# Requirements Analysis

Requirements are needs or conditions to which the system must conform. This discipline does not give any solution to the system under consideration (SuD); rather, it acts as a pool between user and the system developer by clarifying the system functionalities and the user expectations. Nevertheless, the act of analysis should not be intoxicated with any technical ideas or implementation aspects.

## 1. Type of Requirements: FURPS+ model

Some consider two broad categories of requirements: functional and non-functional requirements. The functional requirements are modelled by use cases and the non–functional are explained in Supplementary Specifications. But there is a broadly accepted model of requirements for OO analysis, known by the acronym FURPS+ model.

- **Functional requirements**: features, capabilities, security
- **Usability**: human factors, help, documentation
- **Reliability**: frequency of failure, predictability, recoverability
- **Performance**: response time, throughput, accuracy, availability, resource usage
- **Supportability**: adaptability, maintainability, internationalization, configurability

'+' means sub factors like

- **Implementation**: resource limitations, languages and tools, hardware
- **Interface**: constraints imposed by interfacing with external factors.
- **Operations**: system management in its operational setting
- **Packaging**
- **Legal**: licensing and so forth.

## 2. Use Case Model

- Excellent technique to understand and describe requirements of a system
- It is the model of system's functionality and environment.
- Introduced first by Ivar Jacobson in 1986.
- Use cases record goals and stories of different users and stakeholders of the system.
- Use cases are text documents and use case modelling is primarily an act of writing text, not drawings.
- Use cases are not object oriented. So, it can equally be applied to requirment analysis of non-OO projects too.
- UML defines use case diagrams by showing actors and use cases with their relationships.

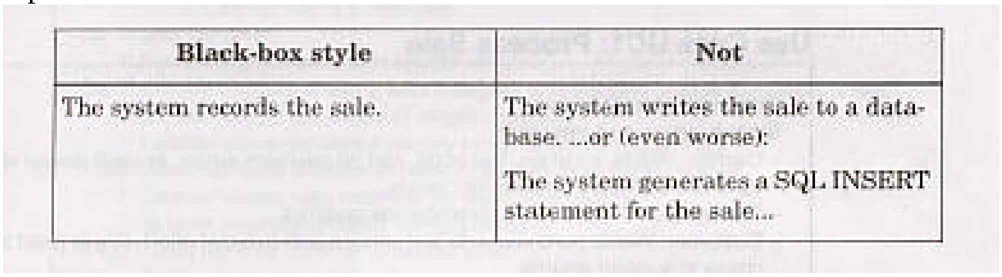
### 2.1. Use case contents

**An Actor**: is external entity that has a role in the system (behaviour) e.g. person, computer, external system etc.

**A scenario** : an instance of a use case, which is collection of actions and interactions between actors and the system. This may also be termed as a story to fulfil a specific goal or of using the system. A scenario may have both success and failure stories.

