Business Rules

**Types of business rules**

* Structural Assertion
  + A defined concept or fact of an aspect of the structure of an enterprise.
  + **Terms:** A word of phrase with specific meaning for an enterprise
    - Course, section, rental car, passenger.
  + **Facts:** an association between two or more terms. A simple declarative statement.
  + **Example:** A customer may request a model of car from a rental branch on a particular date.
* Action Assertion
  + A statement of **constraint** or condition that limits the action of the enterprise.
  + **Example:** A car **must** have a registration number
* Derivation
  + A statement of knowledge that is derived from other knowledge in the business
  + **Derived Fact:** this is created by inference or mathematical calculations from terms, facts, etc.
  + **Example:** The total cost of the rental is calculated from the sum of its insurance amount, rental amount and late charge.

**Business Rules Overview**

* A rule that defines or constrains one aspect of your business that asserts business structure or influences the behavior of the business.
* **Constraints:** Conditions that must hold true in specific situations
* Defines the conditions under which a process is carried out.
* Defines what **MUST** be the case. It is **NOT** a description of the process.
* List of statements of what you may or may not do.
* Helps make decisions.

**Business Rule Functions**

* Often Focus on access control issues (Teachers are allowed to modify student marks, but only their students.)
* May pertain to business calculations. (A teacher must convert a mark to a letter grade)
* Focus on policies of your organization.

**Power of Business Rules**

* They have the ability to make direct business-level statements that become part of an operational system.
* They are multiple simple statements that are grouped together to have a great impact.

**Analyst’s Job**

* To specify a series of clear logic statements for a business.
* Making the business logic understandable
* Don’t use technical terms. Keep it simple.

**Business Rule Rules…**

Business Rules must be:

* Atomic: Cannot be broken down any further without losing information
* Precise: Have only one obvious interpretation
* Declarative: statements of policy
* Compact: Single short sentence
* Consistent: Together provide a coherent description
* Distinct: Not redundant
* Compatible: Use same terms as the rest of the business model.

**Gathering Business Rules**

* Organizational documents such as manuals, policies, contracts, marketing brochures.
* Interview notes from individual and group information systems.
* Description of business functions, events, policies, units, etc.
* Ask questions such as
  + Is this always true?
  + Are there circumstances when an alternative occurs?
  + Is there a need to keep history of those?

**Expressing Rules**

* <Subject> must <constraint>
* The subject and constraint can be complex
* Start to form the entities that are used to create your classes

**Potential Problems**

* Using facts
  + Avoid vague terms like “there”. There may be no more than …..
  + Avoid obscure facts or terms. Must identify party involved
  + Question the terms used. Make sure they match the business.
  + Use a fact model. Rules can relate to other parts of the business model.
* Simple constraints
  + Avoid starting with “if”
  + Avoid complex rules
  + Avoid “and” and “or”.
  + Avoid padding. (adding extra words that mean nothing)
* Quantification and qualifications
  + Avoid plural forms.
  + Use “each” and “every”
* States and events
  + Avoid events as subjects
    - By close of business, all withdrawals must be notified to the head office
  + Avoid ambiguous states
    - The outstanding balance on a secondary account that is being closed must be transferred to the primary account before it is terminated.
  + Avoid ambiguous time frames
    - A new account must be approved by at least two managers in the final quarter
  + Avoid “when”.
* Actors
  + Question actors. Are they necessary?
  + Avoid actors as subjects because actors change and roles that do certain jobs change as well.
    - A customer representative may issue a replacement charge card only if the old card expired within the last 30 days.
* Dangerous verbs
  + Avoid command verbs
  + Avoid action verbs that create unclear definitions
  + Look carefully at CRUD words.
    - Words that imply implementations
    - update the account balance… the account balance is…
* Computation
  + Avoid embedded computations
    - Make them 2 separate rules.
  + Make computations the subject of the rule.
* Structure and consistency
  + Check for overlapping, duplications, inverted rules
  + Check for contradictory rules as well.

Intro to Systems Analyst

**Causes of Problems in Systems**

* Lack of methodology
* Poor communication
* Complex systems
* Unreasonable expectations about the development process
* Poor or no documentation

**Other Issues**

* Software is inherently complex
  + There may be interdependencies between components and it may be complex data intensive systems
* Conformity
  + Software must conform to existing hardware/software platform, and pre-existing information systems.

**Goal of a System**

* The primary goal is to create value for an organization

**What is a System?**

* A collection of interrelated components that function together to achieve an outcome.
* Key Components
  + People
  + Processes
  + Hardware
  + Software
  + Data
* Example: Moodle, Omnivox, Payroll System

**Characteristics of a System**

* A level of complexity
* Has definite purpose or function
* May have subsystems or be part of a larger system
* Components arranged according to some plan or structure

**Analysis**

* The resolution of anything complex into simple elements.
* The exact determination of its components

**Methodology**

* A process model composed of tasks, work products and roles which help to consistently and cost-effectively achieve a specific objective.

**Lifecycle / Methodology**

* Uses phases where each phase has a standard set of outputs
* **Planning**
  + Identifies business values, develop a work plan, staff the project, control and direct.
* **Analysis**
  + Information gathering, process modeling, data modeling
* **Design**
  + Physical design, interface design, database and file design, program design
* **Implementation**
  + Construction, installation
* **Post-Implementation (Maintenance)**
  + Bug fixes, on going maintenance, hardware maintenance, enhancements, etc.

Project Teams

**Project Team Skills**

* Members of a team are change agents who find ways to improve their organization
* Skills involve interpersonal, business, analytical, technical, management, and ethical skills.

