同济大学软件学院
School of Software Engineering, Tongji University

Software Testing Foundation Level

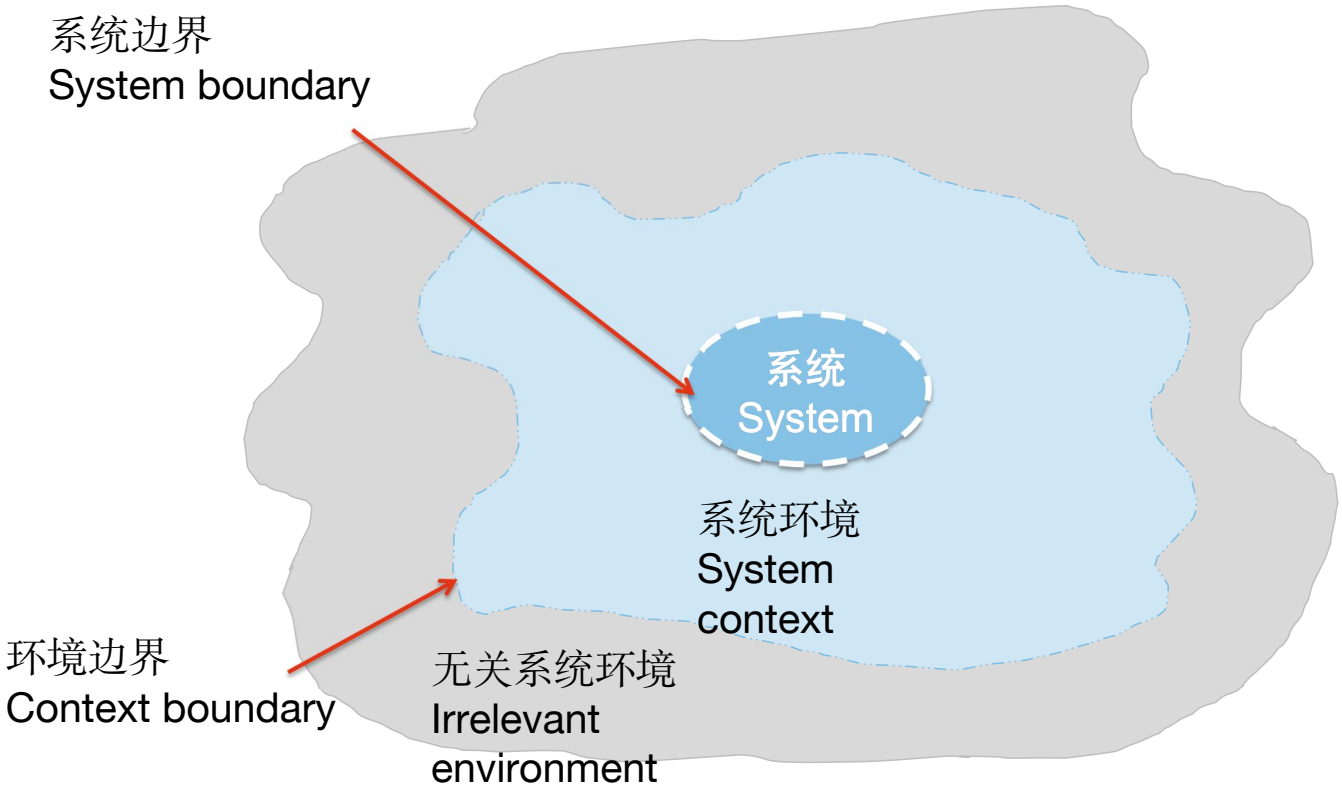
Chapter 4: Test Design Techniques

刘琴 Qin Liu

4. Test Design Techniques

1. Identifying and designing test cases
2. Categories of test design techniques
3. Specification-based or black-box techniques
4. Structure-based or white-box techniques
5. Experience-based techniques
6. Choosing test techniques

Review: System and System Boundary



Requirements engineering *focuses on the specification of requirements for systems consisting of software components, technical elements (computer hardware, equipment, sensing, etc.) and organizational elements (people, jobs, business processes, etc.)*

Review: 5 transformations of natural language

1. Nominalization

“ In case of a system crash, a restart of the system shall be performed”

2. Nouns without reference index

“The data shall be displayed to the user on the terminal”

“the data” What data?

“the user” Who?

“the terminal” Which terminal?

Example: “The system shall display the billing data to the registered user on the terminal she is logged in to”?

3. Universal quantifiers

“The system shall show all data sets in every submenu ” , “all data sets” ?

“every submenu” ? Really?

Review: 5 transformations of natural language

4. Incompletely specified conditions

"The restaurant system shall offer all beverages to a registered guest over the age of 20 years"

"all beverages" Does that include alcoholic and non-alcoholic? What do you drink if you are "20 years old and under"?

"All alcohol-free beverages to any registered user younger than 21 years"

"All beverages including all alcoholic beverages to any user over the age of 20"

5. Incompletely specified process verbs

There are verbs that describe a process. A certain adjective or adverb is needed to assist in the semantic integrity. Sometimes it is better to use the active voice than the passive voice to avoid the lack of semantics.

Example:

"The data needs to be transmitted"

What data? From where to where?

"To log a user in, the login data is entered"

Where do I log in? What information needs to be entered into what and where?

