Requirements Engineering Defined

Figure 1 *Retrieved from:*
http://faculty.winthrop.edu/dannellys/csci521/lectures/10_Reqs_Gathering.ppt

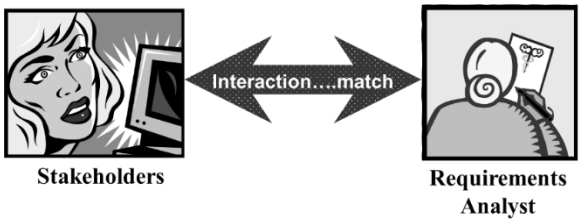


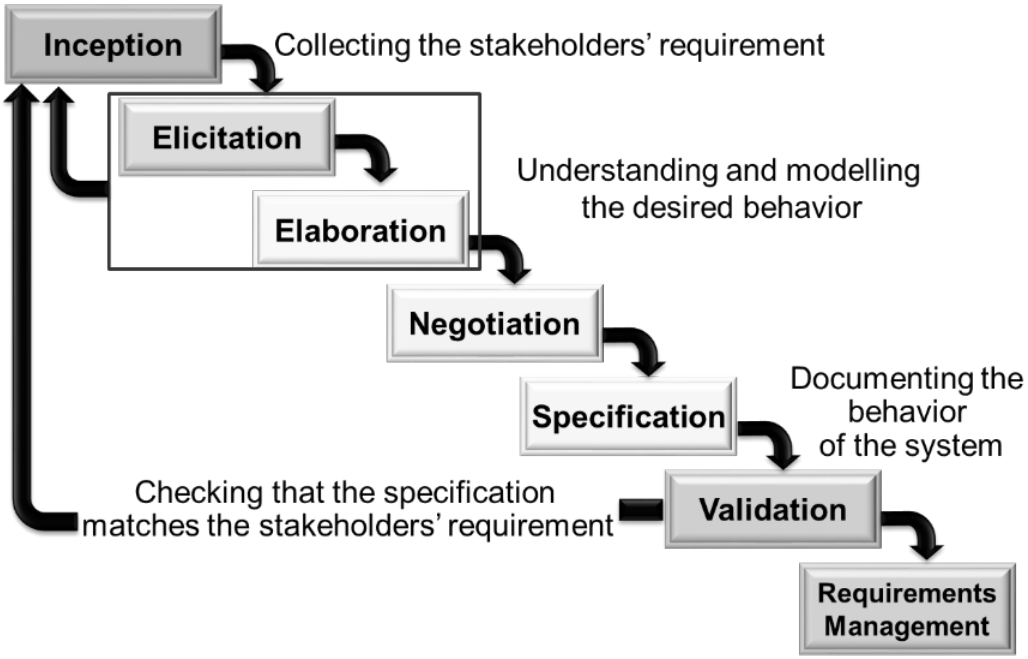
Figure 2 *Retrieved from:*
http://www.cs.ru.nl/~ths/sdm1/theo2010/RequirementAnalyse.ppt

- **Stakeholder**
  - Three (3) Main Categories:
    - Acquirers of the software product
    - Suppliers of the software product
    - Other stakeholders

- **Requirement**
  - It is a feature of the system or a description of what the system should do to fulfill its purpose, the services that it provides and the constraints on its operation
  - The following are the key fields, which should be part of the functional requirements specifications document:
    - Purpose of the Document
    - Scope
    - Business Processes
    - Functional Requirements
    - Data and Integration
    - Security Requirements
    - Performance
    - Data Migration & Conversion

- **Expectation**
  - It defines what services the system is expected to provide to system users and the constraints under which it must operate
    - Execution qualities
    - Evolution qualities
  - The following are the key fields, which should be part of the non-functional requirements specifications document:
    - Baseline testing
    - Compatibility testing
    - Compliance testing
    - Endurance testing
    - Load testing
    - Localization testing
    - Internationalization testing
    - Performance testing
    - Recovery testing
    - Resilience testing
    - Security testing
    - Scalability testing
    - Stress testing
    - Usability testing
    - Volume testing

