Chapter 6
System Engineering

Software Engineering: A Practitioner's Approach, 6th edition
by Roger S. Pressman

本章要点

- 系统工程 的层次性
- 产品工程 的层次性
- 系统建模

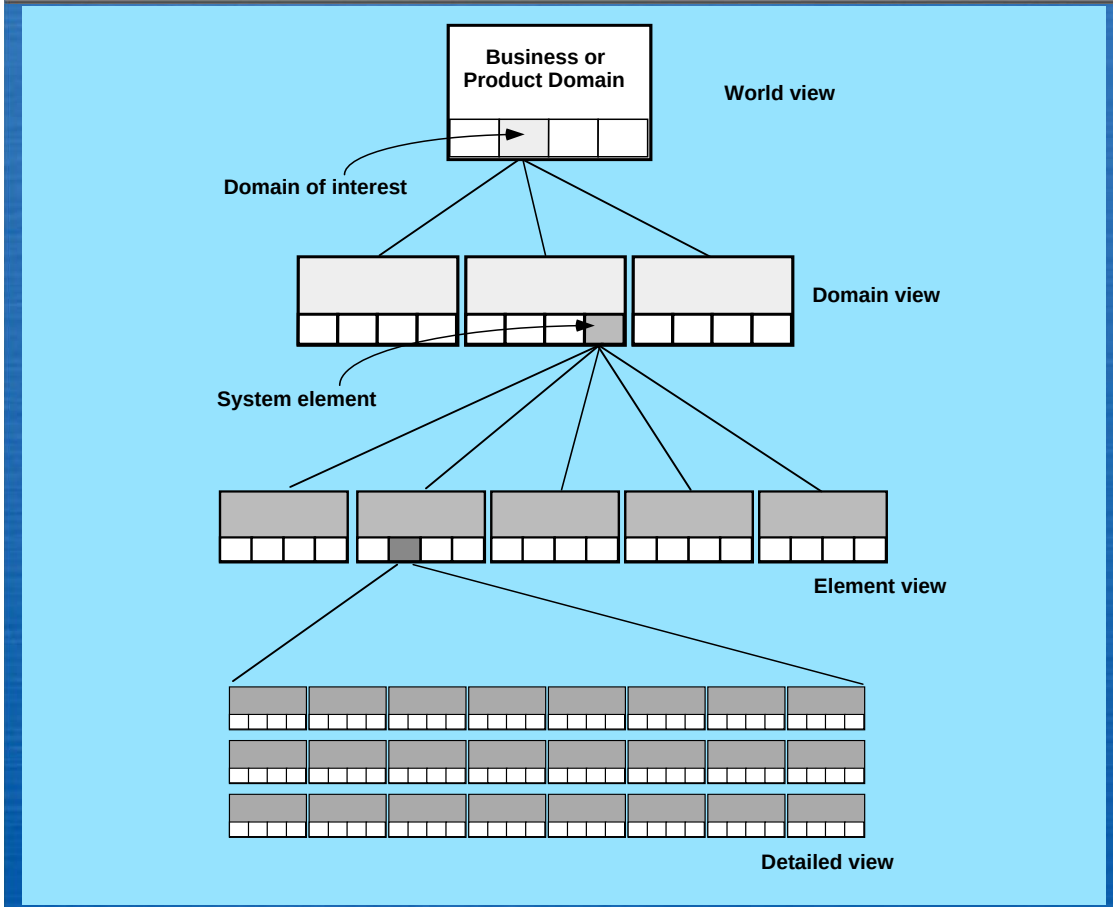
背景

- 软件工程化之前必须了解软件所处的外部“系统”。为此，必须确定系统的整体目标，必须识别硬件、软件、人员、数据库、规程和其它系统要素的角色，而且必须对有效需求进行提取、分析、说明、建模、确认和管理，这些活动都是系统工程的基础。
- 要做到“既见树木、又见森林”

System Engineering

- Elements of a computer-based system
  - Software
  - Hardware
  - People
  - Database
  - Documentation
  - Procedures
- Systems
  - A hierarchy of macro-elements[宏要素]

The Hierarchy of System Engineering



全局视图(WV)由领域集合组成:
WV=(D1,D2,D3,…,Dn)

