2019 级本科

软件工程

Software Engineering

张昕

zhangxin@cust.edu.cn

计算机科学技术学院软件工程系

Software Requirement Analyzing Tools

Review to Requirement Analysis

- Requirements analysis is the last stage of the software definition period
  - The basic task is to accurately answer "What must the system do?"
  - Requirements analysis task
    - Determine what the system must complete, that is, put forward complete, accurate, clear and specific requirements for the target system
      - Not yet sure how the system will do its job
    - Before the end of the requirements analysis phase, the system analyst should write a software requirements specification, which accurately describes the software requirements in written form

    - In the process of analyzing software requirements and writing software requirements specifications, both analysts and users play a key and indispensable role
      - Only users really know what they need, but they don't know how to use software to achieve their needs. Users must describe their needs for software as accurately and concretely as possible.
      - Analysts know how to use software to achieve people's needs, but they are not very clear about the needs of users at the beginning of the needs analysis. They must communicate with users to obtain the needs of users for software.

Review to Requirement Analysis

- Requirements analysis and specification is a very difficult and complicated task
  - There is a lot of content that needs to be communicated between users and analysts
    - Misunderstandings or omissions are prone to occur during the exchange of information between the two parties, and there may also be ambiguities
  - Throughout the needs analysis process
    - Use proven communication technology
    - To ensure that a concentrated and meticulous work process is carried out
    - The results of the verification requirements analysis must be strictly reviewed

Review to Requirement Analysis

Working Principles of Needs Analysis

- Four "must"
  - Must understand and describe the information domain of the problem,
    - Data model should be established
  - The functions that the software should complete must be defined
    - Build a functional model
  - Must describe software behavior as a result of external events
    - Modeling behavior
  - Models describing information, functions, and behaviors must be decomposed, showing details in a hierarchical manner

Entity-Relation Diagram

- In order to describe the user's data requirements clearly and accurately, system analysts usually build a conceptual data model (also called an information model)
  - Conceptual data model is a problem-oriented data model, which is a model built on data according to the user’s point of view
    - It describes the data seen from the user's perspective, reflects the user's real environment, and has nothing to do with the implementation method in the software system
  - The data model contains 3 types of interrelated information
    - Data object
    - Attributes of data objects
    - The relationship between data objects

Entity-Relation Diagram

- Data object
  - It is an abstraction of compound information that software must understand
    - The so-called compound information refers to things with a series of different properties or attributes, and things with a single value (for example, width) are not data objects
    - The data object can be
      - External entities (for example, anything that generates or uses information), Things (for example, reports), Behavior (for example, phone call), Event (for example, sound an alarm), Role (e.g. teacher, student), Unit (for example, Accounting Section), Location (for example, warehouse), Structure (for example, file), etc.
    - Data objects are related to each other
      - For example, teachers "teach" courses, students "learn" courses, the relationship of teaching or learning represents a specific connection between teacher and course or student and course
    - The data object only encapsulates the data without a reference to the operation applied to the data
      - It is the significant difference between the data object and the object-oriented paradigm in the "class" or "object"
  - Entities that can be defined by a set of attributes can be considered as data objects

