QA Testing Boot Camp

Chapter 3 – Software Requirements Analysis and Specification

Learning Outcomes

- Understand the Software Requirements Specification
- Understand the importance of the SRS
- Understand how the SRS impacts the testing phase

Software Requirements

Different Terms

- SRS – Software Requirements Specification
- BRD – Business Requirements Document

Building Software is Complex

A solution to the problem of scalability in Software Engineering

Becomes an agreement with the end user

SRS

software requirements specification (SRS) is a description of a software system to be developed, laying out functional and non-functional requirements

Why Use a SRS?

- Helps the developer to understand the objectives
- Allows for decomposition of the tasks to manageable “chunks”
- Reduces the overall costs associated with the development cycle – reduces refactoring and rework

Functional vs. Non-Functional

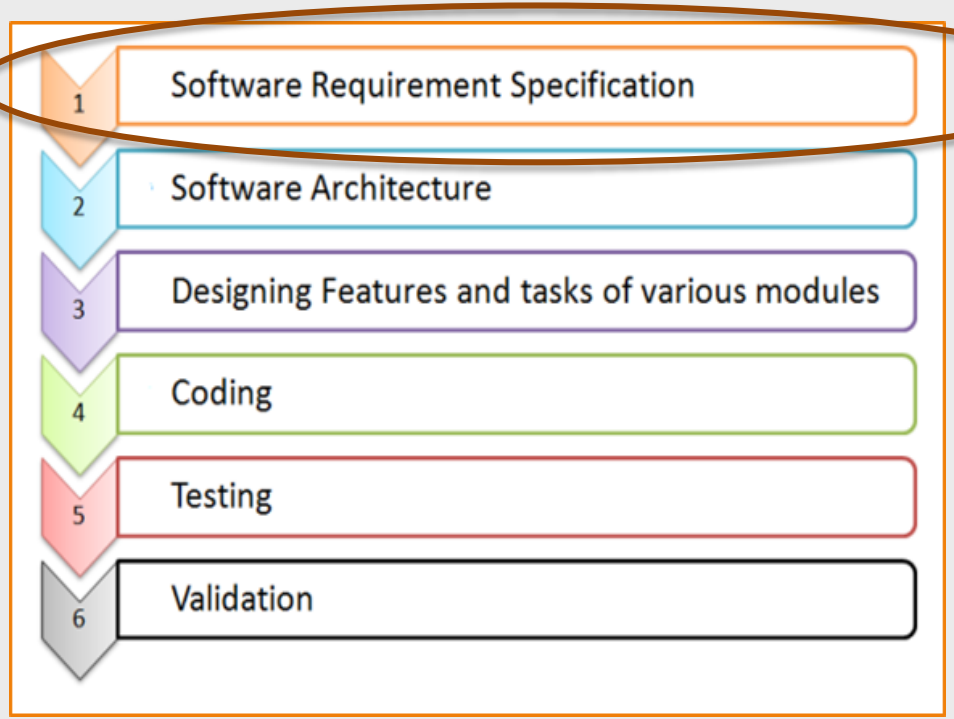
Functional Requirements

- Features that the user interacts with

Non-functional Requirements

- Features that work behind the scenes
- Includes: database, security, performance, etc.

