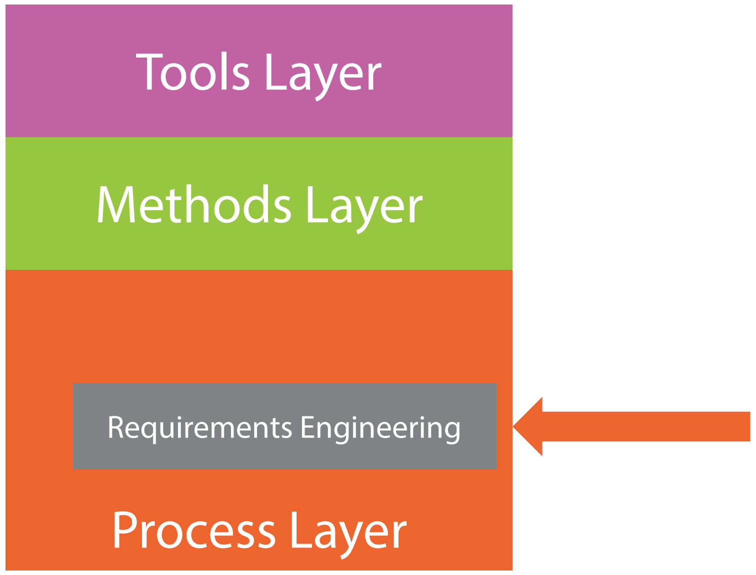
Requirements Engineering

Mohamad Halabi
@mohamadhalabi

hardcore dev and IT training

Introduction



User and System Requirements

- **Two abstraction levels of requirements:**
  - User Requirements
  - System Requirements
- **Each level is suitable to a different stakeholder group**