每个领域(Di)由特定要素组成:
Di=(E1,E2,E3,…,Em)

每种特定要素(Ei)由完成特定功能的构件(Ci)来实现:
Ei=(C1,C2,C3,…,Ck)

System Modeling

- define the processes that serve the needs of the view under consideration. [定义在所考虑视图中满足需要的过程]
- represent the behavior of the processes and the assumptions on which the behavior is based.[描述过程行为和该行为所依据的假设]
- explicitly define both exogenous and endogenous input to the model.[明确定义模型的外在和内在输入]
  - exogenous inputs link one constituent(要素) of a given view with other constituents at the same level or other levels; endogenous input links individual components of a constituent at a particular view.
- represent all linkages (including output) that will enable the engineer to better understand the view.[描述有助于工程师理解视图的全部联系(包括输出)]

Business Process Engineering(BPE)

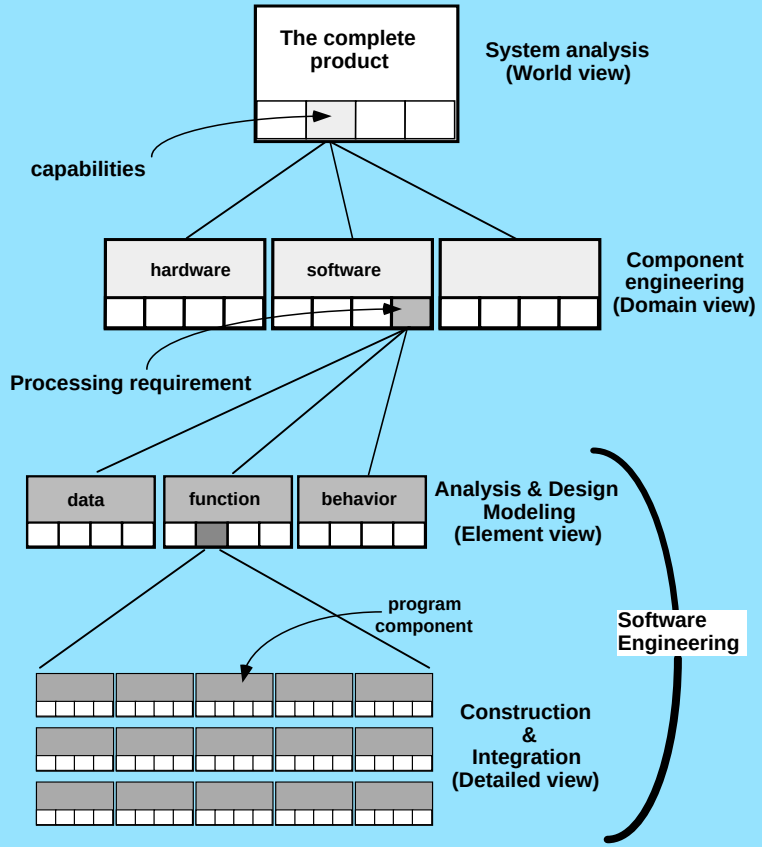
BPE的目标是定义一个能有效利用信息进行业务活动的体系。

- uses an integrated set of procedures, methods, and tools to identify how information systems can best meet the strategic goals of an enterprise
- focuses first on the enterprise and then on the business area
- creates enterprise models, data models and process models
- creates a framework for better information management, distribution, and control

System Architectures

- Three different architectures must be analyzed and designed within the context of business objectives and goals:
  - data architecture
  - applications architecture
  - technology infrastructure
- data architecture provides a framework for the information needs of a business or business function
- application architecture encompasses those elements of a system that transform objects within the data architecture for some business purpose
- technology infrastructure provides the foundation for the data and application architectures