- **Requirements Engineering**
  - It provides the end users the opportunity to define the scope and boundaries of what a system should do to the requirements analyst



- **Inception**
  - Stakeholder's or End user's Motivation
  - End user's Perception
  - Effectiveness of Communication

- **Elicitation**
  - Problems of scope
  - Problems of understanding
  - Problems of volatility

- **Elicitation Method**
  - Collaborative Requirements Gathering
  - Quality Function Deployment
    - Normal Requirements
    - Expected Requirements
    - Exciting Requirements

- **Classification of Priorities**
  - Must Have
  - Should Have
  - Could Have
  - Won't Have

- **Elaboration**
  - Requirements Model
  - Analysis Model

- **Negotiation**
  - Recognize that it is not competition
  - Map out a strategy
  - Listen actively
  - Focus on the other party's interests
  - Don't let it get personal
  - Be creative
  - Be ready to commit

- **Specification**
  - This is where the final work product in the form of software requirement specification is constructed.

- **Validation**
  - Requirements Validation Checklist
    - Is each requirement consistent with the overall objective for the system/product?
    - Have all requirements been specified at the proper level of abstraction? That is, do some requirements provide a level of technical detail that is inappropriate at this stage?
    - Is the requirement really necessary or does it represent an add-on feature that may not be essential to the objective of the system?
    - Is each requirement bounded and unambiguous?
    - Does each requirement have attribution? That is, is a source (generally, a specific individual) noted for each requirement?
    - Do any requirements conflict with other requirements?
    - Is each requirement achievable in the technical environment that will house the system or product?
    - Is each requirement testable, once implemented?
      - Approaches: Demonstration, actual test, analysis, or inspection
    - Does the requirements model properly reflect the information, function, and behavior of the system to be built?
    - Has the requirements model been "partitioned" in a way that exposes progressively more detailed information about the system?
    - Have the requirements pattern been used to simplify the requirements model? Have all patterns been properly validated? Are all patterns consistent with customer requirements?

- **Requirement Management**
  - It is where set of activities are performed to identify, control, and track requirements and changes to the requirements at any time as the project progresses