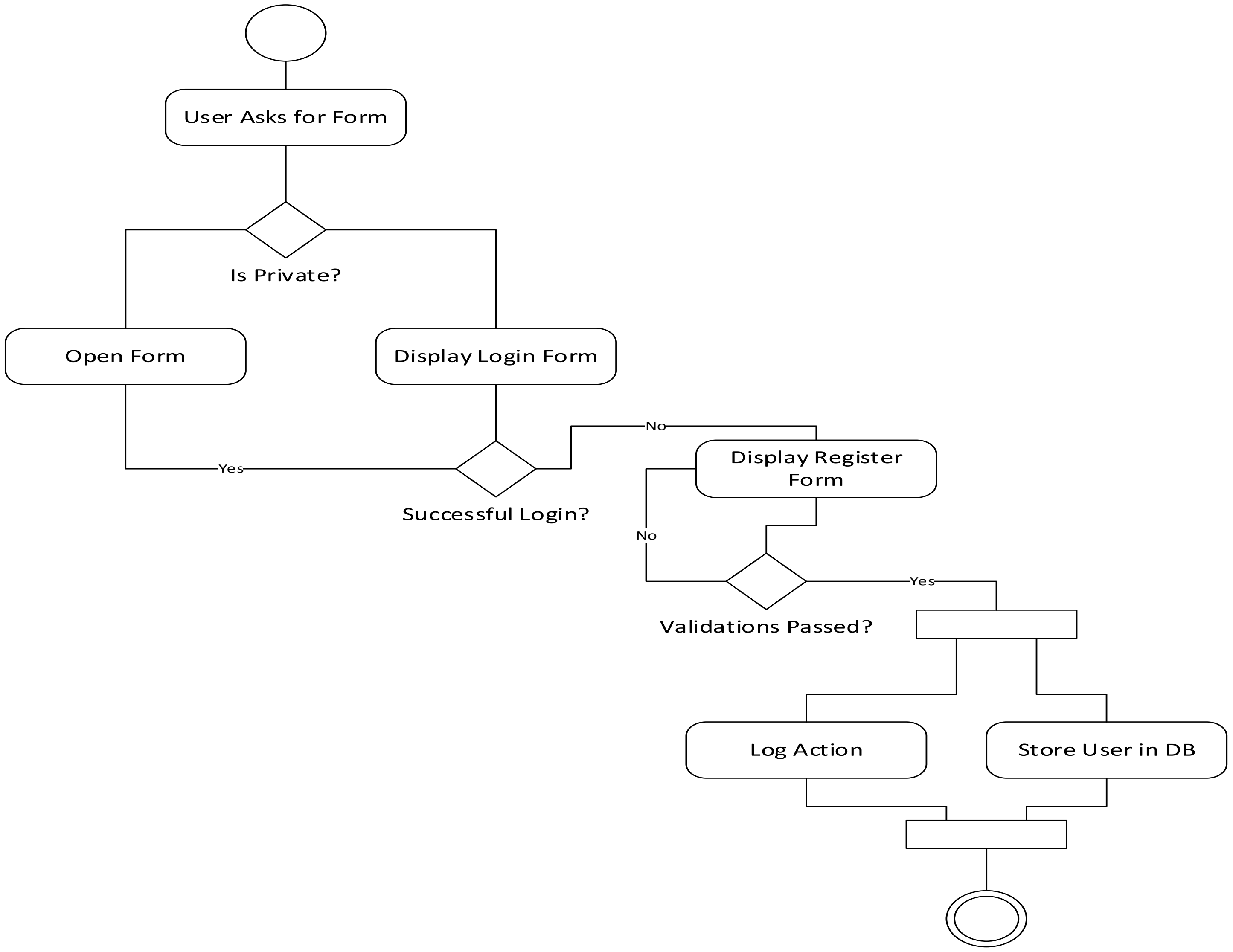
User Requirements

- **Requirements are described in natural language**
  - Possibly with simple notations such as tables and diagrams
- **Audience:**
  - Not interested in advanced modeling
  - Audience interested in high-level services and constraints
  - Ex: High-level business stakeholders and managers
- **Example: "*In addition to its public forms, System X will contain private forms that can only be accessed by authenticated users*"**

System Requirements

- Define exactly what will be implemented
- System Requirements document contains models about: services, use-cases, behavior, constraints, business processes, etc…
- Forms a contract between implementation team and customer
- Critical input to design, development, and testing activities

System Requirements



Functional and Non-Functional Requirements

Functional

Define system's external behavior

- how the system interacts with its environment
- how it responds to inputs
- what output to generate
- and how to behave under certain conditions

What the system must do and not do

Non-Functional

Quality attributes or constraints

Quality Attributes examples: security, performance, and availability

Technology constraint: system must be Java-based

Process constraint: RUP must be used

Example

- **Airline Reservation system:**

Functional

‘a user can search for a flight. If a flight is selected, the user has 2 minutes to confirm booking or the flight will be released’

How the system will interact and respond to use?

Non-Functional

‘the response time for a search transaction under peek load of 10,000 users must not exceed 2s’

‘the throughput of the system is 100 search transactions per second’

What are the quality attributes of the system?

Quantifying Non-Functional Requirements

- **Customers almost never give quantitative non-functional requirements**
- **Instead of:**
  - “*the response time for a search transaction under peak load of 10,000 users must not exceed 2s*”
  - Customers will say “*the system should be fast when number of users is high*”
- **Instead of:**
  - “*the throughput of the system is 100 search transactions per second*”
  - Customers will say “*the system must accept high number of search commands*”
- **Requirements such as “*I want a secure system*” and “*I want a system that is easy to use*” will surface all the time**