Requirements is the 1st Step



It is essential to have a clear understanding of what is to be built

54% of all defects are found after unit testing

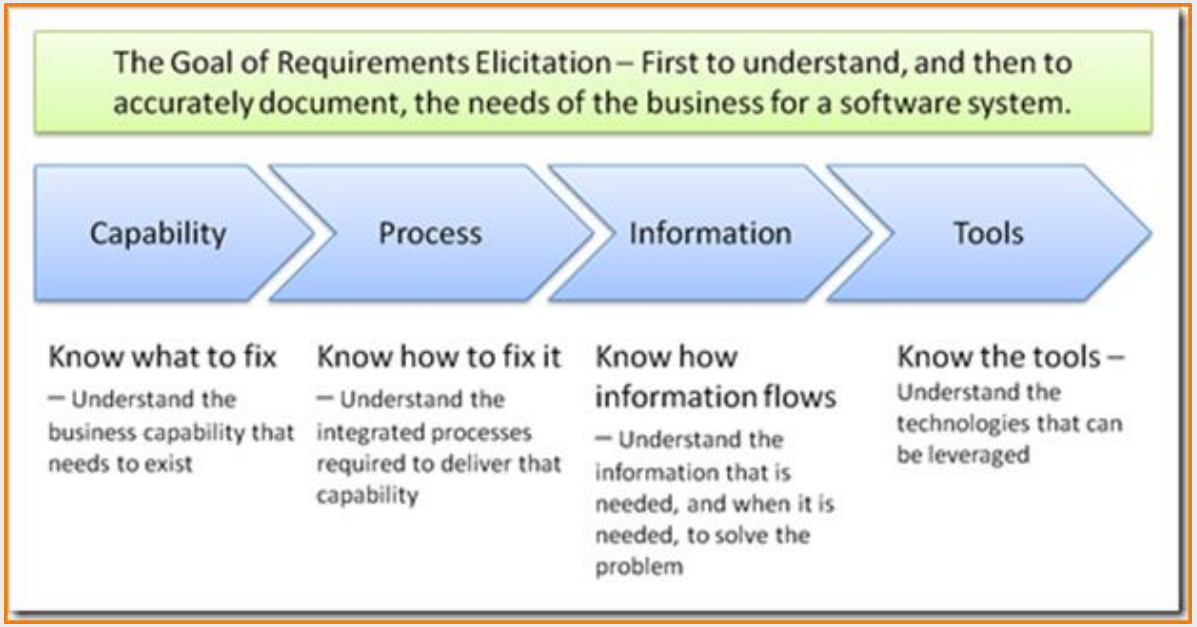
Of those found, 45% originated from a defect in the SRS.

25% of all defects found were introduced during requirements

GRAND CIRCUS
·DETROIT·

Requirements Elicitation

Goal: Understand the needs of the user



GRAND CIRCUS
·DETROIT·

Requirements Elicitation (2)

