- **What is Software?**

  Computer software, or simply software, aka computer programs, is the intangible (formless, 무형의) component of computers. It includes all computer program-related files such as executable files, libraries, documents and even scripts. That is, A collection of computer programs and related data.

- **Generic Software**

  Developed to be sold to two or more general customers. Ex. Office Program, Music Player.

- **Bespoke (or Custom) Software**

  Developed for a single customer according to their specification. Ex. Student Management Sys in Univ.

- **Software Life Cycle**

Requirements	Development
Analysis	
Design	
Implementation	
Testing (Validation)	
Post-delivery Maintenance	Maintenance

- **Attributes of Good Software**

  The software should have functions and performance that the customer expects and requires.

  - Maintainability (유지보수성)

    Software must evolve to meet changing needs.

  - Dependability (의존성)

    Software must be trustworthy (reliable, 신뢰성).

  - Efficiency (효율성)

    Software should not make wasteful use of system resource.

  - Usability (사용성)

    Software must be usable by the users for which it was designed.

- **Software Engineering is**

  An engineering field that is concerned with creating and maintain software products by applying technologies and prosses and by managing peoples and projects.

- **Goal of Software Engineering**

  Deliver the products on time and within budget. / Satisfy customer's needs (requirements). / Produce and maintain high-quality software.

- **Software Engineers should**

  Apply efficient and suitable approaches to their work. / Use appropriate tools and techniques to solve given problems. / Consider the development constraints and the available resources.

- **Software Life Cycle: Requirements**

  Identify customer's (user's) demands and development constraints.

  - Type of demands

    Functionality: Features, Design, etc.

    Quality: Performance, Reliability, Maintainability

  - Type of constraints

    Deadline, Cost, etc.

  - Difficulties in requirements

    Users do not know what they want in advance. Requirements are invisible; thus, usually ambiguous, incomplete, and contradictory.

- **Software Life Cycle: Analysis**

  Analyze and refine the requirements. The requirements must be expressed in the language of the client. The outcome of analysis is a specification document (Specifications) that is a kind of a contract between customer and developer.

  Complete and clear specifications is essential for Validation and Maintenance of the software.

- **Software Life Cycle: Design**

  Propose an optimal solution that satisfies the given requirements. / Determine programming language and tool. / Architectural design and component design.

  These are difficulties in design what select only one among many feasible (가능한) solutions that can satisfy the given requirements, and discover conflicts on requirements.

- **Software Life Cycle: Implementation**

  Ideally transform the design into a program and remove errors from the program. Clear and well-defined design specifications make easy implementation.

- **Software Life Cycle: Verification and Validation**

  Verification and validation is intended to show that a system conforms to its specification and meets the requirements of the customer.

  Involves checking and testing. Testing Involves executing the software for various test cases that are derived from the specifications, by using real data.

- **Software Life Cycle: Maintenance**

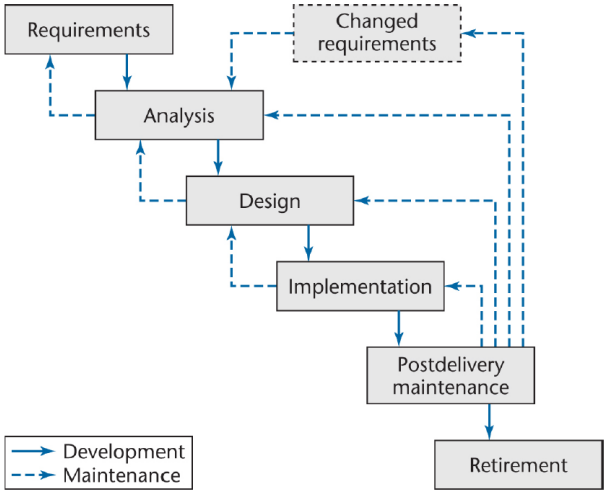
  Behaviors after release of the software. Upgrade (Perfective) with new features or Fixes (Corrective) problems that should have been solved before.