Quantifying Non-Functional Requirements

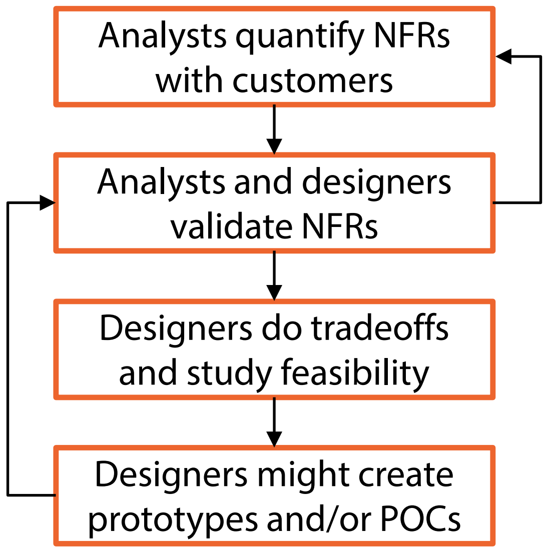
- **How to quantify non-functional requirements to read:**
  - *"10,000 users peak load"*
  - *"2s response time"*
  - *"100 Tx per second throughput"*
  - *"user must learn how to perform a search in operation after 3 attempts"*
- **Requirements Engineering process transforms vague non-functional requirements into more quantitative form**
- **Non quantitative requirements are difficult to design and implement**

Analyst/Designer (Architect) Collaboration

- **Most of the times, analysts and designers (architects) must collaborate to quantify non-functional requirements**
- **Designers know (based on system's architecture) if non-functional requirements are feasible**
- **Ex: An analyst considers a '*response time of 2 s*' a quantifiable, clear, and unambiguous non-functional requirement (and it is!)**
  - Designer might judge that the number '2 s' is not feasible
  - Without collaboration, analyst approves the requirement, only for the designer to deem it unfeasible later in the process

Analyst/Designer (Architect) Collaboration

- **Analysis and Design – especially for non-functional requirements (NFR) – are highly related and iterative**

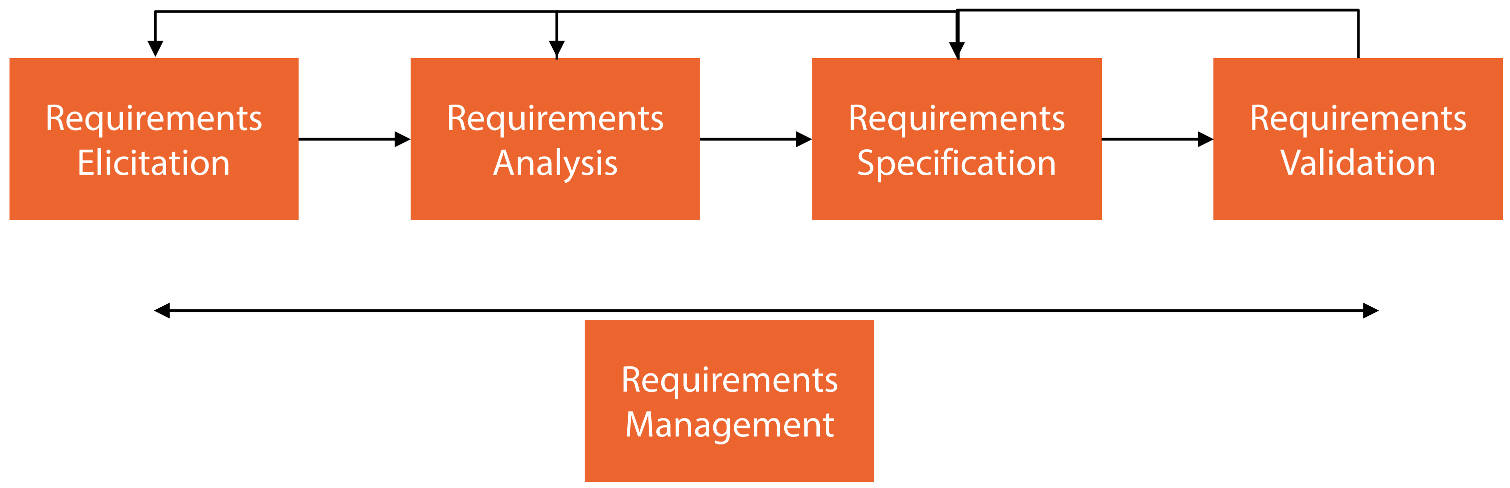

Analysts quantify NFRs with customers
Analysts and designers validate NFRs
Designers do tradeoffs and study feasibility
Designers might create prototypes and/or POCs