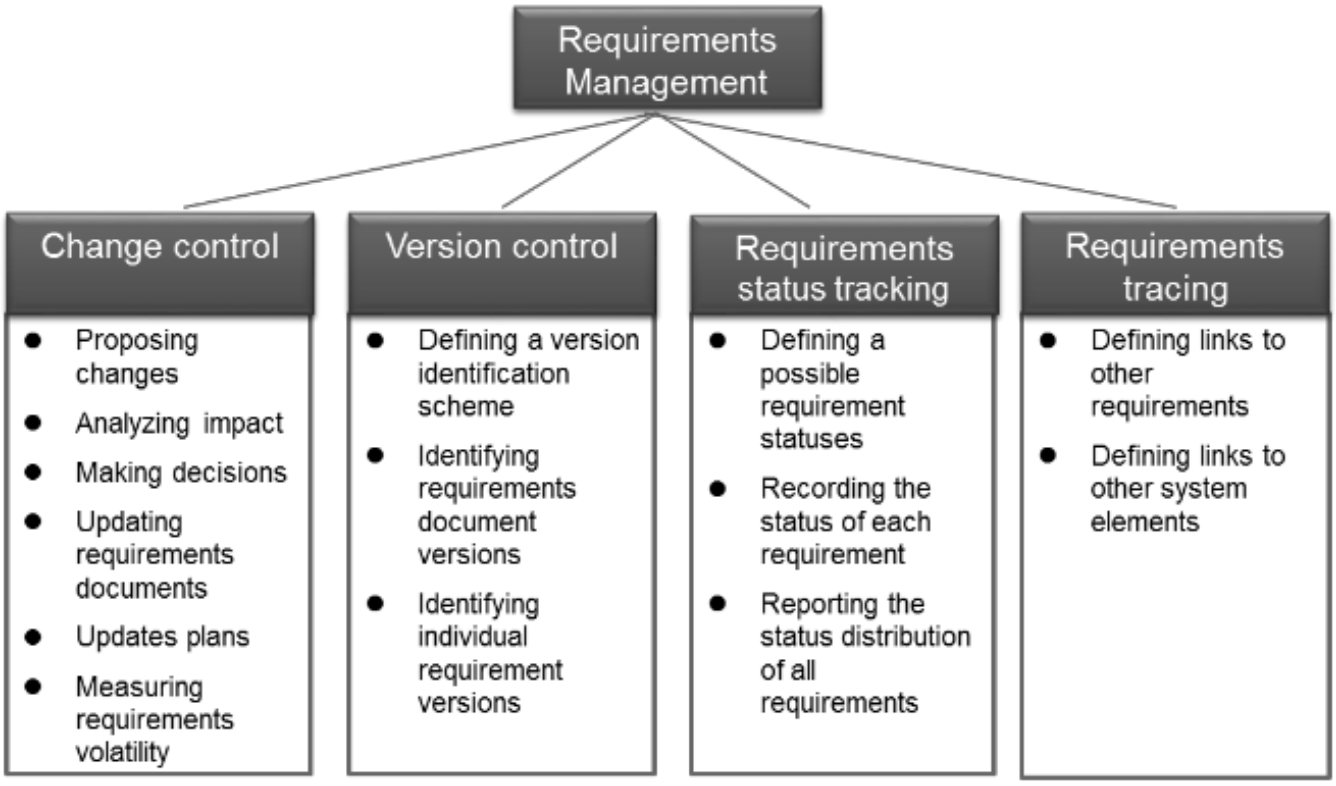


Figure 3 Requirements Management *Retrieved from:* http://csis.pace.edu/~marchese/CS775/Lectures/775L11b.ppt

Requirements Analysis and Model

- **Three (3) Elements of Requirements Model**
  - **Use Case Model**
    - Use Case Diagram
    - Use Case Specifications

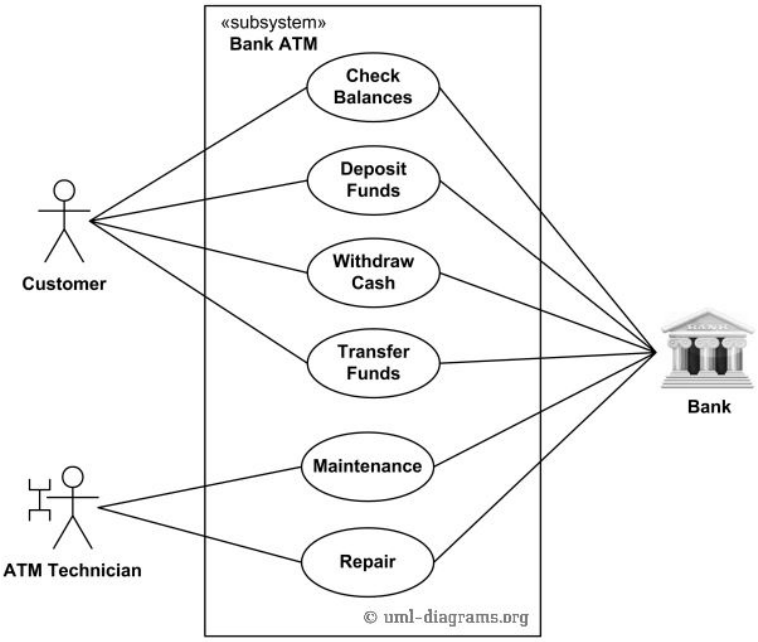
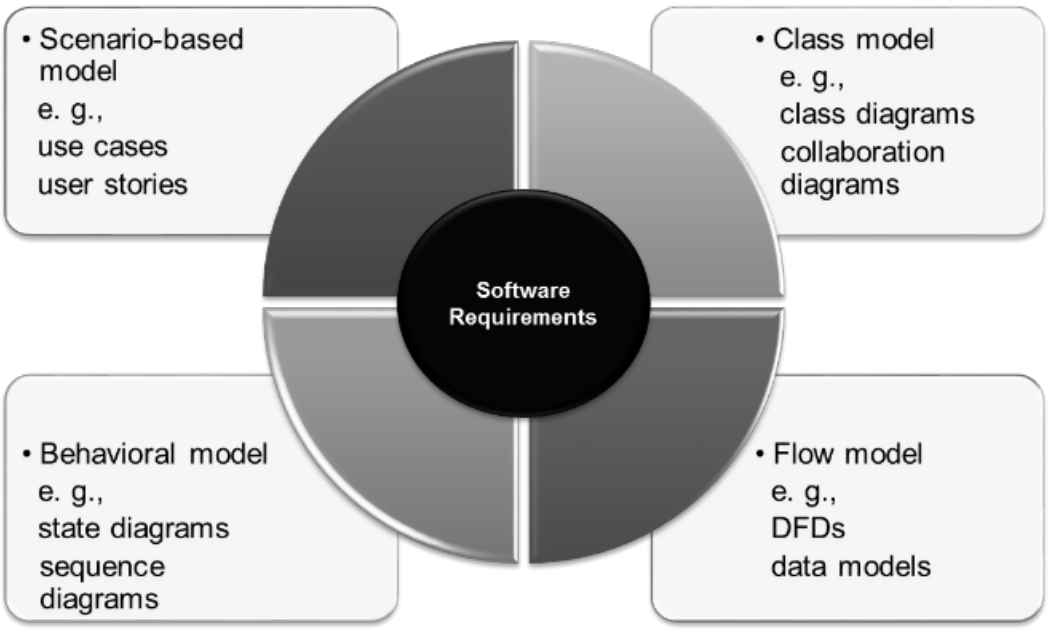


