CEN4088.01 **Brandon Thompson**
Prof. Elish Due Date: 12/11/19

Final Review

1 Designing for Quality

1.1 Industry Frameworks

Software design is the most important phase of software development as it is the last chance to uncover issues with security and functionality.
Two activities that are conducted during design for functionality are:

- Static design
- Dynamic design

1.1.1 Static Design

Notation consists of images, pictures and drawings in the form of UML and wireframe drawings or small prototypes.

1.1.2 Dynamic Design

Dynamic design breaks software requirements into smaller software modules.

- Application concept
  - Decomposes the application into separate domains.
  - List the concepts that make up the use case.
  - Associate the concept by connect concepts using business reasons.
- Application interaction: how each concept (user, application, resources) interacts with the others.
- Application sequence: System sequence diagrams model the communication between concepts.

1.2 Organizational Resources

Design artifacts (conceptual models, interaction diagrams, sequence diagrams) are not the end of the analyst's role. The developer needs to understand these models because they are the last chance to make sure you code the right stuff.

- Collaborate with the analysts
  - Whiteboard meetings
    - Provide open communication between analysts and developers.
    - Clear definition of prototypes and wireframes.
    - clear definition of data access.
    - Clear definition of system's sequence of events.
  - Asking the right questions.
    - Start by asking open ended questions to understand the analyst's vision.
    - Next switch to questions about GUIs and get information about the interfaces.
    - Finally, get type of transactions that are needed to satisfy the user's request.
  - Finalize the drawing
    - Design meetings will produce a lot of drawings and notations on the whiteboard.
    - Complete UML diagrams and distribute them to everyone.
  - Map the data
    - Data mapping is the best activity you can do to ensure that you have complete requirements.
    - The **inputs** are the data elements that are coming into the application from another source (screen, input device).
    - The **outputs** are the data elements coming from the application that were requested, updated, or created based on the input request.
  - Map the inputs
    - Developer need to define every input parameter. Input cleansing is the key for secure code.

  - Map the outputs
    * Determining if the data exists
      · Wireframe or printout of the report that is to be created, check off all output data elements that are required.
    * Designing the queries
      · Writing SQL may not be an option in all environments.
      · Querying the database is the only method possible that will identify if all the tables and columns are needed to satisfy the requirement.
      · Designing query now reduces development time and makes sure you have all necessary inputs.
    * How to execute the query
      · Dynamic SQL is the ability to have create a SQL query based on a list of inputs.
      · Static SQL is hard-coded SQL.
    * Justify the cost of running each query
      · Designing the queries so that they perform acceptably in the database is a developer's responsibility.
      · Measure database CPU time or MIPS (million instructions per second).

1.3 Quality in the Cube

Last call with business analyst is to make sure all business logic is intact, make sure there is a universal understanding of the business requirements, update the system documentation and print the documentation.

- Business logic bridging the GUIs to the Database
  - All requirements are written down (sometimes to much information can can be to much information.
  - human factors are the primary causes of design disasters.
  - Understand everything about the application before you design it.
- Universal Understanding

  - Form requirements into simple logical statements of expectation so that anybody who reads the instructions will be able to understand the requirements.

- Updating the system documentation
  - Done!
  - Print wireframes, database schemas, and the final use case.

1.4 Summary

- Software design is the most important phase of software development
- Static design activities are used to help the analyst and the developer understand the overall application and the software requirements.
- Dynamic design activities produce documentation that is used to describe the behavior and interaction of systems and components.
- Human factors are the primary cause of design disasters

2 Designing for Security

2.1 Industry Patterns

- First define what it is you are securing
  - What software modules need to be created?
  - What other programs will use those modules and how?
  - What types of properties go into those modules?
- Separate use cases by unit of work (function)
  - UML diagrams provide visual representation of the software so that each use case can be broken down and analyzed for common secure design patterns and reusable components.
  - Secure requirements include the use of SSL and/or digital certificates.
- Secure design activities are
  - Interaction diagrams.