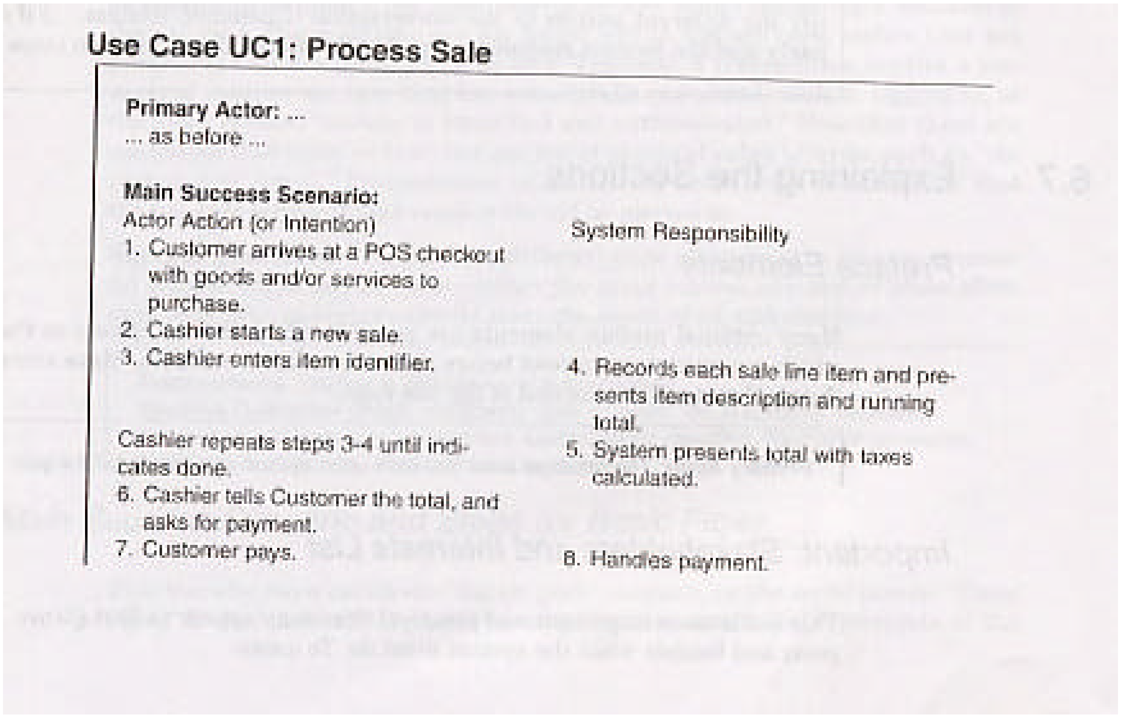
## 2.2. Use case types and formats

Most common type is Black box type that describes "what" should the system do, not "how". They do not describe the internal procedures of the system; rather define responsibility to represent the detailed functions.



| Black-box style | Not |
|---|---|
| The system records the sale. | The system writes the sale to a database. ...or (even worse): The system generates a SQL INSERT statement for the sale... |

Use cases can be written in two-column format also - separating the actions and the responsibilities in two different columns.



**Use Case UC1: Process Sale**

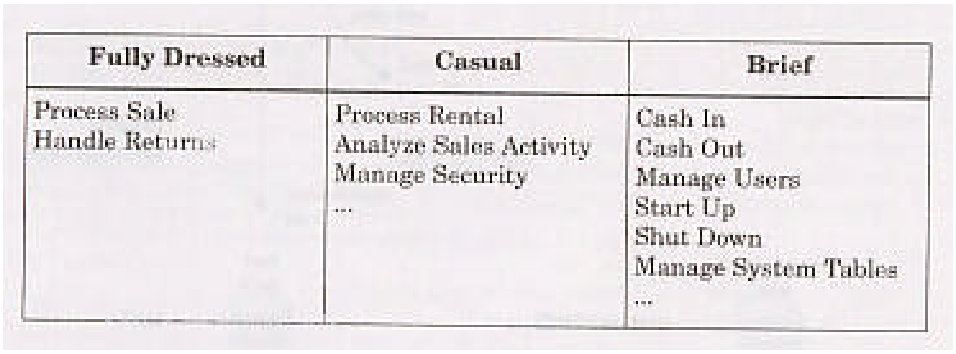
**Primary Actor:** ...
... as before ...