Figure 4 An example of use case diagram for Bank ATM subsystem – top level use cases *Retrieved from: Source image: http://www.uml-diagrams.org/bank-atm-uml-use-case-diagram-example.html*

  - **Supplementary Specifications**
  - **Glossary**

- **Elements of Analysis Model**



- **Scenario-based model**

Use Case 1.1 ***Withdraw money from an ATM***

Withdraw money from an ATM

Primary actors: Customer
ATM Technician
Bank

Preconditions: Network connection is active
ATM has available cash

Basic flow of events:
1. Bank customer inserts debit card and enters PIN.
2. Customer is validated.
3. ATM displays actions available on ATM unit. Customer selects Withdraw Cash.
4. ATM prompts account.
5. Customer selects account.
6. ATM prompts amount.
7. Customer enters desired amount.
8. Information sent to Bank, inquiring if sufficient funds/allowable withdrawal limit.
9. Money is dispensed and receipt prints.

Alternative flows
2a. Customer is not validated.
2a1. ATM displays error message.
7a. Customer selects invalid amount.
7a1. ATM prompts user to re-enter valid amount.
8a. Customer has insufficient funds.
8a1. ATM displays error message.
8a2. ATM shows available withdrawal limit, redirects to step 6.
9a. ATM has insufficient cash.
9a1. ATM Technician is alerted.
9a2. ATM displays error message and phone number to call.
9b. Cash gets stuck in dispensing.
9b1. ATM displays error message.

Figure 5 An example of Use Case Scenario *Retrieved from:* https://www.lucidchart.com/pages/use-case-scenario-example-and-template-UML