Chapter 4:
Test Design Techniques

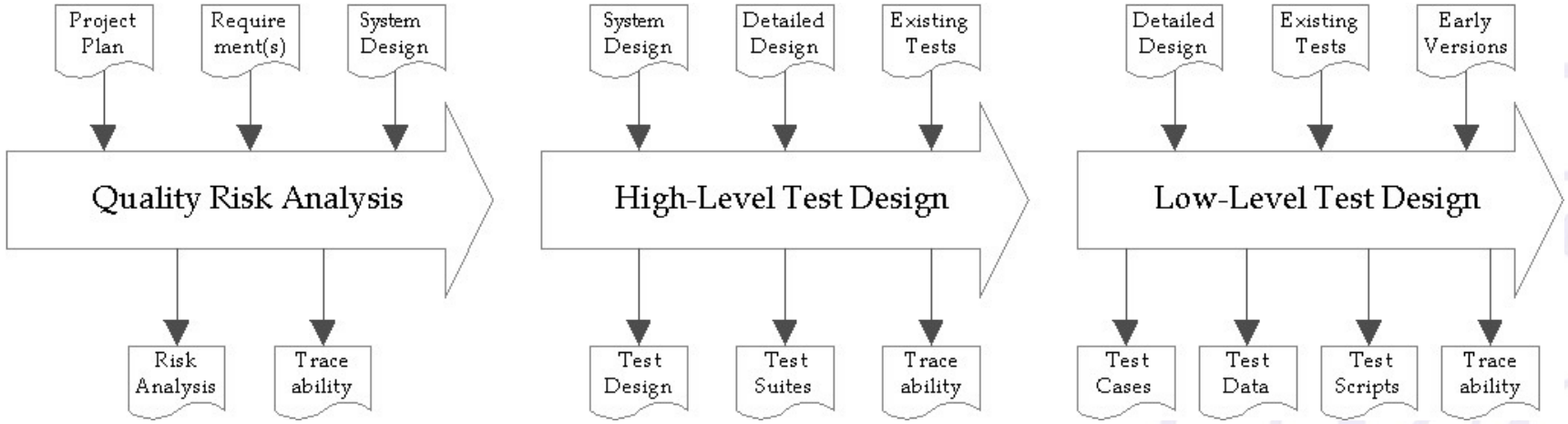
Section 1:
The test development process

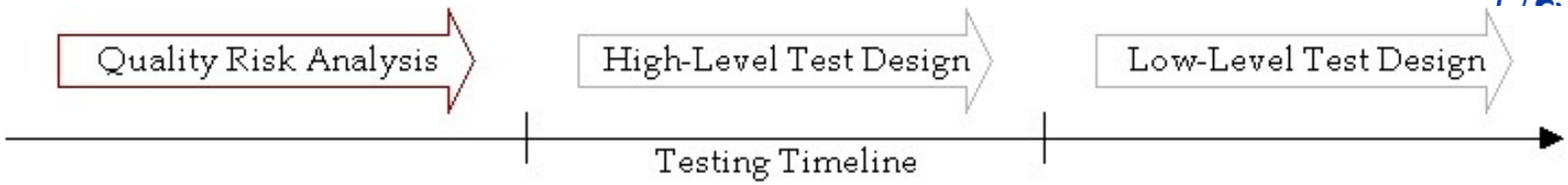
Identifying and Designing Test Cases

- Key concepts
  - Base testing on risk analysis
  - Determine level of risk with likelihood and impact
  - Specify test designs, cases, procedures
  - Translate test cases into test procedures
  - Relate test cases and test procedures
  - Develop test execution schedule
- Terms to remember

Phases of Test Development

- Test development often proceeds in phases
  - (Quality risk) analysis
  - High-level test design
  - Low-level test design (implementation)
- External inputs used to create internal deliverables (testware)



