Jordan Brown

Oral Exam Study Guide

3/15/2016

Contents

CSCI-5220 Research Methods

Measurement Types

Non-interval (qualitative) scales

- **Nominal**: unordered (e.g. N/S/E/W)
    - Example: Meyers-Briggs
- **Ordinal**: ordered (e.g. good/fair/poor)
    - Example: coin grading, Doc Habanero's scale

Interval-based scales (quantitative) scales

- **Interval**: don't start at 0 (e.g. Fahrenheit)
    - Example: + or – average goals by team/game
- **Ratio**: start at 0 (e.g. Kelvin)
    - Examples: anemometer (mph), rain gauge, Scoville Heat Unit Scale

Data Reliability Types

- **Interrater reliability** – extent to which different people evaluating a set of data give identical judgments
    - Example failure: in assessing a paper for publication, three reviewers return different opinions: accept with minor changes, accept with major changes, reject
- **Internal consistency reliability** – extent to which related items within an instrument give consistent results
    - **Instrument** – artifact for obtaining data (e.g. survey, questionnaire)
    - Example failure: students with inconsistent performance on an exam's questions (the exam is the instrument)
- **Equivalent forms reliability** – extent to which distinct versions of an instrument give consistent results
    - Example failure: instructor gives two different versions of an exam, version A and version B; students that took version A consistently earned better scores than students that took version B
- **Test-retest reliability** – extent to which the outcome of multiple trials agrees
    - Example failure: a medical study claims that students who chew gum during math tests earn higher grades than those who do not chew gum; a subsequent study found no such results

Data Validity Types

- **Face validity** – degree to which an instrument appears to measure a particular characteristic

- **Content validity** – extent to which a process yields a "typical" sample of the domain being measured
- **Criterion validity** – extent to which measurements from an instrument with other related measurements
    - e.g. shyness and introversion
- **Construct validity** – extent to which measurements of an immeasurable characteristic directly reflect the sought-after characteristic
    - e.g. motivation, creativity, racial bias

Qualitative Study Types

Ethnographic – immersion and study of a culture or organization

Case studies – a particular instance of something used or analyzed to demonstrate a thesis or principle

Phenomenological – focuses on people's perception of the world

Grounded theory – attempts to generate/discover a theory via systematic data collection and analysis

Content analyses – systematically determine key features of a body of literature

Research Publication Types

- White papers
    - Reports that present current developments in some area of computing (not official research)
    - Can be informative or persuasive
- Technical reports
    - Fast, no refereeing, quality varies wildly
    - Published in-house (universities, industrial labs), distributed via Internet
    - Often contain gritty details omitted in formal papers
- **Technical notes**
    - Special interest group periodicals (e.g. SIGPLAN, SIGCOMM)
    - Contain wide range of unreviewed, specialized papers
        - Some half-papers, papers that can't find a conference, bibliographies, reviews, etc.
    - Same variable quality as technical reports
- Conferences/symposia
    - Papers published as conference proceedings
    - Quality depends on conference, but varies less for better conferences
- Journals
    - Written for experts, by experts

    - Often best written, due to refereeing, but slow to process
- **Technical magazines**
    - Flagship magazines (e.g. Communications of ACM, IEEE Computer)
        - Articles of general interest with varying quality and appeal
        - Targeted for professionals, but not necessarily experts
    - Specialized magazines
- Theses
    - Most detailed papers
    - Quality depends on committee oversight, time pressures
    - Often published as technical reports
- **Annuals**
    - Surveys of published papers but with high quality of journals
    - Now seem outdated
- Bibliographies
    - Categorized sources with short synopses (often by-products of thesis research)

Writing Concepts

Comma splice – use of a comma *alone* to join two independent clauses; to correct a comma splice, use a coordinating conjunction or separate the independent clauses into two sentences

Independent clause – an idea that can stand alone as a sentence

Parallel construction – show that two or more ideas are equally equivalent by stating them in grammatically parallel form

Indicative abstract – explains only a paper's metadata (e.g. purpose, scope, and research methodology)

Informative abstract – explains a paper's material contents as well as its metadata

Compare and Contrast

Inferential vs. descriptive statistics

- Inferential statistics – make inferences about a population via a sample
- Descriptive statistics – summarize data numerically/graphically that characterizes an entire population

Random sampling vs. convenience sampling

- Random sampling – subjects truly chosen at random
- Convenience sampling – subjects chosen that are the easiest to access

Validity vs. reliability

- Validity – are the results true?
- Reliability – are the results repeatable?

Internal vs. external validity

- Internal validity – are the results valid for the initial scope of the study?
- External validity – can the results generalize to related scopes?

Proportional random stratified sampling vs. quota sampling

- Proportional random stratified sampling – select randomly from categorized populations according to population size, i.e. the larger the category population size, the more subjects chosen
- Quota sampling – sample proportionally from populations but no attempt to randomize individuals selected (typically, another form of convenience sampling)

Random stratified sampling vs. cluster sampling

- Random stratified sampling – select random individuals from categorized populations
- Cluster sampling – random sampling from arbitrary but relatively uniform subpopulations

Assent vs. **consent**

- Assent – consent from a child (7-17 years old) to participate in research
- Consent – consent from an adult (18+ years old) to participate in research
- In general, assent and consent required for children, consent required for adults

Type I experimental error vs. **Type II** experimental error

- **Type I** – claim that hypothesis is true, but it's actually false
- **Type II** – claim that hypothesis is false, but it's actually true

Belmont Report Principles

Belmont Report - U.S. guidelines for sponsored human subjects research

1. **Respect for persons**
    a. Individuals should be treated as autonomous agents
    b. Persons with diminished autonomy are entitled to protection
2. **Beneficence** (welfare of the research participant must be a goal)
    a. Human subjects should not be harmed
    b. Research should maximize benefits and minimize harm
3. **Justice**

a. The benefits and risks of research must be distributed fairly

CSCI-5230 Software Project Management

Selected Terms and Concepts

William Ury's 5 steps to negotiation

1. **Go to the balcony** – stop your reaction, the first barrier to real-world negotiation
2. **Step to their side** – listen carefully to the person's point of view
3. **Don't reject, reframe** – don't outright reject the person's point of view; instead, reframe it to an agreeable state for both parties
4. **Build a golden bridge** – make it easy as possible for the person to say yes
5. **Bring them to their senses, not their knees** – educate the person about the consequences of refusing to negotiate; do not threaten, simply state the facts

Best alternative to negotiated agreement (BATNA) is the starting point for effective negotiation and a fallback position for when negotiation fails. Also considered the greatest leverage a person has in negotiation.

Types of software prototypes

- Throwaway – use prototype to elicit requirements, then discard
- Evolutionary – prototype that evolves over time into a product
- **Operational** – hybrid approach; some prototypes discarded, some evolve

Process quality improvement techniques

- **Lean** – an approach to process quality improvement that focuses on eliminating or reducing waste
- **Six Sigma** – an approach to process quality improvement that focuses on eliminating or reducing variation in product quality
    - **Voice of the customer** sought as standard of quality
- **Capability Maturity Model (CMM)** – process quality can be gauged via a five level framework
    - Initial – organization has no formal processes
    - Repeatable – organization operates with a stable set of processes, but has not formally defined them
    - Defined – organization has formally defined its processes and uses *qualitative* measures to assess their operation
    - Managed – organization has formally defined its processes and uses *quantitative* measures to assess their operation
    - Optimizing – organization uses automatically generated feedback to improve its processes
- **Capability Maturity Model Integrated (CMMI)** – extension of CMM to 34 aspects of organizational processes

Lewis's 7 workplace motivators

- Fear
  - Good: creates a sense of urgency when needed
  - Bad: diminishes creativity and initiative
- Greed
  - Good: teammates can achieve a well-defined goal
  - Bad: turns teammates into competitors, motivation becomes more expensive over time
- Guilt
  - Good: regulates inappropriate behavior
  - Bad: turns adults into children
  - Overall, shouldn't be used in a workplace, period
- Avoiding boredom
  - Good: increases energy and enthusiasm
  - Bad: none
- Need for approval
  - Good: establishes yourself as a leader, inspires hard work, fosters creativity and initiative
  - Bad: creates perception that you're a phony/sycophant
- Exclusivity
  - Good: inspires loyalty, hard work, team trust
  - Bad: creates us vs. them attitude, can antagonize rest of company
- Need for self-esteem
  - Good: most important beneath approval and exclusivity
  - Bad: none
  - The most effective of the 7 basic motivators but can't be accessed directly

Lewis's five styles of decision making

- Authoritarian – fast, cheap, and appropriate for crises
- Consensus – slow, expensive, messy, but high buy-in
- Consultation – mainstay of effective leadership; balances speed, cost, buy-in while maximizing quality
- Delegation – always delegate to someone with more expertise than you; be willing to live with a bad decision
- Democracy – use only as a fallback to consensus when authority to decide is lacking

According to Lewis, empathy is the single most important element of an effective communication.

Types of procurement contracts

- Fixed-price – buyer/seller agree on fixed price
- Cost reimbursable – buyer pays seller cost, plus fee
- Time and material – aspects of both

According to Drucker:

- An organization suffering from *excess meetings* has poor structure.
- An organization suffering from *recurring crises* has a lack of system and/or foresight.
- *Time* is an enterprise's scarcest and most valuable resource.
- A knowledge worker's main output is *results*.

Goal alignment is the single most important factor in team formation.

Scrum is a software development methodology with user-story-driven, value-prioritized, time-boxed iterations.

Spiral is a software development methodology focused on risk management.

Risk management strategies

- For threats
    1. **Avoid** – change plan to completely eliminate threat
    2. **Mitigate** – change plan to reduce a threat's probability of occurrence and/or probable impacts
    3. **Transfer** – shift some or all of impact to another party
    4. **Accept** – do nothing special to address the threat
- For opportunities
    1. **Exploit** – change plan to seize opportunity
    2. **Enhance** – change plan to increase an opportunity's probability of occurrence and/or probable benefits
    3. **Share** – allocate some or all benefits to another party
    4. **Accept** – do nothing special to exploit the opportunity

Plan-do-check-act (PDCA) is the heart of all iterative methods for project management.

Function points are a program-size-based metric for estimating project effort.

Feature points account for program inputs, outputs, files, interfaces, queries, and algorithmic complexity.

Cockburn's two key goals of software development

1. Deliver useful, working software
2. Set up for the next game, which may be to alter or replace the current system or create a neighboring system

Software quality is value to a person. More quality for one person may be less quality for another.

According to Richard Gabriel, *worse* is sometimes better if interface design is sacrificed for simplicity of implementation and first to market.

Active listening is a communication strategy that emphasizes restatement of a speaker's points along with requests for clarification.

Opportunity cost – the consequences of deciding to do one project instead of another

How to sequence tasks

1. Identify dependencies
    a. **Mandatory** – unavoidable; inherent in nature of work
    b. **Discretionary** – dependencies that should be observed but are optional
        i. May reflect best practice or special considerations
    c. **External** – non-project tasks
2. Assign types to dependencies
    a. **Finish-start**, e.g. can't start training until after software is installed
    b. **Start-start**, e.g. start training users and creating accounts concurrently
    c. **Finish-finish**, e.g. quality control can't finish until production ends
    d. **Start-finish**, e.g. can't finish a production cycle until transport leases begin

Three-point strategy for project estimation – weighted average of pessimistic, average and optimistic estimates

Working environment is the single most important determinant of a software organization's productivity.

Gantt Chart – graphical depiction of task start/end dates and their dependencies

Resource levelling – adjusting task start/end dates according to resource constraints (e.g. resources work 40 hours per week)

Compare and Contrast

Crashing vs. fast-tracking

- Crashing – add more resources to a schedule to attempt to speed it up
- Fast-tracking – complete serial tasks in parallel for at least a portion of their duration

Process WBS vs. product WBS vs. hybrid WBS

- Process WBS – decomposes a process into small, manageable tasks
- Product WBS – decomposes a product into its constituent parts
- Hybrid WBS – contains alternating layers of processes and products

Total float vs. free float

- **Total float** – max delay in a task's start relative to *project delivery* without impacting the schedule
- **Free float** – max delay in a task's start relative to *immediate successor activities* without impacting the schedule

Defined vs. managed levels of the CMM

- Defined – uses qualitative measures to assess operation of processes
- Managed – uses quantitative measures to assess operation of processes

Reversioning vs. rollback

- **Reversioning** – ability to recover originals of changed artifacts relative to the last commit (i.e. git checkout .)
- **Rollback** – ability to recover any version of any artifact (i.e. git checkout <commit> <file>)

Critical path scheduling vs. critical chain scheduling

- **Critical path scheduling** – identify path through schedule that takes most time (assuming unbound resources), then allocate resources along this path
- **Critical chain scheduling** – same as critical path, except that it focuses on the longest path *after* resource allocation

Parametric vs. analogy-based strategies for project estimation

- **Parametric strategies** use input data and statistical models to deliver estimates.
- **Analogy-based strategies** use data gathered from similar projects to deliver estimates.