Product Engineering



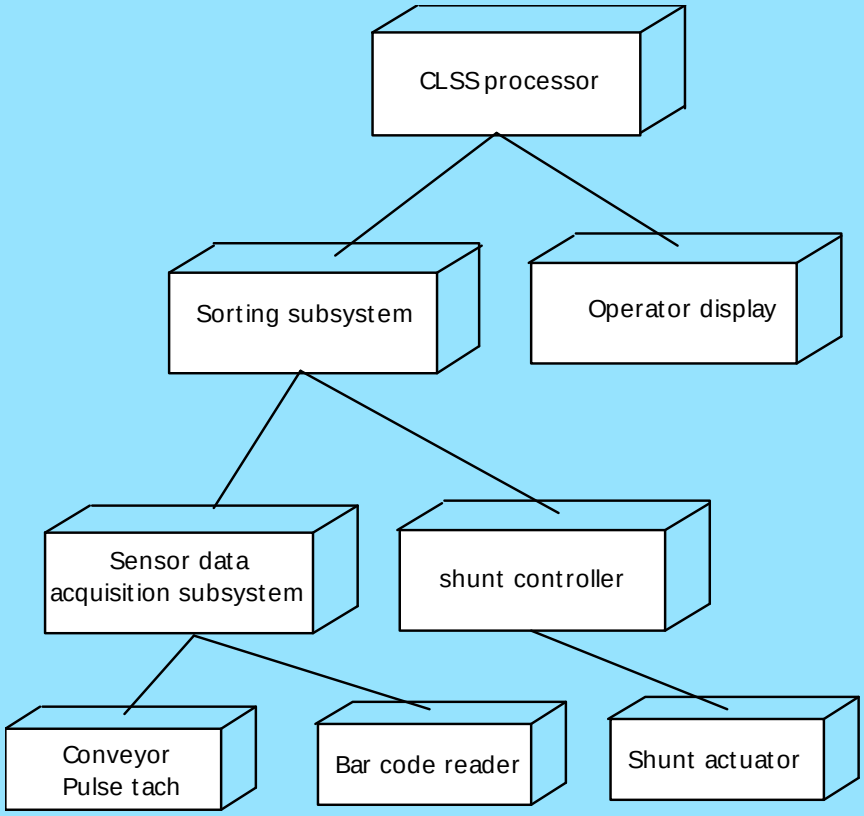
PE的目的是将用户期望的已定义的一组能力转变成真实产品。

System Modeling with UML

- UML提供了大量图表表示法,它们用于在系统和软件层次进行分析和设计.
  - Deployment diagrams
    - Each 3-D box depicts a hardware element that is part of the physical architecture of the system
  - Activity diagrams
    - Represent procedural aspects of a system element
  - Class diagrams
    - Represent system level elements in terms of the data that describe the element and the operations that manipulate the data