- **Risk of finding unfeasible NFRs later in the design phase is not 100% mitigated, but greatly reduced**

Derived Requirements

- **Functional requirements can be derived from non-functional requirements**
- **Ex: "*the system must be secured against unauthorized access*"**
  - Is implemented through user login and user registration use-case

The Requirements Engineering Process



Customers and Users

Customer	User
Sponsor of the system	Uses (interacts) with the system
Ex: HR Manager sponsoring an HR system	Ex: HR employees will use the system
Ex: Company owner asking for an e-Commerce web site	Ex: Online users buying items

Analyst

- **Responsible for the Requirements Engineering process**
- **Analyst defines the "what" not the "how"**
  - Defines functional and non-functional requirements
  - Documents models in the System Specification document
  - Gathers information from both customers and users
- **Analyst is not only a passive information collector!**
  - Observes users at work and translates their actions into requirements
  - Interprets users' work patterns and looks for methods to improve
  - Has a vision about the customer requested to-be state

Stakeholders

- **It's essential to identify stakeholders**
- **Entities that have interest in a project**
  - Affected by or affect project outcome (positively or negatively)
  - Inside or outside an organization
- **Stakeholder identification part of larger Stakeholder Management**
  - Includes identification and managing influence, expectations, and communication needs
  - Typically part of the project management methodology
- **Analysts identify and interact with stakeholders**

Requirements Elicitation

- **Information discovery and collection**
- **Work is observed and improved methods are sought after**
- **Information is translated into requirements in Requirements Analysis step**

The Problem Domain

- **The analyst must understand the problem (business) domain**
- **Ex: For an Airline Reservation system, analyst must understand flight rules, regulations, and common processes**
- **Knowledge must be acquired for domains which are new to analysts**
- **This allows the analyst to:**
  - Advise stakeholders
  - Perform tradeoffs
  - Resolve conflicts
  - Play the role of customer ‘champion’

Stakeholder Identification

- **Part of Stakeholders Management process**
  - Part of Project Management umbrella activity
- **However, as domain experts, analyst will (almost) always identify stakeholders**
- **Regardless of the method of identification, Requirements Elicitation cannot commence before relevant stakeholders are identified**

Information Sources

- **Various sources:**
  - Interviews and user discussions
    - For mass users (ex: online users), user segments must be identified
  - Existing documentation
  - Marketing surveys and user questionnaires
  - Observation
  - Facilitated meetings
    - Requires a facilitator to prevent favoring of outspoken/senior people requirements

Requirements Analysis

- **Customer and user needs are deeply understood**
- **Elicitation information is used to:**
  - Organize requirements
  - Identify ambiguous, conflicting, and inconsistent requirements
  - Identify relationships between requirements
  - Ranking requirements
- **Requirements must be precise to enable validation, cost estimation, and implementation verification**

Complete Requirements

- **A requirement clearly states system behavior:**
  - Under all range of inputs and outputs
  - Under all possible constraints
- **Ex: "*a loan request is accepted if the applicant's age is above 20*" is not complete**
  - What is the age upper limit?
  - How the system responds if age is out of range? Request rejected or placed in queue?

Consistent Requirements

- **A requirement does not conflict with any other requirement**
- **Ex:**
  - “*identification Id is required in the user registration process*”
  - Conflicts with “*identification Id is optional in the user registration process*”
- **Analyst must resolve conflicting stakeholders’ conflicting requirements**
  - Stakeholder power grid
  - Negotiation
  - Conflict resolution

Implementation-free Requirements

- **Requirements specify the ‘what’ not the ‘how’**
  - The ‘how’ is specified in the Design activity
- **Ex: “*the system must accept requests regardless of the format and protocol of the sender*”**
  - Does not specify patterns (Pipe and Filter, Publish-and-Subscribe, etc…)

Unambiguous Requirements

- **Each requirements can be interpreted in a single way**
- **Ambiguous requirements can be implemented in various valid ways**
- **Ex: "*an applicant can edit non-essential information of an existing loan request*"**
  - What is 'non-essential' information?
  - Implementers might interpret this different to what business users intended