CVCSes vs. DVCSes

- CVCS – one-tier check-in system with centralized repository
- DVCS – two-tier check-in system with distributed repository

Leadership vs. management

- **Leadership** – discuss the results you want, people achieve them
- **Management** – directing people to do specific activities

V- and W- models of software development

- V-model – postpones testing until after product implementation
- W-model – includes testing throughout development and adds review-based testing

McConnell's cones and clouds of uncertainty

- **Cones** – best-case upper/lower boundaries of project estimates during any phase of the software lifecycle

- **Clouds** – estimate boundaries of projects that are not conducted in a way that fully reduces variability (i.e. mismanagement)

Know X and Y axes, intents, and at least one type of person that fits into each quadrant

Discussion Questions

In what ways do Ury's strategy for preparing for a negotiation resemble the three-point strategy for project estimation?

- **What could you live with?** – pessimistic estimate
- **What would you be content with?** – average estimate
- **What do you aspire to?** – optimistic estimate

Which of Ury's strategies would you use to counter the following estimation games?

- Double dummy spit – Use one of Ury's five strategies for deflecting attacks, e.g. reframe past wrongs to future remedies
- Spanish inquisition – Use one of Ury's four strategies for going to the balcony, e.g. go back to your desk and decide a feasible estimate
- X-plus – Use one of Ury's three strategies for getting around stone walls, e.g. take the stone wall seriously, but test it – did a high-up executive really request this deadline?

Which of Lewis's five decision making styles would Drucker favor?

- Consultation since it allows for disagreement and saves time.

Project Math

Budget at Completion (BAC) – total budget allocated for tasks

Planned Value (PV) – scheduled cost of work for a given time frame

Earned Value (EV) – money earned from completed work in a given time

Actual Cost (AC) – actual amount of money spent to date

CPI – Earned Value / Actual Cost

TCPI – (BAC – EV) / (BAC – AC)

CSCI-5200 Software Systems Engineering

Test 1

What does it mean for a software process to be agile?

- Iterative software lifecycle instead of linear
- Working software valued over comprehensive documentation
- People and interactions valued over processes and tools
- Late requirements changes are welcome

Why study software systems engineering?

- Learn to balance cost, schedule, and quality of software projects.
- Enables effective teamwork.
- Avoid common hazards that can cause a software project to fail.

Why choose an agile process model?

- Unstable requirements
- Small, experienced development team
- Customer wants to be involved in project
- Strict deadlines, not enough time to document

Differentiate between incremental and iterative development.

- Incremental – develop product one stable, complete feature at a time until complete
- Iterative – build and refine a feature (or entire product) repeatedly until complete

Explain test-first development and refactoring w.r.t. extreme programming.

- Test-first development – write automated test cases before production code; test cases become a requirements document
- Refactoring
    - Refactoring is the process of modifying a software system's internal structure without affecting its external behavior
    - With respect to XP,
        - make concentrated efforts to remove redundancy, eliminate unused functionality, and rejuvenate obsolete designs
        - keep design simple and avoid needless clutter

Why are technical reviews considered some of the most effective software quality assurance mechanisms?

- Developers learn others' code and obtain a better understanding of the software system
- Technical reviews expose bugs outside of automated testing environments
- Developers can offer suggestions to refactor code

Why is the cost of making requirements changes more expensive as time passes?

- As a project grows older, requirements changes may break existing code and force developers into reworking its design and implementation.

Name and explain all of the generic software development phases.

1. Problem identification and definition – identify the software's business value
    a. Give project a name
    b. Identify sponsors
    c. Identify business need
    d. Identify high level functionalities
    e. Identify the expected value, i.e. feasibility analysis
    f. Identify any special issues
2. Software specification
    a. Analyze the client's circumstances as precisely as possible
    b. Determine the client's software needs
3. Software design and implementation
    a. Architecture, detailed, interface design
    b. Design mode, document produced
    c. Write code, unit tests, integration tests
4. Software validation – ensure software meets specifications
    a. System testing
    b. Acceptance testing
5. Software deployment and evolution – delivery and maintenance
    a. There will always be change – corrective, perfective, or adaptive
    b. More acceptance testing
    c. Regression testing

Why does adding more people to a late project not complete it sooner?

- New developers must be trained before they can become productive

Test 2

Differentiate between functional and non-functional requirements.

- Functional requirements describe how a system should react to particular inputs or behave under particular circumstances.
- Non-functional requirements describe constraints under which a system must operate.

Name 3 process areas in CMMI for development.

- Product integration
- Requirements development
- Requirements management

Name 5 maturity levels from the staged CMMI representation.

1. Initial
2. Managed
3. Defined
4. Quantitatively managed
5. Optimizing

Name the 6 capability levels.

- Incomplete
- Performed
- Managed
- Defined
- Quantitatively managed
- Optimizing

What is a requirement?

- A requirement is a precise statement describing how a system should react to particular inputs or what constraints should be placed on a system.

What are the disadvantages of obtaining CMMI certification?

- CMMI requires a significant amount of effort and documentation to complete, which would be unsuitable for a small development team using agile methods
- Continuous CMMI representation would not assign a maturity level to the company
- CMMI certification guarantees that a company is using formal methods but does not guarantee that the company is delivering quality products and services

What are the 3 main inputs of the requirements engineering process?

- Stakeholder needs
- Regulations
- Domain information

What is the difference between capability level and maturity level?

- Capability levels are assigned to process areas
- Maturity levels are assigned to companies

What items should be included in a problem identification and definition document?

- Project name
- Sponsors
- Business need
- High level functionalities
- Expected value, i.e. feasibility analysis
- Special issues

Name 3 of the software process assessment approaches discussed in-class.

- CMMI
- SAMPI
- CBA IPI

Test 3

What is a use case description's normal flow?

- A generalized description of a user's actions in a use case.

What is a scenario?

- Scenarios are actual examples of how a system is used, i.e. an instantiation of a use case description.
- Same as the relationship between a class and an object.

What is an operation w.r.t. a UML class?

- An operation is an action that a class can perform
- Syntax: visibility name (parameter-list) : return-type { property-string }

What is a system model?

- A diagram that identifies system actors, high-level functionalities, inputs, and outputs.

What are actors, data elements, and actions w.r.t. functional requirements?

- **Actors** – people or external systems that interact with the system being described
- *Data elements* – information being accessed by actors
- Actions – describe how actors interact with a system
- Ex. "The system shall allow **library patrons** *to search* for a book by title, author, or ISBN"

What do the following multiplicities mean?

- 1 – exactly one
- 0..1 – zero or one
- 1..* - 1 or many
- * - zero or many

Why are "will", "should", and "may" problematic when writing requirements?

- "shall" is government best practice
- "will", "should", and "may" do not express a requirement

What is it important to build a prototype rapidly when eliciting requirements?

- Prototypes should be built rapidly to receive continuous feedback from customers over successive iterations.

Test 4

What is a data model, functional model, and behavioral model?

- All three comprise an **analysis model**.
- Data model – depicts data elements and their relationships
- Functional model – depicts functions that transform data
- Behavioral model – depicts system behavior

Why should a requirements engineer create an analysis model?

- To identify producers and consumers of data
- To evaluate data flow and content
- To define and elaborate all software functions that transform data
- To understand how a system should respond to external stimuli
- To establish characteristics of a system's interface
- To uncover additional design constraints

What is an object-oriented analysis?

- Object-oriented analysis produces a definition of a software system via the object-oriented paradigm
    - Data modeling: class diagrams
    - Functional modeling: interaction diagrams, activity diagrams
    - Behavioral modeling: state-machine diagrams, activity diagrams

What are the desirable characteristics of a requirement?

- Consistent – can be satisfied w.r.t. the remainder of a system's requirements
    - i.e. two requirements are inconsistent if they cannot be satisfied simultaneously
- Correct – meets the client's needs
- Complete – fully defined; i.e. it contains
    - One or more actors
    - An action performed
    - An objective of the action
    - A trigger
- Unambiguous – can be interpreted in only one way
- Traceable – can be uniquely identified; source can be easily identified; related requirements can be easily found
- Testable – can be tested via test cases

Test 5

What is a requirements specification?

- A precise statement of a requirement a system must satisfy

What is the major problem with using natural language for specifications?

- Natural language is inherently ambiguous
- It is difficult to use natural language to identify and separate system elements

What semi-formal methods can be used for requirements specification?

- Backus-Naur form (BNF) – context-free grammar notation useful for specifying input requirements
- Pseudocode – imperative approach to specifying requirements; uses natural language in a structured way
- Decision table – truth table depicting system's behavior under combinations of different conditions
- State-machine diagrams, activity diagrams, data flow diagrams, class diagrams

What are the advantages of using formal methods?

- Formal methods remove ambiguity

- Formal methods encourage greater rigor in early phases of software engineering
- Formal methods support completeness, correctness, and consistency checks of system requirements

What are the disadvantages of using formal methods?

- It is difficult to use formal methods to represent behavioral aspects of a system
- Some requirements can only be obtained through prototyping
- Formal methods do not solve the problem of how to construct requirements
- Formal methods lack adequate tool support

What is the difference between a state and an operation?

- State – collection of data that a system can modify
- Operation – action that a system can perform, which may change the system's state

What is an invariant?

- An invariant defines what is guaranteed not to change

What is a pre-condition?

- A pre-condition defines the circumstances in which a particular operation is valid

What is a post-condition?

- A post-condition defines what happens when an operation has completed its action

Test 6

What is OCL?

- OCL is a formal (but easy to use) declarative language for describing system constraints
    - Unambiguous; natural language instead of mathematical notation
    - Not a programming language! No control flow or side effects (i.e. state changes)
- OCL cannot be used independent of UML
- OCL is standardized across organizations

OCL describes expressions and constraints on UML models

- An expression is a specification of a value
- A constraint is a restriction on a set of values in an object-oriented model
    - Constraints must be predicates

What is the difference between requirements analysis and requirements V & V?

- Requirements analysis – have we got the right requirements, i.e. does the system meet stakeholder needs?
- Requirements V & V – have we got the specified requirements right, i.e. do the specifications agree with what was produced during requirements analysis?

Problems that may be discovered through requirements V & V:

- Lack of conformance to quality standards
- Ambiguous requirements specifications
- Errors in system models or the problem to be solved
- Requirements conflicts which were not detected during the analysis process

Pre-review checking – person familiar with organization's requirements standards, but not involved in the project, reviews requirements document

The pre-review checking process

1. Receive requirements specification document
2. Check document by…
    a. Checking document completeness
    b. Checking document against standards
    c. Running automated checkers
3. Issue a problem report
4. Repeat from #2 until document is up to organization standards

Requirements inspection (i.e. review) process

- Complete pre-review checking
- Distribute the document
- Prepare for review (individuals look for problems)
- Hold review meeting
- Decide follow-up actions (do not correct errors during meeting)
- Revise the document

Why is it beneficial to develop an early user manual?

- Developing a user manual forces a detailed requirements analysis, thus revealing problems within a requirements document

Why is a traceability matrix beneficial?

- Ensures that requirements are not lost during the requirements engineering phase

- Ensures that all requirements defined for the system are tested (personally, I've seen this used as a basis for customer acceptance testing)

CSCI-5300 Software Design

Quiz 1: Basic Python

What value is used for the result of an arithmetic operation exceeding sys.float_info.max? What about sys.int_info.max?

- *inf* is used to represent values exceeding sys.float_info.max
- Integers do not have a maximum since Python supports arbitrary precision integers

Explain the concepts of mutability and immutability and how these affect Python's data structures.