These and other UML models will be discussed later

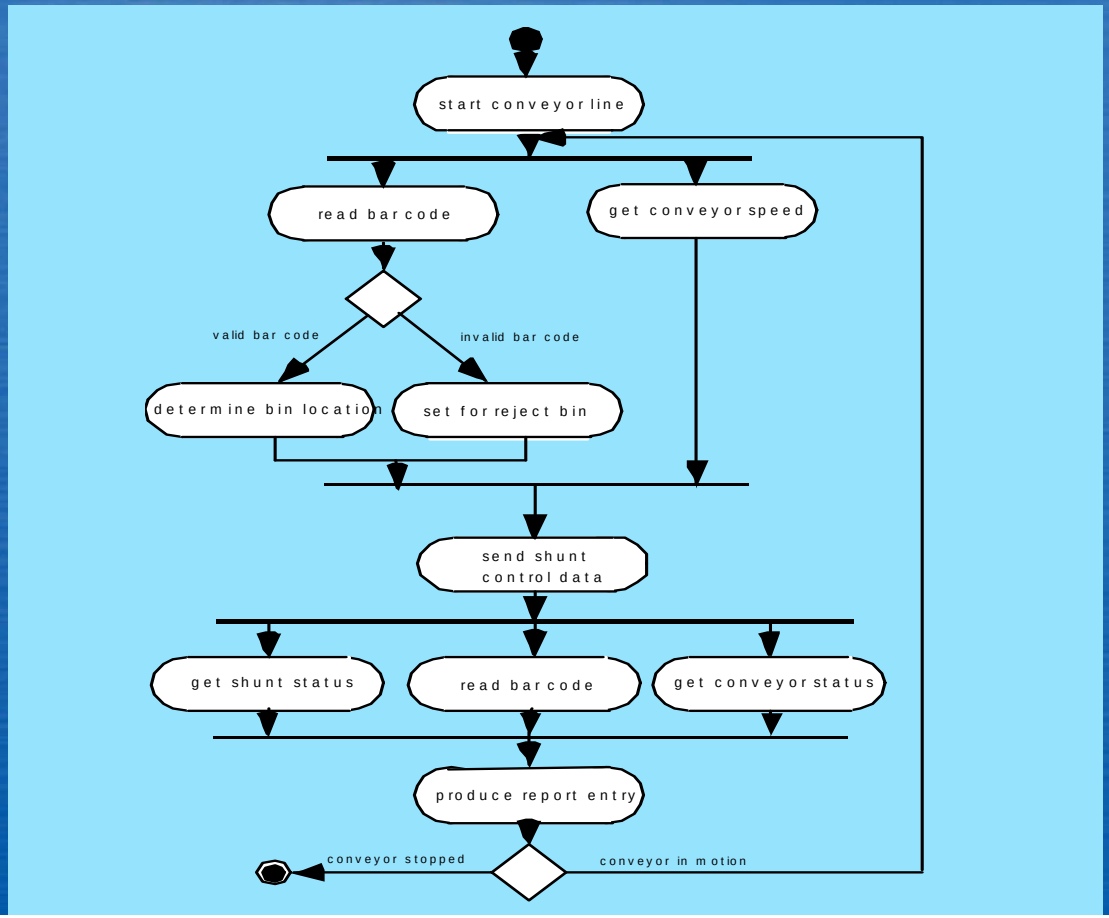
Deployment Diagram



CLSS: 传送带分类系统

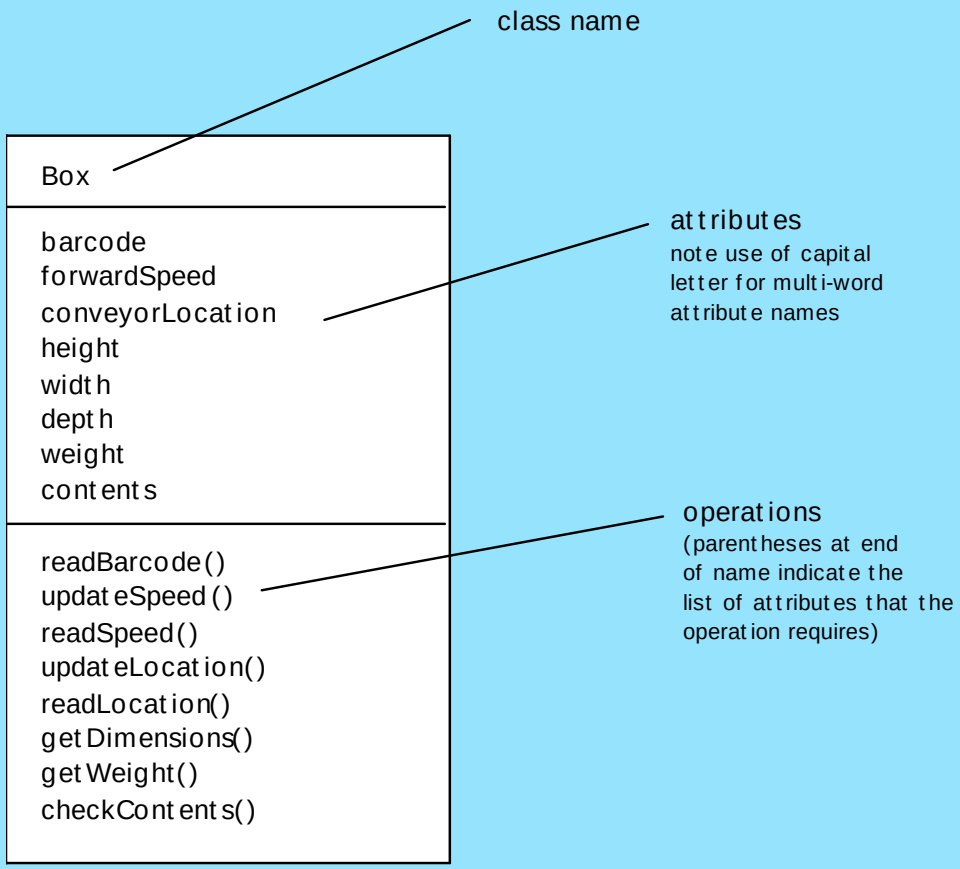
每个三维方盒描述了一个属于系统物理架构的硬件要素.

