Testing Review 1-3

Day 1 Transition from Java to Testing

Re-introduction

Dr. Gregory Laidlaw

- Vice President ASI Systems
- Adjunct Professor UDM
- 25 Years SME Experience
- Security/Networking/System Integration
- Dissertation: “An Agile Methodology for implementing SOA in small and medium sized organizations”

GRAND CIRCUS
·DETROIT·

What we should know from Java

- Basics of coding-loops If/then basic syntax
- Structure of a program
- How to break down a problem/code
- Eclipse Environment

GRAND CIRCUS
·DETROIT·

Why Did We Learn Java first?

- Popular Language
- Basics of OO programing
- Speak the same language as programmers
- Career Options

Overview of Testing Topics

Overview of Software Development

- ~~The Software Life Cycle~~
- ~~Software Quality Assurance~~
- ~~Requirements Analysis~~
- ~~Project Management~~
- Risk Management
- Quality Assurance Management

Overview of Testing Topics

Quality Assurance(QA) Testing

- Fundamentals of Testing
- Continuous Verification and Validation
- Levels of Testing
- Testing Types
- Test Design Principles

Overview of Testing Topics

Quality Assurance(QA) Testing

- Black Box Testing
- White Box Testing
- Managing the Test Cycle
- Configuration Management, Risks and Incidents

Overview of Testing Topics

Automated Testing

- Why
- Types
- JUnit
- Selenium

Chapter 1: The Software Life Cycle

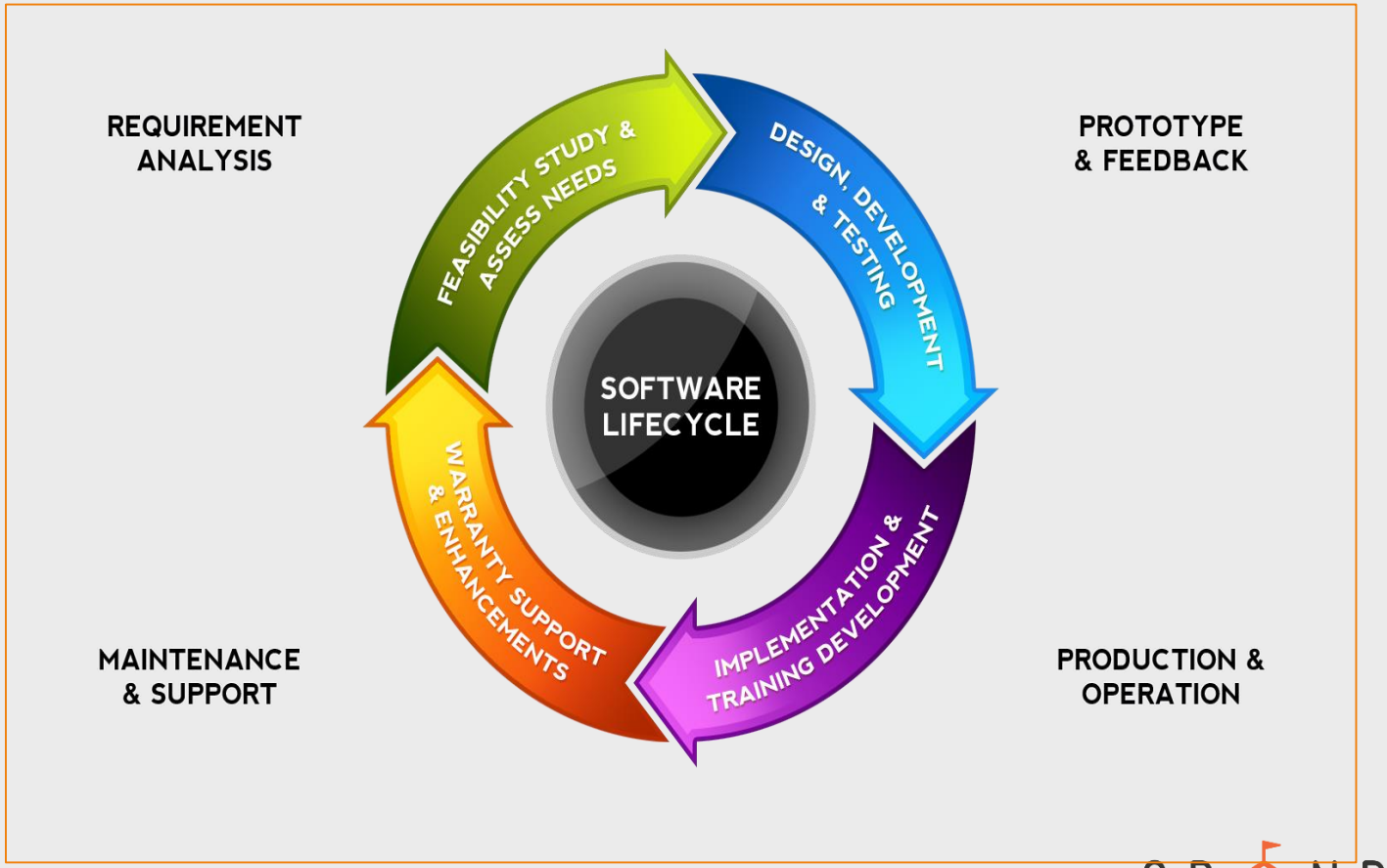
- Project Life Cycle
- Needs Assessment
- SDLC
- Development Models & Methodologies