  Why developers do maintenance is customers change their minds / environmental changes / better technologies or methods become available.

- **Software Process Model**

  The primary functions of a software process model. It is need to determine the order of the stages (procedures) involved in software development and evolution / to establish the transition criteria for progressing from one stage to the next.

- **Software Process Model: Waterfall**



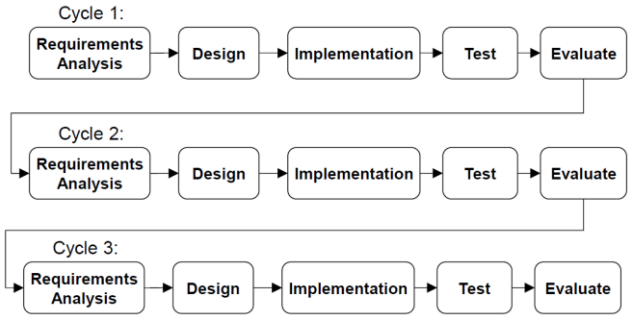
Waterfall model requires that every stage should be completely perfect and correct. It is appropriate when the requirements are well-defined and understood.

Specify the requirements completely, consistently, and unambiguously on the first attempt. / Design the system completely and correctly on the first attempt. / Write all of the program interfaces and internal details (program codes) correctly on the first attempt / Test and verify the system at the last step.

Advantages are: Easy to understand and relatively easy to manage. / Tasks in every stages may be assigned to specialized teams / Project progress is evaluated and approved at the end of each stage. Results obtained from every stage are reliable.

Disadvantages are: Difficult to know forthcoming (곧 다가올) problems and situations in advance. / Specialized teams of each stage have different experiences and skills. So, different analysis and understanding for client demands. / Difficult to accommodate (accept) modification and change after the process is started. / Difficult to respond to changing customer requirement once the process is started. / Many SW project cases have limitations in partitioning projects into distinct stages. / No demonstrations and tests before the development is finished. / Other team members of later stages can be idle while the requirements are being completed.

- **Software Process Model: Evolutionary**



Used when requirements cannot be specified in advance and requirements can be changed several times. Repeat software life cycle stages from requirements to validation.

Initial version is developing functions, for which customer rarely change requirements. *Intermediate version* is developing other functions, reflecting customer's changing requirements. *Final version* is completing software products.

Advantages are: Customers do not make a complaint. / Because customers continuously participate in evolutionary cycles, they understand the delivered product well and certainly use them.

Disadvantages are: There can be massive iterations during a software project. / It can require high cost if management and plans are weak. / Products are often poorly structured.

Applicable to: Small or medium size systems. / Parts of a large system. / Systems having short lifetime.

- **Software Process Model: Formal System**

Requirements Definition → Formal Specification → Formal Transformation → Integration and System Testing.

Implement software according to formal mathematical specification. Complete an executable program while transforming a mathematical specification. Trans formations should be correct.

Advantages are: Does not involve high complexity rate. / Discovers ambiguity, incompleteness, and inconsistency in the software. / Offers defect-free software. / Formal specification language semantics verify self-consistency.

Disadvantages are: Need for specialized skills and training to apply the technique. / Difficult to formally specify some aspects of the system such as the user interface (systems having risks of many variations). / Time consuming and expensive. / Difficult to use this model as a communication mechanism for non-technical personnel.

Applicable to: Critical system, especially where a safety or security case must be made before the system is put into operation.

- **Software Process Model: Reuse-based**

Requirements Specification → Component Analysis → Requirements modification → System Design with Reuse → Development and Integration → System Validation.

Based on systematic reuse. Useful when systems can be integrated from existing components or Commercial-off-the-Shelf (COTS, 규격품) systems. This approach is becoming more important but still limited experience (cases) with it.