**Main Success Scenario:**

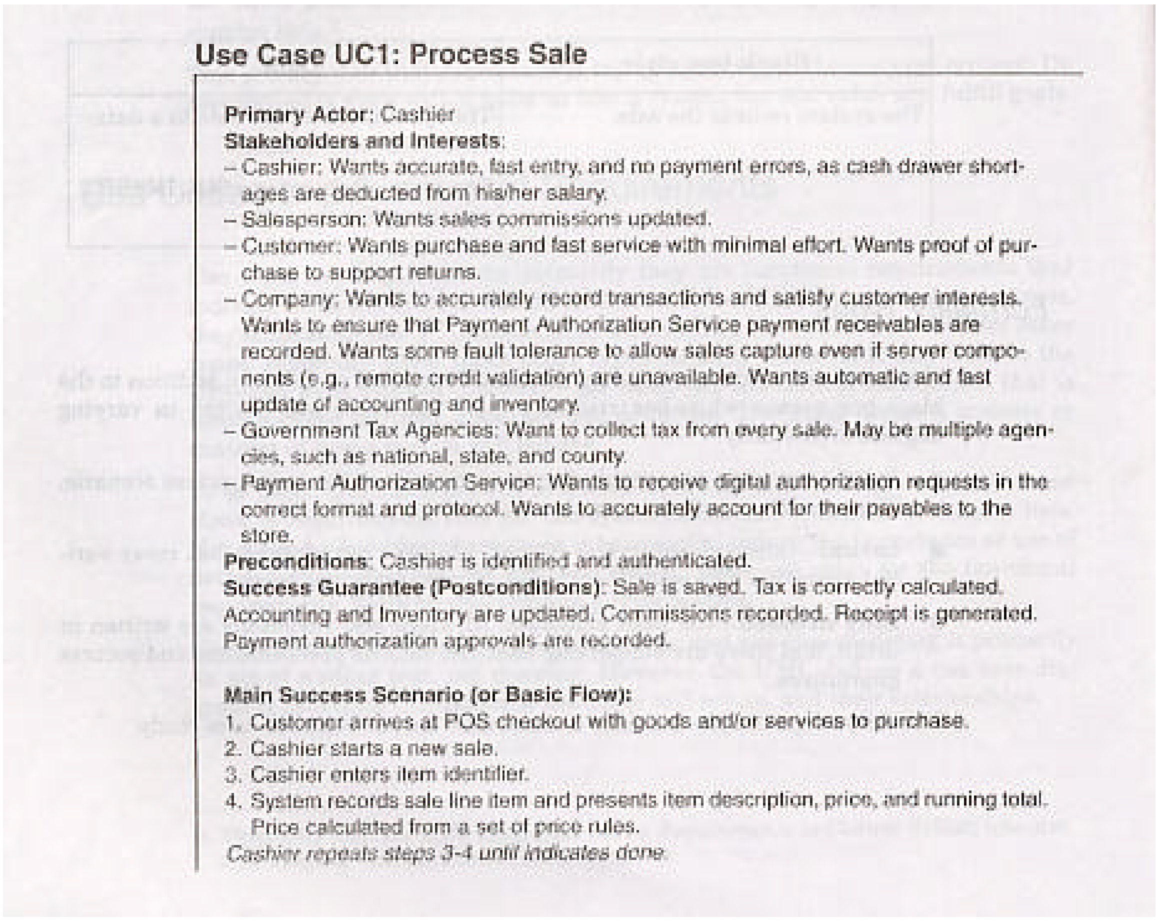
| Actor Action (or Intention) | System Responsibility |
|---|---|
| 1. Customer arrives at a POS checkout with goods and/or services to purchase. | |
| 2. Cashier starts a new sale. | |
| 3. Cashier enters item identifier. | 4. Records each sale line item and presents item description and running total. |
| Cashier repeats steps 3-4 until indicates done. | 5. System presents total with taxes calculated. |
| 6. Cashier tells Customer the total, and asks for payment. | |
| 7. Customer pays. | 8. Handles payment. |

Depending upon the detail of the scenario, there are commonly three formats of use cases in practice: -

- Brief
- Casual
- Fully dressed.



| Fully Dressed | Casual | Brief |
|---|---|---|
| Process Sale | Process Rental | Cash In |
| Handle Returns | Analyze Sales Activity | Cash Out |
| | Manage Security | Manage Users |
| | ... | Start Up |
| | | Shut Down |
| | | Manage System Tables |
| | | ... |