Entity-Relation Diagram

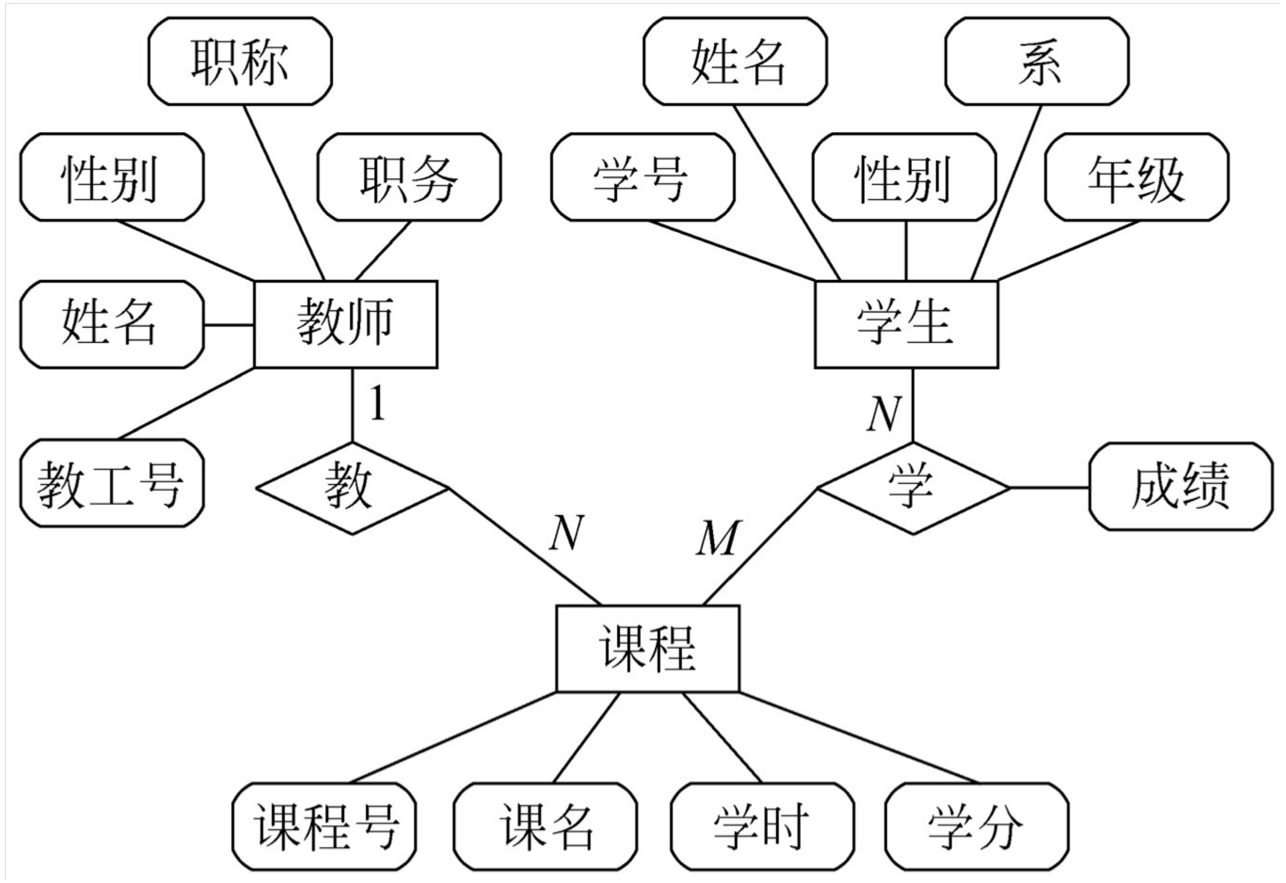
- Attributes of data objects
  - Properties define the nature of the data object
  - One or more attributes must be defined as "identifiers"
    - That is, when you want to find an instance of a data object, you can use the identifier attribute as a "keyword" (usually simply referred to as a "key").
  - A suitable set of attributes of a particular data object should be determined based on the understanding of the problem to be solved

Entity-Relation Diagram

- The relationship of the data object, i.e., the relation
  - The way in which data objects are connected to each other is called a connection, also called a relationship
  - Contact can be divided into 3 types
    - One-to-one contact (1:1)
      - For example, if a department has a manager, and each manager only works in one department, the contact between the department and the manager is one-to-one
    - One-to-many connection (1:N)
      - For example, there is a one-to-many relationship "teaching" between teachers and courses in a school, that is, each teacher can teach multiple courses, but each course can only be taught by one teacher
    - Many-to-many connection (M:N)
      - For example, the connection between students and courses ("learning") is many-to-many, that is, one student can learn multiple courses, and each course can have multiple students to learn
  - Relationn may also have attributes
    - For example, the results obtained by a student "learning" a certain course are neither the attributes of the student nor the attributes of the course
    - Since "score" depends on both a specific student and a specific course, it is the attribute of the connection between the student and the course "learning"