- **Software Process Model: Rapid Prototyping**

  Initially develop core functions of the system. Help to more specify user requirements.

  Advantages are: Provides a working model to the user early in the process, enabling early assessment (평가) and increasing user's confidence. / The developer gains experience and insight by developing a prototype there by resulting in better implementation of requirements. / The prototyping model serves to clarify requirements, which are not clear, hence reducing ambiguity and improving communication between the developers and users. / Helps in reducing risks associated with the software.

  Disadvantages are: High expectation. / Demand for cost reduction.

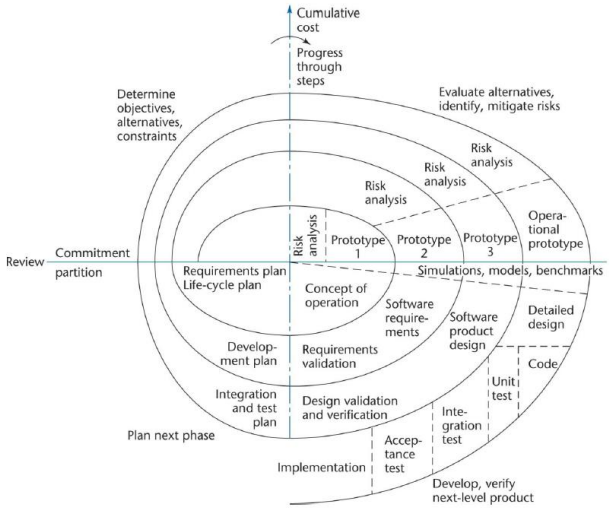
- **Software Process Model: Incremental**

  User requirements are prioritized; the highest priority requirements are included in early builds. Design is partitioned into a series of prioritized builds.

  Advantages are: Can release the first version of product as fast as possible. / Can dominate the market while investigating user reaction. / Low risk of overall project failure.

  Disadvantages are: Requires planning at the management and technical level. / Becomes invalid when there is time constraint on the project schedule or when the users cannot accept the phased deliverables.

- **Software Process Model: Spiral**



Each loop in the spiral represents a phase in the process. Each phase consists of Planning / Determining Objectives and Methods / Risk Analysis / Development and Validation.

Advantages are: Risk reduction by risk analysis. / Development of high quality software satisfying customer's requirement.

Disadvantages are: May cost too much. / Developers should be skilled at and familiar with risk analysis and planning. / Very careful and cautious development; may fail to dominate commercial markets.

Applicable to: Big size software products. / Software having high risks.

- **Necessary Works on: Requirement and Analysis**

  High-level statement for customers, more detailed system specification for developers.

  Necessary works are: Feasibility (possibility) study. / Requirements elicitation (obtaining, deriving), analysis, specification and validation.

- **Necessary Works on: Design**

  Design a software structure that realizes the specification.

  Necessary works are: Architectural design which is sub-system (modules) and their relationships. / Interface design which is interface between each sub-system and other sub-systems (I/O, params, etc.) / Component design which is necessary components that can be applicable for main system and its interfaces with main system. / Data structure design. / Algorithm design.

  The design can be described using graphical models such as: Data-flow model / Entity-relation-attribute model / Structural model (system components and interactions) / Object-oriented model (inheritance, how objects interact with each other).

- **Necessary Works on: Implementation**

  Translate the software structure (design) into an executable program. The activities of design and implementation are closely related.

  Necessary works are: Translating a design into a program and removing errors of program. / Programming is a personal activity; so it is important to cooperate with other programmers according to their rules while developing a big size of software. / Programmers carry out testing to discover faults in the program and remove these faults whit the debugging process.

- **Necessary Works on: Validation**

  Verification and validation is to show that a system satisfies its specification and meets the requirements of the customer.

  Necessary works are: Checking functions and testing the performance and operation. / Executing the system with several test cases that are derived from the specification of the real data.

- **Software Project Management**

  Concerned with activities to make certain that software is delivered on time and on schedule, satisfying requirements of both customers and developers.