Basic Process – Analysis

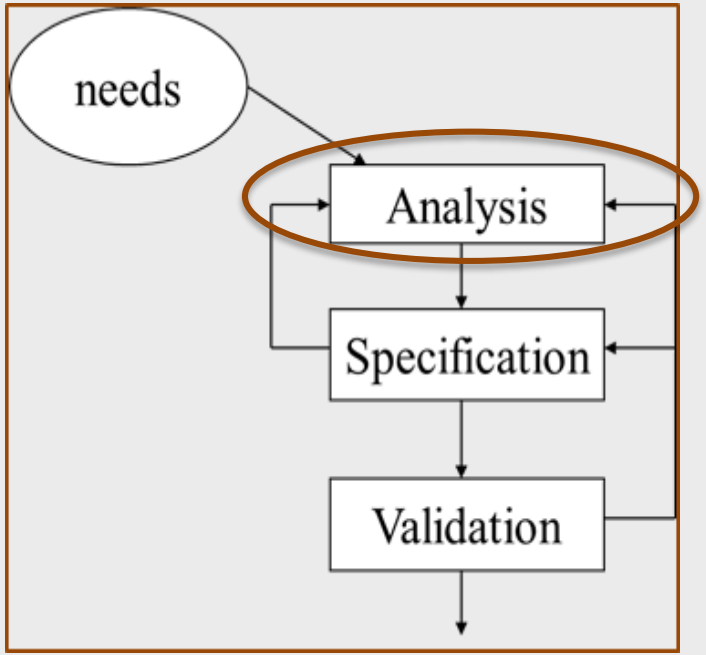
Problem Analysis is the hardest

It is an iterative and parallel process

Gain understanding of the user's needs

Usually conducted by a Business Analyst

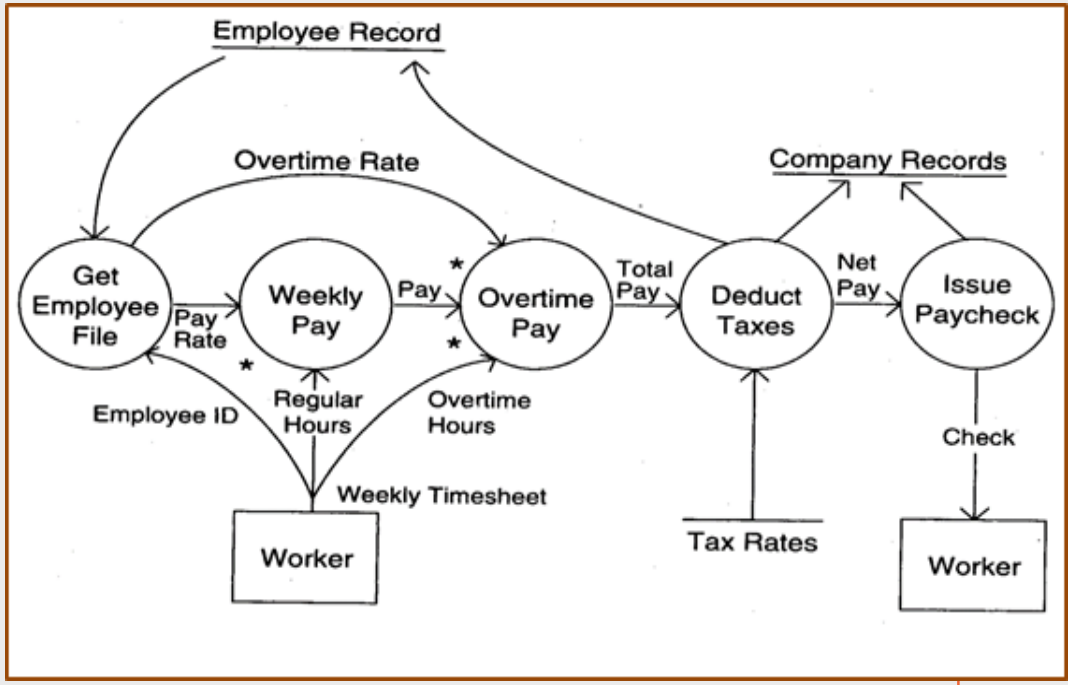
Divide and Conquer Approach used



Requirements Elicitation (3)

Data Flow Diagrams (DFD)

- Complex projects
- Focus Functions & Data Transforms
- Can become very large



Issue Pay Check

DFD – Steps to Create

1. Identify inputs/outputs, sources and sinks for the system.
2. Work consistently through inputs to outputs to identify a few high-level transforms to capture full transform.
3. Reverse direction if you get stuck.
4. When the high-level transforms are defined, decompose each to a more detailed transform
5. If you start thinking in loops or conditions (what ifs) stop and restart.
6. Label each circle and arrow.
7. Identify inputs/outputs of each transform.

Requirements Elicitation (4)

Data Dictionary

- Further decomposition of DFD
- Identifies data structures
- Uses regular mathematical expressions to express data

Data Dictionary

	A	B	C	D	E	
1	Variable / Field Name	Form Name	Section Header	Field Type	Field Label	Choices, Calculations, OR Slider Labels
2	participant_id	demographics		text	Participant ID	
3	enroll	demographics		text	Date subject signed consent	
4	fname	demographics		text	First Name	
5	lname	demographics		text	Last Name	
6	city	demographics		text	City	
7	state	demographics		text	State	
8	zip	demographics		text	Zipcode	
9	sex	demographics		dropdown	Gender	0, Female | 1, Male
10	given_birth	demographics		radio	Has the subject given birth before?	0, No | 1, Yes
11	num_children	demographics		text	How many times has the subject given birth?	
12	race	demographics		checkbox	Race	1, Caucasian | 2, African American | 3, Asian | 4, Other
13	race_other	demographics		text	Please describe:	
14	dob	demographics		text	Date of birth	
15	age	demographics		calc	Age	round(datediff([enroll],[dob],"y"),1)
16	height	demographics		text	Height (cm)	
17	weight	demographics		text	Weight (kilograms)	
18	bmi	demographics		calc	BMI	round([weight]*10000/([height]*[height]),1)
19	pcp	demographics		dropdown	Does patient have a primary care physician?	1, Yes | 2, No
20	upload	demographics		file	Upload record documents	

