- These are then fed back to the stakeholders to resolve them through the negotiation process
- Analysis is interleaved with elicitation as problems are discovered
- A checklist may be used to support analysis. Each requirement may be assessed against the checklist

55

55

Analysis Checklists

- Premature design
  - Does the requirement include premature design or implementation information?
- Combined requirements
  - Does the description of a requirement describe a single requirement or could it be broken down into several different requirements?
- Unnecessary requirements
  - Is the requirement 'gold plating'? That is, is the requirement a cosmetic addition to the system which is not really necessary.
- Non-standard
  - Does the requirement mean that non-standard hardware or software must be used? (To make this decision, you need to know the computer platform requirements.)

56

56

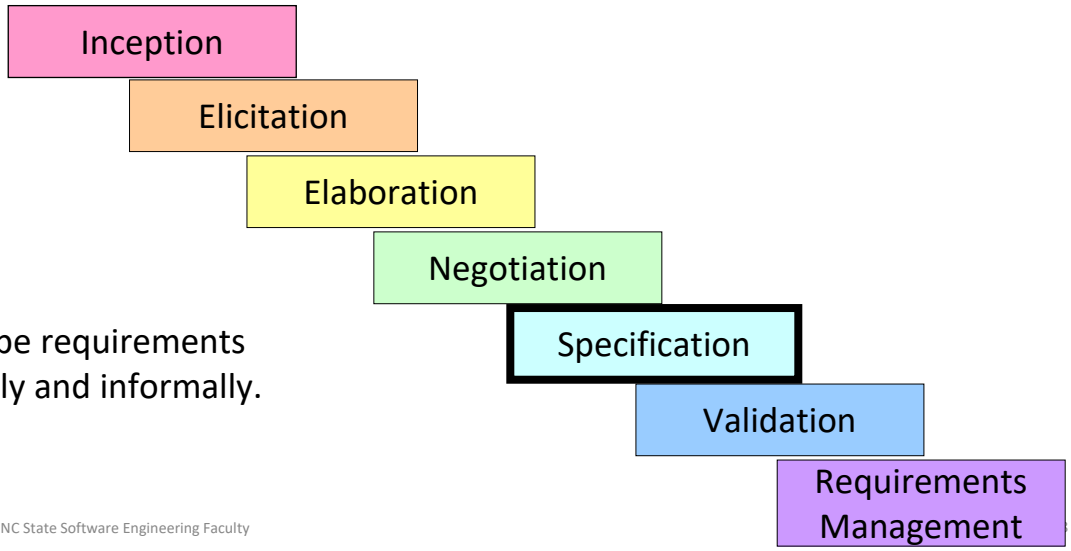
Analysis Checklists

- Conformance with business goals
  - Is the requirement consistent with the business goals defined in the introduction to the requirements documents?
- Requirements ambiguity
  - Could the requirement be understood differently by different people?
- Requirements realism
  - Is the requirement realistic given the technology which will be used to implement the system?
- Requirements testability (verifiability)
  - Is the requirement testable, that is, is it stated in such a way that test engineers can show if the system meets that requirement?

57

57

Task #5: Specification



Describe requirements formally and informally.

© NC State Software Engineering Faculty

58

Specifications (Generally)

- A broad term that means definition
- A statement of agreement (contract) between
  - producer and consumer of a service

59

59

Specifications (Generally)

- Document many aspects of projects:
  - What the users need (requirements)
  - The features offered by a system (functional spec)
  - The performance characteristics
  - The design of the software (e.g. architecture)
  - External behavior of a module
  - Internal structure of a module

60

60

Software Requirements Specifications (SRS)

- Statement of user requirements
- Major failures occur through misunderstandings between the producer and the user
  - Client: “But I expected that the system would be able to handle that.”
  - Developer: “Well, that’s not the impression you gave me. I thought that you wanted it to operate like this.”

61

61

Specification Qualities

- Consistent
- Complete
- Verifiable
- Traceable
- Precise, clear, unambiguous

62

62

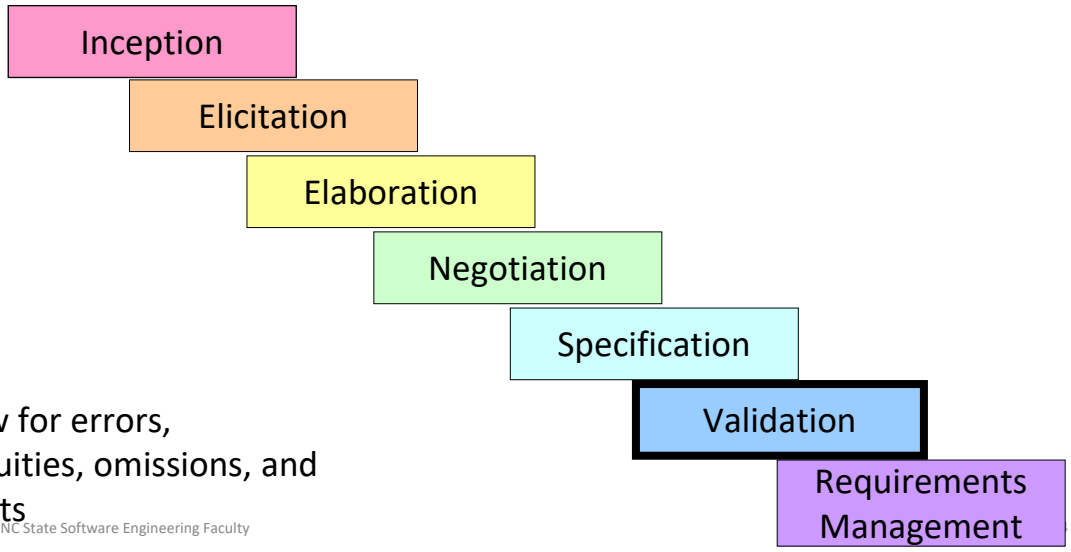
Technically Speaking, "requirement" ≠ "specification"