Entity-Relation Diagram

- The relationship of the data object, i.e., the relation



Entity-Relation Diagram

Entity-connection diagram symbol

- Use entity-relationship diagrams to build data models, referred to as E-R diagrams
    - Correspondingly, the data model depicted by E-R diagram can be called E-R model
- The E-R diagram contains 3 basic components
    - Entity (i.e. data object)
        - Usually represented by a rectangular box
    - relation
        - Represented by diamond-shaped boxes connecting related entities
    - Attributes
        - Represented by an ellipse or a rounded rectangle
        - A straight line connects the entity (or relationship) with its attributes

Data normalization

- Reason
  - Software systems often use a variety of long-term stored information, which is usually organized and stored in a database or file in a certain way. In order to reduce data redundancy, avoid insertion or deletion exceptions, and simplify the process of modifying data, it is usually necessary to change the data Structure standardization
- Usually use "normal forms" to define the degree of elimination of data redundancy
  - The first normal form (1 NF) has the highest degree of data redundancy, and the fifth normal form (5 NF) has the smallest degree of data redundancy
    - The higher the paradigm level, the more tables that need to be decomposed to store the same data. Therefore, the process of "storing itself" is more complicated.
  - As the level of the paradigm increases, the degree of matching between the storage structure of the data and the structure based on the problem domain also decreases.
    - Data stability is poor when demand changes
  - The increase in the paradigm level will increase the number of tables that need to be accessed, so the performance (speed) will decrease
    - From a practical point of view, the third normal form is more appropriate for most occasions

Data normalization

- Paradigm
  - The degree of normalization is usually distinguished according to the dependencies between attributes
  - Dependency between attributes meets different levels of requirements for different paradigms
    - Meet the minimum requirements is the first normal form
    - In the first paradigm, the one that further satisfies some requirements is the second paradigm.
    - The rest and so on
  - First normal form
    - Each attribute value must be an atomic value, that is, it is just a simple value without internal structure
  - Second normal form
    - Satisfy the first normal form conditions
    - And each non-keyword attribute is determined by the entire keyword (not by a part of the keyword)
  - Third normal form
    - Meet the conditions of second normal form
    - Each non-keyword attribute is determined only by keywords
    - And a non-keyword attribute cannot just be a further description of another non-keyword attribute
      - That is, a non-keyword attribute value does not depend on another non-keyword attribute value

State transition diagram

- The behavior model of the software system should be established in the process of requirement analysis
- State transition diagram (referred to as state diagram for short)
  - Express the behavior of the system by depicting the state of the system and the events that cause the transition of the system state
  - The state diagram also indicates what actions the system will do as a result of a specific event (for example, processing data)
- The state diagram provides a behavior modeling mechanism, which can meet the requirements of the third analysis criterion of demand analysis
  - Must describe software behavior as a result of external events

State transition diagram

- State
  - It is any observable system behavior pattern
  - A state represents a behavior mode of the system
- State defines how the system responds to events
  - System response to events
    - It can either do one (or a series of) actions
    - It can also just change the state of the system itself
    - It can also change the state and perform actions
- The states defined in the state diagram mainly include
  - Initial state (i.e. initial state)
  - Final state (i.e. final state) and
  - Intermediate state
- There can only be one initial state in a state diagram, and the final state can have 0 or more
- The state diagram can represent the cyclic operation process of the system, and it can also represent the one-way life cycle of the system
  - When depicting the running process of a loop, you usually don't care how the loop is started
  - When depicting the one-way life cycle, it is necessary to indicate the initial state (the initial state when the system starts) and the final state (the final state when the system ends)

State transition diagram

- Event
  - Is something that happened at a certain moment
  - It is an abstraction of external events that cause the system to act or (and) transition from one state to another
    - For example, the internal clock indicates that a certain period of time has passed, and user movement or mouse clicks are all events
- In the state transition diagram, the event is the control information that causes the system to act or (and) transition the state