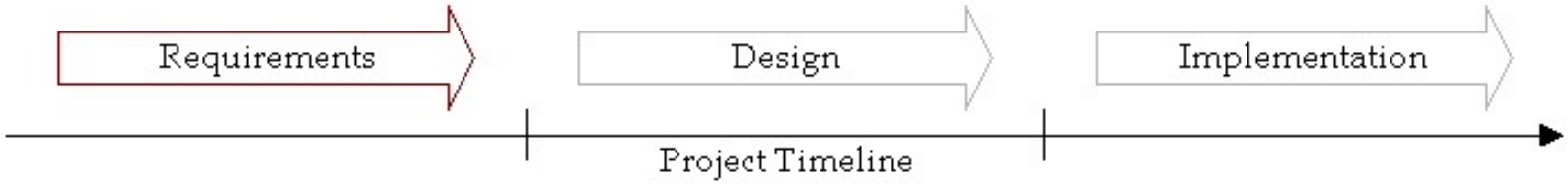


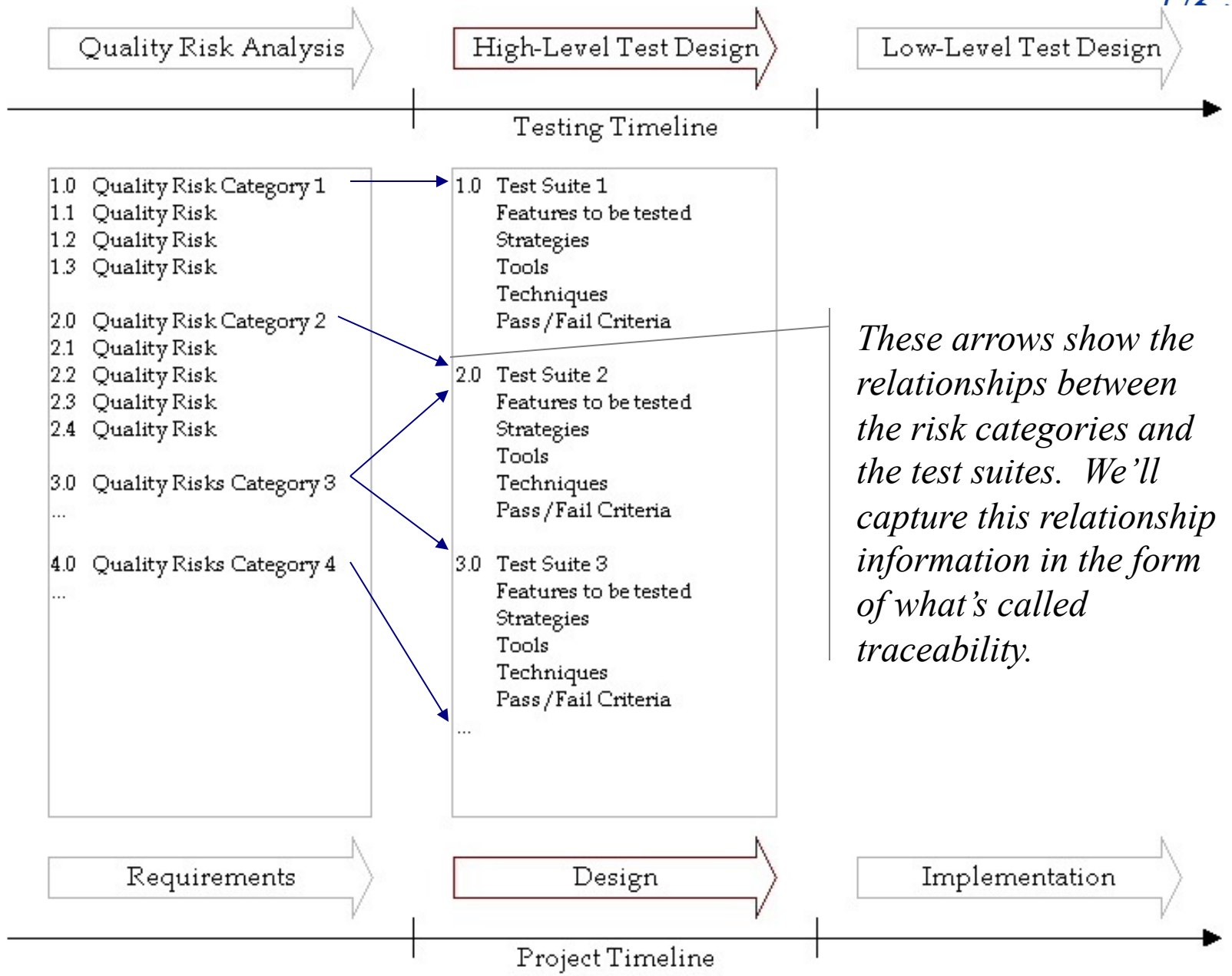
1.0 Quality Risk Category 1
1.1 Quality Risk
1.2 Quality Risk
1.3 Quality Risk

2.0 Quality Risk Category 2
2.1 Quality Risk
2.2 Quality Risk
2.3 Quality Risk
2.4 Quality Risk

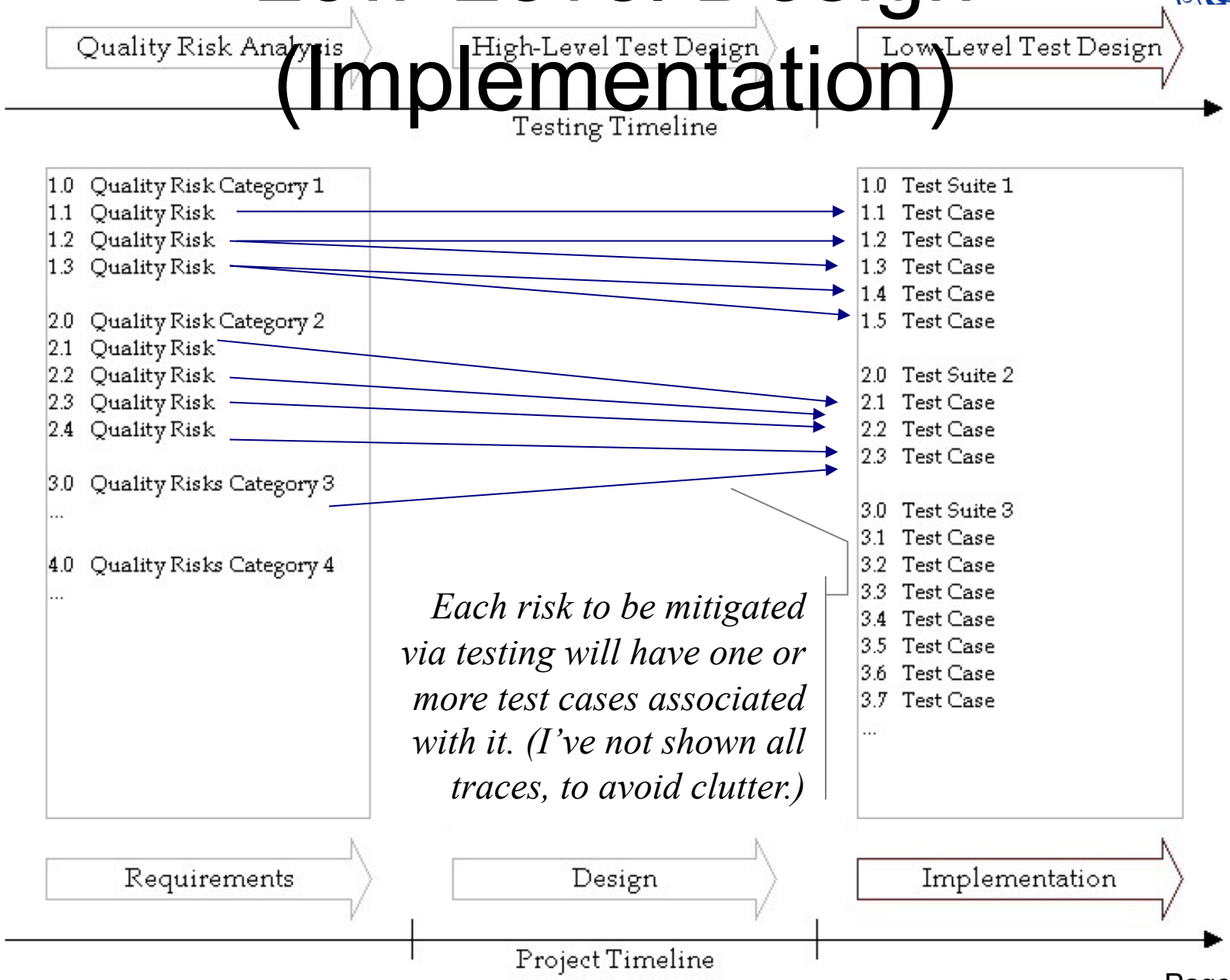
3.0 Quality Risks Category 3
...