- Requirement – understanding between **customer and supplier**
- Specification – what the **software** must do

  → we'll be writing specifications, next class!
- Requirements that are not in the SRS
  - Costs
  - Delivery dates
  - Acceptance procedures
  - etc

Task #6: Validation



Review for errors, ambiguities, omissions, and conflicts

© NC State Software Engineering Faculty

Validation Objectives

- Certify that the requirements document is an acceptable description of the system to be implemented
- Check a requirements document for
  - Completeness
  - Understandability, organization
  - Conformance to standards
  - Requirements conflicts, redundancy, ambiguity
  - Technical errors
  - Traceability and Verifiability

65

65

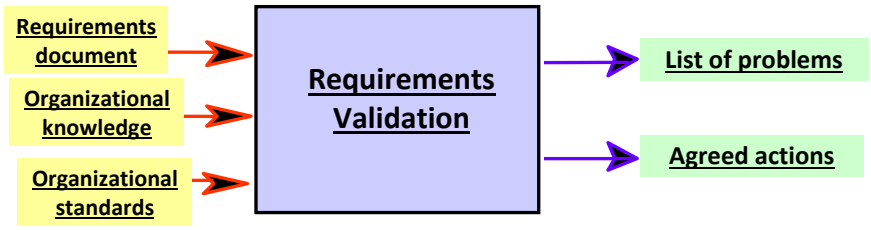
Analysis and Validation

- Analysis works with raw requirements as elicited from the system stakeholders
  - The requirements are known to be incomplete and expressed in an informal, unstructured way
- Validation works with a final draft of the requirements document, i.e., with negotiated and agreed-on requirements
  - Remove additional sources of incompleteness, inconsistency, and so forth

66

66

Validation Inputs and Outputs



67

67

Validation Inputs

- Requirements document
  - Should be a complete version of the document, not an unfinished draft.
- Organizational knowledge
  - Knowledge on the part of the organization, that may be used to judge the realism of the requirements
- Organizational standards
  - Local standards, e.g., for the structure of the requirements document

68

68

Validation Outputs

- Problem list
  - List of discovered problems in the requirements document
- Agreed-on actions
  - List of agreed-on actions in response to requirements problems
  - Some problems may have several corrective actions

69

69

Requirements and Testing

- Testing doesn't occur until later in the lifecycle, when code is available.
- Still, each functional requirement must be testable
  - i.e. it should be possible to define tests to check whether or not that requirement has been met.
- Inventing requirements tests is an effective validation technique as missing or ambiguous information in the requirements description may make it difficult to formulate tests
- Testers can begin developing tests as soon as requirements specs are available

70

70

(example) Test Case for SQRT

Test 1.

PURPOSE:

- Test the invocation of sqrt without arguments, and the operation of sqrt on an integer in the expected range.

SETUP:

- None needed

INPUTS and EXPECTED OUTPUTS:

inputs	expected output
"sqrt"	"Enter number:"
"4"	"2" followed by command-line prompt