- An immutable object cannot be modified and can be used as a key
- A mutable object can be modified and cannot be used as a key

Explain the function of a Python for loop's else clause. When can that clause be bypassed?

- A for loop *else* clause executes after a for loop finishes a traversal of a collection
- A for loop *else* clause can be bypassed by breaking the loop before it completes

What value does a Python function return that lacks a return statement?

- *None*

Name two statements that instantiate Python objects, and one statement that destructs them.

- instance = MyClass()
- raise RuntimeError("")
- del(instance)

What Python data type is used to realize a group of varargs parameters? A group of keyword parameters?

- A list is used for varargs parameters
- A dict is used for keyword parameters

What are the two primary differences between eval and exec?

- Eval returns the result of the expression executed
- Eval only accepts a single expression
- Exec does not return a value
- Exec can accept entire code blocks

How do generators differ from ordinary functions in format and use?

- Generators use the *yield* keyword to return elements of a collection one at a time
- Repeated calls of a generator will pick up at the generator's last executed statement
- Generators can be finite or infinite

Do Python classes support data hiding?

- No; data hiding is a convention in Python rather than an enforced constraint

How do Python static methods and instance methods differ?

- Instance methods can access an object's state via the *self* parameter
- Static methods can access class attributes but cannot access an object's state

How do Python class variables and instance variables differ?

- Instance variables belong to instances of a class; they are declared in the class's __init__ method
- Class variables belong to a class; they are declared in the immediate body of the class's definition

Explain the difference between shallow copying and deep copying.

- Shallow copy creates a new name to alias an object
- Deep copy creates a new object with data identical to the old object

Quiz 2: Cohesion and Coupling

Cohesion – relatedness of module elements

Constantine's hierarchy of functional cohesion

- Functional – 1 well-defined task
- Sequential – 1 sequence of subtasks for a well-defined task
- Communicational – 1 set of subtasks for a well-defined task
- Procedural – 1 sequence of subtasks across 2 or more sequential tasks
- Temporal – 1 set of subtasks across 2 or more concurrent tasks
- Logical – multiple, superficially related tasks
- Coincidental – multiple, unrelated tasks

Coupling – interreliance among modules

Constantine's hierarchy of coupling

- Sharing inside function
    - Null – no shared data (i.e. no parameters)
    - Data – sharing of individual data elements via a function's interface

        - Stamp – sharing of data structures via a function's interface, including items that need not be shared
            - Hybrid – special instance of stamp coupling where multiple meanings are attached to a single datum
            - Example – values [-256, 1] are error codes, [0, 255] are ASCII chars
            - Introduced in the 70's for efficiency; avoid
        - Control – sharing of parameters or return values that direct behavior
            - Avoid Boolean / flag parameters and error codes
- Sharing outside function
    - External – sharing via persistent public data stores (databases, files, environment variables)
    - Common – sharing via common areas
        - Common areas = blocks of shared data explicitly imported into program modules
        - Example - mixins
    - Global – sharing via global variables
    - Content – sharing via internal workings of other functions
        - Example – jumping into middle of another function's code

Quiz 3: Software Architecture

What are the five essential characteristics of an object (IBLIS)?

- Identity (i.e. a referent)
- Behaviors
- Lifetime
- Interface
- State

What problem do mutual (i.e. circular) dependencies create for software development?

- Mutual dependencies make testing more difficulty – which module should be tested first?

Define module fan-out. What problem, if any, does low fan-out pose for software development?

- Module fan-out is the number of child modules that inherit from a single module.
- Low fan-out poses no issue. High fan-out is troublesome due to the miller limit—too much information to keep in short-term memory at once.

What 3 factors affect software project quality?

- Cost, schedule and quality. Favoring any two may negatively impact the third.

What is a viewpoint? A view?

- **Viewpoint** – set of related concerns shared by stakeholders
- **View** – application of a viewpoint to a system

What 3 viewtypes should characterize a software architecture?

- **Component-and-connector** – depict a system's runtime behavior
- **Module** – depict a system's static composition
- **Allocation** – maps software units to deployment environments

Explain the difference between tiered computing and client-server architectural styles.

- In client-server, the server is the only back-end service responding to requests.
- In tiered computing, the server acts as a client to other back-end services (e.g. a database) to respond to requests.

Explain the difference between the client-server and broker architecture styles.

- In broker architecture
  - servers register with a broker
  - clients query the broker to receive a list of available servers
  - clients begin exchanging messages with the servers directly
- Client-server does not support the concept of a broker and assumes clients already know the addresses of servers they wish to contact.

Explain the difference between the broker and proxy architecture styles.

- In broker architecture, clients exchange messages with servers directly after retrieving server addresses from a broker.
- In proxy architecture, clients exchange messages with servers indirectly through a proxy.

Explain the difference between the **repository** and **blackboard** architecture styles.

- **In repository architecture**, system elements communicate via a passive persistent store.
- **In blackboard architecture**, system elements communicate via an active persistent store that triggers events in the elements.

Explain the difference between the repository and publish-subscribe architectural styles.

- In publish-subscribe architecture, data sent to the server by producers is persisted just long enough to be read by consumers.
- In repository architecture, data is persisted to a data store indefinitely.

Name and describe the three elements of the MVC architecture style.

- Model – persistent store for domain objects
- Views – presentation of the model's content
- Controllers – accept event-driven commands to query the model and return views

Explain how functional and non-functional requirements differ.

- Functional – specifies a system's output as a function of input
- Non-functional – specifies how a system must run

How does **system response** differ from **system throughput**?

- **Response** = total amount of time taken for a system to respond to a request
- **Throughput** = number of responses delivered by a system within a certain timeframe

Describe the strategy Dr. Barrett recommends for managing non-functional requirements.

- Model NFRs via class diagrams
- Compose base classes to represent soft (i.e. high-level) goals
- Compose subclasses to refine soft goals to specific goals, then to measurable statements

Define the **RPC** and **REST** architectures. State one key difference between them.

- **RPC** – remote procedural call
- **REST** – representational state transfer
- RPC directs requests to procedures; REST directs requests to resources

Quiz 4: UML

How do **aggregation** and **composition** differ?

- **Aggregation** (white diamond)
    - Aggregate is permitted to share part with other classes
    - Aggregate can exist independently of its part
- **Composition** (black diamond)
    - Aggregate is not permitted to share part with other classes
    - Aggregate cannot exist independently of its part

What is an **association class**?

- An association class captures characteristics of an association between two classes (think intersection table in ERD).

What is a **template**?

- A **template** allows a class to be parameterized, e.g. for collection classes.

What is a **stereotype**? (<< >>)

- A **stereotype** is an annotation that offers more information on class responsibility.

Compare and contrast state diagrams and activity diagrams.

- Activity diagrams can depict parallel computation, state diagrams cannot
- Both have start/end states

Compare and contrast package diagrams and deployment diagrams.

- Deployment diagrams depict physical environments for running software; package diagrams do not
- Both diagrams depict high-level representations of software components

Quiz 5: Class Design Strategies

What is **RTTI**?

- **RTTI** = run-time type inference, a coding scenario where an object's possible types are checked to determine the proper action to take

What is **dependency inversion**?

- **Dependency inversion** is the process of refactoring a class to use an interface representation of a dependency rather than the dependency itself

RTTI and dependency inversion attempt to address a common problem. What is it?

- Deciding how to act according to an object's type

Explain which of these two strategies is less maintainable and why.

- RTTI is less maintainable because
    - if-else chains are verbose and difficult to read
    - if-else chains must be updated if a module's types are changed

What specific maintainability-related concern do the **Laws of Demeter** attempt to address?

- The Laws of Demeter attempt to eliminate designs that "reach through" one object to another (i.e. tight coupling)

How do the Laws of Demeter attempt to address this concern?

- A method m(p1, p2..pm) of an object O with components c1, c2..cn should limit itself to invoking
    - m itself
    - methods of any objects m instantiates

    - methods of p1, p2..pm
    - methods of c1, c2..cn

How do the **strong** and **weak Laws of Demeter** differ?

- Strong and weak Laws of Demeter have different rules on inheritance
- Strong – c1, c2..cn limited to member data in O
- Weak – c1, c2..cn can include member data in O's base class

What is **Liskov subtyping**?

- **Liskov subtyping** defines sub-classing based on substitutability
- Idea: treat B as a subtype of A iff B can be used wherever A can be used
- Classes: every attribute a of A must have a corresponding attribute b in B such that b is a subtype of a
    - Scalars: for scalar attribute b to be a subtype of a, b's range must be a subrange of a's range
    - Sequence: for a sequence b to be a subtype of a = (x1, x2..xk), b must have at least k elements
    - Functions: rules are complex and aren't worth typing, but the key idea is you can't plug b in for a if b rejects any arguments that a would accept

Define model inheritance.

- Model inheritance reflects is-a relationships between abstractions in a model.

Define variation inheritance.

- Variation inheritance describes a class in terms of differences with another class.

Define software inheritance.

- Expresses software relationships with no obvious real-world counterpart (e.g. types of sockets)

Quiz 6: Python Class Concepts

What data sets are instance methods, static methods, and class methods supposed to operate on?

- Instance methods should operate on instance variables via the "self" argument.
- Class methods should operate on class variables via the "myclass" argument.
- Static methods should operate on class variables via a class name.

How does Python handle the classic "**diamond**" **problem** introduced by multiple inheritance? Is it possible to alter this behavior?

- When the diamond problem is present, Python prioritizes inherited names from left to right. For example, if class D inherits B and C, which both inherit from A, and class D's definition is *class D (B, C)*, B's members would take priority.

- This behavior could be overridden by calling C.<name> and manually passing in an instance of D as self.

How do Python's __getattr__ and __getattribute__ differ?

- __getattr__ - called when a client attempts to access a non-existing variables within an object.
- __getattribute__ - called when a client attempts to read *any* variable within an object.

Define **latent (i.e. duck) typing**.

- Latent typing is a language design choice where names are used as a substitute for typed attributes, allowing objects to imitate types they do not inherit.
- In other words, latent typing occurs when an object contains identical names to a type to imitate inheritance.

State a circumstance where __repr__ can't fully capture an object's state.

- __repr__ won't work as intended if an object is dependent on values stored in one or more class variables, assuming these class variables are not initialized via a constructor
- (wasn't in the quiz, but I think __repr__ would also fail to capture an object's methods)

Quiz 7: Design Patterns

What are the 3 categories of design patterns?

- Creational – focus on creating objects while hiding creation logic
- Structural – focus on class and object composition
- Behavioral – focus on communication between objects

What pattern supports the execution of a sequence of handlers, one at a time, that continues until one handler responds to a request?

- Chain of Responsibility

What pattern breaks the construction of an object into parts and decouples the concept of a part from the logic that makes it?

- Builder

What pattern allows an object to substitute for another object, fielding requests on the second object's behalf?

- Proxy

What pattern uses an existing object as a basis for instantiating an object?

- Prototype

Briefly describe two differences between the **Composite** and **Decorator** patterns.

- Composite – ideal for constructing hierarchies of objects that act as a single, composite object; does not enforce a strict ordering of its objects
- Decorator – ideal for constructing a linked list of objects that act as a single, embellished object; enforces a strict ordering of its objects

Describe one similarity and one difference between the **Decorator** and **Chain of Responsibility** patterns.

- Similarity – both involve the construction of a linked list of objects
- Difference – chain of responsibility is used to have requests serviced by a dynamically varying set of handlers; decorator is used to embellish objects with additional functionality

Describe one similarity and one difference between the **Iterator** and **Visitor** patterns.

- Similarity – both enact a traversal of a data structure
- Difference – iterator exposes objects to the caller; visitor runs code provided by the caller, but does not expose objects to the caller

Describe the **Bridge** pattern.

- The bridge pattern is used to separate an abstraction hierarchy from an implementation hierarchy.
- “Decompose the component's interface and implementation into orthogonal class hierarchies. The interface class contains a pointer to the abstract implementation class”

Describe the **Adapter** pattern.

- The adapter pattern allows an object’s interface to be accessible via another interface. It’s typically used to integrate a module without modifying its source code.

Describe the **Observer** pattern.

- A subject maintains a list of dependents and notifies them of any state changes.

Describe the **Singleton** pattern.

- The singleton pattern restricts the instantiation of a class to one object.

What’s the difference between the **Factory** and **Abstract Factory** patterns?