4.0 Quality Risks Category 4
...




Quality Risk Analysis
High-Level Test Design
Low-Level Test Design
Testing Timeline
1.0 Quality Risk Category 1
1.1 Quality Risk
1.2 Quality Risk
1.3 Quality Risk
2.0 Quality Risk Category 2
2.1 Quality Risk
2.2 Quality Risk
2.3 Quality Risk
2.4 Quality Risk
3.0 Quality Risks Category 3
...
4.0 Quality Risks Category 4
...
1.0 Test Suite 1
Features to be tested
Strategies
Tools
Techniques
Pass/Fail Criteria
2.0 Test Suite 2
Features to be tested
Strategies
Tools
Techniques
Pass/Fail Criteria
3.0 Test Suite 3
Features to be tested
Strategies
Tools
Techniques
Pass/Fail Criteria
...
These arrows show the relationships between the risk categories and the test suites. We'll capture this relationship information in the form of what's called traceability.
Requirements
Design
Implementation
Project Timeline

Low-Level Design (Implementation)



What's a Quality or Product Risk?

- Risk
  - The possibility of a negative or undesirable outcome
  - The likelihood of a risk becoming an outcome is…
  - >0, <1 in the future…
  - 0 or 1 in the past
- Quality or product risk
  - The possibility that the system will fail to satisfy customers, users, or other stakeholders
  - A family of possible bugs are behind quality risks

How Can We Analyze Quality Risks?

Informal	ISO 9126	Failure Mode and Effect Analysis
Start with the classic quality risk categories	Start with six main quality characteristics	Start with categories, characteristics, or subsystems
Functionality, states and transactions, capacity and volume, data quality, error handling and recovery, performance, standards and localization, usability, etc.	Functionality, Reliability, Usability, Efficiency, Maintainability, Portability (FRUEMP), then decompose into key subcharacteristics for your system	Key stakeholders list possible failure modes, predict their effects on system, user, society, etc., assign severity, priority, and likelihood, then calculate risk priority number (RPN)
Set priority for testing each quality risk with key stakeholders	Set priority for testing each subcharacteristic with key stakeholders	Stakeholders use RPN to guide appropriate depth and breadth for testing

Regardless of technique the keys are cross-functional stakeholder participation, consensus, and a best-possible-outcome outlook

Quality risks are potential system problems which could reduce user satisfaction

Risk priority number: Aggregate measure of problem risk

Business (operational) risk: Impact of the problem

Technical risk: Likelihood of the problem

Tracing information back to requirements, design, or other risk bases

Quality Risk	Tech. Risk	Bus. Risk	Risk Pri. #	Extent of Testing	Tracing
Risk Category 1					
Risk 1					
Risk 2					
Risk *n*					

A hierarchy of risk categories can help organize the list and jog your memory.

1 = Very high
2 = High
3 = Medium
4 = Low
5 = Very low

The product of technical and business risk, from 1-25.

1-5 = Extensive
6-10 = Broad
11-15 = Cursory
16-20 = Opportunity
21-25 = Report bugs

Tips for Risk Analysis

- Use a cross-functional brainstorming team
- Identify the risk items, then assign the level of risk
- Only separate risk items when necessary to distinguish between different levels of risk
- Consider technical and business risk
  - Technical risk: likelihood of problem, impact of problem on system
  - Business risk: likelihood of usage, impact of problem on users
- Follow up and re-align risk analysis, testing, and the project at key project milestones

A2.1.6: Omninet Risk Analysis

- Based on your reading of the Omninet Marketing Requirements Document, the Omninet System Requirements Document, and your experience with testing and bugs, perform a risk analysis for Omninet
- Discuss.

IEEE 829 Test Design Specification

- The test design specification describes a collection of test cases and the test conditions they cover at a high level, and includes the following sections
  - Test design specification identifier
  - Features to be tested (in this test suite)
  - Approach refinements (specific techniques, tools, etc.)
  - Test identification (tracing to test cases in suite)
  - Feature pass/fail criteria (e.g., test oracle, test basis, legacy systems, etc.)
- This collection of test cases often called a test suite
- Sequencing (test suites and cases within suites) often driven by risk and business priority and affected by project constraints, resources, and progress

IEEE 829 Test Case Specification

- A test case specification describes the details of a test case, and includes the following sections
  - Test case specification identifier
  - Test items (what is to be delivered and tested)
  - Input specifications (user inputs, files, etc.)
  - Output specifications (expected results, including screens, files, timing, etc.)
  - Environmental needs (hardware, software, people, props…)
  - Special procedural requirements (operator intervention, permissions, etc.)
  - Intercase dependencies (if needed to set up preconditions)
- In practice, test cases vary significantly in effort, duration, and number of test conditions covered