Traceable Requirements

- **Each requirements must be traceable by being assigned a unique id/number**
- **Links the requirement:**
  - Corresponding models in the specifications document
  - Design entities
- **Facilitates maintenance and change management**

Quantitative Requirements

- **Especially relevant to non-functional requirements**
  - Customers usually give vague non-functional requirements
- **Ex:**
  - *“the system must be fast ”*
  - Must be transformed to *“the average response time for a transaction under peak load of 1000 concurrent users is 1s*”

Achievable Requirements

- **Requirements must be technically feasible**
  - Requires analyst & designer collaboration
- **Ex: “*the system must have the same average response time for employees in the head office and employees in other branches*”**
  - Not feasible due network latency of branch employees
- **Designers often help in identifying technical feasibility**
  - Often a single ‘Analysis and Design’ iterative phase is mentioned

Testable (Verifiable) Requirements

- **Requires collaboration between analysis and testing teams**
- **Ex: "*the system must be user friendly*" is not testable**
- **Analyst transforms this to "*the system must provide the user with tips. Controls appear based on a menu selection thereby reducing the amount of displayed controls*"**

How to Achieve These Attributes?

- How to achieve the discussed attributes (consistent, unambiguous, implementable, testable, clear, etc…)?
- What notations we use to interact with different stakeholder groups (with different backgrounds)?
- Modeling tackles these questions

The Importance of Modeling

- **How to guarantee effective Requirements Analysis?**

Condition #1

Analyst must thoroughly understand the system

- Interaction with environment
- Services for different user segments
- Information model
- Behavior under certain conditions

Condition #2

Effective communication with various stakeholders groups

- Various groups have various interests and backgrounds
- Groups must envision the system – each from its own perspective

- **Modeling helps analysts achieve these conditions**

The Importance of Modeling

- **Modeling describes the system from different abstraction levels**
- **Viewpoints tackle different concerns**
  - Explain the system from different perspectives
  - Used for efficient communication with various stakeholder groups
- **Ex: Understanding system's behavior**
  - Behavioral models are used
  - Analyst uses models to better understand and refine the system
  - Analyst uses models to communicate with stakeholder group interested in behavioral aspects
- **Models also become the basis of Design activity**
- **Models also are used to design Acceptance Tests**

Requirements Specification

- **The process of production, review, evaluation, and approval of the System Requirements Specification document**
  - Document contains User and System Requirements
- **Requirements Specification step is closely related to Requirements Analysis**

Outline

- Introduction: Purpose, Scope, Definitions, Acronyms, and References
- Overall Description: User Requirements, Constraints, and Assumptions
- Models: Viewpoints (Context, Data, Functional, Behavioral, etc…)
- Validation Criteria: Test Cases that will verify requirements
- References: Relevant documents (elicitation, vendor specific, standards, etc…)
- Appendix