- Factory focuses on returning types of a single product.
- Abstract factory focuses on returning a family of products.

Describe the **Flyweight** pattern.

- The flyweight pattern is used share immutable state across a large number of objects, usually to save memory.
- Extrinsic state – mutable state unique to individual objects
- Intrinsic state – immutable state that should be shared

CSCI-5250 Database Design

Midterm Concepts

What is an entity-relationship diagram?

- A visual representations of entities, their attributes, and the relationships between entities.

What is an entity? Attribute? Relationship?

- Entities – people, places, objects, things, events, or concepts about which data is being collected
- Attributes – properties of entities
- Relationships – rules governing associations between entities

What is **modality**? **Cardinality**?

- Both terms describe relationship constraints.
- **Modality** – specifies whether an instance of an entity can exist without a related instance of the other entity
- **Cardinality** – specifies how many instances of one entity are associated with an instance of the other entity

What is an associative entity?

- An associative entity models intersection data between two entities.

What is the difference between a strong entity and a weak entity?

- Strong entities can exist independently
- Weak entities cannot exist without an "owner" entity
    - Records cannot be uniquely identified with weak entity's attributes alone
    - Weak entity must use a foreign key in conjunction with its own attributes as a primary key

What is an EER?

- An EER (enhanced ERD) adds sub/superclasses, specialization/generalization, and union/category type to traditional ERD notation.

What is a logical model?

- A logical model convert's an ERD into a set of 3NF relations.

What is normalization?

- The formal process of creating well-structured relations that are relatively free from insertion, update, and delete anomalies while having minimum data redundancy.

What are the normal forms? What is the industry-standard normal form?

- **Unnormalized** – relation contains multi-valued attributes
- **1st normal form** – each attribute is atomic, no multi-valued attributes
- **2nd normal form** – first normal form plus removal of partial dependencies
    - Partial dependency – non-key attributes are dependent on part of a primary key
    - Solution – decompose relation into two relations
- **3rd normal form (industry standard)** – second normal form plus removal of transitive dependencies
    - Transitive dependency – non-key attributes are dependent on other non-key attributes
    - Solution – decompose relation into multiple relations

Why are the pros and cons of denormalization?

- Pros
    - Faster performance by avoiding joins
    - In data warehousing, denormalized tables may be easier to understand for end users
- Cons
    - Possible introduction of data anomalies

What is a transaction?

- A transaction is a logical unit of work in a database.

What are the ACID properties of a transaction?

- Atomic – a transaction is a complete unit of work, performed entirely or not at all
- Consistent – a transaction's correct execution takes a database from one consistent state to another
- Isolated – a transaction must appear to execute by itself without interference from other transactions
- Durable – a transaction's changes to a database must be permanent

What are the 3 possible states of a transaction?

- START – marks transaction beginning
- COMMIT – marks successful completion of a transaction; all changes saved to the database
- ROLLBACK – unsuccessful end of a transaction; all changes made are undone

What is concurrency?

- Simultaneous read/write operations on a single database by multiple users

What problems can happen during two concurrent operations?

- Interleaving transactions may generate incorrect or inconsistent results
- Lost updates – results of one transaction overwrite results of the other
- Dirty reads – one transaction reads data that is being modified by the other
- Non-repeatable reads – one transaction reads data, then reads it again and finds the data has been modified or deleted
- Phantom reads – one transaction reads rows and finds newly inserted rows that meet filter criteria

What is a **schedule**? What are the **3 types of schedules**?

- A schedule is a sequential ordering of operations within multiple transactions.
- **Serial schedule** – operations within transactions are executed sequentially; no concurrency
- **Serializable schedule** – operations within transactions are executed concurrently, but the transactions produce a consistent state equivalent to a serial schedule
- **Non-serial schedule** – operations within transactions are interleaved and may conflict

What tradeoffs are involved with **lock granularity**?

- Finer locks provide more concurrency, but more locking overhead.
- Lock granularities include:
    - Database
    - Table
    - Row / column
    - Cell

What are **isolation levels**?

- **Isolation levels** are DBMS configuration options for controlling locks that occur when selecting data
- Each level represents a trade-off between consistency and concurrency; more concurrent = less consistent, and visa-versa

How do database systems handle deadlocks?

1. Detect a deadlock (prevention too hard)
2. Abort a transaction participating in the deadlock, rollback its operations
    a. May require cascading rollbacks
3. Roll the aborted transaction forward after the other transaction completes

What is an index? What are the properties of an index?

- An index is an optional data structure that maps key values to physical locations.

What types of indices exist?

- Primary vs. secondary
    - Primary index - based on a table's key attributes that are used to physically order data
    - Secondary index - based on non-ordering attributes, both unique and non-unique
- **Sparse** vs. **dense**
    - Sparse index – every nth record in a table maps to a single index entry; during a search, the largest index record <= the lookup key is found, then the actual record is located via a linear search
    - Dense index – each record in a table maps to a single index entry; during a search, a lookup key should map to an exact index record, avoiding a linear search
- **B-tree**
    - Suited to high-cardinality (i.e. large amounts of variance) attributes, e.g. number column
    - Default in most cases
    - Properties – ascending vs. descending, unique vs. non-unique
    - Uses a large amount of space, index can be larger than table being indexed
    - Useful for OLTP
- **Bitmap**
    - Suited to low-cardinality (i.e. little or no variance) attributes, e.g. boolean column
    - Maps key values to row ids
    - Useful for OLAP

What is **domain integrity**?

- **Domain integrity** is specifying the appropriate data type for each item.
- Controls
    - Nullability – can a column contain null values?
    - Check constraint – should column values be restricted to a set list?
    - Unique constraint – is the column unique?
    - Default constraint – should there be a default value if nothing is entered?

What is entity integrity?

- **Entity integrity** is ensuring that each entity has a primary key.
- Controls
    - Decide tradeoffs between natural vs. artificial key columns
    - Avoid NULL values in primary key columns

What is referential integrity?

- **Referential integrity** ensures that foreign key values reference valid primary key values in an associated table
- Controls
    - Turn on referential integrity checks in DBMS
    - Specify RESTRICT, NULL, DEFAULT, CASCADE options for each foreign key

What is policy integrity?

- **Policy integrity** ensures that business rules are enforced
- Simple rules can be enforced through domain, entity, and referential integrity
- Complex rules require triggers or an application

What strategies could you use to boost database performance *without* changing logical design?

- Create indices
- Create views
- Use horizontal partitioning
- Use vertical partitioning

What strategies could you use to boost database performance *with* changing logical design?

- Use artificial keys
- Store derived data
- Combine 1:1 relationships
- Use repeating groups
- Use denormalization
- Minimize data type storage requirements

Post-Midterm Concepts

What are common problems faced by RDBMSs?

- RDBMSs struggle to handle “big data” via distributed transaction processing
- Difficult to develop OO applications for relational databases

What are the differences and tradeoffs between **vertical** and **horizontal scaling**?

- Vertical scaling – move to larger computers
    - Expensive
    - Limited
    - Vendor lock

- Horizontal scaling – partition data across 2 or more databases
    - Flexible
    - Easily scaled
    - More complex

What are the **BASE** properties of a NoSQL database system?

- **Basic-availability** – system guarantees availability
- **Soft-state** – state may change over time, even without input
- **Eventual consistency** – system will become consistent over time, given that it does not receive more input

What is the **CAP theorem**?

- **Consistency**, **availability**, and **partition tolerance** are 3 desirable properties of a distributed system
    - Availability – clients can always read/write
    - Consistency – clients always have the same view of data
    - Partition tolerance – system works well despite physical network partitions
- Can only embrace 2 of the 3

What is map-reduce?

- Map-reduce is an architecture for distributed processing of large datasets.
- Map – phase where new work is delegated
- Reduce – phase where completed work is aggregated

Why use a DB procedural language (e.g. PL/SQL, T-SQL)?

- More powerful than traditional SQL (conditional statements, loops, etc.)
- Decouples business logic from middle tier or GUI
- Maintain in one place
- Security
- Speed implications

What types of subprograms exist in PL/SQL?

- Procedure – no return value
- Function – returns value
- Trigger – executed by DBMS

What types of triggers exist in PL/SQL?

- AFTER | BEFORE | INSTEAD OF

- DELETE | INSERT | UPDATE
- [FOR EACH ROW]

What are decision support systems (DSS)?

- Decision support systems are designed to aid managers in decision-making tasks (e.g. budgeting, forecasting, planning)

Compare and contrast **OLTP** vs. **OLAP** databases.

- **OLTP** – support application systems used by company employees for everyday operational tasks
- **OLAP** – support data analysis and business decisions.

What are the four common properties of data warehouses?

- Subject-oriented – data is organized around subject areas (e.g. finance, marketing, sales)
- Integrated – data is collected from several transactional databases
- Time variant – data is identified within particular time periods
- Non-volatile – data is stable; new data added, but rarely updated

What is the difference between a **data warehouse** and a **data mart**?

- **Data warehouse** – large-scale data repository; incorporates aggregated data for an entire company, division, or business unit
- **Data mart** – small-scale data repository serving the needs of one department; constructed from a few transactional databases or a subset of data warehouse data

What is drill-down analysis? Slice and dice?

- Drill-down analysis – extracting data from higher to lower hierarchies of business dimensions
- Slice and dice – extracting data from two hierarchies

What is a fact table?

- Fact tables store important business measurements at the intersection of radiating dimensions

Describe the **ETL** process.

- **Extract** – copy data from a variety of transactional databases according to batch schedule; use API to obtain data from incompatible databases
- **Transform** – format, consolidate, and clean data in preparation for a load
- **Load** – data loaded into data warehouse

CSCI-5150 Distributed Systems

Midterm Concepts

What is **strong** vs. **weak mobility**?

- **Weak mobility** – program's code is moved to another machine for execution from a pre-defined start position (usually the beginning)
- **Strong mobility** – running process is stopped, then moved to another machine where execution resumes

What are the two most difficult issues in implementing strong mobility?

- Heterogeneous systems – distributed systems are often constructed on a heterogeneous collection of platforms, each having their own operating system and machine architecture; in strong mobility, each platform must be able to represent the transferred process's execution state
- Local resources – all resources used by a process, such as files, printers, devices, and other processes, must be available to the process when it is transferred to another machine

What is the difference between a **distributed operating system** and a **network operating system**?

- **Network operating system** – collection of machines that cooperate in some, but not all ways, via loose coupling; i.e., a typical network setup with basic functions, such as file and printer sharing
    - Usually not capable of process migration, load balancing, transparency
- **Distributed operating system** – has tightly-coupled control of all machines in the system; global knowledge can take time if services are distributed, but is faster if services are centralized

What is a distributed scheduling algorithm?

- A distributed scheduling algorithm is an algorithm that affords distributed operating systems the ability to balance workload across multiple machines

Differentiate between the following **dimensions of communication**:

- **Two-way** vs. **multiway**
    - Two-way – one sender, one receiver; normal communication between two processes
    - Multiway – one sender, multiple receivers; either **broadcast**, send to everyone, or **multicast**, send to a list of interested parties
- **Synchronous** vs. **asynchronous**
    - Synchronous – sender blocks waiting for a response from receiver
    - Asynchronous – sender does not block
- **Transient** vs. **persistent**
    - Transient – messages discarded when they cannot be delivered

    - Persistent – messages stored for future delivery
- **Receipt-based** vs. **delivery-based** vs. **response-based**
    - Receipt-based – sender waits for acknowledgement of message arrival only, not for a response
    - Delivery-based – sender waits for acknowledgement of message delivery to receiver, but still not for a response
    - Response-based – sender waits for response (i.e. the "normal" case)

What is a **client proxy**?

- In client-server architecture, the client and the server should package communication code as a separate module. This module is called a proxy.
- A client proxy is in charge of packaging data, sending requests, and receiving/unpacking responses.
- A server proxy operates similarly.

How is broker architecture initialized?

1. Launch broker
2. Launch servers, have servers register with the broker
3. Clients can now contact the broker to receive a list of available services

What is **X.500**? **LDAP**? What are their differences?

- X.500 and LDAP are directory services protocols. **Directory services** are data stores that allow users to locate distributed resources and services on a network.
- X.500 was criticized for being slow due to its incompatibilities with TCP/IP.
- LDAP accesses transport-level TCP/IP services directly, increasing performance.
- LDAP replaced X.500 over the span of a few decades.