IEEE 829 Test Procedure Specification

- A test procedure specification describes how to run one or more test cases, and includes the following sections
  - Test procedure specification identifier
  - Purpose (e.g., which tests are run)
  - Special requirements (skills, permissions, environment, etc.)
  - Procedure steps (logging, set up, start, proceed [steps themselves], measurement of results, shutdown/suspension, restart [if needed], stop, wrap up/tear down, contingencies)
- Test procedures are often embedded in test cases
- A test procedure is sometimes referred to as a test script, and may be manual or automated

Test Case Coverage (Traceability)

- Measure or correlate tests against areas of concern
- Used as a way to measure and enhance the tests
- Practical types
  - Requirements specifications
  - Design specifications
  - Functional areas
  - Quality risks
  - Configurations
- This is a commonly-used technique to ensure thorough test coverage

	Test Case													
	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	2.1	2.2	2.3	2.4	2.5	Total
Spec														
1.1	1	1		1		1		2						6
1.2	2			2	1				1		2			8
1.3														0
1.4		2									2			4
1.5	1			1				1	1					4
1.6						2								2
1.7			2		1		2			2				7
1.8					2							1		3
1.9	1			1							2			4
1.10		1					1	1			1			4
Total	5	4	2	5	4	3	3	4	2	2	7	1	0	

- One technique
  - Use spreadsheet
  - List test cases and coverage area to measure
  - 0: none; 1: indirect; 2: direct

Exercise: Omninet Test Design Specification

- Based on your quality risk analysis for Omninet, outline a set of test suites to address the important areas of risk.
- For each test suite, list briefly:
  - What the test would cover
  - How you would recognize passing or failing tests
- Establish the relationship between your test suites and the risks they will cover.
- Based on your risk analysis, how would you sequence the test suites? What other considerations would affect the sequencing of test suites?
- Discuss.

Chapter 4:
Test Design Techniques

Section 2:
Categories of test design techniques

Categories of Test Design Techniques

- Key concepts
  - Reasons for specification-based (black box), structure-based (white box), and experience-based tests
  - Common black box and white box techniques
  - Characteristics and differences between specification-based testing, structure-based testing, and experience-based testing
- Terms to remember

Three Types of Test Design Techniques

- Specification-based
  - Create tests primarily by analysis of the test basis
  - Look for bugs in the way the system behaves
- Structure-based (white box)
  - Create tests primarily by analysis of the structure of the component or system
  - Look for bugs in the way the system is built
- Experience-based (attacks, checklists, exploratory)
  - Create tests primarily based on understanding of the system, past experience, and educated guesses about bugs
  - Look for bugs in the places other systems have bugs
- Black-box tests are specification- or experience-based, functional or non-functional

Specification-Based or Black Box

- Common elements include:
  - Formal or informal models used to specify the problem to be solved, the software or its components
  - Test cases derived systematically from these models
- Examples include:
  - Equivalence partitioning and boundary value analysis
  - State transition diagrams
  - Decision tables

Structure-Based or White Box

- Common elements include:
  - System structure (e.g., code, design, etc.) used to derive the test cases, for example code and design
  - The extent of structural coverage can be measured for existing other test cases
  - Further test cases can be derived systematically to increase coverage
- Examples include:
  - Statement coverage
  - Branch coverage

Experience-Based

- Common elements include:
  - The knowledge and experience used to derive test cases
  - Can consider knowledge and experience of the software, its usage and its environment or…
  - …knowledge about historical and likely defects and their distribution
- Examples include:
  - Attacks
  - Checklists
  - Exploratory

Exercise : Omninet Test Techniques

- Refer to your outline of test suites for Omninet.
- For each test suite, identify whether one, two, or all three of the following categories of test techniques would be useful in designing test cases:
  - Specification-based (black box)
  - Structure-based (white box)
  - Experience-based
- Discuss.

Chapter 4:
Test Design Techniques

Section 3:
Specification-based or black box techniques

Specification-based Techniques

- Key concepts
  - Writing test cases from given software models using equivalence partitioning, boundary value analysis, decision tables, and state transition diagrams
  - The main purpose of each technique and how coverage may be measured
  - Use case testing
- Terms to remember

Equivalence Partitioning

- Divide the inputs, outputs, behaviors, and environments into classes you think will be handled equivalently
- Define at least one test case in each partition, or use boundary values in partitions that are ranges
- Can use marketing info to favor classes

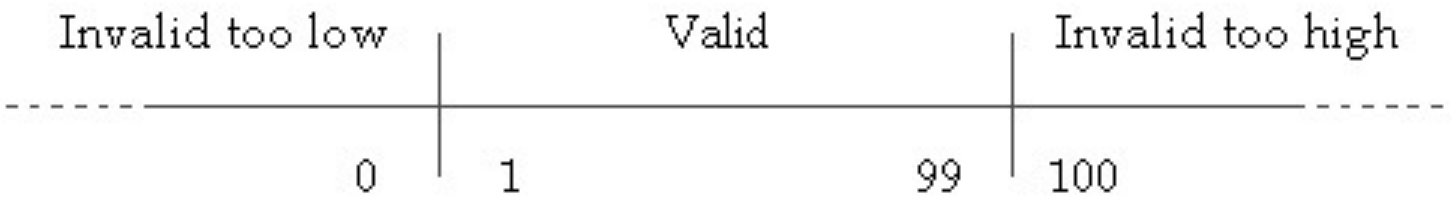
- Ex: Testing supported printers
  - Physical interface: Parallel, serial, USB 1.1, USB 2.0, infrared, Firewire, SCSI, others?
  - Logical interface: Postscript, HPPL, ASCII, others…
  - Image application: Laser jet, ink jets, bubble jets, dot matrix, line printers, letter quality, pen plotters, others…