  PM is necessary because software development is always controlled by budget and schedule constraints that are determined by customers and developers.

  Activities are: Proposal writing / Project planning and scheduling / Project costing / Project monitoring and reviews / Personnel selection and evaluation / Report writing and presentation.

- **Project Planning**

  Probably the most time-consuming activity in project management. Continuing activity from requirement to delivery.

  Validation plan describes the approach, resources and schedule used for system validation.

  Maintenance plan predicts the maintenance requirements of the system, maintenance costs and effort required.

  Staff development plan describes how 소 skills and experience of the project team members will be developed.

- **Project Scheduling**

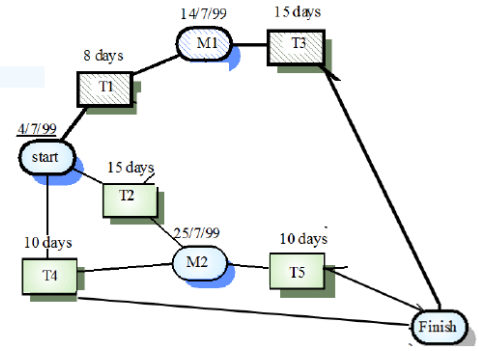
Split a big project into small tasks. Tasks should not be too small or too big. / Estimate time and resources required to complete each task. / Assign employees to appropriate tasks. / Minimize task dependencies to avoid delays caused by one task waiting for another to complete. / Greatly affected by project manager's skill and experience.

Software requirements → Identify activities → Identify activity dependencies → Estimate resources for activities → Allocate people to activities → Create project charts → Back to Estimate or Result activity charts and bar charts

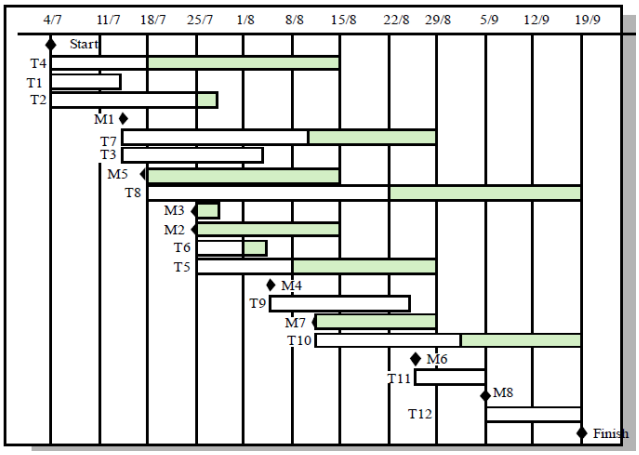
- **Project Scheduling: Activity chart**

Task	Duration (days)	Dependencies
T1	8	
T2	15	
T3	15	T1 (M1)
T4	10	
T5	10	T2, T4 (M2)

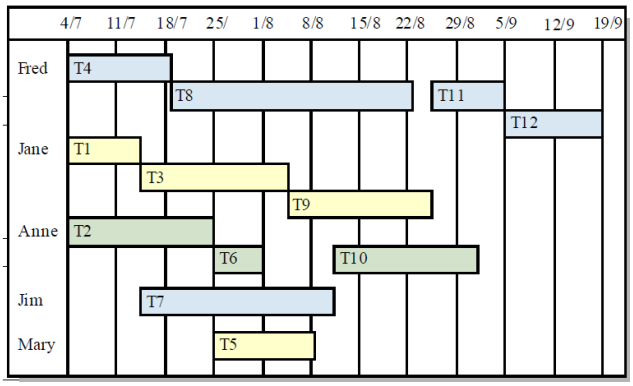
(T : task , M : milestone)



- **Project Scheduling: Activity bar chart**



- **Project Scheduling: Staff allocation**



- **Risk Management: Identification**

Identify potential risks including project, product and business risks. The result is list of potential risks.