What is an overlay network?

- An overlay network is a graph such that nodes represent processes and edges represent possible communication channels (i.e. logical links).
- Essentially, an overlay network is a software-driven network built on top of an underlying network architecture.

What is **flat naming**?

- Flat naming is the use of random identifiers (i.e. bit strings) to assign names to entities.

What is a distributed hash table? How can it be used for name resolution in a flat name space?

- A distributed hash table is a decentralized overlay network that provides a lookup service comparable to a hash table.
- Introduces a variety of complex problems

    - What happens if a node enters? Leaves? Crashes?
    - How are keys in the DHT's key space delegated among nodes?
    - What is the DHT's lookup algorithm?
- In **Chord**'s DHT implementation, a **hash function** is used to assign unique identifiers to nodes and keys for fast lookup.

What is the end-to-end argument?

- The end-to-end argument states that a low-level communication library cannot provide all the services needed for solving communication problems.
- End-to-end argument applies: authentication would have to be written in the application layer
- End-to-end argument doesn't apply: provided a key, encryption could be implemented at the packet layer

What is IBM's MQ Series? What does it do to enable persistent, asynchronous communication? What's an example use, other than email?

- IBM's MQ series enables persistent communication by using queue managers to store messages in send and receive queues.
- IBM's MQ series enables asynchronous communication by using message channels and message channel agents to transfer messages asynchronously between queue managers.
- Log aggregation is an example use.

What are the pros and cons of **two-level** vs. **three-level** architecture?

- **Two-level architecture** – client and server only
    - Simpler design since only two types of machines exist – a client and a server
    - Must consider tradeoffs between fat vs. thin clients
        - Fat clients can support rich user interfaces but may suffer from client platform issues
        - Thin clients tend to be less problematic but at the cost of user experience
- **Three-level architecture** – client, application, and database
    - Application and database layers run on separate machines
    - Ideal since application layer is decoupled from database layer
    - Architecture is more complex; communication between application and database machines must be established

What are the pros and cons of **TCP** and **UDP**?

- TCP uses error correction to guarantee that messages are delivered in their entirety, without errors. TCP is slower than UDP and should be used for important data, such as web pages.
- UDP does not use error correction. Messages sent by UDP may be incomplete or contain errors, but UDP is much faster than TCP. UDP should be used for streaming media, such as music or movies.

What are the pros and cons of a fixed-size thread pool server vs. a new thread per request server?

- A fixed-size thread pool server yields faster response times since client requests avoid thread creation overhead. However, developers must decide the server's thread pool size, which may be difficult to determine.
- A new thread per request server yields slower response times since client requests must wait for thread creation. However, the server has a simpler design and may save memory if client traffic is slow.

What is **marshalling**? How is it different from **serialization**? What issues must be handled when marshalling parameters for RPC?

- Marshalling and serialization both transform objects into a primitive form for transfer or storage. The difference is that marshalling includes objects' codebase, which allows the objects to be reconstructed in their original form once unmarshalled. Serialization does not include objects' codebase, meaning that new objects are constructed once deserialized.
- Marshalling must handle big-endian/little-endian and data representation differences between machines.

When would a multicast name resolution method work well? What drawbacks would it have?

- Multicast name resolution is useful for locating a machine that is a member of a multicast group. However, multicast name resolution can only be performed over a local-area network.

What is an **agent**?

- An **agent** is a program designed to move between machines to handle requests on the best network node possible. In this context, "best" means the agent can easily get some information on that node or carry out some computation there.

What is the **ontology problem**? Why is it a big drawback for creating general agent systems?

- The **ontology problem** refers to domain-specific terminology that has no meaning or has alternate meanings in other domains.
- General agent systems would require a comprehensive domain dictionary to be effective.

What steps would a client take to use a service via a service discovery system like Jini?

- Jini is a Java-based implementation of broker architecture.
- Steps to use a service:
    1. Client finds a lookup service by broadcasting or asking another client.
    2. Client queries the lookup service to find the service it needs.
    3. Client downloads a server object that provides proxy services for the actual service.
    4. Client uses reflection to discover and use the proxy's interface.

What is **copy-restore parameter passing?** What can go wrong when it's used as a substitute for pass-by-reference?

- In **copy-restore parameter passing**, parameters do not receive new assignments until the procedure ends and new values are copied into memory.
- If a parameter passed by copy-restore is expected to update mid-execution (as is the case for pass by reference), the procedure may assign an incorrect value to the parameter when it finishes.

What is the difference between **iterative** and **recursive name resolution**?

- **Iterative name resolution** – client's name resolver passes labels to name servers, one at a time, until the entire name is resolved
- **Recursive name resolution** – client's name resolver passes name to the root server, which contacts other name servers recursively until the name is resolved

Final Concepts

What is the network time protocol (NTP), i.e. Cristian's algorithm?

- In NTP, clients contact a time server, then estimate and factor out transmission delays.

What is a WWV station? Name three sources of delay that could be introduced between WWV broadcasting the time and processors in a distributed system setting their clocks.

- A WWV station is a shortwave radio station that broadcasts a short pulse at the start of each UTC second.
- Latency between receiver and processer, atmospheric conditions, and bouncing of radio waves could all cause delays

What is the Berkeley algorithm? How does it differ from NTP?

- The Berkeley algorithm takes a distributed approach to clock synchronization as opposed to NTP's centralized approach.
- In the Berkeley algorithm, a coordinator contacts all machines asking for their time, then takes an average and distributes the time to all machines.
- Remember that actual time is not important here. What is important is that all machines agree on *a* time.

Explain the concept of Lamport's logical clocks? What is their significance?

- Logical clocks are based on two premises:
    - If two processes do not interact, their clocks do not need to be synchronized
    - It doesn't matter what time processes agree on; what matters is that they agree on an order in which events occur

Explain totally-ordered multicasting. What type of consistency does it produce?

- Totally-ordered multicasting is an expensive multicasting algorithm that maintains sequential consistency.

Explain causally-ordered multicasting using vector clocks. What type of consistency does it produce?

- Casually-ordered multicasting is the use of vector clocks to enforce causal consistency.

What are the pros and cons of centralized mutual exclusion? Decentralized mutual exclusion? Token ring mutual exclusion?

- Centralized mutual exclusion – simulates a non-distributed environment
    - Pros
        - Guaranteed mutual exclusion
        - Fair (requests handled in order received by coordinator)
        - Easy to implement
        - Low message traffic, yielding higher performance
    - Cons
        - Single point of failure
        - Processes can't distinguish between a dead coordinator and "permission denied"
- Decentralized mutual exclusion – each resource replicated n times; each replica represented by its own coordinator; replicas stored in a distributed hash table for fast lookup; to access a resource, a process must obtain a majority vote > n/2 coordinators
    - Pros
        - More reliable, no single point of failure
    - Cons
        - Architecture is "probabilistically" reliable; if > n/2 coordinators crash, mutual exclusion can no longer be guaranteed
        - Inefficient in comparison to centralized mutual exclusion
        - Denied processes are not queued, which can result in starvation

What's the difference between **synchronization** and **mutual exclusion**?

- **Synchronization** is the general problem of having concurrent processes/threads cooperate to achieve objectives.
- **Mutual exclusion** is a specific problem within synchronization that deals with the prevention of race conditions.

Does distributed deadlock differ much from how non-distributed deadlock is handled?

- Distributed deadlock is handled by methods similar to non-distributed systems.

What conditions must be true for a deadlock to occur?

1. **Mutual exclusion** – only one process can access a resource at a time
2. **No preemption** – processes cannot force other processes to give up resources

3. **Hold and wait** – processes will not voluntarily give up their resources
4. **Circular waiting** – processes wait on each other to give up resources

What solutions exist for breaking deadlock?

- Do nothing –the usual case for operating systems
- Detect deadlock, then break it
    - Detecting deadlock – **resource allocation graph**
        - Create a directed, unweighted graph
        - Create nodes for processes and resources
        - Create an edge from each process requesting a resource to the resource requested
        - Create an edge from each resource to the process currently holding it
        - If a cycle exists in the graph, a deadlock is present
    - Breaking deadlock – a few options available
        - **Wound-wait** (older job can kill younger job to take resources)
        - **Wait-die** (younger job kills itself if it requests resources from an older job)
- **Prevent deadlock** by breaking one of the 4 conditions, e.g. have processes time out on resource requests (hold and wait)
- **Avoid deadlock** by allocating resources carefully, e.g. banker's algorithm
    - Too expensive for practical considerations

What is a **schedule**? **Serial schedule**? **Serializable schedule**?

- Schedule – sequential ordering of instructions within multiple transactions
- Serial schedule – transactions run sequentially, no concurrency, always consistent
- Serializable schedule – transactions run concurrently, rules followed to ensure consistent results
    - Instructions inside a transaction cannot be re-ordered
    - Data elements must be locked before accessing

What is **two-phase locking**?

- Locking in two phases
    - All locks needed by a transaction are acquired
    - Transaction releases locks gradually or all at once at the end
- Two-phase locking is guaranteed to produce serializable schedules

What problem do election algorithms attempt to solve?

- Many distributed algorithms require one process to act as a coordinator or some other special role. In most implementations, it doesn't matter which process takes the responsibility, but one of them has to do it.
- In general, election algorithms attempt to locate the process with the highest PID an designate it as coordinator.

Describe the bully election algorithm.

- When a process p notices a coordinator is no longer responding to requests, it initiates an election
- P sends an election message to all processes with higher PIDs
    - When a higher PID process receives an election message, it acknowledges the sender's message to indicate it is alive and will take over
    - The receiver then holds an election, unless he is already doing so
    - Alternatively, if no one responds, p wins the election and becomes coordinator
- Eventually, all processes give up but one. The winner announces its victory by sending a special message to all processes indicating it is the new coordinator.

What happens if two processes initiate an election simultaneously in the bully algorithm?

- The process with the highest key still wins.

Describe the ring election algorithm.

- Assumes processes are ordered and keyed in a ring, and that each process knows all other processes in the ring
- Algorithm
    - A process initiates an election, circulates an array where live processes put their PIDs
    - When the array returns to the initiator, a new message is circulated indicating the new coordinator
- Multiple elections will not harm the algorithm; the process with the highest key still wins

How are elections held in wireless environments?

- Algorithm is complicated; essentially, it's done through a breadth-first search

What is the difference between **consistency** and **correctness**?

- Consistency = same order of updates across replicas
- Correctness = the order to be used

What is a conit?

- A conit specifies the unit over which consistency is to be measured (e.g. in a stock market, a single stock).

What is a consistency model?

- A consistency model is a contract between software and memory. It states that memory will work correctly if the software obeys certain rules.

List and explain consistency models in the order of most to least consistent.

1. **Strict consistency** – All reads/writes executed in an order derived from some global wall clock time. This model is impossible in practice since distributed systems cannot support perfection synchronization of global time.
2. **Linearizability** – Attempts to achieve strict consistency by synchronization machine clocks tightly to use as a universal time stamp. This model is impractical due to performance hits.
3. **Sequential consistency** – Any valid interleaving of read and write operations is acceptable behavior, but all processes see the *same interleaving of operations*. In other words, this is a total ordering of read/writes across replicas according to logical time, not real time.
4. **Causal consistency** – Writes that are potentially causally related must be seen by all processes in the same order. Concurrent writes may be seen in different orders on different machines. Often, writes performed in sequence by one process must be seen in that order by all other processes.
5. **Entry consistency** – Processes acquire and release mutex locks to read and write to data elements
6. **Eventual consistency** (client-centric) – Ideal for systems where processes read from databases and rarely write to them. Updates are propagated in a lazy fashion, meaning that a reading process will see an update only after some time has passed. All replicas converge toward identical copies of each other.

What are some common rules for enforcing eventual consistency?

- Monotonic reads – If a process reads a value x, it will never see an older version of x at a later time
- Monotonic writes – A write to x is only performed if x has been brought up-to-date by a client's previous writes
- Read your writes – a write operation is always completed before a successive read operation by the same client
- Writes follow reads – any successive write operation by a process on x will be performed on a copy of x that is up-to-date with the value most recently read by that process

What two aspects of replica management should be considered when constructing a distributed data store?