Activity Diagram



活动图表示系统实现各种功能时的具体步骤，圆角矩形表示特定的系统功能，箭头表示系统的流程，菱形表示分支(从菱形出发的箭头都需要标注)，水平实线表示并发事件.

Class Diagram



类是从问题描述中提取的. 对CLSS来说, 候选的类可能是: 箱子、传送带、条形码阅读器、分流控制器、操作者请求、报告、产品以及其它。

Chapter 7
Requirements Engineering

Software Engineering: A Practitioner's Approach, 6th edition
by Roger S. Pressman

本章要点

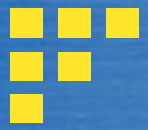
- 需求工程主要任务

概述

- 需求工程帮助软件工程师更好地理解他们将要解决的问题。
- 具体步骤：首先定义将要解决的问题范围和性质；然后是引导、帮助客户定义需要什么；接下来是精炼需求，精确定义和修改基本需求。

- “构建一个软件系统最困难的部份是确定构建什么. 其它部份工作不会像这部分工作一样, 在出错之后会如此严重地影响随后实现的系统, 并且在以后修补竟会如此的困难. ”

---Fred Brooks



Requirements Engineering Tasks

- **Inception**—Establish a basic understanding of the problem and the nature of the solution.
- **Elicitation**—Draw out the requirements from stakeholders.
- **Elaboration**—Create an analysis model that represents information, functional, and behavioral aspects of the requirements.
- **Negotiation**—Agree on a deliverable system that is realistic for developers and customers.
- **Specification**—Describe the requirements formally or informally.
- **Validation**—Review the requirement specification for errors, ambiguities, omissions, and conflicts.
- **Requirements management**—Manage changing requirements.

1 Inception

- Ask “context-free” questions
  - Who is behind the request for this work?
  - Who will use the solution (product/system)?
  - What will be the economic benefits?
  - How would you characterize “good” output from the system?
  - What problems does this solution address?
  - What environment will the product be used in?
  - Are you the right person to answer these questions?
  - Are these question relevant?
  - Can anyone else provide additional information?
  - Should I be asking you anything else?
  - ……

2 Eliciting Requirements

- Why is it so difficult to clearly understand what the customer wants?
  - Scope
    - The boundary of the system is ill-defined.
    - Customers/users specify unnecessary technical detail that may confuse rather than clarify objectives.
  - Understanding
    - Customers are not completely sure of what is needed.
    - Customers have a poor understanding of the capabilities and limitations of the computing environment.
    - Customers don’t have a full understanding of their problem domain.
    - Customers have trouble communicating needs to the system engineer.
    - Customers omit detail that is believed to be obvious.
    - Customers specify requirements that conflict with other requirements.
    - Customers specify requirements that are ambiguous or untestable.
  - Volatility
    - Requirements change over time.

2.1 CRG Meeting[协同的需求收集会议]

- Meetings are attended by all interested stakeholders.
- Rules established for preparation and participation.
- Agenda should be formal enough to cover all important points, but informal enough to encourage the free flow of ideas.
- A facilitator[推动者,发起人] controls the meeting.
- A definition mechanism (blackboard, flip charts[活动挂图], etc.) is used.
- During the meeting:
  - The problem is identified.
  - Elements of the solution are proposed.
  - Different approaches are negotiated.
  - A preliminary set of solution requirements are obtained.
  - The atmosphere is collaborative and non-threatening[胁迫的].