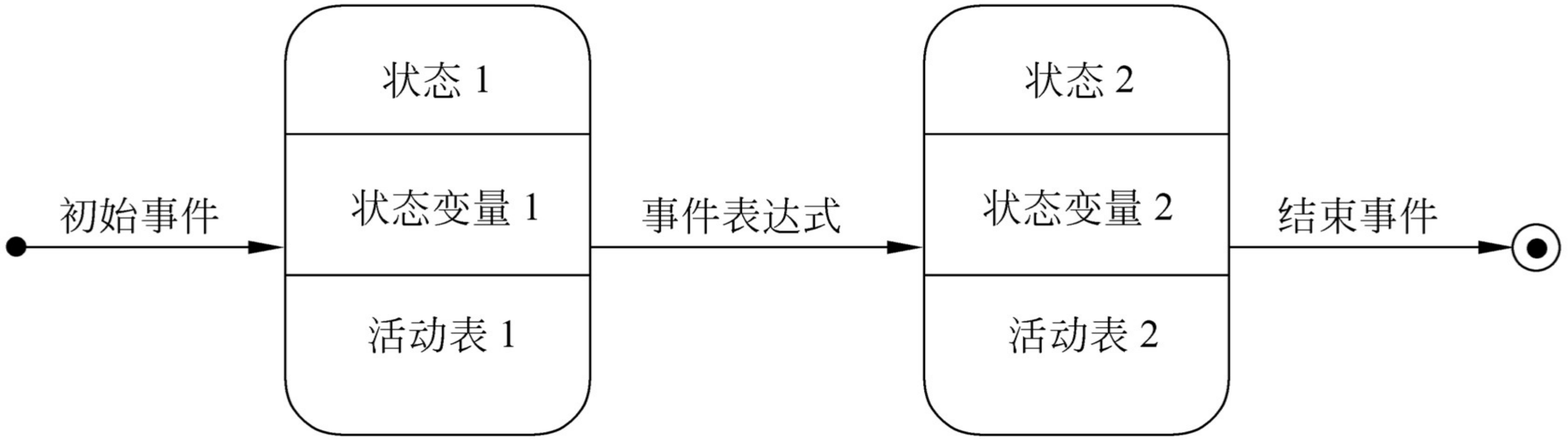
State transition diagram

- Symbolic expression
    - The initial state is represented by a filled circle
    - The final state is represented by a pair of concentric circles (the inner circle is a solid circle)
    - The intermediate state is represented by a rounded rectangle, which can be divided into upper, middle and lower parts by two horizontal lines
        - The upper part is the name of the state, which is required
        - The middle part is the name and value of the state variable, which is optional
        - The following part is the activity table, which is optional
- Syntax format of activity table
    - Event name (parameter table)/action expression,
        - "Event name" can be the name of any event
            - The following 3 standard events are often used in the activity table: entry, exit and do
            - The entry event specifies the action to enter the state
            - The exit event specifies the action to exit the state
            - The do event specifies the action in this state
    - You can specify the parameter table for the event when needed
    - The action expression in the activity table describes the specific action that should be done