- **Class-oriented model**

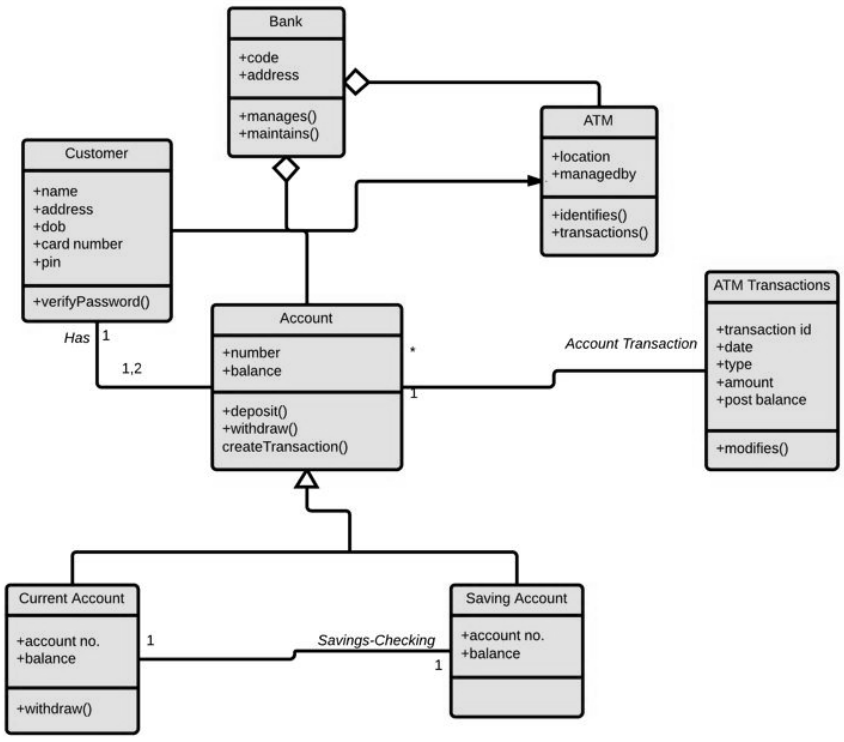


Figure 6 Sample Class Diagram for an ATM Process *Retrieved from:* https://www.lucidchart.com/pages/class-diagram-for-ATM-system-UML

- **Behavioral model**

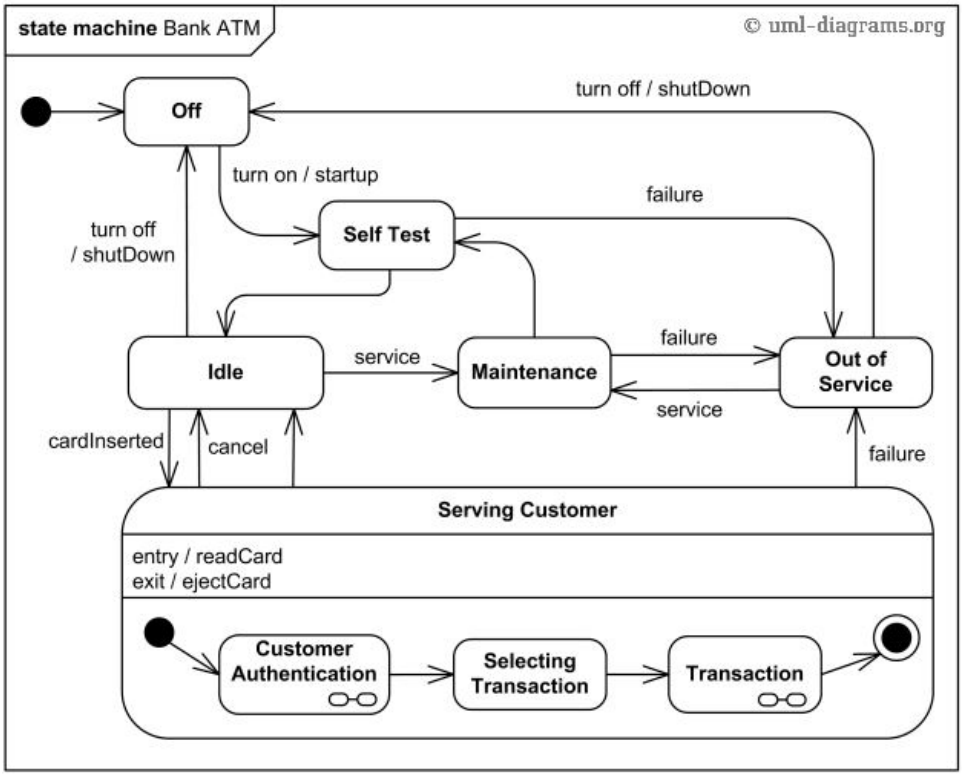


Figure 7 *Retrieved from:* http://www.uml-diagrams.org/bank-atm-uml-state-machine-diagram-example.html