2.2 QFD: Quality Function Deployment

- 将客户需求转化为技术需求. QFD的目的是最大限度地让客户从软件工程过程中感到满意. 为此, QFD强调理解“什么是对客户有价值的”.
  - Function deployment determines the “value” (to the customer) of each function required of the system. [用于确定系统所需的每个功能价值]
  - Information deployment identifies data objects and events.[确认系统必须使用和产生的数据对象和事件]
  - Task deployment examines the behavior of the system.[在合适的环境下检查系统或产品的行为]
  - Value analysis determines the priority of requirements.[帮助确定上述三个部署中需求的相对优先级别]

2.3 Elicitation Work Products

- Statement of need and feasibility.[必要性和可行性陈述]
- Statement of scope. [系统或产品的范围说明]
- List of participants in requirements elicitation. [参与需求导出的客户、用户和其它共同利益者的列表]
- Description of the system's technical environment.[系统技术环境的说明]
- List of requirements and associated domain constraints.[需求列表,以及每个需求适用的领域限制]
- List of usage scenarios.[一系列使用场景]
- Any prototypes developed to refine requirements.[任何能够更好地定义需求的原型]

3 Developing Use-Cases

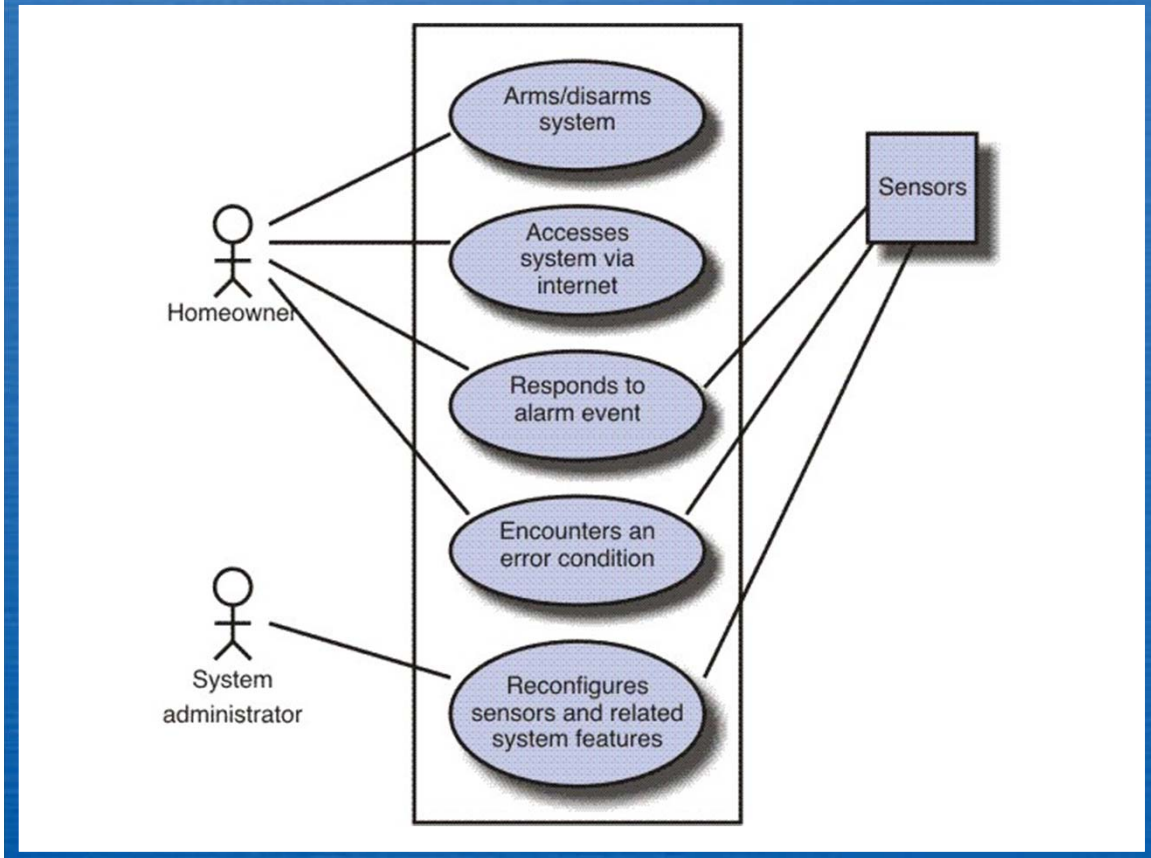
- A use-case scenario is a story about how someone or something external to the software (known as an actor) interacts with the system.
- Each scenario answers the following questions:
  - Who is the primary actor, the secondary actor(s)?
  - What are the actor's goals?
  - What preconditions should exist before the story begins?
  - What main tasks or functions are performed by the actor?
  - What exceptions might be considered as the story is described?
  - What variations in the actor's interaction are possible?
  - What system information will the actor acquire, produce, or change?
  - Will the actor have to inform the system about changes in the external environment?
  - What information does the actor desire from the system?
  - Does the actor wish to be informed about unexpected changes?