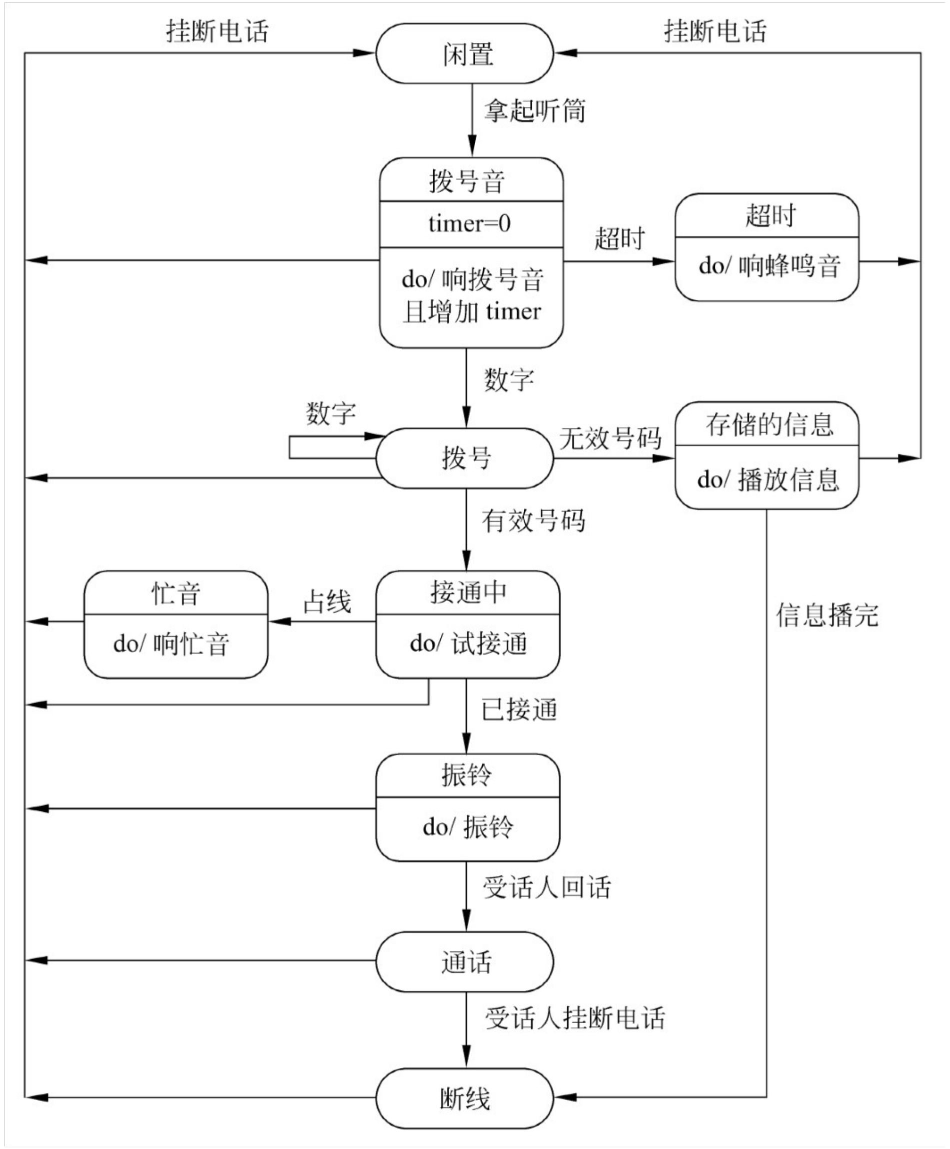
State transition diagram

- Symbolic expression (cont.)
  - The connection line with arrows between the two states in the state diagram is called state transition, and the arrow indicates the direction of the transition.
  - State transitions are usually triggered by events
    - In this case, the event expression that triggers the transition should be marked on the arrow line representing the state transition
    - If the event is not marked on the arrow line, it means that the conversion is automatically triggered after the internal activity of the source state is executed.
    - The syntax of the event expression is as follows:
      - Event description [Guard condition]/action expression
      - The syntax of the event description is: event name (parameter list).
      - Guard condition is a boolean expression
      - If both event description and guard conditions are used at the same time, the state transition will occur if and only if the event occurs and the Boolean expression is true
      - If there is only a guard condition and no event description, the state transition will occur as long as the guard condition is true
      - The action expression is a procedural expression, which is executed when the state transition starts