**Use Case UC1: Process Sale**

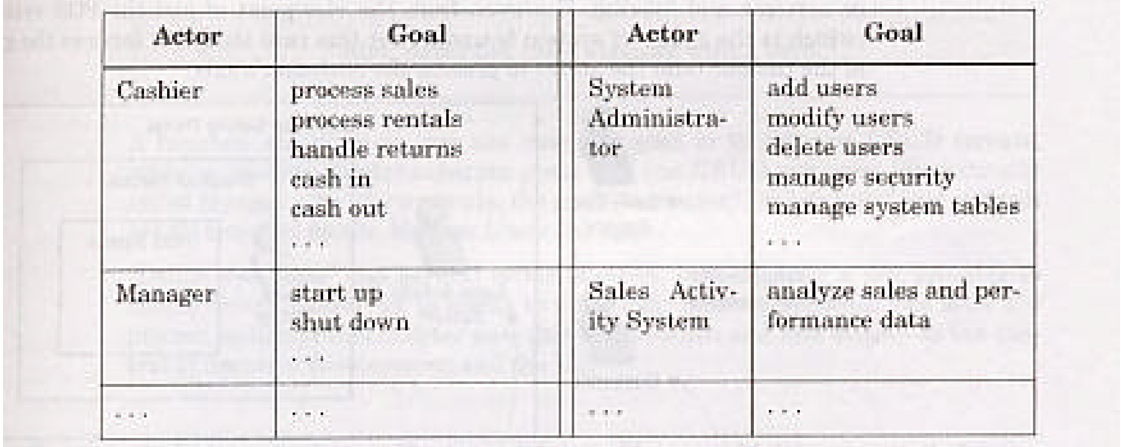
**Primary Actor:** Cashier
**Stakeholders and Interests:**
– Cashier: Wants accurate, fast entry, and no payment errors, as cash drawer shortages are deducted from his/her salary.
– Salesperson: Wants sales commissions updated.
– Customer: Wants purchase and fast service with minimal effort. Wants proof of purchase to support returns.
– Company: Wants to accurately record transactions and satisfy customer interests. Wants to ensure that Payment Authorization Service payment receivables are recorded. Wants some fault tolerance to allow sales capture even if server components (e.g., remote credit validation) are unavailable. Wants automatic and fast update of accounting and inventory.
– Government Tax Agencies: Want to collect tax from every sale. May be multiple agencies, such as national, state, and county.
– Payment Authorization Service: Wants to receive digital authorization requests in the correct format and protocol. Wants to accurately account for their payables to the store.
**Preconditions:** Cashier is identified and authenticated.
**Success Guarantee (Postconditions):** Sale is saved. Tax is correctly calculated. Accounting and Inventory are updated. Commissions recorded. Receipt is generated. Payment authorization approvals are recorded.

**Main Success Scenario (or Basic Flow):**
1. Customer arrives at POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules.

*Cashier repeats steps 3-4 until indicates done.*

**Sections in fully dressed use cases: -**
- Preface elements
- Stakeholders and interests list
- Preconditions and postconditions
- main success scenario and steps
- Extensions
- special requirements
- technology and data variation list

## 2.3. Actor, goals and use cases



| Actor | Goal | Actor | Goal |
|---|---|---|---|
| Cashier | process sales<br>process rentals<br>handle returns<br>cash in<br>cash out<br>. . . | System Administrator | add users<br>modify users<br>delete users<br>manage security<br>manage system tables<br>. . . |
| Manager | start up<br>shut down<br>. . . | Sales Activity System | analyze sales and performance data |
| . . . | . . . | . . . | . . . |

### 2.3.1. Actor

- Primary actors
- Supporting actors
- Offstage actors

### 2.3.2. Goals

- Use cases for an application are discovered from different level of tasks called goals. Therefore there may be revealed sub-function goals if we perform event analysis.
- Different levels of granularity do exist in a business application as guided by elementary business processes (EBP).
- Each goal can be represented by a use case e.g.
    - Negotiate a supplier contact
    - Handle returns
    - Log in