4 Building Analysis Model

4.1 Elements of the Analysis Model

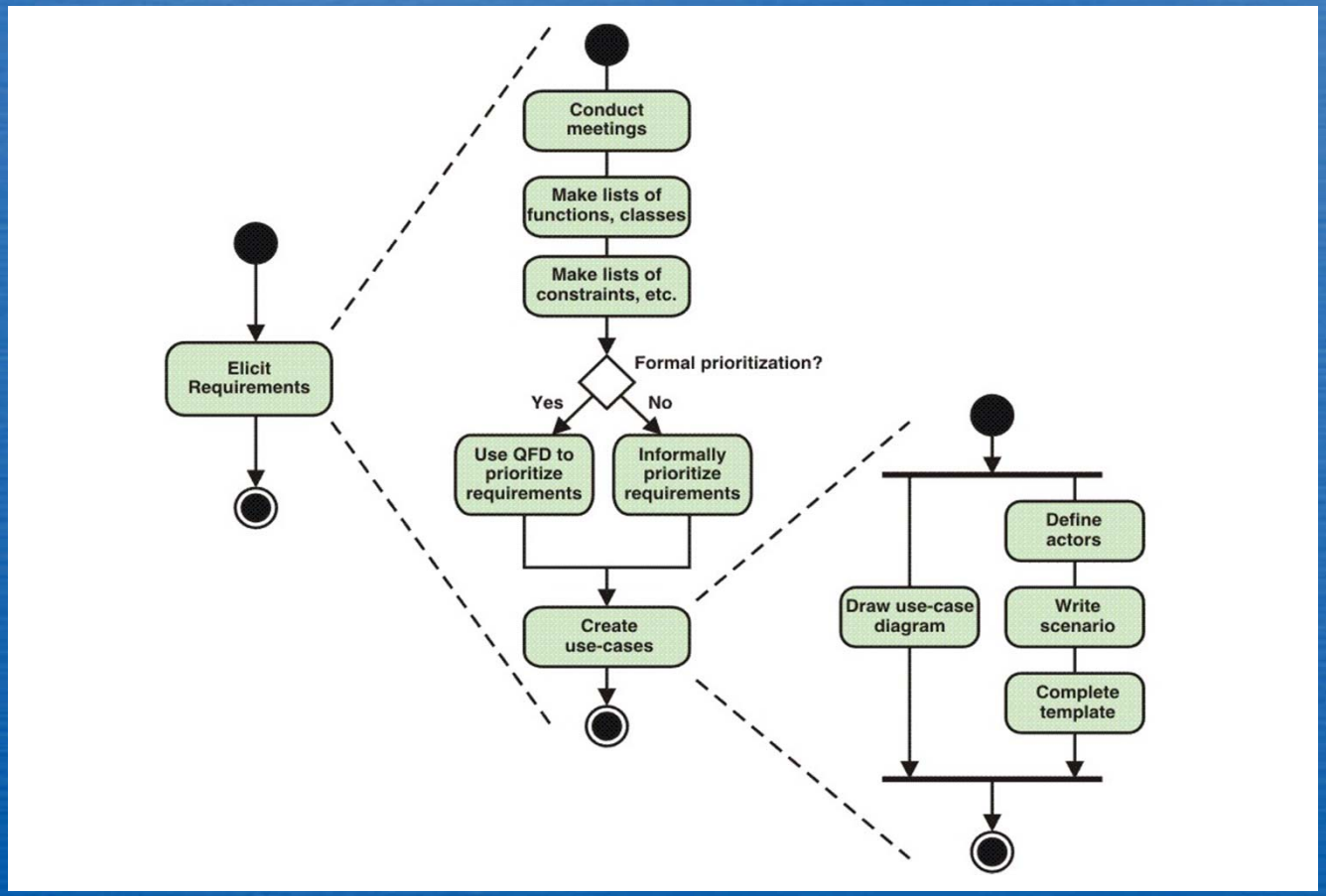
- Scenario-based elements
  - Use-case—How external actors interact with the system (use-case diagrams; detailed templates)
  - Functional—How software functions are processed in the system (flow charts; activity diagrams)
- Class-based elements
  - The various system objects (obtained from scenarios) including their attributes and functions (class diagram)
- Behavioral elements
  - How the system behaves in response to different events (state diagram)
- Flow-oriented elements
  - How information is transformed as if flows through the system (data flow diagram)