State transition diagram



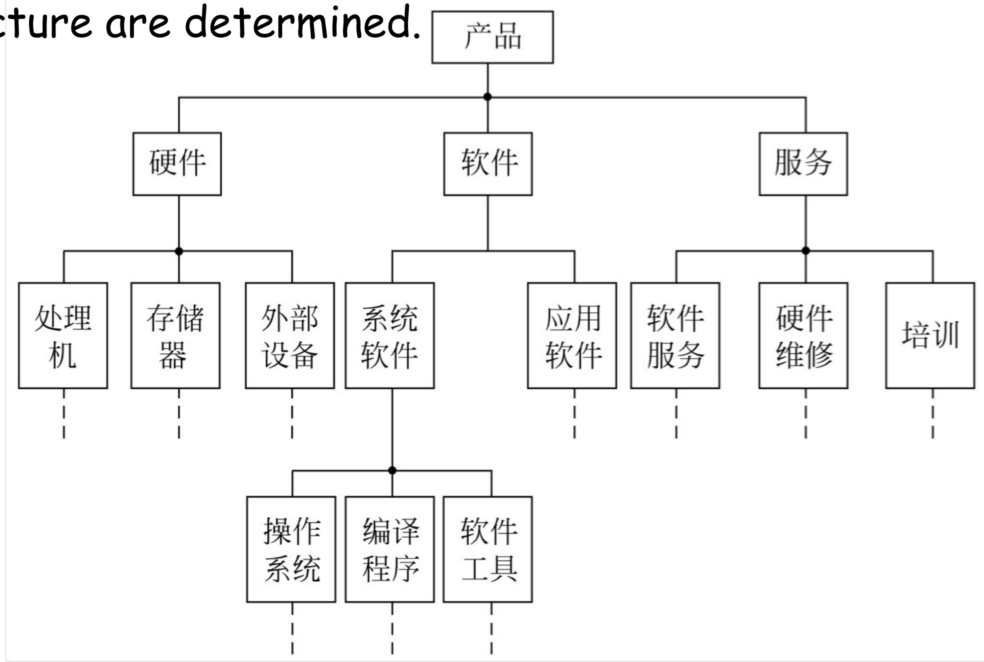
State transition diagram - Example



- The figure shows
  - The phone is idle when no one is calling;
  - When someone picks up the handset, it enters the dial tone state. After reaching this state, the behavior of the phone is to ring the dial tone and time;
  - At this time, if the person who picked up the receiver changes his mind and does not want to call, he puts the receiver down (hangs up), and the call goes back to the idle state;
  - If you pick up the receiver and do not dial for a long time (timeout), it will enter the timeout state;
  - ...

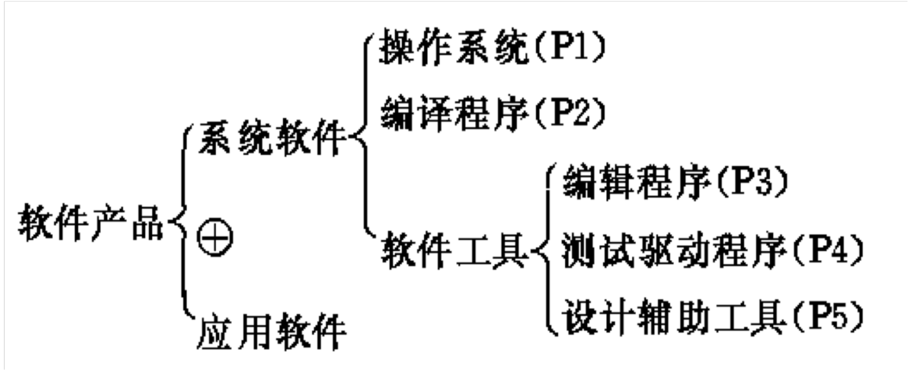
Hierarchical block diagram

- Use a series of multi-level rectangular boxes in a tree structure to depict the hierarchical structure of data
  - The top level of the tree structure is a single rectangular box, which represents the complete data structure
  - The following rectangular boxes represent a subset of this data
  - The boxes at the bottom represent the actual data elements that make up this data (elements that cannot be divided)
- With the refinement of the structure, the hierarchical block diagram depicts the data structure in more and more detail. This mode is very suitable for the needs of the requirements analysis stage.
  - The system analyst starts with the classification of the top-level information, and repeats the refinement along each path in the diagram until all the details of the data structure are determined.
- For example
    - depict the data structure of all products of a computer company