**Customer Roles**

* Financial Stakeholders
  + People who have a financial stake in a project.
  + They pay the bills
  + Users and system owners. Although may sometimes not be the users.
* Subject Matter Experts
  + Provide business knowledge to analysts on business process and methodology
  + Provide feedback on system on whether it meets all business requirements.
    - Legal, policy, corporate, standard
* System/Business Owner
  + Person responsible for the system.
  + Person who must accredit the system to run as part of the corporate infrastructure.
  + Not usually a technical person
  + Provides funding on project
* Stakeholders
  + Customers or developers
  + Day-to-day work is affected by the success of the project.
* Committees/Boards
  + Executive Committee
    - Approve project and provide support
  + Steering committee
    - Control the direction of the project
  + Change Control Board
    - Responsible for changes in project. (Release, update, etc)

**Project Roles**

* Project Manager
  + Manages the team of analysts, programmers, technical writers, testers, etc.
  + Develop and monitors the project plan
  + Assigns resources
  + Primary point of contact
* Business Analyst
  + Analyzes the key business aspects of the system
  + Identifies how the system brings value to the company
  + Designs business process and policies
* Systems Analyst
  + Identifies how technology can improve the business process
  + Designs the information system
  + Problem solver rather than programmer
  + Ensures that the system conforms to information system standards
  + What do they do?
    - Research and understand the problem
    - Verify that the benefits of solving a problem outweigh the costs
    - Define requirements for solving the problem
    - Develop a set of possible solutions
    - Decide which solution is best (make a recommendation)
    - Define details of chosen solution
    - Implement the solution
  + Skills Needed
    - Technical knowledge – databases, files, hardware, software, networks
    - Business knowledge – how organizations are structured, work that goes on in an organization
    - Interpersonal skills – effective communicator, interviews, technical reviews, presentations.

* Infrastructure Analyst
  + Ensures that the system conforms to infrastructure standards
  + Identifies infrastructure changes needed to support the system
  + Knowledgeable about networking, database admin, hardware and software products
* Architects
  + Solution Architect
    - Defines the “big picture” of the solution
  + Technical Architect
    - Defines technical components of business solution
  + System Architect
    - Responsible for how the system fits into overall business model
* Change Management Analyst
  + Develops and executes the change management plan
  + Sometimes develops and executes user training
  + Needs to control the change to make sure it does not affect system delivery
* Testers
  + Provide quality control along all the steps of development
  + Recommends ways to improve processes
* Developers
  + Develop and test the system as specified during the analysis and design
  + Integrate various system components to form a complete system or subsystem.

System Development Life Cycle

Why do systems fail?

* Poor requirement gathering
* Insufficient analysis
* Poor management

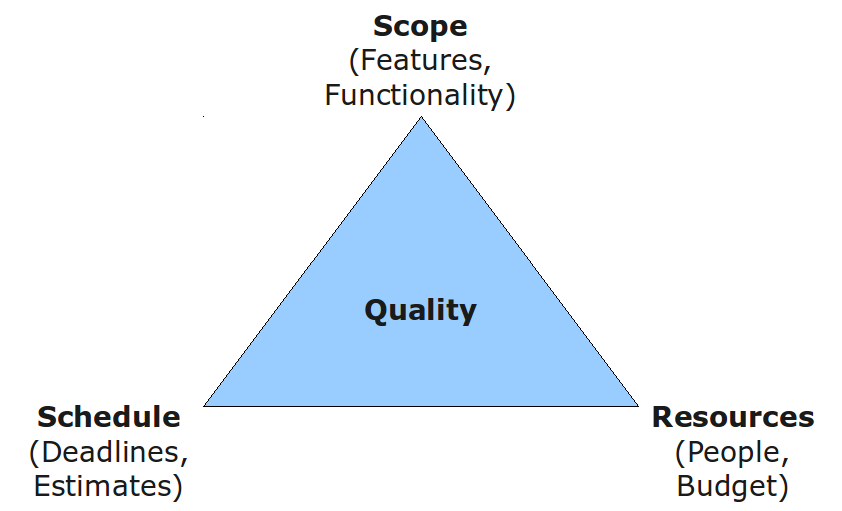
**Phases of SDLC**

* Planning
  + Project initiation
    - Develop a system request
    - Conduct feasibility analysis
  + Project Management
    - Develop a work plan
    - Staff the project
    - Control and direct the project
  + Why should we build that system?
* Analysis
  + Develop analysis strategy
  + Gather requirements
  + Develop a system proposal
  + What should the system do for us? Where and when will it be used?
* Design
  + Develop a design strategy
  + Design architecture and interfaces
  + Develop databases and file specifications
  + Develop the program design
  + How will we build the system?
* Implementation
  + Construct System
  + Install the system
  + Establish a support plan
  + Implement training plan for users
  + Build the system!

Characteristics of SDLC

* Each step leads to specific deliverables
* Phases are interactive
* **What is a deliverable?**
  + Proof that you have completed the current phase and you can proceed to the next.
  + Examples
    - Planning -> Project plan
    - Analysis -> System Proposal
    - Design -> System specifications
    - Implementation -> New system and maintenance plan

**Project Management Triangle**



**Methodology**

* Formalized approach to a series of steps
* Well known methodologies include waterfall development, agile, scrum, etc.

**Methodology Categories**

* Predictive, structured development
  + Projects move methodically from one to the next step
  + Very controlled approach
  + A step is finished before the next begins
  + Recommended for high risk systems
    - Waterfall
      * A: Identifies system requirements long before programming begins and minimizes changes
      * D: Design must be complete before implementation begins. Long time lapses
    - Parallel
      * Design for the project is divided into subprojects (Pilot projects)
      * They are all delivered one at a time
      * Once all projects are delivered they are integrated together to deliver the system.
* Rapid application development (RAD)
  + Phased
  + Prototyping
    - Create a “quick and dirty” look at the system
    - Minimal features
    - Show it to others and get their feedback
    - Re-analyze, re-design, re-implement
    - A: Quick feedback from users
    - D: Users constantly not happy, when are you done? May sacrifice careful analysis
  + Throwaway prototyping
    - When you don’t really understand what others want.
    - Thorough analysis, move to design but do a prototype to make sure you understand.
    - A: feedback on difficult misunderstood parts
    - D: Extra time required.
* Adaptive, agile development
  + 4 core values: Communication, simplicity, feedback, courage.
  + Rapid feedback between developers
  + Simple, rapid iterations of application
  + **Emphasizes:**
    - **Individuals and interactions** over processes and tools
    - **Working software** over comprehensive documentation
    - **Customer collaboration** over contract negotiation
    - **Responding to change** over following a plan
  + Extreme programming
  + Scrum
    - 15 minutes daily meeting where each person says
      * What they’ve done
      * What they’re going to do,
      * Any problems/coordination/help
    - Scrum projects work as a series of sprints.
    - A **sprint** is a duration of time no longer than 30 days
      * A sprint has a specific goal
      * Sprint team has final say in estimating and determining what they can accomplish