- Weekly_timesheet = employee_name + id + [regular_hrs + overtime_hrs]*
- Pay_rate = [hourly | daily | weekly] + dollar_amt
- Employee_name = last + first + middle
- Id = digit + digit + digit + digit

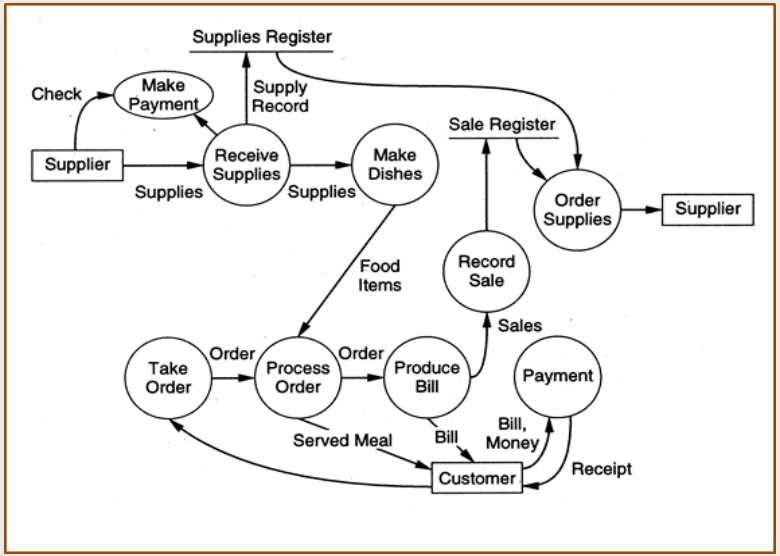
GRAND CIRCUS
·DETROIT·

Requirements Elicitation (5)

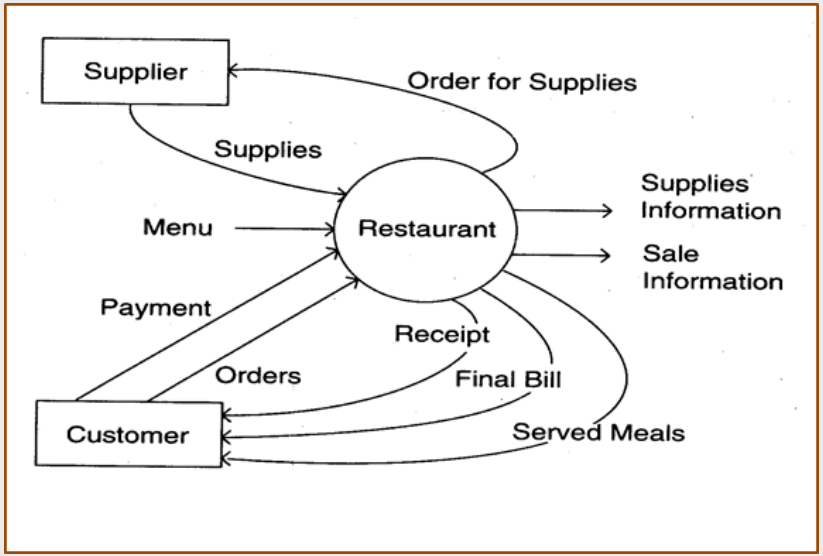
Context Diagram

- Full system viewed as a full system transform
- Basically a DFD with 1 transform - the entire system
- Used to understand the entire system
- Often used to compare proposed to current system