Warnier diagram

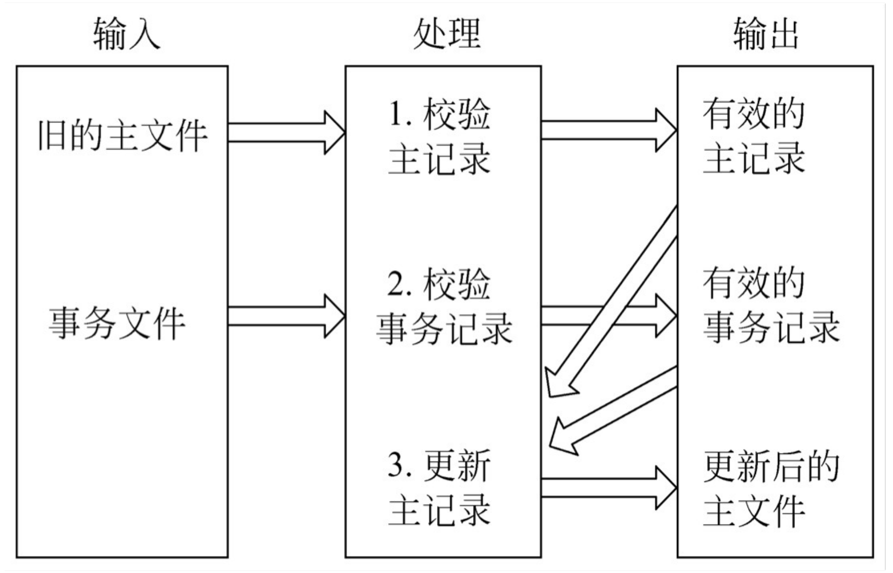
- Similar to the hierarchical block diagram, the Warnier diagram also uses a tree structure to depict information, but this graphical tool provides a richer depiction method than the hierarchical block diagram
- Warnier diagram can show the logical organization of information
  - Can point out that a type of information or an information element is repeated
  - It can also indicate that specific information appears conditionally in a certain type of information
- Because repetition and conditional constraints are the basis for explaining the software processing process, it is easy to turn the Warnier diagram into a software design tool



- Example: a software product is either system software or application software
  - There are P1 operating systems in the system software
  - P2 compilers, in addition to software tools
  - Software tool is a kind of system software
    - It can be further subdivided into
      - Edit program
      - Test driver
      - Design aids