Factors in selecting the proper methodology

* Clarity of user requirements
* Familiarity with technology
* System complexity
* System reliability
* Short time schedules
* Schedule visibility

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Usefulness for** | **Waterfall** | **Parallel** | **Phased** | **Prototyping** | **Throwaway Prototyping** | **Agile** |
| Unclear user requirements | Poor | Poor | Good | Excellent | Excellent | Excellent |
| Unfamiliar technology | Poor | Poor | Good | Poor | Excellent | Excellent |
| Complex systems | Good | Good | Good | Poor | Excellent | Poor |
| Reliable systems | Good | Good | Good | Poor | Excellent | Good |
| Short time schedule | Poor | Good | Excellent | Excellent | Good | Excellent |
| Schedule visibility | Poor | Poor | Excellent | Excellent | Good | Good |

Project Vision

**Conducting an Interview**

* Appear professional
* Record all information
* Separate facts from opinion
* Give interviewee time to ask questions

**Interview Practical Tips**

* Don’t worry be happy
* Pay attention
* Be honest
* Summarize key points

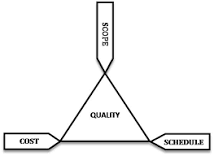
**Post Interview Follow Up**

* Prepare interview notes
* Prepare interview report
* Look for gaps in questions

**Vision**

* Used in both agile and non-agile world.
* Used to answer:
  + What problem will the system solve?
  + What features and benefits will it provide?
  + Who does it provide it for?
  + What performance, reliability, etc. will it deliver?

**Project Constraints**

* Constraints are outside of your control
* Imposed by client, organization or government
* 3 main types:
  + Cost
  + Time
  + Scope

Requirements Analysis

**The systems analysis phase**

* Identify need for a system (System request)
* Perform feasibility analysis (not learned yet)
* Gather information using requirement gathering techniques
* Define system requirements
* Prioritize requirements
* Prototype/evaluate

What is a requirement?

* It’s a statement of what the system must do and what characteristics it must have
* It is the “What” of the system. It’s written in a business perspective
* They later become the “How” of the system, but only later.

**Functional VS Non-Functional requirements**

* Functional: relates directly to a process the system must perform or information it needs to contain
* Non-Fictional: refers to behavioural properties

**Non-Functional Requirements**

* FURPS+
  + Functionality
  + Usability
  + Reliability
  + Performance
  + Supportability
  + Operational
* Operational: Deals with the physical and technical environment that the system will operate
* Performance: Deals with speed, capacity, and reliability of system
* Security: Deals with access. Who can access the DB, who can see what and do what.
* Cultural and political: legal requirements that affect the system, political factors.
  + Currency
  + Time zones

**Vision**

* This is a document used in both Agile and non-agile development. It answers:
  + **What problem** will it solve
  + **What features and benefits** will it provide
  + **Who** does it provide it for?
  + **What** will it offer in terms of **performance, reliability**, etc.
  + **What platforms, standards,** will it support?

Feasibility Analysis

**Purpose:** To discover the business value of a system.

**Traits of a Successful Project**

* **Cost:** The project did not cost more than was originally allocated
* **Schedule:** Delivered on the original delivery date
* **Performance/Quality:** The project includes all of the functionality originally wanted.

**Project Identification**

* Projects are driven by business needs and are identified by:
  + Business people
  + IT people
  + Both
* **Project Sponsor:** believes in the system and wants it to succeed. Usually a business person and has authority to move things forward.

**Initiating Projects**

* Projects begin when someone sees an opportunity to create business value from IT

**Identifying Business Value**

* Small businesses are easier to plan for since they can take advantage of a lot of new opportunities
* Large businesses must be looked into carefully into the future outcome.

**Feasibility Analysis**

* **Feasibility Analysis:** Used to aid in the decision of whether or not to proceed with the project. (is it worth it?)
* Identifies: *Cost, Risk, Benefits*
* **Types of feasibility analysis:**
  + Technical (Can we build it?)
    - Familiarity with technology
    - Familiarity with application
    - Project size
    - Compatible with current systems?
  + Economic (Should we build it?)
    - AKA Cost-Benefit
    - Identify financial risks
    - Cost:
      * Development
      * Operational
    - Benefit:
      * Tangible: Can be quantified and measured (2% increase in revenue)
      * Intangible: cannot be quantified and measured (improved customer service) It is more of an intuitive feeling.
    - **Steps in Economic Feasibility:** 
      * Identify Costs, Identify Benefits, Identify Technique (ROI, Break-even)
      * ROI: (Total Benefits – Total Cost) / Total Cost
      * BEP: (Yearly Net Cash Flow – Cumulative Net Cash Flow) / YNCF
  + Organizational (Do we want to build it?)
    - Stakeholders consider:
      * Project sponsors
      * Managers
      * System users
    - Is the project right for the business?
      * Strategic alignment with business
    - Questions to ask:
      * Will the system result in layoffs?
      * Will the system require training?
      * Will performance decline?
  + Schedule (Do we have time to build it?)
    - Can the project be implemented in the proper time frame?
    - Questions to ask:
      * Can the company control factors that affect schedule feasibility? (Do we have slack?)
      * Will there be a project manager to manage the time?
* Evaluating Feasibility
  + Identify system requests that are not feasible
  + Determine if the system is really necessary
  + Look at alternatives such as other technologies, purchasing a solution, etc.
* Project Selection
  + Once analysis is done, it’s submitted to approval committee.
    - They either approve, decline or request more information
  + Once approved, it enters project management process.
* Approval Committee
  + Senior Management
  + Meet regularly to discuss new and existing projects
  + Resource allocation
  + Discuss benefits of overall project.

**Classifying a project**

* Size, Cost, Purpose, Length, Risk, Scope, ROI

**How NOT to select a project**

* First in, first out
* Any method that does not conduct an analysis
* Political clout of project inventor

Requirements Gathering

What is a requirement?