- Replica-server placement – determine geographic locations to deploy replicas
- Content-replication and placement – how should replicas be created?
    - Permanent replicas – static, initial set of replicas that constitute a distributed data store
    - Server-initiated replicas – data store copies created programmatically to enhance performance
    - Client-initiated replicas – done via client-side caching

What are the pros and cons of distributed shared memory?

- Pros
    - Programmers don't have to handle distributed aspects of shared memory; similar naming and usage patterns deployed as in unshared memory

    - No marshalling of parameters required (distributed operating systems enforce communication protocol)
    - No application-level communication needed
- Cons
    - Programmers still have to use mutual exclusion tools
    - May not scale well due to overhead of messages passed between processes
    - Communication costs are hidden to programmers

Access to shared Java objects can be serialized by declaring its methods as being synchronized. Is this enough to guarantee serialization when such an object is replicated?

- No, only one replica would be protected at a time

Why choose a weak consistency model?

- Performance, scalability

How does replication in DNS take place? Why does it actually work well?

- DNS servers follow an eventual consistency model where, every once in awhile, they trade their name tables. The higher a DNS server is in the tree, the longer the elapsed time between trading since names at the top don't change often.

Why is a contract between the software and data storage needed?

- Programmers need memory guarantees to ensure consistency and correctness.

The weaker the consistency model, the greater the burden on application programmers to preserve consistency. Why?

- Implications of weak consistency models are harder to understand than strong consistency models
- Weak consistency models typically require the use of mutex locks for reads and writes

Why does totally-ordered multicasting not scale well?

- Too many message between replicas

Describe how **quorum-based protocols** work for maintaining sequential consistency.

- Quorum-based protocols require clients to request and acquire permission of a specified minimum number of replicas before reading/writing to a data item.
- Client must contact half of servers plus one (majority) to perform a write.
- Client must contact at least half of the servers to perform a read.
- In other words,
    - Read quorum = collection of replicas a client uses for reading = NR

- Write quorum = collection of replicas a client uses for writing = NW
- NR + NW > N
- NW > N / 2

What are the tradeoffs in granularity size in distributed shared memory (DSM)?

- Small granularity = more parallelism, more overhead of managing data elements, more bandwidth used
- Large granularity = less parallelism, less overhead of managing data elements, less bandwidth used

What is **false sharing**? What solutions exist for false sharing?

- **False sharing** is a scenario where processes share a page of memory but operate on different values within the page, preventing parallelism.
- Solutions: replicate the page, use a smaller granularity, or modify the compiler to allocate memory more intelligently

What is thrashing in single-cpu operating systems? How is thrashing different in DSM?

- Single-cpu thrashing occurs when 1) memory is full and 2) the memory manager keeps swapping pages to disk that are needed by other processes
- In DSM, thrashing occurs when processes constantly need access to a shared page

How would a page fault be handled in a centralized DSM paging system? A distributed DSM paging system?

- Centralized – contact memory manager for page (but single point of failure)
- Distributed – get page from peer, but update all page tables across peers

What properties are desirable in a fault tolerant system?

- Available – system is ready to use immediately
- Reliable – system can run continuously without failure
- Safe – nothing catastrophic happens in case of system failure
- Maintainable – system can be repaired with relative ease

What models exist for describing system failures?

- Crash – server halts, but works correctly until it halts
- Omission – messages somehow lost
- Timing – client expects a response, but it doesn't happen in time
- Response – server sends incorrect response
- Arbitrary – server produces arbitrary responses at arbitrary times (hardware failure)
- Transient – happens once, then disappears
- Intermittent – happens in some cases, but not others (non-determinism)

What are distributed agreement algorithms?

- Distributed agreement algorithms are used to have non-faulty processes reach a consensus on some issue.

What is the two generals' problem? Byzantine generals' problem?

- Two generals' problem – original messages and acknowledgement messages can be lost in-transit, resulting in a potentially infinite series of messages between two entities
- Byzantine generals' problem – how can we know if a process is producing bad outputs?

What is failure masking? What types exist?

- Failure masking hides the occurrence of failure from other processes using redundancy
- Three types:
    - Information redundancy – extra bits are added to allow recovery from garbled bits
    - Time redundancy – an action is performed and, if needed, performed again
    - Physical redundancy – extra processes or equipment added to allow system to tolerate malfunctioning of some components

Can the triple modular redundancy model (TMR) handle Byzantine failures?

- No. Assuming k faulty processes, 2k+1 non-faulty processes are required to handle Byzantine failures.

Does TMR generalize to 5 elements per group?

- Yes, the model works with any odd number > 1. 2 processes can fail under a model with 5 elements.

What is reliable multicasting? Atomic multicasting?

- Reliable multicasting is used to ensure processes receive multicasted messages via acknowledgements.
    - Has scalability issues, but can be helped by feedback suppression (do not send positive acknowledgements; send negative acknowledgements and tell other nodes not to duplicate a negative acknowledgment)
- Atomic multicasting ensures that a message is delivered to all processes or none at all; also ensures sequential consistency
    - Virtual synchrony – if a sender of a message crashes during a multicast, the message is delivered to all processes or none of them

How is the distributed commit problem solved by two-phase commit?

- Coordinator sends a vote request message to all workers
- Each worker replies with a vote commit or vote abort message
- Coordinator collects replies, then sends a global commit (if all are committed) or global abort message to all workers
- Each worker commits if a global commit message is received, else the transaction is aborted

What is the write-ahead log protocol?

- The write-ahead log protocol requires that transaction instructions be written to a log before actually run on production data elements

What's a checkpoint? What's the difference between a fuzzy checkpoint and a sharp checkpoint?

- Checkpoint – point in log where transactions are signed off as committed
- Sharp checkpoint – suspends processes running transactions before log records are written
    - Easier to rollback, but suffers performance hits
- Fuzzy checkpoint – does not suspend running transactions before log records are written
    - Harder to rollback but better performance

Give an example where group communication requires no message ordering at all.

- A read-only distributed data store.

What is stable storage? How is it implemented?

- Stable storage = fault tolerant disk storage (i.e. RAID with error checking)

How do at-least-once semantics, at-most-once semantics, and exactly-once semantics compare?

- At-least-once semantics – guarantees that an RPC has been carried out at least once
- At-most-once semantics – guarantees that an RPC has been carried out at most one time, but possibly none at all
- Exactly-once semantics – guarantees that an RPC has been carried out exactly once; impossible in distributed systems

What properties are desirable in a secure system?

- Confidentiality – system information is disclosed only to authorized parties
- Integrity – alterations to system information can be made only in an authorized way

What's the difference between a **policy** and a **mechanism**?

- A **policy** is a requirement describing which actions system entities are allowed to take and which ones are prohibited
- A **mechanism** is a means by which a policy can be enforced, such as encryption, authentication, authorization, and auditing

What are the tradeoffs between an access control matrix, access control list, and a capability list?

- An access control matrix maps all subjects to all objects in a system
    - Subjects are rows
    - Objects are column

    - Overall, very inefficient in large systems where thousands of users and millions of objects exist
- Access control list – each object carries a list of subjects that can access it
- Capabilities list – each subject carries a list of objects it can access

What's the difference between symmetric and asymmetric encryption?

- Symmetric encryption – both parties use a single, shared, private key
- Asymmetric encryption – both parties have their own set of public and private keys

What are the two essential properties of a hash function?

- One-way – it is computationally infeasible to map a known output to an input
- Collision resistance – given an input-output pair, it is computationally infeasible to locate a second input that produces the same output

What is the challenge-response protocol?

- The challenge-response protocol is used to establish a secure channel with two hosts. It is assumed the two hosts share a secret key. The secure channel is established by transferring encrypted challenge messages between both hosts and verifying that each host can decrypt the message.
- Can be vulnerable to a reflection attack.

What are the advantages of a key distribution center?

- A key distribution center uses *n* keys to establish the challenge-response protocol across *n* hosts, which is much more efficient than n^2 keys required for a decentralized approach

Does it make sense to restrict the lifetime of a session key?

- Yes. A session key could be broken by a hacker analyzing encrypted messages sent between two parties.

How do remote objects work? What's the difference between remote objects and distributed objects?

- Clients can communicate with a remote object via a proxy object that acts as its interface.
- A distributed object resides on 1 or many machines.
- A remote object resides on only 1 machine.

What's the difference between a persistent and transient object?

- Persistent object – an object that continues to exist via secondary storage even if it is not contained in the address space of any server process
- Transient object – an object that exists only as long as the server that is hosting the object

What's an object server?

- An object server is a server tailored to support distributed objects. Services are provided by the objects themselves, not the server.
- Example - EJB

What is it useful to define object interfaces using an interface definition language (IDL)?

- IDL allows one language for all bindings independent of the actual programming language being used to support distributed objects.

Should client/server-side objects for asynchronous method invocation be persistent?

- Yes, but a client may have to wait longer if an RMI is placed into a server-side queue.

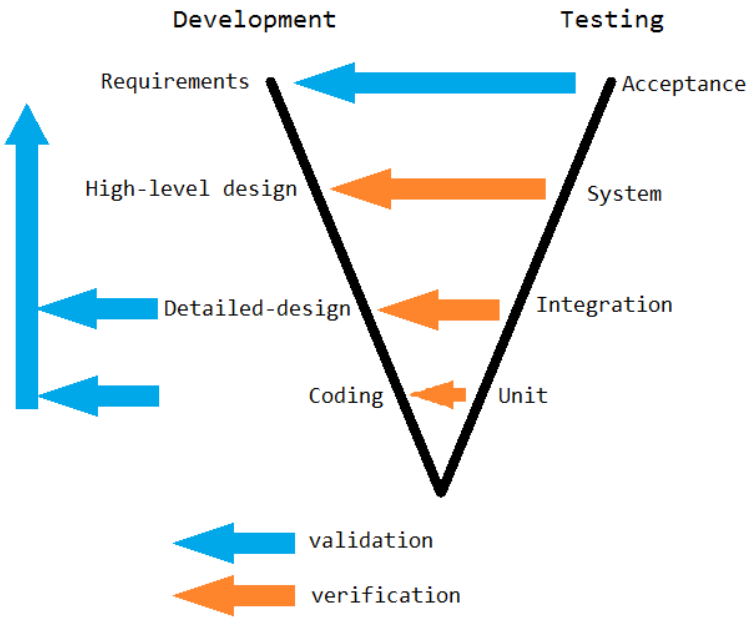
How could exceptions in RPCs / RMIs be implemented?

- Error status codes (think HTTP)

CSCI-5220 Software Verification / Validation

Test 1

Describe Naik and Tripathy's V-model. Which activities are verification? Which are validation?



Explain the difference between a mistake, a fault, an error, and a failure.

- **Mistake** – a human action produces an incorrect step in a segment of code
- **Fault** (or defect) – an incorrect step, process, or data definition; the manifestation of a mistake
- **Error** – the amount by which the output is incorrect, i.e. the degree of failure
- **Failure** – an incorrect output, i.e. the manifestation of a fault

Give a brief description of the following types of testing:

- **Unit testing** – testing programmatic units (methods, classes, etc.) in isolation
- **Integration testing** – testing interaction between programmatic units and interaction with other systems
- **System testing** – testing the complete integrated system according to requirements specifications
- **Regression testing** – testing done during maintenance to ensure that enhancements or fixes do not introduce defects to existing functionality
- **Stress testing** – testing a system beyond the limits of its specified requirements or resources
- **Acceptance testing** – testing done by stakeholders to ensure the system meets their needs

What is the difference between white-box and black-box testing?

- White-box testing – testing code by examining the logic of internal modules
- Black-box testing – testing a system's external interfaces; concerned with inputs and expected outputs, not internal logic

Test 2

What is predicate calculus? First-order logic? How can they be applied to software testing?

- Predicate calculus is the use of sentences that contain predicates and variables to quantify terms.
- First-order logic is applied to things, rather than systems (i.e. higher order logic).
- Predicate calculus can be used to construct formal proofs of software correctness.

What is contrapositive? Contradiction?

- Contrapositive: a ☾ b, -b ☾ -a
- Contradiction: a and –b

What is an axiom?

- An axiom is a self-evident or universally-recognized truth; can be used in other proofs to define advanced systems of logic

What is the overall goal of software testing?

- "The ideal, abstract goal of testing is to reveal all faults in a software system without exhaustively testing the software"

What is an ideal test?

- "An ideal test is a small, proper subset of the entire input domain" that can be shown to ensure program correctness

In test selection, what is the difference between a reliable and valid criterion?