## 2.4. Procedure of finding a use case

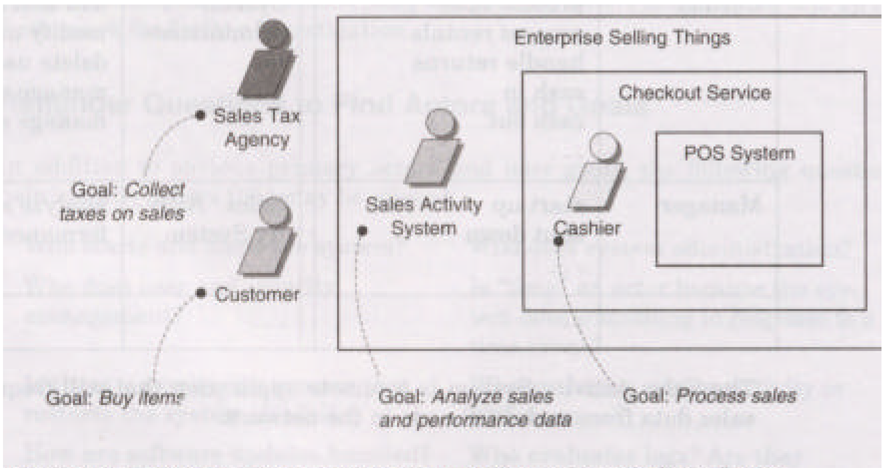
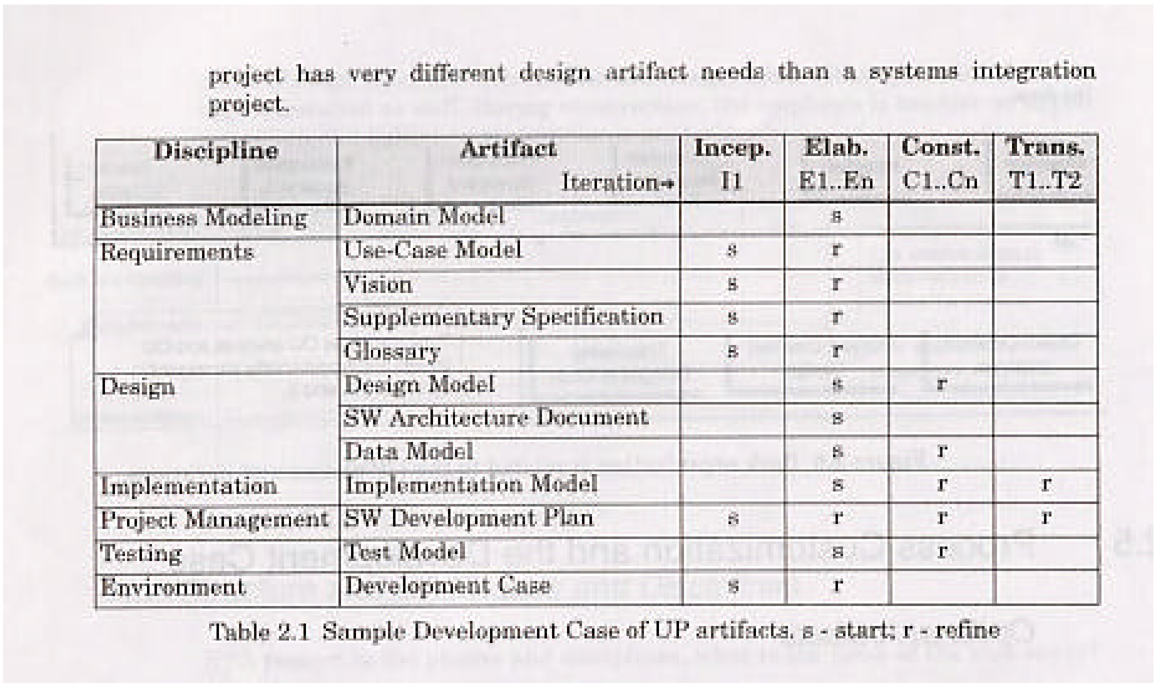


Figure 6.1 Primary actors and goals at different system boundaries.