* It’s a statement of what the system must do and the characteristics it must have
* Written in a business perspective (“what” of system)
* Requirements become more technical along the way

**Requirements Determination**

* Part of analysis where the project team turns a high-level explanation of business requirements into a precise list of system requirements
* AKA requirements discovery

**Requirements Gathering**

* Techniques used during the requirements determination to define the system requirements
* **The Process:** 
  + Establish trust and rapport with the users
  + Get buy-in from the users
  + MUST talk to all key stakeholders (employee, owners, suppliers, customers)
  + Determine ways in which you will gather the requirements such as:
    - Interviews
    - Questionnaire
    - Document analysis
    - Observation
    - JAD sessions
    - Domain research

**Requirement Gathering Techniques**

* **INTERVIEWS**
  + Select Interviewees
    - Based on information needed.
    - Often good to get different perspectives so try to get variety such as managers, users, or ideally all stakeholders
    - **TYPES OF QUESTIONS:**
      * Open (What do you think about the system? What problems do you have?)
      * Closed (How many orders are received a day? Are runs assigned?)
      * Probing (Why, can you explain further, can you give an example?)
  + Design interview questions
    - Unstructured:
      * Used to get broad information
      * Use early in the process
    - Structured:
      * Used to get more specific questions
    - Organize questions in a logical sense
    - Strategies:
      * Bottom up: very specific -> moderately specific -> very general
      * Top Down: very general -> moderately specific -> very specific
  + Prepare for the interview
    - Prepare list of questions, anticipated answers, and follow-ups
    - Confirm areas of knowledge of interviewees
    - Set priorities in case of time shortage
    - prepare interviewees before the interview.
      * Give schedule
      * Inform of reason of interview
      * Inform on areas of discussion
  + Conduct the interview
    - Appear professional
    - Record all information
    - Separate facts from opinions
    - Give interviewees time to ask questions
    - Be sure you understand all issues at hand
    - Be honest
    - Watch body language
    - Summarize key points
    - Pay attention
  + Follow-up with post-interview information
    - Look for gaps and new questions
    - Prepare interview report
    - Prepare interview notes
  + **Advantages**
    - Establishes rapport with users
    - Allows to probe for more information/feedback
    - Questions can be adapted for each interviewee
  + **Disadvantages**
    - Very time consuming
    - Highly dependent on analyst’s interpersonal skills
    - May be difficult due to physical location of interviewees.
* **QUESTIONAIRES**
  + It is a set of written questions for obtaining information
  + Usually used when the system will spread to multiple locations or is used outside the organization.
  + Mass produced and distributed to respondents
  + Possibility to do it electronically
  + **Steps:** 
    - Select participants
    - Design questionnaire
      * Begin with non-threatening and interesting questions
      * Group items logically
      * Put important items at the top, not bottom
      * Don’t crowd a page with too many items
      * Avoid abbreviations
      * Number the questions
      * Pre-test the questionnaire to receive feedback
    - Administer the questionnaire
    - Follow-up on responses.
  + **Completion Rates:** 
    - Are participants completing and returning the questionnaire?
    - How to improve:
      * Explain why it’s being conducted
      * Explain why the respondents have been selected
      * Provide a completion date
      * Provide an incentive to complete such as giveaways
      * Offer to provide a summary of responses
  + **Advantages:**
    - Can be answered quickly
    - Relatively inexpensive
    - Allow individuals to remain anonymous
    - Responses can be gathered quickly and automatically
  + **Disadvantages:** 
    - Responses are usually low
    - Answers may not be detailed enough
    - Users cannot ask follow-up questions
    - Tough to prepare good questions
    - Questions can be misinterpreted
* **DOCUMENT ANALYSIS**
  + Provides clues about the “as-is” system
  + Typical documents are:
    - Forms
      * Look for unused form elements
      * Look for additions to form
    - Reports
    - Policy manuals
  + **Advantages:** 
    - Provides good insight of as-is system
    - Can provide background information on requirements
  + **Disadvantages:** 
    - may only tell formal system not the real system
    - left up to interpretation of the analyst
* **OBSERVATION**
  + Since managers don’t always remember what they do, then go watch them
  + Behaviors change when people are watched
  + Avoid ignoring periodic activities
  + Do not influence the person working
  + **Advantages:**
    - Data is very reliable
    - Able to see exactly what is being done
    - Relatively inexpensive
    - Analyst can do the work measurements
  + **Disadvantages:**
    - People can feel uncomfortable
    - You must observe for a long time
    - Can be inconvenient to schedule
    - Tasks observed are subject to interruption
    - Workers can perform tasks differently when being watched.
* **DOMAIN RESEARCH**
  + Involves researching relevant locations, websites and determine what the other competitors are doing
  + Determine requirements based on what others are offering
  + See what is good and bad (the things you want to do vs the things you want to avoid)
* **JOINT APPLICATION DEVELOPMENT (JAD)**
  + Allows project team, users and managers to work together to identify the requirements
  + Highly structured meeting to gather requirements
  + Often the most useful method for collecting information from users\
  + Scope creep is reduced by 50%
  + Avoids requirements being too specific or too vague
  + 10-20 participants led by a facilitator
  + E-JAD is becoming popular
  + Key Roles:
    - Facilitator: sets the meeting agenda and guides the discussion
      * Keep session on track
      * Help with technical terms
      * Record group input
      * Help resolve issues
    - Scribe: records notes, making copies, etc.
  + JAD Setting
    - U-Shaped seating
    - Away from distractions
    - Whiteboard/flip chart
    - Prototyping tools
  + Lasts about 5 – 10 days over a three-week period
  + Prepare questions as with interviews
  + Formal agenda and ground rules
  + **Managing problems in JAD sessions**
    - Encourage non-contributors
    - Reduce domination
    - Side discussions
    - Agenda merry-go-round
    - Use humor
    - Violent agreement

**Requirements for Prototyping**

* Convert basic information into something tangible for the user
* Give it to the user to spark further conversation
* May make user think of other things such as missing fields, too many fields, data calculations
* **Prototyping is useful when:**
  + Requirements are unclear
  + Few users/stakeholders
  + Complex designs need to be validated
  + History of communication problems
  + Tools are readily available
* **Prototyping Drawbacks**
  + Replace formal documentation
  + Geared toward small set of users
  + Built as standalone system without accounting for security, sharing data, interactions, etc.
  + Users may think it’s the final answer!