Project risks: affect schedule or resources / *Product risks*: affect the quality or performance of the software / *Business risks*: affect the profits of organization (company) developing the software.

Staff resignation:	Project and Product
Management change:	Project
Hardware unavailability:	Project
Requirements change:	Project and Product
System underestimation	Project and Product
CASE tool underperformance	Product
Technology change	Business
Product competition	Business

Causes of Risk	Example of Risks
Technology	- The database system cannot process as many transactions per second as expected. - Software components that should be reused contain defects.
People	- It is impossible to employ experts having the required skills. - A key staff is sick and unavailable at critical times.
Tools	- The code generated by CASE tools is inefficient. - CASE tools produce inaccurate results.
Requirements	- An additional requirement requires a major design modification.
Estimation	- The time required to develop the software is underestimated. - The size of the software is underestimated.
Organization	- Organization's financial problems force into reducing the project budget.

- **Risk Management: Analysis**

  Analyses the seriousness and negative effects of the risk. The result is prioritized risk list, rank of risks.

  Probability may be: very low, low, moderate, high or very high.
  Risk effects may be: disastrous, serious, tolerable or trivial.

Example of Risks	Probability	Seriousness
Software components that should be reused contain defects.	Low	Disastrous
A key staff is sick and unavailable at critical times	High	Tolerable
The code generated by CASE tools is inefficient.	Moderate	Serious
The size of the software is underestimated.	Very Low	Trivial
Organization's financial problems force into reducing the project budget	Very High	Serious

- **Risk Management: Planning**

  Make plans to avoid or minimize the effects of the risk. The result is risk avoidance and contingency plans.

  Consider each risk and develop a strategy to manage that risk. Avoidance strategies is how to avoid risks and how to reduce the probability of risk. Contingency plan is how to deal with the risk when the risk arises.

- **Risk Management: Monitoring**

  Monitor the risks while the project progresses. The result is report of risk assessment including effect of the risk (ex, scale of damage).

  Asses the effects of the risk when a risk occurs. Report the risk assessment to efficiently manage the risk when it occurs once more.

- **Computer-Aided Software Engineering (CASE)**

  Software tools used for developing high-quality, defect-free, and maintainable software.

- **Software Requirement**

  Requirement is the description of the system services (functions) and constraints. It may range from a high-level abstract statement of system services or constraints to a detailed mathematical functional specification.

- **Software Requirement: User Requirements**

  Defined using natural language, tables and diagrams. Written for customers. Should describe *functional* and *non-functional* requirements so that they are understandable to users who don't have detailed technical knowledge.

  Functional requirements describe system functionality or services (What the system does).

  Ex) The user should be able to access to the database. / The system should provide appropriate viewers for the user to read documents.

  Non-functional requirements describe system quality and constraints (how well the system works)

  Ex) The system should operate in real time (Execution time). / The system should be easily used without any user training (Ease of use). / The system should achieve higher accuracy than 90% (Reliability, Accuracy).

- **Use Case based Modeling**

  Support to define functions required for the system. Investigate the functionality of the system from the user's perspective (viewpoint) and the interaction between the user and the system.

  The user is called an actor. Functional requirements can be easily defined by actors and their roles. Actor can be entity (object) and Roles can be functions.

- **Software Requirement: System Requirements**

  More detailed specifications of user requirements. A structured document including detailed descriptions of the system services.

  For example,

  User Requirement statement

  The software must provide a means of representing and accessing external files created by other tools.

  Corresponding *System Requirement* statements

  1. The user should be provided with facilities to define the type of external files.
  2. Each external file type may have an associated tool which may be applied to the file
  3. Each external file type may be represented as a specific icon on the user's display.
  4. Etc…

- **Software Requirement: Software Specification**

  A detailed software description. Be used as a basis for a design or implementation. Written for developers include both user requirements and system requirements.