Context Diagram



Current System



Proposed System

GRAND CIRCUS
DETROIT

Issues in Problem Analysis

- Difficulty in obtaining the necessary information
- Limited interactions with end users resulting in a gap in understanding of how the user interacts with the system
- A large amount of information is often collected and therefore becomes overwhelming often resulting in a ‘watered-down’ understanding of the system
- There is often difficulty in ensuring the completeness of the analysis – all areas of the system have been examined
- There is often difficulty in ensuring the consistency (unambiguous) view of the analysis
- It is often difficult for the analysis to avoid creating the design the final system

Requirements Elicitation

Other Approaches

- Prototyping
  - Evolutionary (Functional)
  - Throw-Away (Non-Functional)
- Object Oriented Analysis (OOA)
  - Define objects (classes, attributes, methods)

Requirements Elicitation

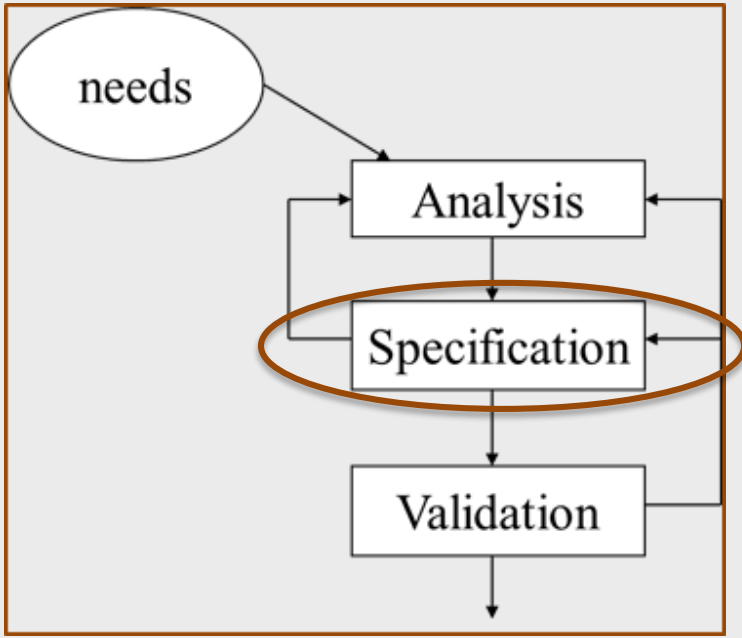
Basic Process – SRS

IEEE 830 Serves at the outline of the SRS

Creates a comprehensive document of the proposed system

DFD, Context and Data Dictionary are the input

Attributes of a good SRS:
Correct, complete, unambiguous, verifiable, consistent, traceable, & prioritized



Attributes of a Good SRS

- Correctness
- Completeness
- **Unambiguous**
- Verifiability
- Consistent
- Traceable

Software Requirements Specification

Contents:

- Functionality – The actions of the system. This includes what the user will see (functional) and what occurs behind the scenes (non-functional). It should be noted that the non-functional requirement of performance is also included.
- Design Constraints – This includes any confines that must be addressed such as legal, regulatory, business rules, as well as hardware.
- External Interfaces – These are the interactions that occur beyond the system boundaries of the proposed software. These interfaces could be within or outside of the organizational unit.

IEEE 830-SRS Outline

Customizable based on project and organization

Section 1

- Identifies the scope of the software deliverable.
- Presents a system overview.
- Defines the purpose.
- System overview

IEEE 830-SRS Outline

Section 2

- SRS identification
- Product perspective
- Product functions
- User characteristics
- Assumptions
- Constraints