- **Flow-oriented model**

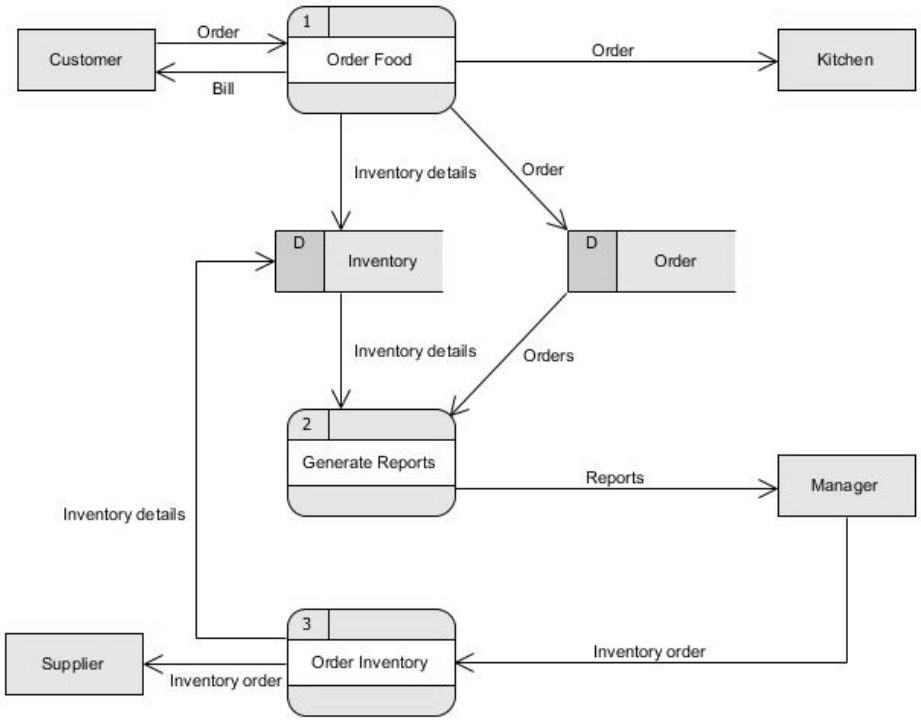


Figure 8 Data Flow Diagram Food Ordering System *Retrieved from:* https://www.visual-paradigm.com/tutorials/data-flow-diagram-example-food-ordering-system.jsp

- **Analysis Rules of Thumb**
  - The model should focus on requirements that are visible within the problem or business domain; the level of abstraction should be relatively high
  - Each element of the analysis model should add to an overall understanding of software requirements and provide insight into the information domain, function and behavior of the system
  - Delay consideration of infrastructure and other non-functional models until design
  - Minimize coupling throughout the system
  - Be certain that the analysis model provides value to all stakeholders
  - Keep the model as simple as it can be

References:

Agarwal, B. B, S. P Tayal, and M Gupta (2010). *Software Engineering & Testing*. 1st ed. Sudbury, Mass.: Jones and Bartlett.
Pfleeger, S. L., & Atlee, J. M. (2009). *Software engineering: Theory and practice* (4th ed.). Boston, MA, United States: Pearson Prentice Hall.
Pressman, RS. (2010). *Software engineering: a practitioner's approach (7th ed.)*. McGraw-Hill/Higher Education
Software Engineering Tutorial. Retrieved January 16, 2017, from https://www.tutorialspoint.com/software_engineering/software_engineering_tutorial.pdf
Software Project Management Requirements Analysis (2010) Retrieved from: http://www.cs.ru.nl/~ths/sdm1/theo2010/RequirementAnalyse.ppt
Sommerville, I. (2010). *Software engineering* (9th edition) (9th ed.). Boston: Addison-Wesley Educational Publishers.