- Reliable criterion – all tests selected by the criterion are successful or none are successful
- Valid criterion – whenever a SUT contains a fault, the criterion selects a test that reveals the fault

What is Dijkstra's fundamental limitation of software testing?

- Software testing can only reveal the presence of defects, never the absence.

Explain the following faults:

- **Requirements fault** – failure to capture stakeholders' actual requirements
- **Design fault** – failure of a software design to satisfy an understood requirement
- **Construction fault** –failure to correctly implement a software design

- **Logic fault** – program produces incorrect results independent of performance resources
- **Performance fault** – program does not produce expected results within resource limitations

What is the difference between **fault seeding** and **program mutation**?

- **Fault seeding** – typical faults are introduced into the program; test cases are evaluated based upon the percentage of faults found
- **Program mutation** – mutations, i.e. variants of a program that may or may not contain faults, are created; test cases are evaluated based upon the number of faulty mutations found

Test 4

Describe NUnit's constraint-based assert model.

- NUnit's constraint-based assert model uses a single method (*That*) for all assertions.
- Example: Assert.That(result, Is.EqualTo(8))

Differentiate between the following NUnit attribute tags:

- SetUpFixture – Used on a class containing methods to run before/after a namespace's tests
- SetUp – If used in a SetUpFixture, runs before a namespace's tests; if used in a TestFixture, runs before every test in the class
- TearDown – If used in a SetUpFixture, runs after a namespace's tests; if used in a TestFixture, runs after every test in the class
- TestFixture – Used on a class containing test cases
- TestFixtureSetUp – Runs once, before all tests in the class
- TestFixtureTearDown – Runs once, after all tests in the class
- Test – Used to denote a test case

What are test doubles and why are they needed?

- Test doubles are "fakes" used to replace a program unit's dependencies with dependencies that produce simplified and expected behavior. This allows program units to be tested in isolation.

What properties are desirable in well-written unit test cases?

- Tests should be high quality (readable, maintainable, etc.)
- Tests should be isolated from other tests
- Tests should be independent of run order
- Tests should have a repeatable and expected (i.e. deterministic) outcome
- Tests should evaluate small, discrete behaviors
- Tests should execute quickly

What is the **arrange, act, assert** pattern?

- Arrange, act, assert is a pattern for composing a test case
- **Arrange** – create a mock object, pass the mock into the SUT
- **Act** – execute the SUT
- **Assert** – verify the SUT's interaction with the mock object

What is the difference between a **mock** and a **stub**?

- A **stub** replaces a method with code that returns a specified result
- A **mock** is a stub with an assertion that the method gets called

Test 6

What is the difference between an **inspection** and a **walkthrough**?

- **Inspection** – step-by-step *group review* of a work product, with each step checked against pre-determined criteria
- **Walkthrough** – *author leads the team* through a manual or simulated execution of a work product using predefined scenarios

What are the roles of code review participants?

- Moderator – guides the review process, should be from an unrelated project
- Author – person who has written code to be reviewed
- Presenter – someone other than the author who understands the code; presents code to group
- Recordkeeper – documents problems found and follow-up actions suggested
- Reviewers – experts in the subject area of the code under review
- Observers – people who want to learn about the code but do not participate in the process

What are the six steps of a code review?

- Readiness
- Preparation
- Examination
- Rework
- Validation
- Exit

What is the recommended length of a code review?

- 1-2 hours

What is **control-flow testing**?

- Testing the paths of a program unit

What is **data-flow testing**?

- Testing the flow of data values (programmatic state) along a path of program execution

What is **domain testing**?

- Areas having potential for domain errors are identified; test data is selected to catch the faults

What is the **McCabe complexity measure**? How is it calculated?

- The McCabe complexity measure outputs a program unit's cyclomatic complexity.
- To calculate the McCabe complexity measure:
    - Construct a graph where each node represents a sequence of adjacent statements and each edge represents flow of control
    - Use the formula v = e – n + 2 where v = cyclomatic complexity, e = number of edges, and n = number of nodes

Test 7

How is control flow testing useful?

- Control flow testing is useful for generating test case inputs necessary to cover paths within a program unit

What is the difference between a **feasible** and **infeasible path**?

- A feasible path can be executed given a specific set of program unit inputs.
- An infeasible path cannot be executed regardless of program unit inputs.

What is a **control flow graph**?

- A control flow graph is a graphical representation of a program unit's control flow.
- Nodes represent instructions, decision points, and merge points
- Edges represent control flow, i.e. the flow from one instruction to the next

What are the symbols used in a control flow graph, and what do they mean?

- Square = sequential instruction, no fork in control flow
- Diamond = decision point, control flow forks based upon output of a condition
- Circle = merge point, two or more paths meet
- Remember to assign a number to each node so that paths through the CFG can be easily identified

Explain the following **path selection criteria**:

- **All paths** – all paths covered by test cases, which can detect all faults except for missing path errors; all paths is desirable, but difficult to achieve in practice
- **Statement coverage** – all statements executed at least once by test cases; considered the weakest coverage criterion in software testing
- **Branch coverage** – all branches (i.e. edges in a CFG) covered by test cases
- **Predicate coverage** – all possible combinations of truth values for predicates (i.e. truth table) covered by test cases

Test 8

How is data flow testing useful?

- Data flow testing produces program unit inputs for selecting paths that cover data definition and use patterns

Describe the following **data flow anomalies**:

- **Defined and then defined again** (type 1) – a variable receives two consecutive assignments without using the first assignment (usually caught by IDE tools)
- **Undefined but referenced** (type 2) – use of a variable that has not received an assignment (usually results in compile-time or runtime errors)
- **Defined but not referenced** (type 3) – creation of an unused variable (throws a compiler warning in C#)

What is a **c-use**? **P-use**?

- A **c-use** is the use of a variable to compute a new value of the same variable or another variable
- A **p-use** is the use of a variable in a predicate

What is a **data flow graph** constructed?

- The entry point defines each input parameter, including instance/global variables if applicable
- Nodes are definitions and c-uses
- Edges are p-uses
- The exit point is a return statement

What are some **notable differences** between a control flow graph and a data flow graph?

- A dataflow graph does not have decision or merge points. Every node is treated as an instruction.
    - Decisions should be represented as a NULL node to represent a no-op instruction; edges leaving the instruction represent conditions
        - Conditions should be considered atomic; don't break at AND or OR
    - Merging edges can simply point to the node they meet at

- If a condition does not exist between two statements, TRUE should be used
- Sequences of c-use statements should be combined into a single node (excluding the entry node)

What is the **all-defs** testing criteria?

- Complete path – path from entry point to exit point
- Global definition – variable receives definition, and the definition is used in a subsequent node or edge
- The **all-defs** testing criteria covers a set of complete paths that cover one global c-use or p-use of every global definition of every variable

Test 9

What is the purpose of integration testing?

- Integration testing's objective is to assemble a reasonably stable system in an incremental manner. This is done by testing interactions between modules.

What are some of the interface errors that can occur when assembling software modules?

- Inadequate functionality – a module assumes another module provides a function when in reality it does not
- Misunderstanding of interfaces – misunderstanding of method preconditions and postconditions (e.g. passing an unsorted array into binary search)
- Inadequate error processing – calling module does not handle an error properly
- Timing/performance problems – inadequate synchronization among communicating processes (e.g. race conditions, improper sequencing)
- Improper coordination of changes – failure of developers to communicate module changes to other developers

What granularities can be used in integration testing?

- **Intrasystem testing** – incrementally construct and test individual components in a distributed system
- **Intersystem testing** – test *end-to-end* interaction between components in a distributed system
- **Pairwise testing** – only *two* distributed components tested at a time

Explain the following **integration testing techniques**:

- **Top-down**
    - Assumes modules arranged in a hierarchy
    - Start with root module, test hierarchy by moving downwards and removing 1 mock at a time
- **Bottom-up**
    - Assumes modules arranged in a hierarchy

    - Beginning with leaf modules, test hierarchy by moving upwards and removing 1 mock parent (i.e. test driver) at a time
- **Sandwich**
    - Use top-down testing for the root module and its immediate children
    - Use bottom-up testing for leaf modules and their immediate parents
    - Use big bang testing for remaining middle layers
- **Big bang**
    - Unit test all modules, then combine modules all at once to test the system as a whole
    - Not recommended for large systems due to difficulty in locating sources of failure

How do the above integration testing techniques rank from most to least effective?

1. Top-down
2. Big bang
3. Sandwich
4. Bottom-up

Test 11

What are the 3 benefits of automated testing as discussed by Naik and Tripathy?

- Automated tests are repeatable and reusable
- Automated tests can test the application using methods that would otherwise be tedious or difficult
- Automated tests can shorten the test cycle of an application

What does it mean for an automated test to be independent and self-sufficient?

- Independent and self-sufficient test cases can be run in any order and do not depend on one another to function properly
- The state of the SUT before each test is equivalent to the state of the SUT after each test

Describe the 6 structures of a typical automated test case.

- Setup – check SUT's environment; configure SUT parameters required by the test case
- Drive the test – Run the test by providing input data to the SUT
- Capture the response – SUT's response is captured and saved
- Determine the verdict – Compare actual outcome with expected outcome
- Log the verdict – log detailed record of the results to a log file
- Cleanup – restore SUT to its original state so that the next test case may be executed

Other

What is the difference between a **computation error** and a **domain error**?

- A computation error causes the program to execute a correct path but produce an incorrect output.
- A domain error causes the program to execute an undesired path.

What is the difference between an **open** and **closed boundary**?

- A boundary is said to be **closed** if points on the boundary are included in the domain of interest, i.e. ==, <=, and >=.
- A boundary is said to be **open** if points on the boundary are not included in the domain of interest, i.e. > and <

How should open and closed boundaries be tested?

- Closed boundary – ON-OFF-ON (i.e. in domain, not in domain, in domain)
- Open boundary – OFF-ON-OFF (i.e. not in domain, in domain, not in domain)

CSCI-5620 Analysis of Algorithms

Introduction

What is an algorithm?

- An algorithm is a well-defined computational procedure that takes input and produces output.

What is the objective of algorithms analysis?

- Determine if the algorithm is correct and as fast as possible
- Time, space complexity in worst case, average case, and best case scenarios
    - Average case can be long-term average (e.g. amortized analysis) or performance for a random input

Proof by Induction

What is proof by induction?

- Proof by indication is a proof technique used to prove a statement for all natural numbers.

What are the components of a proof by induction?

- **Base Case** – prove the given statement for the first natural number
- **Inductive Hypothesis** – assume given statement is true for k, state objective for showing k + 1 is true
- **Inductive Step** – show k +1 is true, conclude the proof

Complexity Theory

What is the definition of Big Oh?

- A function f is in O(g) if there exists a c and a k such that |f(n)| <= c|g(n)| for all n > k

What is the definition of Big Oh using limits?

- f is in O(g) if:

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = c \quad \text{for some} \quad c \in \Re^{¿}$$

What is L'Hospital's rule?

- L'Hospital's rule is used to solve a limit that simplifies to an indeterminate form (0/0 or ∞/∞. The rule is applied using derivatives:

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = \lim_{x \to \infty} \frac{f'(x)}{g'(x)}$$

Define the **following notations** related to Big Oh and give an example of each.

- Big Omega – f is in Ω(g) if cg(n) <= f(n) for all n >= k, i.e. best case complexity
  - Ex. n is in Ω(lg(n))
- Big Theta – f is in Θ(g) if f is in O(g) and g is in O(f), i.e. same order complexity
  - Ex. lg(n!) is in Θ(nlgn)
- Little Oh – like big oh, but a strict upper bound (i.e. strictly less than)
- Little Omega – like big omega, but a strict lower bound (i.e. strictly greater than)

Give Big Oh complexity for the following:

- Linear search - n
- Heap sort – n lg(n)
- MST – n^2
- Default matrix multiplication – n^3
- Factoring – 2^n
- Travelling salesman problem – n!

Solving Recurrence Relations – Substitution/Master's Method

What is a recurrence relation?

- A recurrence relation is an algorithm specified recursively, typically breaking down into
  - the actual number of operations for the first step
  - number of pieces the work is divided into for further work
  - multiplier factor, if necessary, on those pieces

What is the difference between an **explicit formula** and a **recursive formula**?