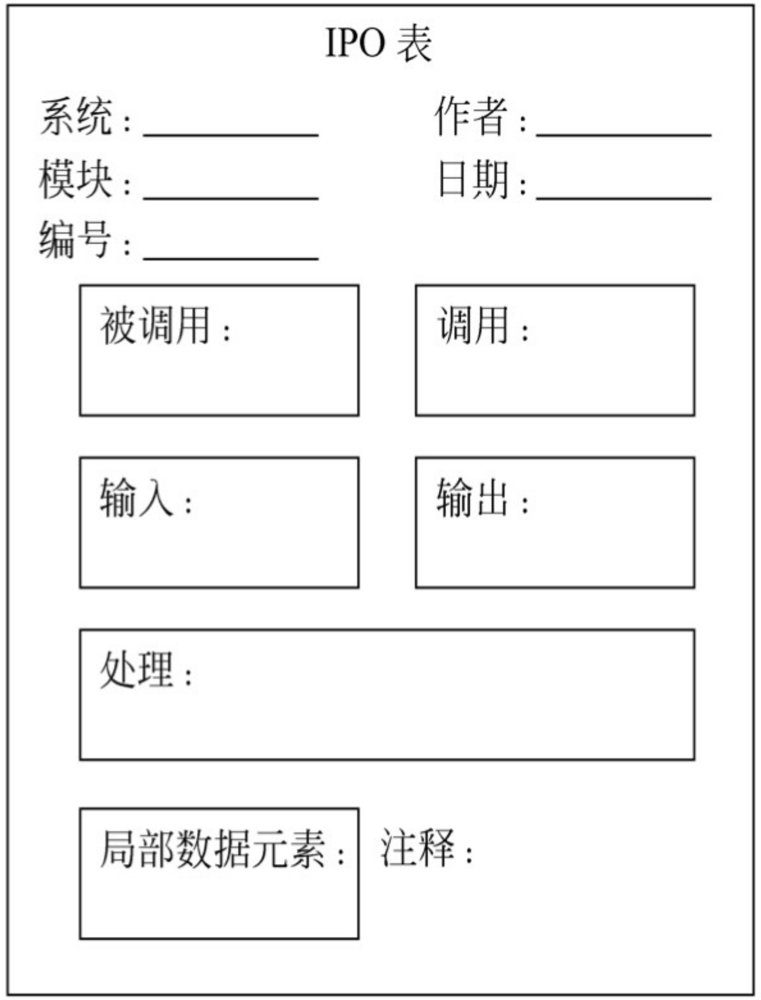
IPO Chart

- IPO diagram is short for input, processing, and output diagram
  - It is a graphical tool developed and perfected by IBM in the United States
  - Can easily describe the relationship between input data, data processing and output data
- The basic symbols used in the IPO chart are few and simple, easy to learn and use
- Basic form
  - List the relevant input data in the box on the left
  - List the main processing in the middle box
    - The order of processing listed in the processing box implies the order of execution
    - But using these basic symbols is not enough to accurately describe the details of the execution of the processing
  - List the generated output data in the box on the right
  - Sometimes thick arrows similar to vector symbols are used to clearly point out the data communication situation



Improved IPO chart (IPO table)

- More useful than the original IPO diagram in the software design process
- In the demand analysis stage, you can use the IPO diagram to briefly describe the main algorithm of the system (that is, the basic algorithm of each processing in the data flow diagram)
    - In the demand analysis stage, many additional information in the IPO chart is not yet available
    - However, these diagrams can be further supplemented and revised in the software design stage as a document in the design stage
- That is, the important advantage of using the IPO diagram as a tool to describe the algorithm in the demand analysis stage


IPO 表

系统：＿＿＿＿ 作者：＿＿＿＿
模块：＿＿＿＿ 日期：＿＿＿＿
编号：＿＿＿＿

被调用：
调用：
输入：
输出：
处理：
局部数据元素：
注释：

Verify software requirements

- The results of the requirements analysis stage are an important basis for developing software systems
    - A large number of statistics show that 15% of errors in software systems originate from wrong requirements
- Requirements analysis must strictly verify its correctness
    - The purpose is to improve software quality, ensure the success of software development, and reduce software development costs

Verify software requirements

- Verify from the following 4 aspects
  - consistency
    - All requirements must be consistent, and any one requirement cannot contradict other requirements
  - Completeness
    - The requirements must be complete, and the specification should include every function or performance required by the user
  - Reality
    - The specified requirements should be basically achievable with existing hardware technology and software technology
      - It is possible to make some predictions about the progress of hardware technology, but it is difficult to make predictions about the progress of software technology. We can only judge the reality of demand based on the current technology level.
  - Effectiveness
    - It must be proved that the requirements are correct and effective, and can indeed solve the problems faced by users

Verify software requirements

- Verify the consistency of requirements
  - When the result of the requirements analysis is written in natural language, there is no other better "testing" method at present except for manual technical review to verify the correctness of the software system specification.
  - Informal specifications are difficult to verify
    - Especially when the target system is large in scale and the specifications are long, the effect of manual review is not guaranteed
    - Redundancies, omissions, and inconsistencies may not be discovered and continue to remain
    - As a result, the software development work cannot be carried out smoothly on the correct basis
  - In order to overcome the above difficulties, a formal method of describing software requirements is proposed
    - When the software requirements specification is written in a formal requirements statement language, software tools can be used to verify the consistency of the requirements, thereby effectively ensuring the consistency of the software requirements

Verify software requirements

- Verify the reality of requirements
  - In order to verify the reality of requirements, analysts should refer to previous experience in developing similar systems and analyze the possibility of using existing software and hardware technologies to achieve the target system
  - When necessary, simulation or performance simulation technology should be used to assist in analyzing the reality of software requirements specifications