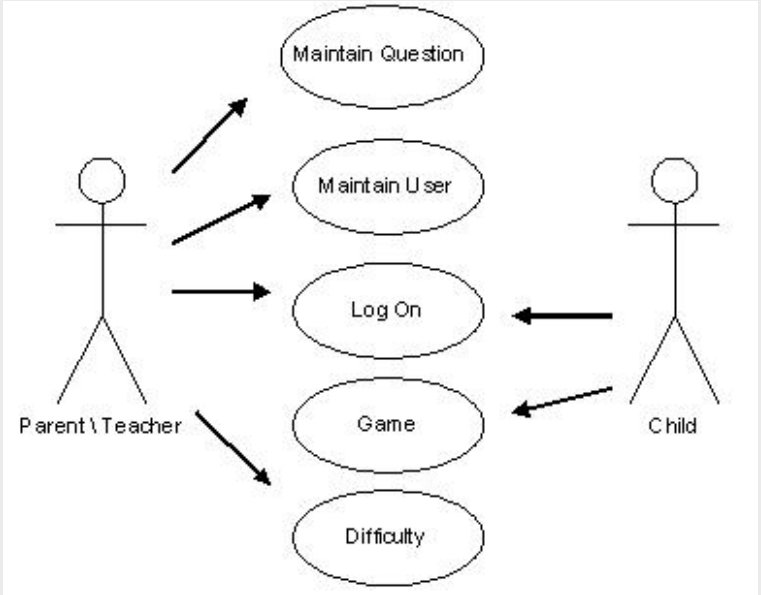
IEEE 830-SRS Outline

Section 3-functional requirements of system

- General System Requirements
- Performance Requirements
- Design Constraints
- External Interfaces

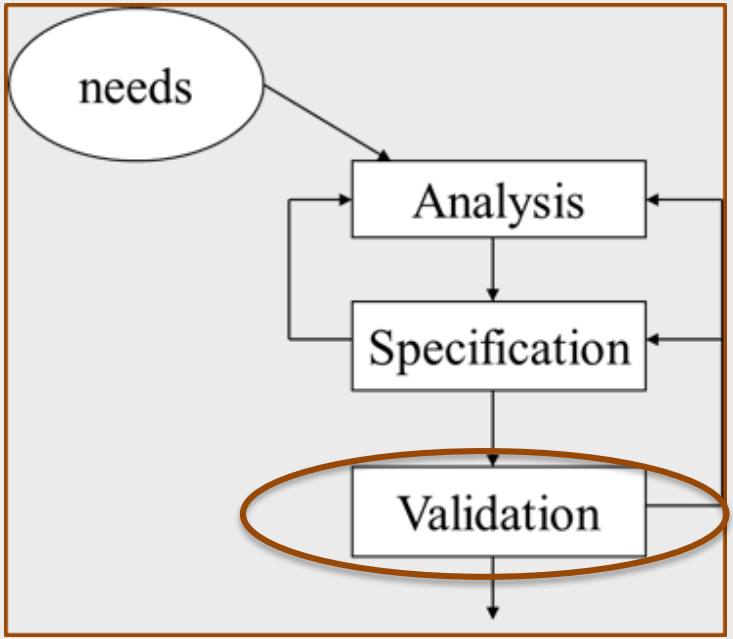
SRS – Use Cases

- Document that specifies the behavior (functionality) and interactions of the system
- Focused on functional specification
- Captures the user interactions and the behavior/response to that interaction
- Useful for interactive systems
- Actor – person or system that interacts with the proposed system
- Scenario – set of actions to achieve some goal



Requirements Elicitation

- Basic Process – Validation
- Avoids Misunderstandings
- Avoids/Prevents Errors
- SEI Study finds:
  - 30% Omission Errors
  - 10-30% Inconsistency Errors
  - 10-30% Incorrect Facts
  - 5-20% Ambiguity
- Should be a group review – author, client, end user, development team



Requirements Elicitation

Requirements Review-Preventative Task

- Can catch approximately 40-80% of requirements errors.
- SRS should have been reviewed by a group of people

Summary

- A good quality SRS is essential.
- 3 major sub-phases (analysis, specification and validation).
- Analysis is the most difficult
- The key quality properties are correct, complete, consistent, traceable and above all must be unambiguous.
- SRS must contain functionality, performance expectations, interfaces and design constraints.
- Use case is a good method to specify the functionality.
- Validation of the SRS occurs to validation techniques such as formal review.

Chapter Assessment

1. Verification of the requirement document achieves which of the following:

a. Verifies that we are building the right system.
b. Verifies that we are building the system right.
c. It is performed by an independent test team.
d. Ensures that it is what the user really wants.