Boundary Value Analysis

- A refinement of equivalence partitioning that selects the edges or end-points of each partition for testing
  - Equivalence partitioning looks for bugs in the code that handles each equivalent class
  - Boundary values are members of equivalence classes that also look for bugs in the definitions of the edges
- Can only be used when the elements of the equivalence partition are ordered
- Non-functional boundaries (capacity, volume, etc.) can be used for non-functional testing, too

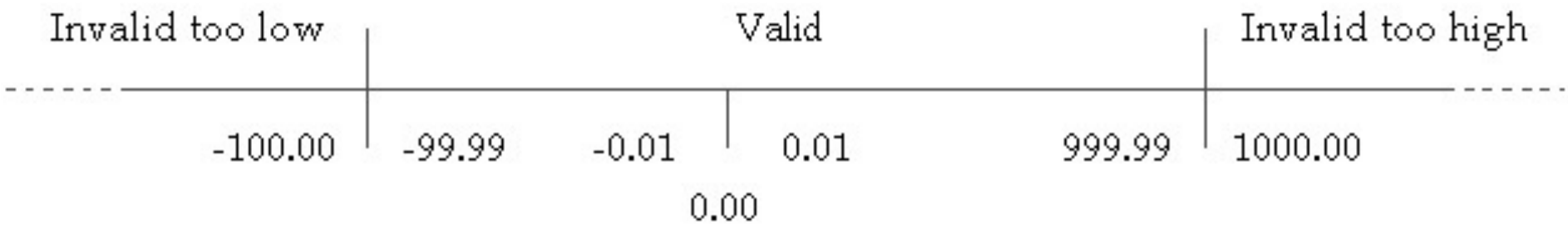
Integer

- “How many items would you like to order?”



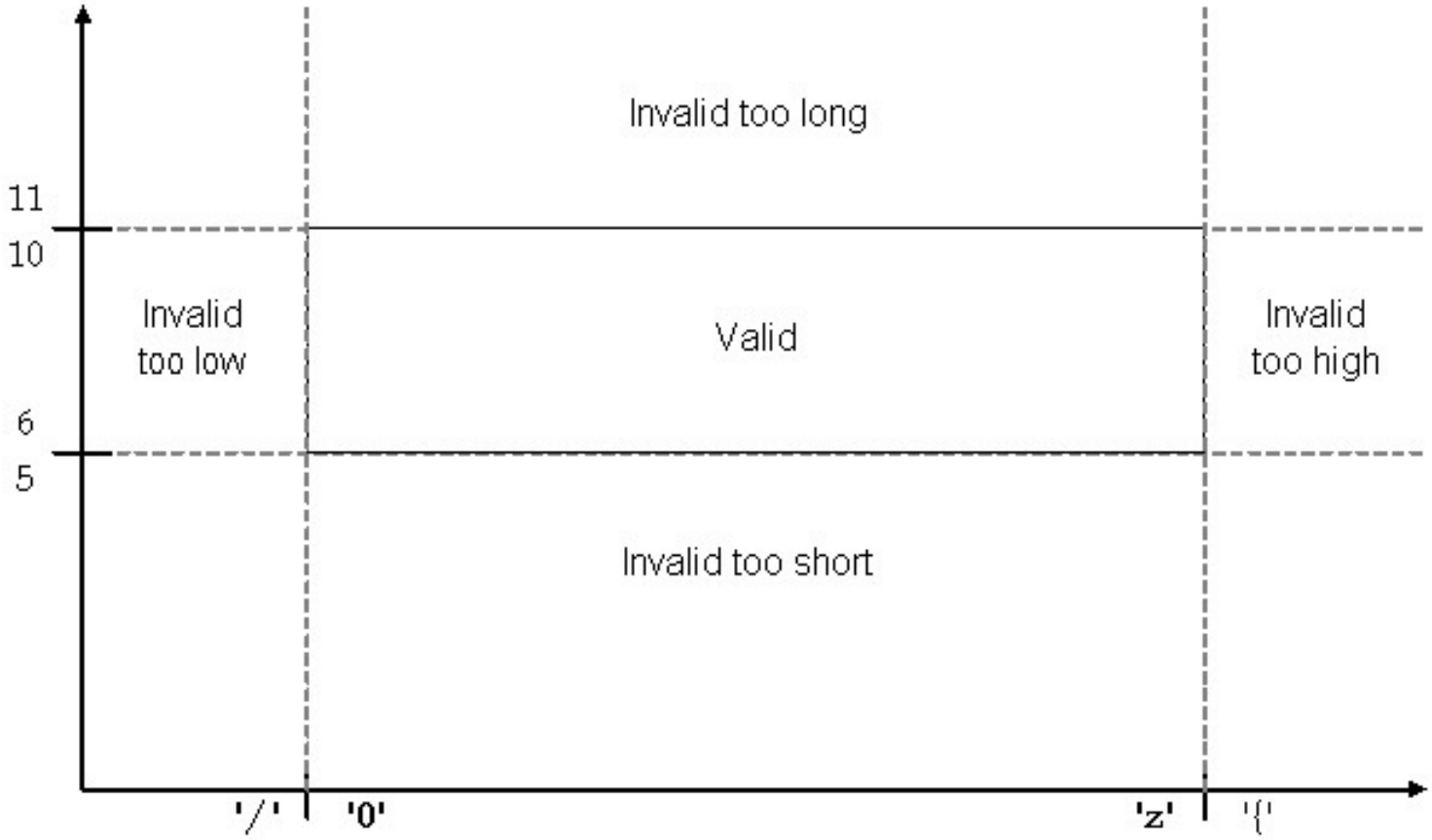
Real-Number

- “What is the average temperature in Duluth?”