  It is necessary because there are potential problems of user requirement and natural language is ambiguous and unclear.

  In principle, *User Requirements* state what the system should do and *System Requirements* describe how it is processed.

  Types of software specification are…
  Form-based specification / Program Description Language (PDL) based specification / Graphical specification / Graphical specification / Mathematical specification

- **From based Specification**

  Definition of the function or entity / Description of inputs and where they come from / Description of outputs and where they go to / Definition of other entities required / Pre and post conditions / The side effects

- **Program Description Language based Specification**

  Requirements are defined using a programming language.

- **Graphical Specification**

  Graphical representation is useful in describing such as System environments / Data structures and flows / State changes and system response to events / Dynamic system behavior.

  Unified Modeling Language (UML) provides an efficient way for graphical specification. It uses sequence diagram that is, the sequence of events that takes place during some user interaction with a system.

- **Who are Involved in Requirement Analysis**

  Technical staff working with customers to define the services that the system should provide and the system's constraints.

  End-users, manages, engineers, domain experts, etc. These are called *stakeholders*.

  Objective of requirement analysis is defining require specification that satisfies demands from all stakeholders.

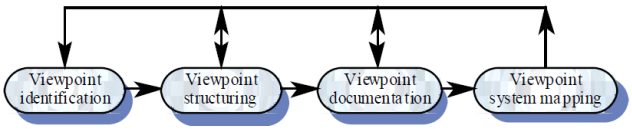
- **Problems of Requirements Analysis**

  Stakeholders don't know what they really want. Different stakeholders may have conflicting requirements. When new stakeholders emerge, the requirements may be changed during the analysis process.

- **Method based Analysis**

  Widely used approach for requirements analysis. A Viewpoint-oriented method is one of representative analysis methods.

- **Viewpoint-Oriented Requirements Definition (VORD)**



  Stakeholders tend to look at a same problem with different viewpoints. VORD is an analysis method oriented by users' viewpoints. This multi-perspective analysis is important as it is impossible to analyses system requirements with a single viewpoint.

  VORD defines viewpoints and services. Viewpoints mean users and service mean functions provided to users. Viewpoint is actor and service is role in use-case modeling.

  Viewpoint Identification discover viewpoints (users) that receive system services. Identify the services that are provided to each viewpoint.

  Viewpoint Structuring group related viewpoints as a hierarchy. Common services are provided at higher-levels in the hierarchy.

  Viewpoint Documentation define the description of the identified viewpoints are services.

  Viewpoint System Mapping transform the analysis results to an object-oriented design. Mapped viewpoint to object, and service to method.

- **In ATM banking system, viewer can be…**

  Bank customers / Hardware and software maintenance engineers / Bank managers and staffs / Database administrators and security staffs / Personnel department / Marketing department

- **Scenario Descriptions**

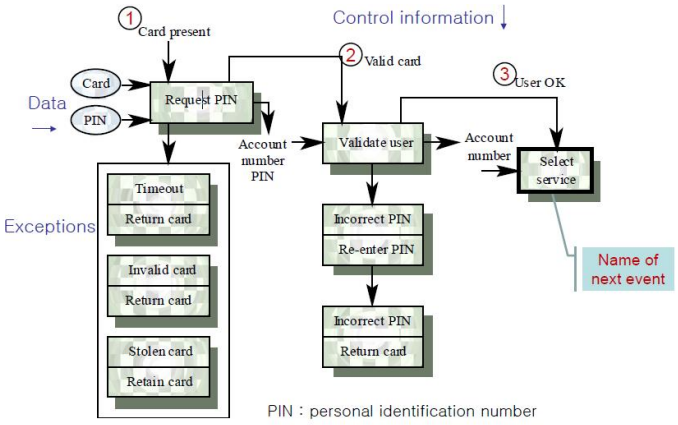
  Scenarios are descriptions of how a system is used in practice. They are helpful in defining requirements more precisely. Flow of events is generally used in the scenario such as *event scenario*.