**Agile Methodology Requirements**

* Continual user involvement
* Agile usage-centered design:
  + Similar to JAD sessions
  + Gather all the experts together for a meeting
* User stories
* Brief (1 or 2 sentences) of each piece of functionality

**Usage-Centred Requirements Gathering**

1. Gather all the experts together
2. Give everyone a chance to vent
3. Determine user roles and write them down (Index cards)
4. Determine tasks per user role
5. Group task cards by similarity
6. Write task description on card
7. Treat each task as a set to be supported by a single interface
8. Prototype and refine each task set

Use Case Descriptions

**Key Ideas**

* Illustrates the activities that are performed by users of a system
* They are logical models they describe the activities of a system without specifying how they are implemented. (What not How)

**Use Case Descriptions**

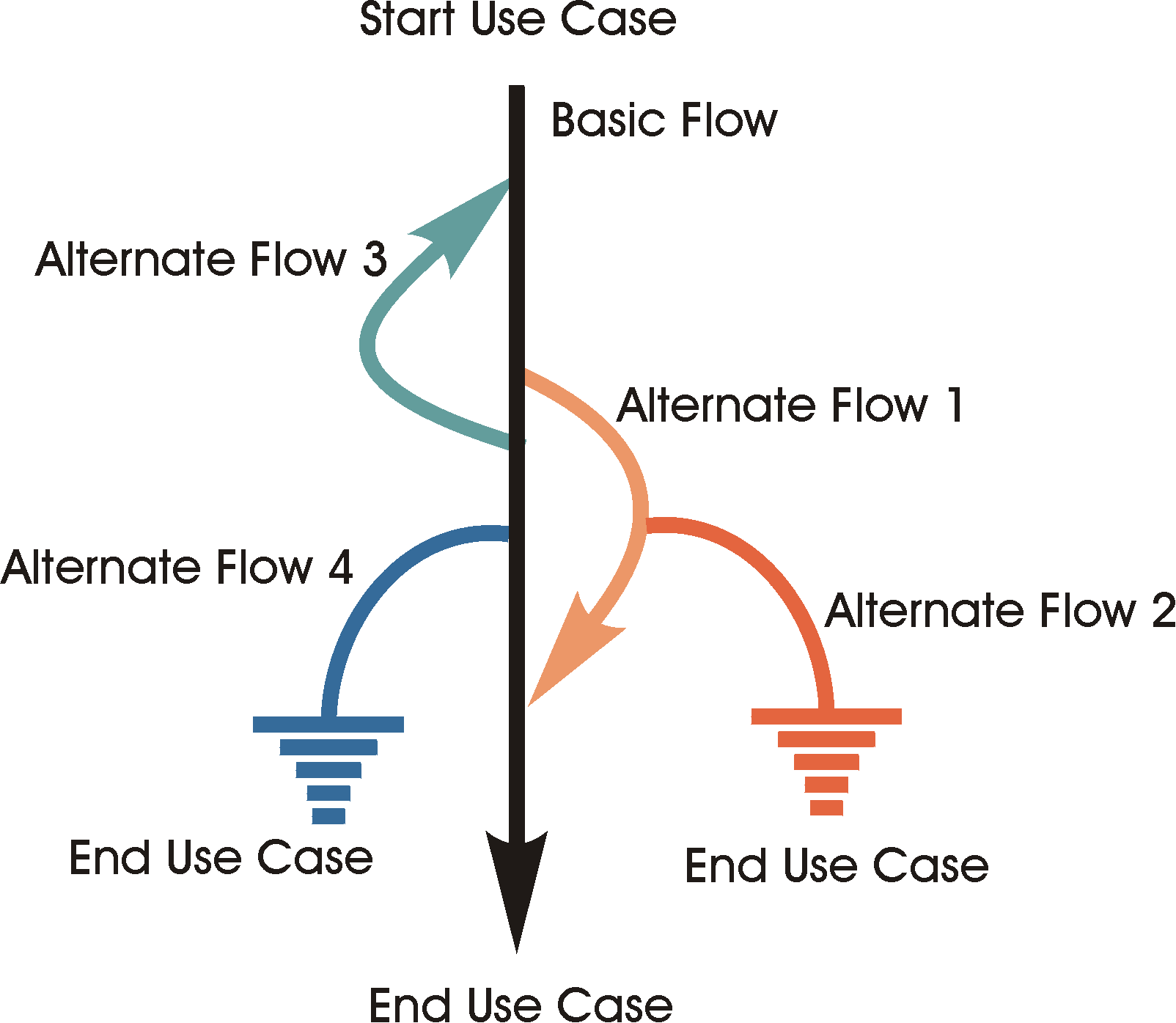
* Describes the basic functions of the system
  + What the user can do
  + How the system responds
* Building blocks for continued design activities
* **“The use case starts when”**
* **“The system does this… the actor does that”**
* **“The system does this…, the use case ends”**

**Creating Use Case Descriptions**

* Two Steps:
  + Write text-based description
  + Translate into a diagram
* Describes only one function but can have many alternative paths

**Types of Use Cases**

* Overview vs Detail
  + **Overview:** Defines high level requirements like the name, actors, and a brief description
  + **Detail:** Detail of all the information needed for the case like primary flow, alternate flow, etc.)
* Essential vs Real
  + **Essential:** Only providing the bare minimum to understand the functionality
  + **Real:** Going into further detail to describe specific sets of steps.

**Use Case Format**

1. Use Case Name
2. Description
   1. Verb-Noun format
   2. State what it is and why it exists
   3. Only needs to be a few sentences
   4. **Bolded words are in the glossary**
3. Actors
4. Pre-conditions
   1. Conditions that must be true about the state of the actor and/or system BEFORE the use case can start
   2. They do not start a use case
5. Basic flow
   1. It is the expected path (Happy path)
   2. Describes the normal way the actor interacts with the case
   3. Describe how the case ends
   4. Start by clearly defining the actor that starts the use case
6. Alternate flow
   1. Less common, optional behaviour or error conditions
   2. Cover exceptional behaviour
      1. Specific: Start at a specific point
      2. General: Can start at any point
      3. Bounded: can only occur in a range
   3. Named and numbered
   4. At {extension point} when/if <something happens>
7. Sub-flows
   1. Self-contained, named sections with their own clear purpose
   2. Complex events can be broken down
   3. The sub flows don’t need to be executed but it improves readability
   4. Needs to be named and numbered
8. Post conditions
   1. Conditions that must be true about the state of the actor and/or system AFTER the use case
9. Special requirements
   1. Requirements don’t fit nicely in a use case
   2. They are often in a separate document but can be in a special section of the use case
   3. Types of supplementary requirements (basically non-functional requirements)
      1. Legal & regulatory
      2. Quality, usability, reliability