- Choose the system boundary
- Identify the primary actors and their goals
    - Questions
    - actor goal list
- Define use cases

## 2.5. Use Cases in UP

- Use cases is an artifact of Requirements discipline of UP and has a vital role across the iterations.
- Use cases start in the inceptiom phase and get refined in the elaboration phase.
- Use cases realization drives the design discipline, which may be influenced by use case driven development.



project has very different design artifact needs than a systems integration project.

| Discipline | Artifact Iteration→ | Incep. I1 | Elab. E1..En | Const. C1..Cn | Trans. T1..T2 |
|---|---|---|---|---|---|
| Business Modeling | Domain Model | | s | | |
| Requirements | Use-Case Model | s | r | | |
| | Vision | s | r | | |
| | Supplementary Specification | s | r | | |
| | Glossary | s | r | | |
| Design | Design Model | | s | r | |
| | SW Architecture Document | | s | | |
| | Data Model | | s | r | |
| Implementation | Implementation Model | | s | r | r |
| Project Management | SW Development Plan | s | r | r | r |
| Testing | Test Model | | s | r | |
| Environment | Development Case | s | r | | |

Table 2.1 Sample Development Case of UP artifacts. s - start; r - refine

### 2.5.1. Use cases in Inception

- has some insight to the problem domain
- identify goals and stakeholders
- finding out tentative system boundaries
- prepare actor goal list
- use cases are written very briefly
- form a high-level picture of system functionality.
- Core, complex and more risky functions (10-20% of the system) are rewritten in fully dressed format in order to decide if the system is worth significant investigating further.

### 2.5.2. Use cases in Elaboration

- Requirements are refined as iterations go on building architecturally important, high valued and risky components.
- Subsets of use cases are also discussed and written down
- More discoveries of requirements is possible.
- Use cases are written in fully dressed format (about 80-90%)

### 2.5.3. Use cases in Construction

- Non-functional and minor use cases might come out during this phase.
- The degree of refinement in the elaorated use cases will be very less, but changes cannot be avoided.
- During the completion of the system, all use cases are finalized iteratively and adaptively.

### 2.5.4. Use Case diagram

- Use cases
- Actors
- relationships

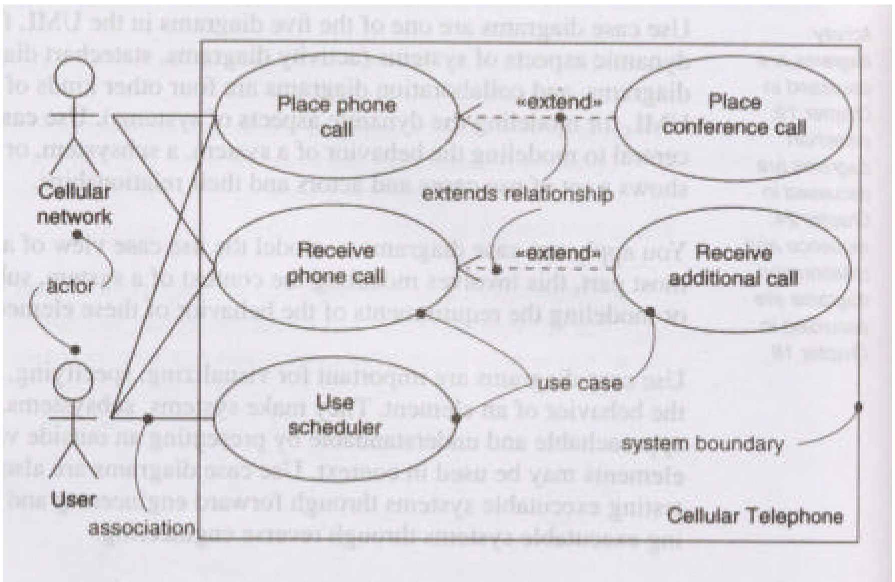


Figure 17-1: A Use Case Diagram

***Sources:***

- Craig Larman, Applying UML and Patterns
- Booch, Rumbaugh and Jacobson: The Unified Modelling Language User Guide