2. What is meant by scope (as in Scope of the Software)?

3. What are software requirements?

4. How can you gather requirements?

5. What is SRS?

6. What are functional requirements?

7. What are non-functional requirements?

Chapter Excercises

3.1 Exercise – Draw the DFD for the Restaurant System

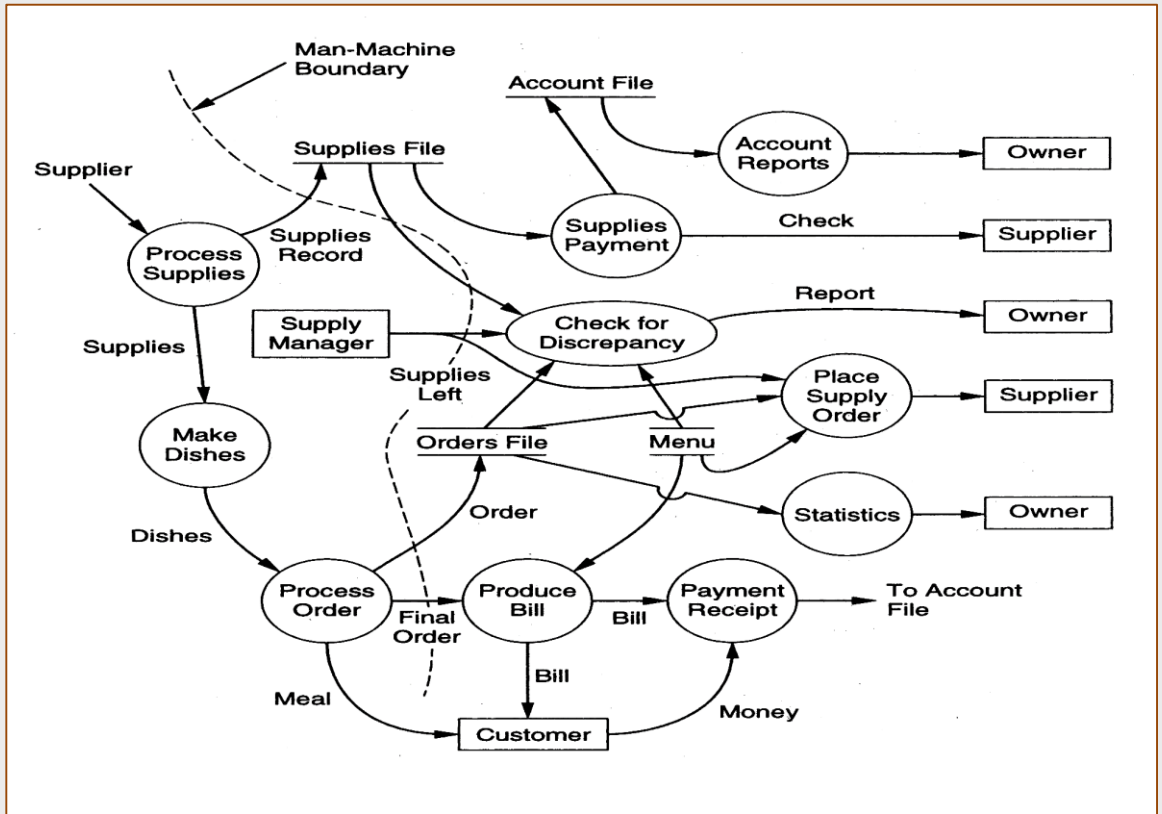
Goals of the new System:

- Automate the order processing – as much as possible
- Automate accounting
- Make supply ordering more accurate in order to reduce end of day waste and to facilitate avoidance of unfulfilled orders
- Help detect employee theft/usage of food products
- Produce sales metrics

Chapter Excercises

3.2 Exercise – Build Example 2 Use Case –Buy a Product

Chapter Exercises



QA Testing Boot Camp

Chapter 3 – Software Requirements Analysis and Specification