**Prioritizing Use Cases**

* High, Medium, Low depending on:
  + Importance to business process
  + Complex, risky or time critical functionality
  + Technology needed to support needs research
* **Higher number = higher priority (0-5)**

**Defining Flow of Events**

* Start with “The use case starts when the…”
* End with “The use case ends”
* DON’T describe the user interface
  + Good: “When the customer selects to browse”
  + Bad: “When the customer clicks the link to browse”
* Start every action with “The actor” or “The system”
* Answer “What” not “How”
* Describe things so an outsider can understand them.
* Keep it simple
* **Things to avoid**
  + **Adverbs:** very, more, rather
  + **Vague terminology:** Information, appropriate, required, relevant
  + **Compound statements:** and…if….then….and….where…..and…..therefore
* Use correct punctuation
* Make glossary terms bold

**Extension Points {this is the format}**

* Named places in the flow of event where additional behaviour can happen
* No naming conventions but make it clear
* Put them in the left margin or their own line
* Can be used once
  + Example: {display product catalogue}
* Or multiple times
  + Example: {out of stock}

**Glossary**

* Supporting documentation for a use case description
* A set of shared definitions of things in the problem domain
* Domain model

**Some guidelines**

* Write in an active voice (“the system validates” over “the amount should be validated”)
* Write in present tense (“The system validates” not “the system will validate”)
* Write in newspaper style (simple, top down sentences)
* Ensure a sensible set of steps
* Use KISS principle

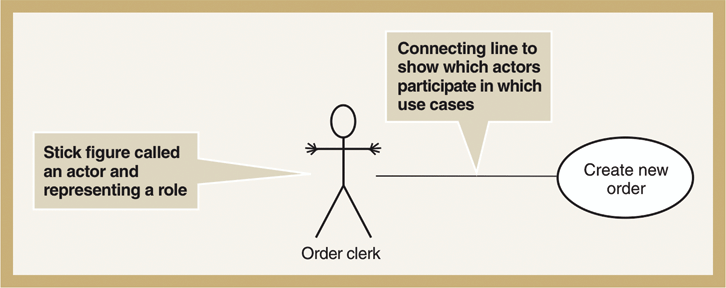
Functional Modeling

**Requirements Challenge**

* Toughest part in a system design is to elicit the correct and necessary system requirements from the stakeholders

**Use case modeling**

* Process of modeling a system’s functions in terms of **business events, who initiated that event, and how the system responds to the events**

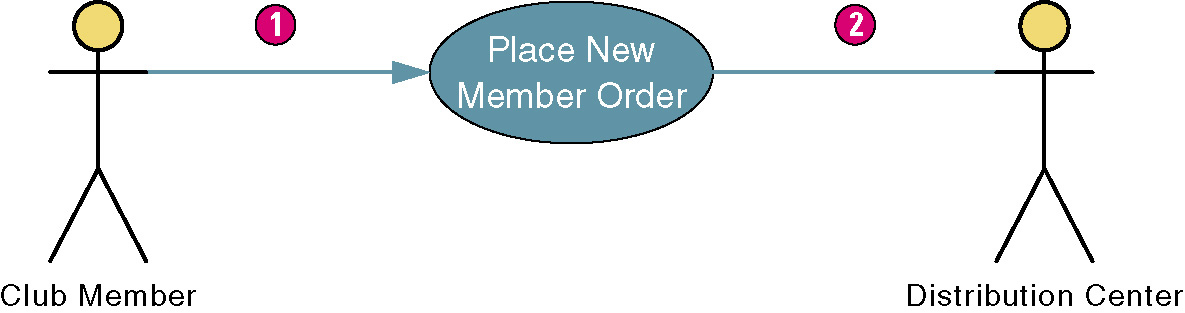
****

**Use-Case modeling**

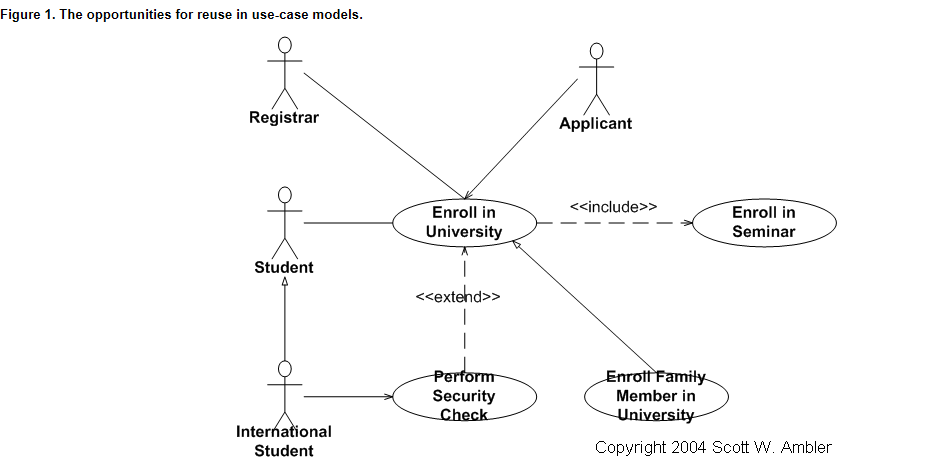
* Use-Case Diagram
  + A diagram that depicts interactions between the system, external systems and users.
  + Graphically represents who uses the systems and what ways the users interact with them.
* Use-Case Description
  + Textual description of the business event and how the user will interact with the system to accomplish a task
* Use-Case
  + A sequence of steps for completing a single business task
  + Represents a major piece of system functionality
  + Subset of overall system functionality
  + Describes the system’s functions from the perspective of an external user.
* Actor
  + A person or system that derives benefit from and is external to the subject.
  + Labeled with role (noun)
  + Anything that interacts with the system to exchange information
  + Can be associated with other actors
  + Types
    - Primary business actors
      * The stakeholder the primarily benefits from the execution of the use case (employee receiving a paycheck)
    - Primary system actor
      * Stakeholder that “triggers” the event (bank teller deposit check)
    - External server actor
      * Stakeholder that responds to a request from a use case (credit bureau authorizing a credit card exchange)
    - External receiver actor
      * Stakeholder that is not primary actor but receives something of value (warehouse receiving a packing slip)
* Illustrates the activities that are performed by users of a system
* Describes basic functions of a system
  + What a user can do
  + How the system responds
* **Benefits:** 
  + Serves as a tool for capturing functional requirements
  + Helps to breakdown system into more manageable pieces
  + Provides a means of communicating with users and other stakeholders to make sure everything is properly understood
  + Serves as aid to determine project scope, effort and schedule
  + Provides a baseline for testing in terms of defining test plans/cases