- **Event Scenarios**

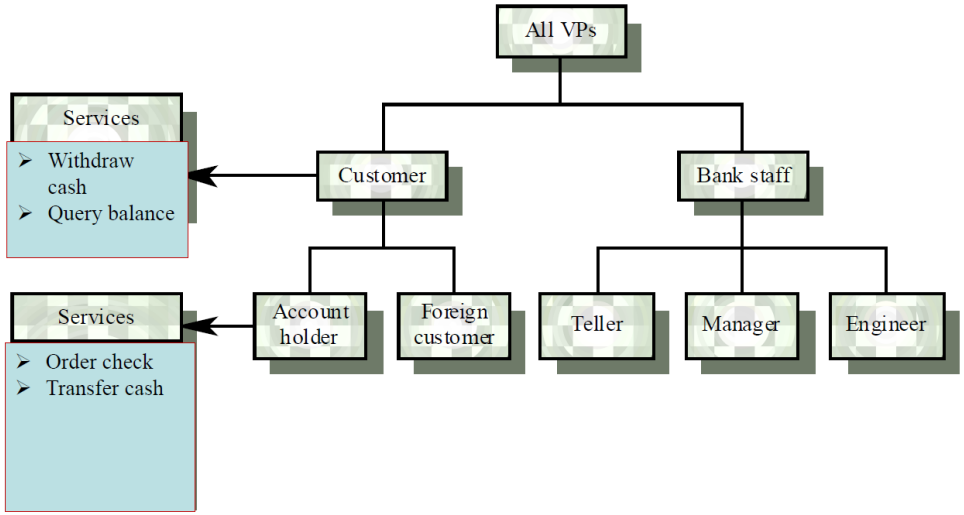
  Event scenarios are used to describe how a system responds to the occurrence of some particular event such as 'start transaction'.

  VORD includes diagrams for event scenarios including Data provided and delivered / Control information / Exception processing / The next expected event.

  Ex) Event Scenario - Start Transaction



- **Results of viewpoint structuring: Viewpoint Hierarchy**



- **Results of viewpoint documentation: VORD standard forms.**

  - Viewpoint template

Reference: Name of a viewpoint
Attributes: Common attributes (properties) of the viewpoint
Events: A set of events induced by the viewpoint
Services: A set of services provided to the viewpoint
Sub-VPs: Names of sub-viewpoints

  - Service template

Reference: Name of a service
Rationale: Reason why the service is provided
Specification: A list of service specifications
Viewpoints: List of viewpoint names receiving the service
Non-functional requirements: A set of non-functional requirements
Provider: A list of system objects which provide the service

- **Results of viewpoint documentation**

  - Ex) Documentation of customer viewpoint
    - Viewpoint template

Reference: Name of a viewpoint
Attributes: Common attributes (properties) of the viewpoint
Events: A set of events induced by the viewpoint
Services: A set of services provided to the viewpoint
Sub-VPs: Names of sub-viewpoints

    - Customer viewpoint

Reference: Customer
Attributes: Account number, PIN
Events: Start transaction, Select service, Cancel transaction, End transaction
Services: Withdraw cash, Query balance
Sub-VPs: Account holder, Foreign customer

- **Results of viewpoint documentation:**

  - Ex) Documentation of 'Withdraw cash' service
    - Service template

Reference: Name of a service
Rationale: Reason why the service is provided
Specification: A list of service specifications
Viewpoints: List of viewpoint names receiving the service
Non-functional requirements: A set of non-functional requirements
Provider: A list of system objects which provide the service

    - Withdraw cash service

Reference: Withdraw cash
Rationale: To provide cash service necessary for customers
Specification: Customers select this service by pressing the cash withdrawal button. They then enter the amount of money. If the balance is sufficient, the cash is delivered.
Viewpoints: Customer
Non-functional requirements: Deliver cash within 1 minute.
Provider: all ATMs