TESTING REQUIREMENTS COVERED: 1 and 6

71

71

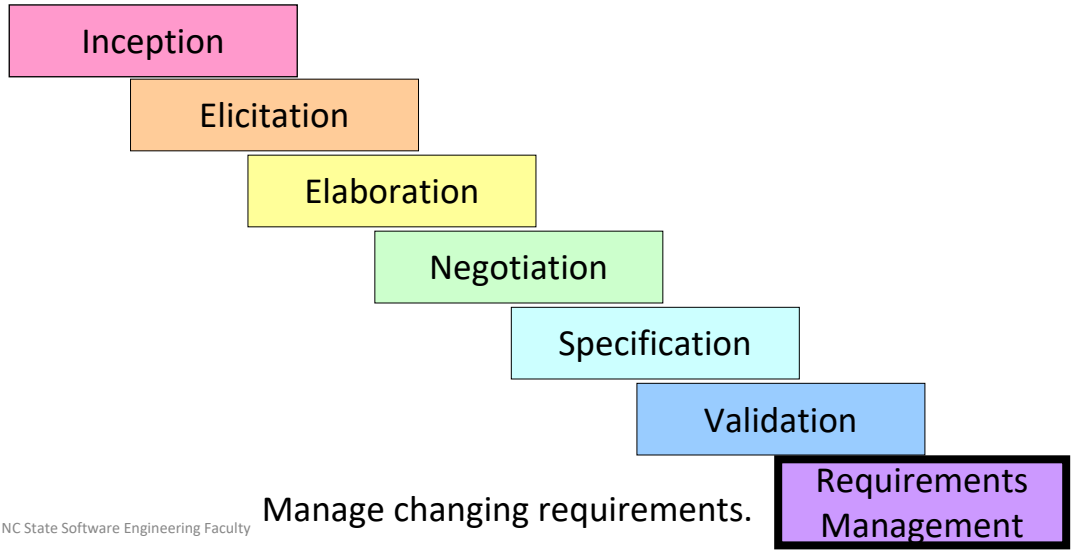
Validation Vs. Verification

- Validation: "Am I building the right product?" checking a work product against higher-level work products or authorities that frame this particular product.
  - Requirements are validated by stakeholders
- Verification: "Am I building the product right?" checking a work product against some standards and conditions imposed on this type of product and the process of its development.
  - Requirements are verified by the analysts mainly

72

72

Task #7: Requirements Management



Manage changing requirements.

© NC State Software Engineering Faculty

73

Requirements Management

- The process of managing change to the requirements for a system
- The principal concerns of requirements management are:
  - Managing changes to agreed requirements
  - Managing relationships between requirements
  - Managing the dependencies between the requirements document and other artifacts produced in the systems engineering process (i.e., design docs, code, tests, user docs….)

74

74

Reasons for Requirements Changes

- Errors and misunderstandings
- New requirements emerge
  - Deeper understanding of the system
- Changing external circumstances
  - Changes of strategy or priority of the business
    - Economic circumstances, new competitors
  - New information about the system's environment
    - E.g., new digital maps for a geographic info system
  - New laws or regulations

75

75

Requirements Traceability

- Requirements cannot be managed effectively without requirements traceability
- A requirement is traceable if you can determine
  - Why the requirement exists (perhaps including who suggested it)
  - What requirements are related to it
  - How that requirement relates to other information such as systems design, code, tests, and user documentation.

76

76

Traceability

- Traceability information is data that helps you assess the impact of requirements change
- It links related requirements and the requirements and other system representations
- Traceability matrices and tables may be used to record traceability information.

77

77

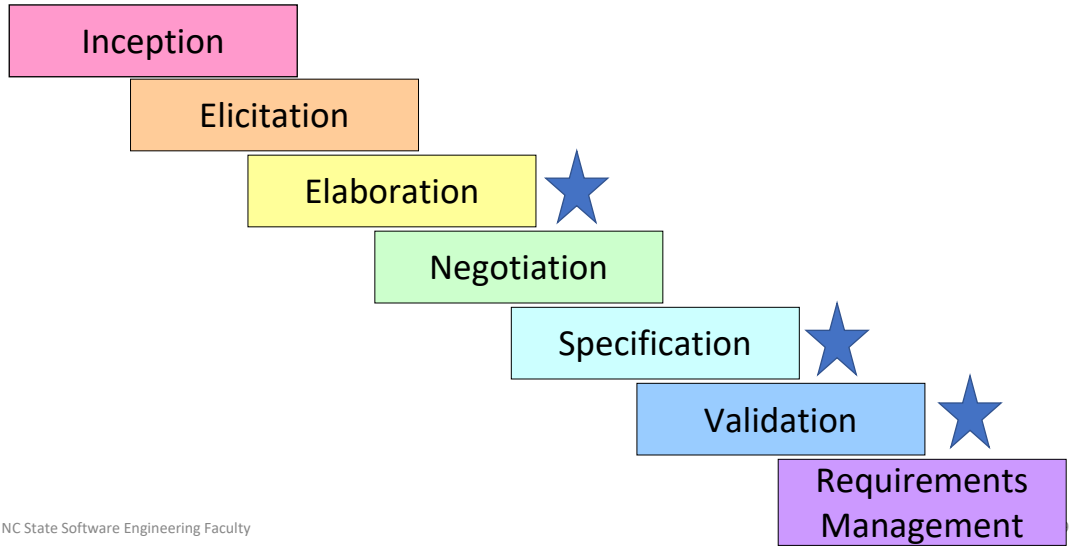
Requirements Management Task

- Project team performs a set of activities to
  - identify,
  - control,
  - track requirements changes at any time as the project proceeds
- Each requirement is assigned a unique identifier
- The requirements are then placed into one or more traceability tables
- These tables may be stored in a database that relate features, sources, dependencies, subsystems, and interfaces to the requirements
- A requirements traceability table is also placed at the end of the software requirements specification

78

78

Requirements Engineering Tasks



© NC State Software Engineering Faculty

79

References:

- Pressman, R. Software Engineering: A Practitioner's Approach. McGraw-Hill, 2005, Chapter 7
- Lecture notes on Software Requirements by Preeti Mishra
- Ian Sommerville, Software Engineering, 5th edition, 1995, Chapter 4

80