**Use Case Syntax**

* Relationships
  + Association (----------)
    - A relationship between an actor and a use case where an interaction happens between them.
    - Can be bidirectional or unidirectional
    - Solid line connecting actor to use case
    - (1) is the initiator
    - (2) is the receiver



* + Include (- - - - - - - -> <<includes>>)
    - Includes mean that a functionality can be re-used
    - It can be use for multiple cases
    - Indicates a dependency on another use case
  + Extend (- - - - - - - -> <<extends>>)
    - Adds functionality it does not re-use any functionality.
    - Indicates
      * Adding new features/capabilities to a use case
      * Any optional use cases
    - Used to add steps to a primary use case
  + Generalization ( ----------------->) aka inheritance
    - Indicates common behaviour between two actors.
    - Student and international student below
    - Inherits from the interactions of its parent.

****

**Steps in Writing Use Cases**

1. **Identify the business actors**
   1. Ask the following questions to identify them
      1. Who or what provides inputs to the system?
      2. Who or what receives outputs from the system?
      3. Are interfaces required to other systems?
      4. Who will maintain that information in the system?
   2. Each actor must have a describing role
   3. Can identify goals of each actor in the description
   4. Needs a description of role
2. **Identify the major use cases**
   1. Identify system boundaries
   2. Identify and write major use cases
   3. Has a verb-noun name and a brief description
   4. Some questions to ask:
      1. What are the main tasks of the actor?
      2. What information does the actor need from the system?
      3. What information does the actor provide to the system?
3. **Expand the major use case**
   1. Choose one major use case to expand
   2. Come up with a use-case template containing flow of events
   3. Come up with happy path
   4. Determine alternative flows
   5. Create sub-flows
   6. Simplify
4. **Confirm the major use case**
   1. Review the current set
   2. Involve users to get feedback
   3. Iterate the entire set of steps until all use cases are defined
5. **Create the use case diagram**
   1. Start with system boundary
   2. Place use cases on the diagram
   3. Place actors on the diagram
   4. Connect the actors with association lines

Interface Design

**What is a user interface?**

* Describes how users interact with a computer system
  + Gets input from the user
  + Provides output to the user

**When does user interface design occur?**

* Design phase
* User-centered design
* Usually done with prototyping

**Principles**

* Layout
  + Each area may be further subdivided
  + Each area is self contained
  + Areas should have a natural intuitive flow
    - Left to right
    - Top to bottom
* Content Awareness
  + Must intuitively answer:
    - Where am i?
    - What am I supposed to be doing here?
  + Applies to sub areas within a form or window
    - Address information is grouped together
    - Report information is grouped together
* Aesthetics
  + Interfaces should be inviting to use
  + Readable
  + Proper colors and fonts
  + Less is more
  + White space is important
  + Information density is proportional to user experience
    - Novice = less
    - Experts = more
* User experience
  + Ease of learning
    - Important for novice users
    - Relevant to systems with large number of users
  + Ease of use
    - Important for expert users
    - Important for specialized systems
  + Sometimes ease of learning and use go hand in hand
  + Types of computer users
    - Expert
      * Experienced in systems and will spend time using specific applications
    - Novice
      * Inexperienced user who will use a computer on a less frequent basis
  + Guidelines
    - Clearly label all controls
    - Select easily understood images to represent actions
    - Provide clear and concise on-screen instructions
    - Make it easy to return to any level in the menu structure
    - Provide shortcuts for experienced users
    - Provide auto complete functionality (like google search)
* Consistency
  + All parts of the system work in the same way
  + Key areas are:
    - Navigation controls
    - Terminology
  + Make it simple so users can predict what is going to happen
* Minimal user effort
  + Interfaces should be designed to minimize effort needed to accomplish tasks
  + Three click rule

**Design Process**

* Interface design prototyping
  + Mock-ups or simulations of computer screens forms and reports
  + Common approaches
    - Storyboard
    - Windows layout diagram
    - HTML prototype
    - Language prototype
* Prototyping the user interface
  + Initial prototype is designed form requirements
    - Mock-ups or simulations
  + User review prototype design and either accept the design or request changes
  + If changes are requested, the cycle repeats until client is happy

**Prototyping**

* Make sure there is a standard
  + Button placement
  + Colour scheme
  + Location of error/help text
* First pass is to verify that all necessary attributes are on the screen
* Final version should be exercised by user
  + Demonstrate how to get help/instructions
  + Demonstrate how security is handled
  + Demonstrate error handling

**Forms**

* A business document that contains pre-defined data and may include some additional areas to fill out
* Is an input to the system.

**Reports**

* A business document that contains only predefined data
* An output of the system
* Used for reading and viewing data

**Creating Forms and Reports**

* User-focused activity
* Follows prototyping approach
* Requirements determination:
  + Gain understanding of the intended user
  + Understand the task objectives
  + Who will use the form/report?
  + When will it be used
  + What is the purpose?