Character and String

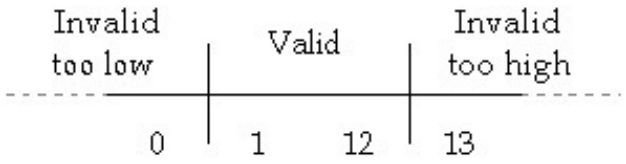
- “Password (6-10 character alphanumeric)”

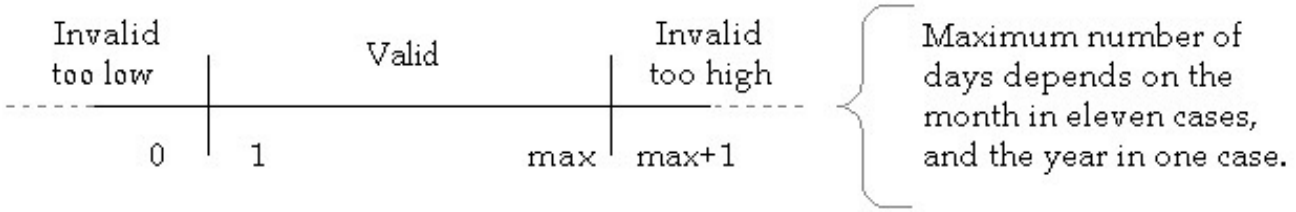


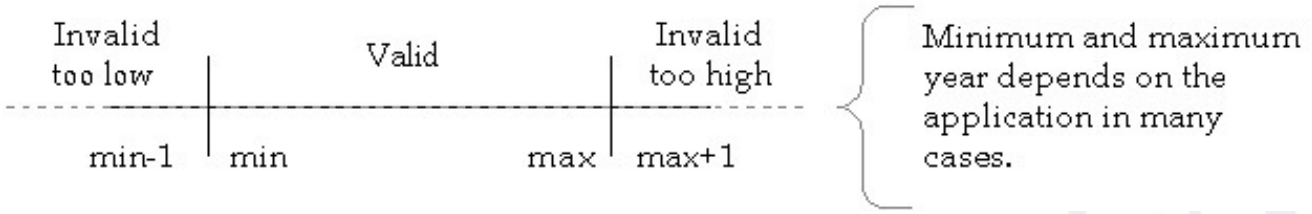
Date

- "Enter the departure date for your flight"

MM/DD/YY



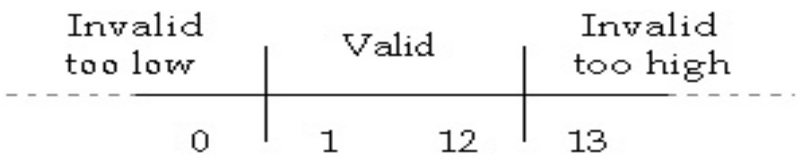


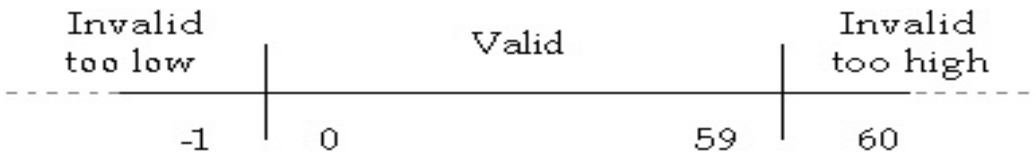


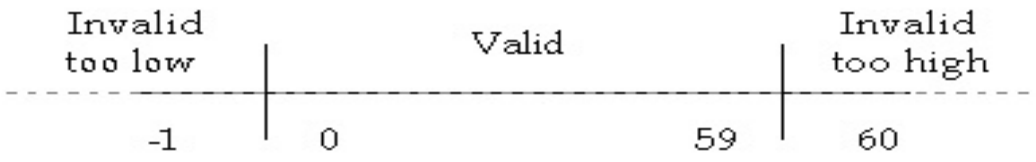
Time

- “Enter the departure time for your flight”

HH:MM:SS

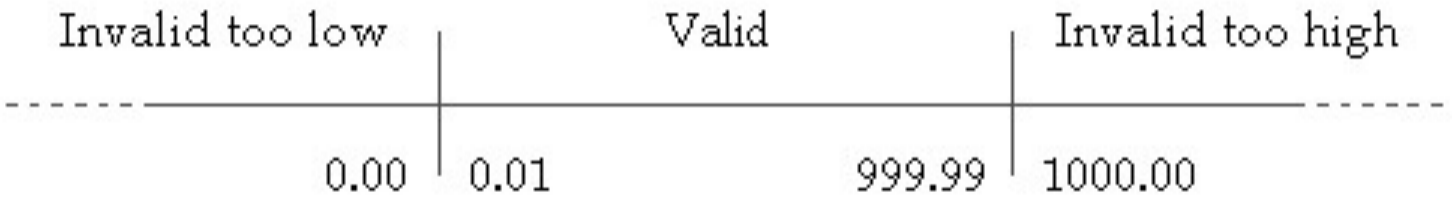






Currency

- “Enter a bid price (under $1000):”