Requirements Validation

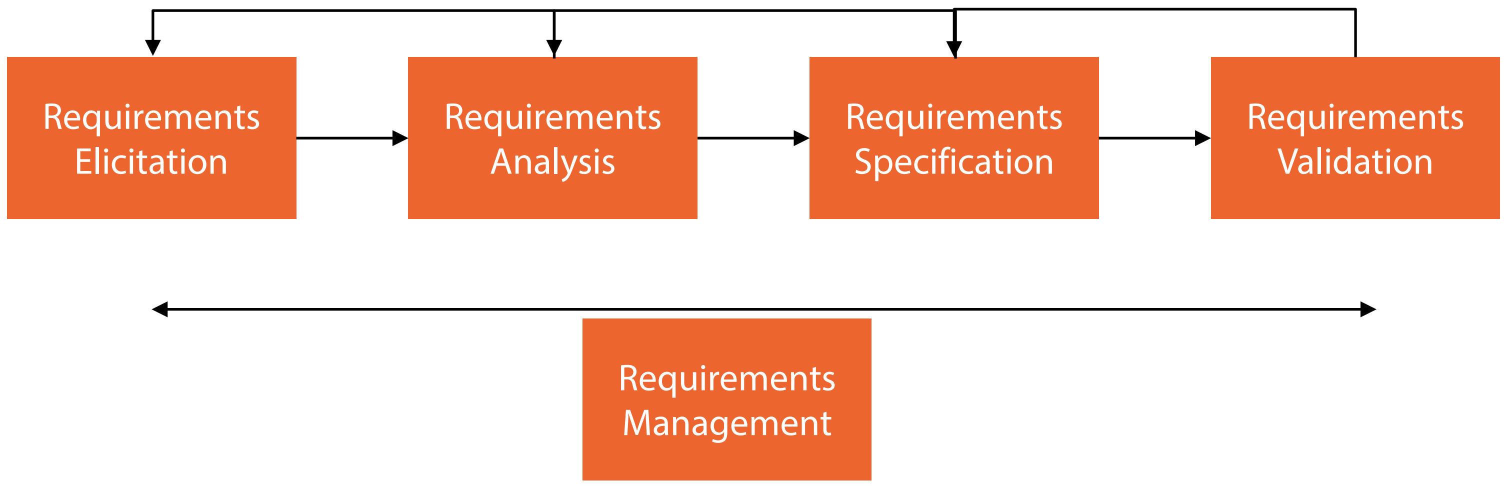
- **Finding potential problems**
  - Reduces the cost of later discovery (design, development, testing)
- **Overlaps with Requirements Specification to verify requirements are complete, consistent, unambiguous, verifiable, achievable, and implementation-free**
- **Validation techniques:**
  - Formal Reviews: Analyst and stakeholders review specifications document
  - Prototyping: Customers and user segment representatives try out an executable model
  - Test Cases: Designing Acceptance Test Cases reveals problems in the requirements (ambiguity, inconsistency, verifiability, etc.)

Requirements Management

- **Relates to Software Configuration Management and Change Management**
- **Controls changes to requirements**
  - Change Control process: impact and cost assessment
  - Tracing assists in impact and cost assessment:
    - Requesting stakeholder (backward tracing)
    - Relation to other requirements (dependency relationship)
    - Linkage to design artifacts (forward tracing)
  - Version Control: document and artifacts approved versions
  - Status Tracking: informing users of status change (proposed, approved, rejected, implemented, verified, deleted)

Requirements Management

- **Involves planning other aspects of the Requirements Engineering process:**
  - What elicitation techniques to use?
  - What specialties are required?
  - What models to create?
  - How to validate the specification document



Requirements Engineering vs. Business Analysis

- **A related discipline is that of Business Analysis**
- **What is the difference?**
  - No unanimous answer
  - Some consider that both eventually lead to requirements specification, analysis, and management
  - Some say there is a difference in scope
  - I will take the second side

Requirements Engineering Scope

- **The scope of Requirements (and Software) Engineering is the delivery of a software system**
  - Recall IEEE's definition: *"…the application of a systematic, disciplined, and quantifiable approach to the development, operation, and maintenance of software…"*

- **Note however, that this <u>does not mean</u> that Requirements Engineering neglect the surrounding environment of a software!**
  - It does not (as you saw in the non-functional requirements section)
  - Requirements Engineering sees the environment as a set of black boxes
  - It studies how the software in scope is affected by these boxes
  - But it is not concerned with the analysis of the entire environment (i.e. organization)

Business Analysis

- **Business Analysis has a wider scope**
- **Wikipedia: “*has a heavy overlap with requirements analysis sometimes also called requirements engineering, but focuses on identifying the changes to an organization that are required for it to achieve strategic goals. These changes include changes to strategies, structures, policies, processes, and information systems*”**
- **International Institute of Business Analysis (IIBA):**
  - *“…the practice of enabling change in an organizational context…”*
  - *“Business analysts work across all levels of an organization and may be involved in everything from defining strategy, to creating the enterprise architecture…”*
  - *“ultimately improves the way they [organizations] do business”*