Use-Case Diagram



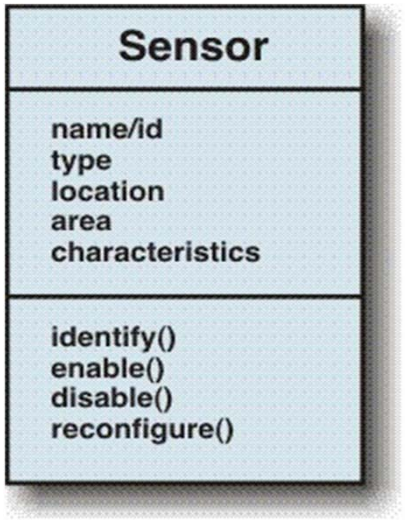
SafeHome住宅安全功能的用例图

Activity Diagram for RE



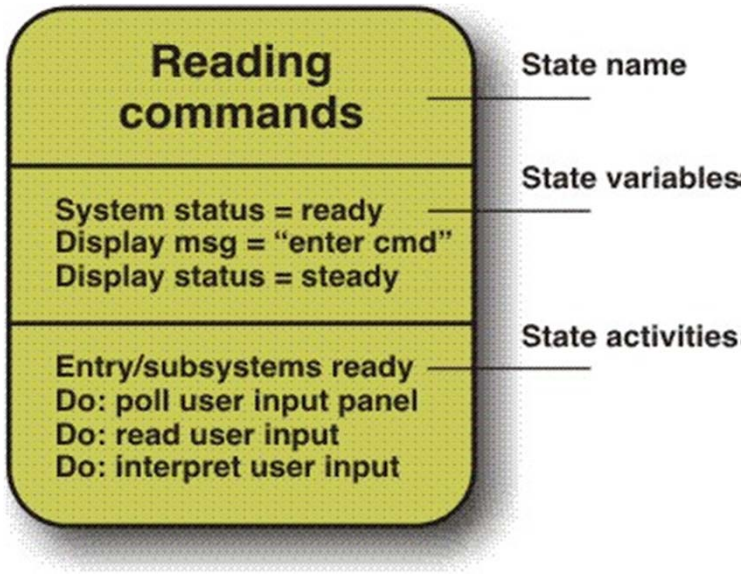
导出需求描绘了一个UML活动图,显示了三层细节内容.

Class Diagram



描绘SafeHome安全功能传感器类的类图.

State Diagram



- ◆状态名：如读指令.
- ◆状态变量：指出状态如何向外界显示自己.
- ◆状态活动：指出如何进入状态(entry/)，以及当处在该状态时可以调动的动作(do:).

5 Negotiating Requirements

- 要让客户以成本和产品投放市场的时间为背景,平衡功能、性能和其它的产品或系统特性。协调过程的目的是保证所开发的项目计划，在满足客户需求的同时反映软件团队所处的真实世界限制（如时间、人员、预算等）
  - Identify the key stakeholders
    - These are the people who will be involved in the negotiation
  - Determine each of the stakeholders “win conditions”
    - Win conditions are not always obvious
  - Negotiate
    - Work toward a set of requirements that lead to “win-win”

6 Validating Requirements

- Is each requirement consistent with the objective of the system?
- Have all requirements been specified at the proper level of abstraction?
- Is the requirement really necessary?
- Is each requirement bounded and unambiguous?
- Does each requirement have attribution[归属]?
- Do any requirements conflict with other requirements?
- Is each requirement achievable in the system's technical environment?
- Is each requirement testable, once implemented?
- Does the model reflect the system's information, function and behavior?
- Has the model been appropriately "partitioned"?
- Have appropriate requirements patterns been used?