        * Focus on how the software objects interact with themselves and other objects.
        * Interaction diagrams are the most important artifacts in design because they are the specifications to class diagrams.
        * Document what is being communicated between objects.
        * Messages link objects and show what objects are being called, what is being communicated and returned.
    - Class diagrams in UML.
        * Software requirements specifications for each module in an application.
        * Documents **Properties** and **Methods**.
    - The printing of the final design for development.
- Organizational regulations for implementing security.
    - Data transmitted
        * Data is sanitized.
        * Double check before use.
        * **Because most applications need to be web enabled, the main protocol to work with is HTTP and TCP/IP**
        * Secure data by using:
            · **Secure Socket Layer (SSL)**
            · **Code Signing**
            · **Personal Digital Certificates**
            · **Two-factor Authentication**
- Autonomous Team Decisions
    - Client-Side Validation
        * Browser executes code on the fly before sending data to the server.
        * Advantages:
            · Data can be validated before being sent to the server.
            · Works well with DHTML.

    * Disadvantages:
        · JavaScript can be disabled.
        · Hard to maintain and debug.
        · Allows users to view validation.
  - Server-Side Validation
    * Browser transmits data data via HTTP to the server so that a server side language can validate the data.
    * Advantages:
        · Logic stored on server.
        · Input data can be validated using database, session or other server attributes.
    * Disadvantages
        · Page requests for form submittal.
        · Unnecessary trips to server.
- SQL Injection

3 Java: Secure Coding Practices

- Rule 1: Ensure Sensitive data is kept secure
- Rule 2: Detect and handle file-related errors
- Rule 3: Generate strong random numbers
- Rule 4: Do not use `Object.equals()` method to compare two arrays.
  Use `Arrays.equals()` instead.
- Rule 5: Validate method arguments
- Rule 6: Avoid division by zero errors
- Rule 7: Do not attempt comparisons with `Double.NaN`
  Instead use `Double.isNaN()`.
- Rule 8: Limit accessibility of fields (private data fields)

4 Testing For Quality and Security

4.1 Concepts and Definitions

Run a battery of test cases with multiple techniques against a specific use case to evaluate if the test case will pass or fail.

- What Testing and Assurance are:
  - Software testing is the discipline of examining the functionality of application software to determine whether it meets the requirements (Verification) and satisfy the need (validation).
  - Software assurance (SwA) is ho effective security is tested and implemented within the software.
- What testing is not:
  - Security measure
  - Quality measure
  - Earlier testing is done the cheaper it is to fix.
- Test Plan
  - How testing should be performed.
  - Who executes the testing.
  - when testing starts / ends.
  - define test scripts.
  - define user community
  - Identify internal resources.
  - Identify external resources.
- Problems with testing in the industry
  - Phase that is most looked over because of budget / time constraints.
  - To much reliance on individual skills
  - Unpredictable users.

4.2 Dynamic testing in the Organization

Executing the current use case using valid inputs and then comparing the real results with expected results for pass or failure.

- Unit testing
  - Lowest level
  - exception handling: manually going into source code and forcing an error or attack.
  - Reviewing error logs
- Integration testing
  - two or more modules are linked together and tested
  - Required testing the while use case rather than an individual piece of software.
- System testing
  - Complete front-to-back execution of the entire application.
  - Each new use case that is added to the system must be tested for quality features (reliability, usability) and security.
- Fuzz testing
  - Feeds random input data into applications to see hat happens.
- Reliability testing
  - Application should produce correct results despite being under attack.
  - requires simulation tools to replicate number of expected uses, time of day, and software environment as if it was the real world.

4.3 Static Testing

Way of evaluating the code without executing the application. Does not evaluate performance.

- Static code analysis
- Code review and inspections
- System documentation.

5 Encryption and Password cracking

- Symmetric Key Encryption
  - Uses same key to encrypt and decrypt data.
  - inherent vulnerability in that key must be transmitted.
  - faster than asymmetric key algorithms.
- Asymmetric Key Algorithms
  - Public key algorithms
  - two keys for encrypting and decrypting