**Commandments of UI Design**

* Understand user and their tasks
* Involve the user in interface design
* Test the system on actual users
* Practice iterative design

**Navigation Design Basic Principles**

* Prevent mistakes
* Simplify recovery from mistakes
* Use consistent grammar order

**Navigation Guidelines**

* The user must always be aware of what to do next
* Tell user what the system expects now
* Notify the user that the data was entered correctly or not
* Explain to the user why there’s a delay in processing
* Tell user what tasks are completed and not
* Display the information in the same general area

**Message Types**

* Error Message
* Confirmation message
* Acknowledgement message
* Delay message

**Message Guidelines**

* Messages should be displayed long enough for a user to view it
* Alert users on lengthy processing times
* Provide onscreen progress reports
* Default values for fields and answers to be entered by user should be specified

**Error Handling Guidelines**

* Anticipate errors users might make
* Do not let user proceed without correcting an error
  + Highlight the error
  + All user to make corrections without retyping the whole thing
* If user does something catastrophic prevent any further input and inform them to call IT
* Provide an undo key or menu choice to reverse an action

**Help Guidelines**

* Ensure help is always available
* Title every help screen
* Provide user-selected help
* Provide context-sensitive help (offer help for task in progress)
* Provide direct route for users to return to point where help was requested

**Tone and Terminology Guidelines**

* **Tone**
  + Use simple, grammatically correct sentences
  + Don’t be funny, cute or condescending!
* **Terminology**
  + Don’t user computer jargon
  + Avoid abbr.
  + Use simple terms
  + Be consistent

**Input Design**

* Best principles
  + Online versus batch processing
  + Capture data at the source
  + Minimize key strokes
* Types of input
  + Free form
    - Text box
    - Number box
  + Selection box
    - Check box (yes or no or multiple options available
    - Radio button (when 1 item can be selected)
    - List box (drop downs, combo box)
    - Sliders

|  |  |
| --- | --- |
| **Validation Type** | **When to Use** |
| **Completeness** | **Ensures all required data has been entered** |
| **Format/datatype** | **Ensures data are of the right type (eg. Numeric) and the right format (eg. A date is month, day, year)** |
| **Range** | **Ensures numeric data are within a correct min and max. (eg. Rejecting birthdates prior to 1880.)** |
| **Check digit/self-checking digit** | **When numeric equations are used to validate information, such as when checking credit card numbers, SIN** |
| **Consistency** | **Ensure that a field makes sense in relation to another field. Eg. when the user enters both a birth date and a date of marriage (birth < marriage). When a form asks user to confirm password (form must contain values for both password and password confirmation, and they must match)** |
| **Limit** | **When there is a min or max, but not both** |
| **Sequence** | **A predetermined sequence exists (regular expression). Eg. Postal code** |

Interface Design (Web Forms)

**7 Goals of UI design:**

1. Path to completion
   * Primary goal
   * Every input requires consideration and action
   * Remove any unnecessary inputs
   * Provide a clear path (better to have aligned vertical)
   * Enable smart defaults
   * Enable flexible inputs (different phone number formats)
   * For long forms, save progress and show (tabs on the left side showing progress)
2. Label Alignment
   * Three choices. The correct type depends on the system.
     1. Top Aligned
        + When data asked for is common (familiar).
        + Minimize time to completion (A)
        + Requires more vertical space (D)
        + Spacing or contrast is vital for the best scanning by user
     2. Left Aligned
        + When data required is unfamiliar
        + Makes it easy to scan labels (A)
        + Reduced vertical space (A)
        + Less clear association between labels and fields (D)
        + Tougher to design layout (D)
     3. Right Aligned
        + Clear association between label and field
        + Requires less vertical space (A)
        + Reduced readability (D)
        + Fast completion times
   * Eye tracking data
     1. Left-Aligned:
        + Easily associated labels with proper input fields
        + Excessive distances can make it hard to read
     2. Right-Aligned:
        + Reduced overall number of fixations by half
        + Form completion is much faster
     3. Top-Aligned:
        + Can capture both label and field with single eye movement
        + Fastest completion time
   * Best Practices
     1. For reduced times & familiar data: Top-Aligned
     2. When vertical space is constrained: Right-Aligned
     3. For unfamiliar data entries: Left-Aligned
3. Proper Validation
   * Provide real-time feedback
   * Suggest valid inputs
   * Help users stay within their limits
   * Indicate required fields
     1. Indicating optional fields is useful when there are few optional fields
     2. Indicating required fields is useful when there are few required but lots of fields
     3. Neither is useful when they are all required.
   * Best Practices
     1. Try to avoid optional fields
     2. If most are required, indicate optional.
     3. If most are optional, indicate required.
     4. Text is best, but \* works for required.
     5. Associate errors with labels.
   * **Field Lengths**
     1. They indicate how much data can be entered.
     2. Appropriate lengths provide enough space for inputs
     3. Random field lengths may add visual noise to a form
     4. Best Practices
        + Provide length appropriate for input or just enough for data that can be entered.
4. Help & Tips
   * Useful when:
     1. Asking for unfamiliar data
     2. To know why info is being requested
     3. There are recommended ways of entering data
     4. Certain data requests are optional
   * Carful not to overwhelm a form with them
   * To avoid overwhelming the client
     1. Provide user activated help so its there if they want it
   * Best Practices:
     1. Minimize number of help & tips
     2. Make the help visible and adjacent to a data request
     3. When lots of unfamiliar data, use dynamic help.
5. Form Organization
   * Grouping content logically provides structure to a form
   * Allows to scan information required at high level
   * Gives a sense of how information is related
   * Best Practices:
     1. Use relevant content grouping to organize forms
     2. Use minimum number of visual elements to group content (horizontal rules)
6. Actions
   * Equal Visual weight
   * Visual Distinctions
   * Primary action must stand out more than secondary
   * Best Practices:
     1. Avoid secondary actions if possible
     2. Ensure clear visual distinctions between secondary and primary
     3. Align primary action with input fields for a clear path to completion
     4. Label the actions in a natural language
     5. Almost never use a reset button
7. Gradual Engagement
   * Progressive Disclosure:
     1. Not all users require all information all at once
     2. Progressive disclosure provides additional options when needed.
     3. Best Practices:
        + Map progressive disclosure to prioritize user needs
        + Most effective when user initiated
        + Maintain consistent approach
   * Selection Dependent Input
     1. If a user click­­­s an option it may need more input which will be prompted on the event.
     2. Exposing Dependent Inputs:
        + Page level (Radio buttons)
          1. Requires additional steps
        + Finger Tabs
          1. Follow path to completion (Scan the line)
        + Horizontal Tabs
        + Group selector
        + Expose below or within
          1. Can be confusing
        + Inactive until selected
          1. Association between primary selection is impaired
        + **Maintain clear relationship between initial selection options**
        + **Clearly associate additional inputs with their trigger**
        + **Avoid “jumping” which disassociates initial selection options.**

**Design Principles**

* Minimize the pain
* Illuminate the path to completion
* Consider the context (familiar vs foreign)
* Ensure consistent communication