- Explicit formula – nth term defined in terms of n, e.g. $a_n = 2^n + n - 1$
- Recursive formula – nth termed in terms of n and previous terms, e.g. f(n) = f(n-1) + f(n-2)

What does it mean to "solve" a recurrence relation? Why is this advantageous?

- Solving a recurrence relation means to produce an explicit formula equivalent to a recursive formula

- After producing an explicit formula, the explicit formula's time complexity can be obtained

How is the substitution method used to solve a recurrence relation?

1. Repeatedly substitute decreasing values of n into the original equation until a pattern becomes visible
2. Generalize the pattern for k
3. Use a base case to define k in terms of n
4. Substitute k in terms of n
5. Simplify and produce an explicit formula

How is the master's method used to solve a recurrence relation?

- Assumption - recurrence is of the form $T(n) = aT(n/b) + f(n)$
- Three cases to memorize, each comparing $f(n)$ to $n^{\log_b(a)}$
- 3rd case requires an additional proof

Divide and Conquer Algorithms

How does a divide and conquer algorithm work?

- A divide and conquer algorithm divides a problem into subproblems recursively until a subproblem is reached that can be solved; subproblem solutions are then combined to solve the original problem
- Examples – binary search, merge sort

How do divide and conquer algorithms relate to recurrence relations?

- Divide and conquer algorithms can be written as recurrence relations in terms of T(n)
- Ex. Binary search => $T(n) = T(n/2) + 1$

Decision Trees

What is a decision tree? How does it relate to comparison sorting algorithms?

- A decision tree is a depiction of comparisons needed to produce all possible outcomes of a comparison sorting algorithm.

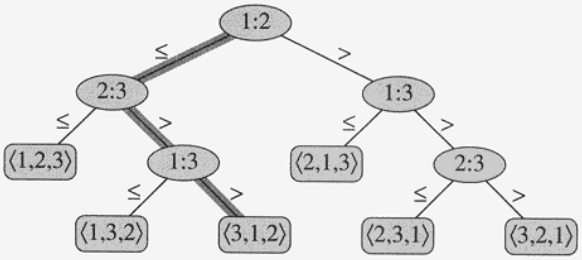


Figure 8.1 The decision tree for insertion sort operating on three elements. An internal node annotated by $i:j$ indicates a comparison between a_i and a_j. A leaf annotated by the permutation $\langle \pi(1), \pi(2), \ldots, \pi(n) \rangle$ indicates the ordering $a_{\pi(1)} \leq a_{\pi(2)} \leq \cdots \leq a_{\pi(n)}$. The shaded path indicates the decisions made when sorting the input sequence $\langle a_1 = 6, a_2 = 8, a_3 = 5 \rangle$; the permutation $\langle 3, 1, 2 \rangle$ at the leaf indicates that the sorted ordering is $a_3 = 5 \leq a_1 = 6 \leq a_2 = 8$. There are $3! = 6$ possible permutations of the input elements, so the decision tree must have at least 6 leaves.

How many leaf nodes does a decision tree of n items have?

- n! since there are n! different ways to arrange the elements

What is the shortest height any decision tree with n items could possibly have?

- lg(n!) = nlg(n)

What do the above rules say about a decision tree with n items in general?

- Ω(nlg(n)) is the best case running time for any comparison sorting algorithm

Loop Invariants and Proof of Correctness

- What is a **loop invariant**?
  - A loop invariant is an assertion that holds true before, during, and after a loop executes
- What **3 steps** are required to prove a loop correct?
  - **Initialization** – prove the loop invariant is correct prior to the loop's first iteration
  - **Maintenance** – prove that if the loop invariant is true at the start of iteration k, it must be true at the start of iteration k+1
  - **Termination** – prove that when the loop terminates, the loop invariant holds a useful property that shows the algorithm is correct

Heaps, Binomial Heaps, and Fibonacci Heaps

How is an element inserted into a heap?

- Insert the element at the end of the heap, then heapify up (A[i]>A[parent[i]], set i=parent[i], repeat)

How is an element extracted from a heap?

- Remove the root

- Move the heap's last element to the top of the heap
- Heapify down

How does a binomial heap differ from a traditional heap?

- A binomial heap B is a collection of binomial trees, such that
  - B_0 is a single node
  - B_k consists of two binomial B_{k-1} linked together
  - B_k has 2^k nodes
  - The height of B_k is k
- Insert – create a B_0 tree, continue merging binomial trees with an equivalent number of nodes until the operation is no longer possible
- Merge – root with lowest key becomes new root, other tree becomes a child of the new root

How does a Fibonacci heap differ from a binomial heap and a traditional heap?

- A Fibonacci heap is similar to a binomial heap, except that insert operations run in O(1) time; merging trees is postponed until a key is extracted or deleted
- Useful when the number of extract/delete operations are low compared to other operations

Minimal Spanning Trees, Kruskal/Prim Algorithms

What is a spanning tree?

- T is a spanning tree of graph G if T is a tree with the same vertices as G and a subset of the edges of G

What is a minimal spanning tree?

- A minimal spanning tree is a spanning tree whose total weight of all the edges is as small as possible

What is Kruskal's algorithm for producing an MST?

- Kruskal's algorithm – repeatedly add edges from low to high until you have added n – 1 edges

What is a disjoint set? How can it help in implementing Kruskal's algorithm?

- A disjoint set is a collection of linked lists; linked lists are easily merged when elements are unioned together; keys must be unique
- A disjoint set can be used to avoid cycles in Kruskal's algorithm; when examining an edge (u, v), add the edge only if u and v are in different sets; merge (u, v) when an edge is chosen

What is Prim's algorithm for producing an MST?

- Start with a single-vertex tree; vertices are either in or "not yet in" the tree; choose the lightest adjacent edge to vertices in the tree when selecting a new vertex; continue until all vertices are in the tree

Amortized Analysis

What is **amortized analysis**?

- Amortized analysis seeks to find the average cost of a data structure's operations in the worst case

What is the difference between the **aggregate method** and **accounting method** of amortized analysis?

- **Aggregate** – analyze a sequence of n operations that take a worst-case T(n) time total; average cost is therefore T(n)/n
- **Accounting** – assign costs to individual operations, then determine the cost to move an element completely through a data structure by "paying" for subsequent operations in advance

Dynamic Programming

How do divide and conquer and dynamic programming algorithms differ?

- In dynamic programming, answers for subproblems are stored and referenced instead of recomputing them
- Dynamic programming is often applied to algorithms which would solve many subproblems repeatedly
- Also applied to optimization problems

What is the time complexity of Dijkstra's shortest path algorithm?

- O(E log V) where E is the number of edges and V is the number of vertices

What is the LCS algorithm?

- The LCS algorithm is a dynamic programming algorithm for finding the longest common subsequence between two strings
- Subsequence = characters that may or may not be adjacent

What are the b and c tables used for in the LCS algorithm?

- B table contains arrows that specify how to navigate the output table to find the LCS
- C table contains numbers that specify LCS length in a particular cell

What is the time and space complexity of the LCS algorithm? Can space complexity be improved?

- The LCS algorithm runs in O(mn) time and takes O(mn) + O(1) space where n and m are the input strings' lengths
- Space complexity can be reduced to O(min(m,n)) + O(1) space

Optimization

What is an optimization problem?

- An optimization problem is a problem where many solutions exist (local minima/maxima) but only one is best (global minima/maxima)

What is an integer program? How is it different from a linear program?

- An integer program is a mathematical formulation of an optimization problem (usually NP-Complete)
- In integer programs, variables are restricted to integers
- In linear programs, variables can be real numbers

NP

How is a decision problem different from an optimization problem?

- A decision problem outputs a simple yes or no for each input
- Decision problems cannot be harder than their optimization problem counterparts, meaning that if the optimization problem is easy, the decision problem must be easy

What is a regular set? How is it different from a regular language?

- A regular set is a subset of A* (all possible finite combinations of characters in the alphabet A) that describes characters matched by a regular expression.
- A regular set is equivalent to a regular language

Describe the following grammar types:

- Type 0 – no restrictions
- Type 1 – in each production a ☾ b, length(a) <= length(b)
- Type 2 (context-free grammars) – in each production, the left-hand side is restricted to a single, non-terminal symbol
- Type 3 (regular grammars) – must be context-free; also, in each production, the right-hand side has at most one non-terminating symbol, which must be to the extreme right of the string

What is the difference between an **LR parser** and a **LL parser**?

- LL parser – left-to-right, leftmost derivation, top-down
    - Leftmost derivation – leftmost non-terminal chosen to be rewritten (i.e. substituted) first
- LR parser – left-to-right, rightmost derivation, bottom-up
    - Rightmost derivation – rightmost non-terminal chosen to be rewritten (i.e. substituted) first

Define the following problem classifications:

- **P** – class of problems that can be solved and checked (verified) reasonably fast in polynomial time
- **NP** – class of problems that can be checked (verified) reasonably fast in polynomial time, but whose solutions may not be easily found (exponential time or worse)

- **Co-NP** – class of problems whose complement is in NP
- **NP-Complete** – the hardest problems in NP; solving any NP-complete problem efficiently would give a solution to any problem in NP
- **NP-Hard** – at least as hard as the hardest problems in NP, possibly harder

What is a verification algorithm?

- A verification algorithm is used to check solutions to a specific problem
- Definition: A(x,y) where x is an instance of a problem and y is a proposed solution to the problem; returns 1 if the solution is correct, 0 otherwise
    - Example: Hamiltonian path – x is a graph G, y is a list of vertices establishing a path
- All NP problems have verification algorithms that run in polynomial time

What is the difference between tractable and intractable problems?

- Tractable – problem can be solved in polynomial time
- Intractable – problem is undecidable or all available algorithms run in exponential time

What is **reducibility**?

- A problem A is **reducible** to problem B if B's solution can be used to solve A in polynomial time
- Example: Hamiltonian path problem reduces to the longest path problem

Classify the following problems as P or NP:

- Shortest path - P
- Longest path - NP
- Critical path - NP
- Graph coloring (2 colors) - P
- Graph coloring (>=3 colors) - NP
- 2-CNF-SAT (also describe this problem) - P
    - Variables arranged as conjunctions of disjunctions (ands of ors); may be negated
    - Can be solved in polynomial time via an implication graph
- 3-CNF-SAT - NP
- Clique (largest complete subgraph) - NP
- Hamiltonian path (also describe this problem) – NP
    - Problem of finding a path that visits every node exactly once
- Hamiltonian cycle - NP
- Minimum vertex cover - NP
- Largest independent set (also describe this problem) – NP

    - Problem of finding the largest set of non-adjacent vertices in a graph
- Traveling salesman problem - NP
- Knapsack problem - NP
- Set coverage (also describe this problem)
    - Problem of finding the smallest collection of subsets whose union is a target set

String Matching / Binary Search Tree

In the KMP algorithm, what is the PI array? How is it used for pattern matching?

- The PI array can be thought of a function that receives an input q and outputs the longest prefix / proper suffix for a string s[0]…s[q]
- The PI array is used in the KMP algorithm to skip over areas of the input text that can't possibly match a pattern

How does the KMP algorithm's runtime compare with brute force pattern matching?

- KMP - O(T) where T is the length of the text
- Brute force – O(TP) where T is the length of the text and P is the length of the pattern

Approximation/Greedy Algorithms

What is a 2-approximation algorithm?

- Approximation algorithms are used to produce close (but not optimal) solutions to NP problems in polynomial time
- A 2-approximation algorithm guarantees that a solution will not be more/less than 2 times the optimal solution

In the Ford-Fulkerson algorithm, what is a residual network? What is an augmenting path?

- A residual network contains all nodes in the original flow network; edges represent residual capacity (i.e. how much capacity is left on an edge in the original flow network)
- An augmenting path is a path from source to sink in a residual network; goal is to find the minimum residual capacity available to increase flow in the flow network

In the Ford-Fulkerson algorithm, how are max flow and min cut related?

- Max flow – maximum amount of flow achievable from source to sink in a flow network
- Min cut – minimum sum of edge capacities when flow network is cut, i.e. partitioned into two sets of vertices with source and sink in separate sets
- Max flow = min cut

What is a fast-fourier transform (FFT) algorithm used for?

- FFT algorithms can multiply polynomials in O(nlg(n)) time, which is much faster than traditional multiplication in O(n) time.