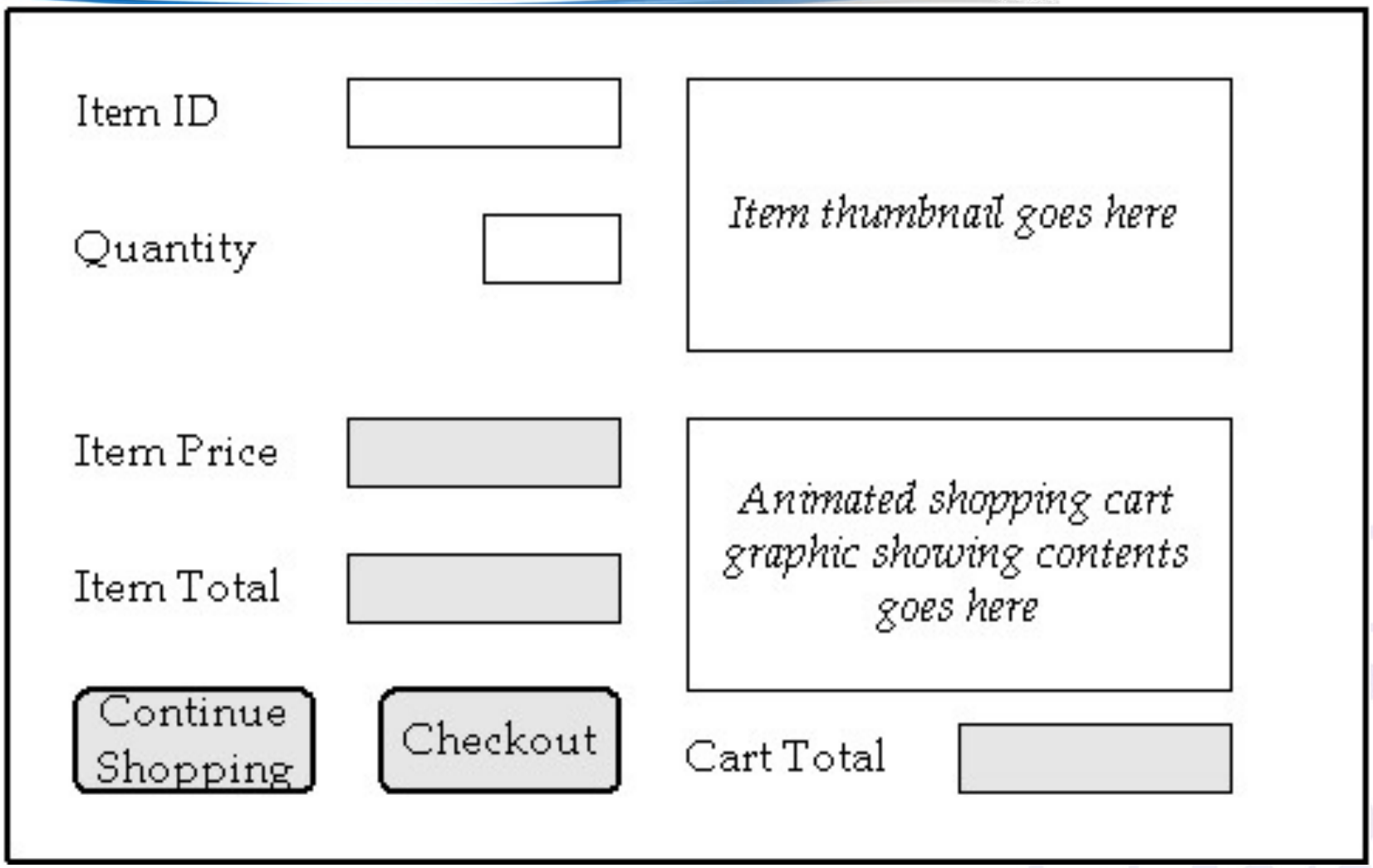


1. rounding! Decimals ! 99.999 != 99.994
2. Currency unit ! The smallest unit might be 20th of the currency (New Zealand). Korean’ won itself.
3. Historical currency 32nd of the currency for old English system.

A2.1.7: Classes and Boundaries

- You are testing an e-commerce site that sells Omninet knick-knacks like baseball caps, jackets, etc.
- Create functional tests for accepting orders
  - The system accepts a five-digit numeric item ID number from 00000 to 99999 (integer / string?)
  - Item IDs are sorted by price, with the cheapest items having the lower (close to 00000) item ID numbers and the most expensive items having the higher (close to 99999) item ID numbers
  - The system accepts a quantity to be ordered, from 1 to 99
  - If the user enters a previously-ordered item ID and a 0 quantity to be ordered, that item is removed from the shopping cart
  - The maximum total order is $999.99
- Use boundary value analysis and equivalence class partitioning to create tests in the following template
- Discuss.

Screen Prototype



a five-digit numeric item ID number from 00000 to 99999

- @ Input 1 ID Nr
  - Eps: Integer/String
  - Integer: BVs (6TCs)
  - 0,99999
  - -1, 100000
  - 1?
  - Max Int (32bits, 64bits?Buffer overflow?)

- String BVs
  - 00000, 99999
  - 0, 9
  - ‘,’; ‘/’,’\’
  - 000000, 999999
  - ‘00 0’,

a quantity to be ordered, from 1 to 99

- 1,99, 0,100, -1,Maxint, Min int

The maximum total order is $999.99

- @output 1

Total order:

0, 999.99

1000

-1

Max int

Min int

Bugs. Logics

- Remove items B.L.EP 1
  - Remove from an empty cart?
  - A normal remove?
- A Full Cart B.L.EP 2
  - Add into a full cart?

Test Template

Test Number	1	2	
Inputs, Actions			
Item ID	00000	00000	
Quantity	2	2	
Con. Shopping	Y	N	
Check out	N	Y	
Expected Results			
Item Price			
Cart Contents	00000*2	0	
Cart Total	00000Price * 2	0	
?			
?			
?			

上海市嘉定区曹安公路4800号，同济大学嘉定校区软件学院