Business Analysis

- **IIBA says that "*Job titles for business analysis practitioners include not only business analyst, but also business systems analyst, systems analyst, requirements engineer, process analyst, product manager, product owner, enterprise analyst, business architect*…"**
- **So IIBA considers a requirement engineer as a Business Analysis practitioner**
  - This shows how different people have different views

Beyond Names

- **It’s the scope that matters!**
- **Business Analysis has a wider scope than Requirements Engineering**
  - Requirements Engineering scope is a software system
  - Business Analysis scope is the business of an entire organization (or business unit)
- **Business Analysis might recommend building software systems that help achieve the organization goals**
  - Requirements (and Software) Engineering is then applied to each software system

Side Note: Business Architecture

- **IIBA considers that a Business Architect is a Business Analyst**
- **Business Architecture is itself a rising discipline**
  - Separate or as part of Enterprise Architecture
- **A discussion between an IIBA's analyst and a Microsoft's architect**
  - http://community.iiba.org/profile.htm?mode=pvb&pid=17&articleId=476
  - http://blogs.msdn.com/b/nickmalik/archive/2012/04/06/the-difference-between-business-architect-and-business-analyst.aspx
  - http://blogs.msdn.com/b/nickmalik/archive/2012/04/08/analysis-synthesis-and-scope-business-architecture-vs-business-analysis-part-two.aspx

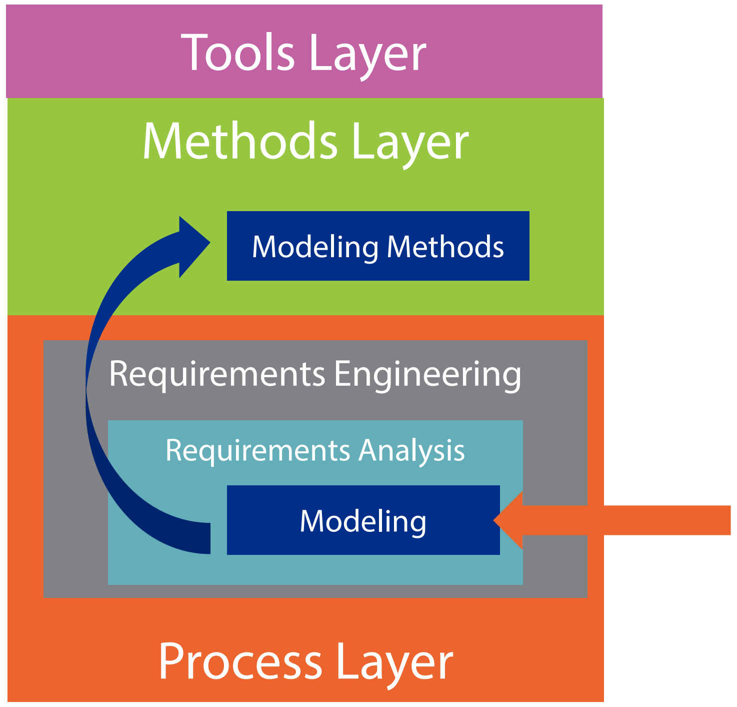
More Resources

- *Gathering Good Requirements for Developers* course by Robert Bogue
- BABOK Guide (http://www.iiba.org/babok-guide/babok-guide-online.aspx)

Summary

- **First core activity of Software Engineering**
  - Requirements Elicitation: information discovery and collection
    - Domain knowledge is important
    - Various techniques, such as interviews, meetings, and observation
  - Requirements Analysis: detailed understanding of requirements
    - Requirements must exhibit various attributes
    - Modeling notations are utilized
  - Requirements Specification: System Requirements Specification document is created, reviewed, evaluated, and approved
  - Requirements Validation: formal validation of requirements
  - Requirements Management: changes are controlled and tasks are planned

What’s Next?


Tools Layer
Methods Layer
Modeling Methods
Requirements Engineering
Requirements Analysis
Modeling
Process Layer