GRAND CIRCUS
·DETROIT·

SDLC Framework

Framework

A framework, describes those activities that are performed at each phase during a software development project



Project Life Cycle – Needs Assessment

Types of Problems

- **Well-structured/Structured** -- constrained problems with convergent solutions, limited number of rules and principles within well-defined parameters.

- **Unstructured** -- multiple solutions, fewer parameters, and contain uncertainty about which concepts and rules.

GRAND CIRCUS
·DETROIT·

Project Life Cycle – Entry and Feasibility Study

Purpose of this phase is to obtain a commitment to change (Entry) and to evaluate whether cost effective solutions are available to address the problem or opportunity that has been identified.(Feasibility)

SDLC

There are following six phases in every Software development life cycle model:

1. **Requirement gathering and analysis**
2. **Design**
3. **Implementation or coding**
4. **Testing**
5. **Deployment**
6. **Maintenance**

1. Requirement Analysis & Design

- Business Requirements are gathered
  - Meetings with Stakeholders
    - Managers
    - Users
    - Designers/Programmers/Analysts
  - Output: Requirements Specification
    - Basis for rest of project

1. Requirement Analysis & Design

1.1 Needs Assessment

1.2 Entry and feasibility studies

1.3 Analysis of the existing system

- Though knowledge and understanding
- Consider organizational culture

2. Design Overview

- Software design from Specifications
- Hardware and system requirements
- System Architecture
- Formulation of strategic requirements
- Organizational and Job Design
- Output: System Design Specifications

3. Implement/Code

- Work divided into modules/units
- Actual coding is started
- Longest phase

4. Testing

- Requirement Testing
- Unit Testing
- Integration Testing
- System Testing

5. Deployment

- Testing phase passed
- System is delivered/deployed
  - Beta testing
  - Parallel testing

6. Maintenance

- After Deployment
  - Discovered problems
  - Enhancements/Changes

Development Models

- Classic Waterfall
- V & V Model (Verification & Validation)
- Prototyping
- Agile (Methodology)
- Iterative: OOD, RAD, Spiral

Chapter Assessment

1. List the advantages and disadvantages of the V&V Model.

2. When would you choose to use the RAD Model?

3. Explain in detail what benefits are achieved with the Prototyping Model.

4. What the differences between the Spiral Model and the Water Fall Model?

5. What is meant by the cost of change when considering the Traditional Methods vs. the Agile Methods?

Chapter 2: Software Quality Assurance

- Define Quality
- What is SQA
- Standards to Reference
- CMMi

Software Quality

- Software Quality – Bug/Defect Free
- ISO 8402-1986 standard defines quality as “the totality of features and characteristics of a product or service that bears its ability to satisfy stated or implied needs.”

Quality – Customer View

- Good design – looks and style
- Good functionality – it does the job well
- Reliable – acceptable level of breakdowns or failure
- Consistency
- Durable – lasts as long as it should
- Good after sales service
- Value for money

Software Quality Assurance

An umbrella activity that is applied throughout the software lifecycle

- Quality management process
- Effective software engineering principles
- Formal technical reviews
- Multilayered testing strategy
- Control of software documentation and changes to it
- Procedures to assure compliance with software development standards
- Measurement and reporting techniques

ITIL – Information Technology Infrastructure Library

- A set of **best practices** for IT service management that focuses on aligning IT services with the needs of the business
- Defines organizational structure and skill requirements needed for an IT organization
- Contains operational management procedures and practices
- Currently 5 volumes in the library – Service Strategy, Service Design, Service Transition, Service Operation, Continual Service Improvement

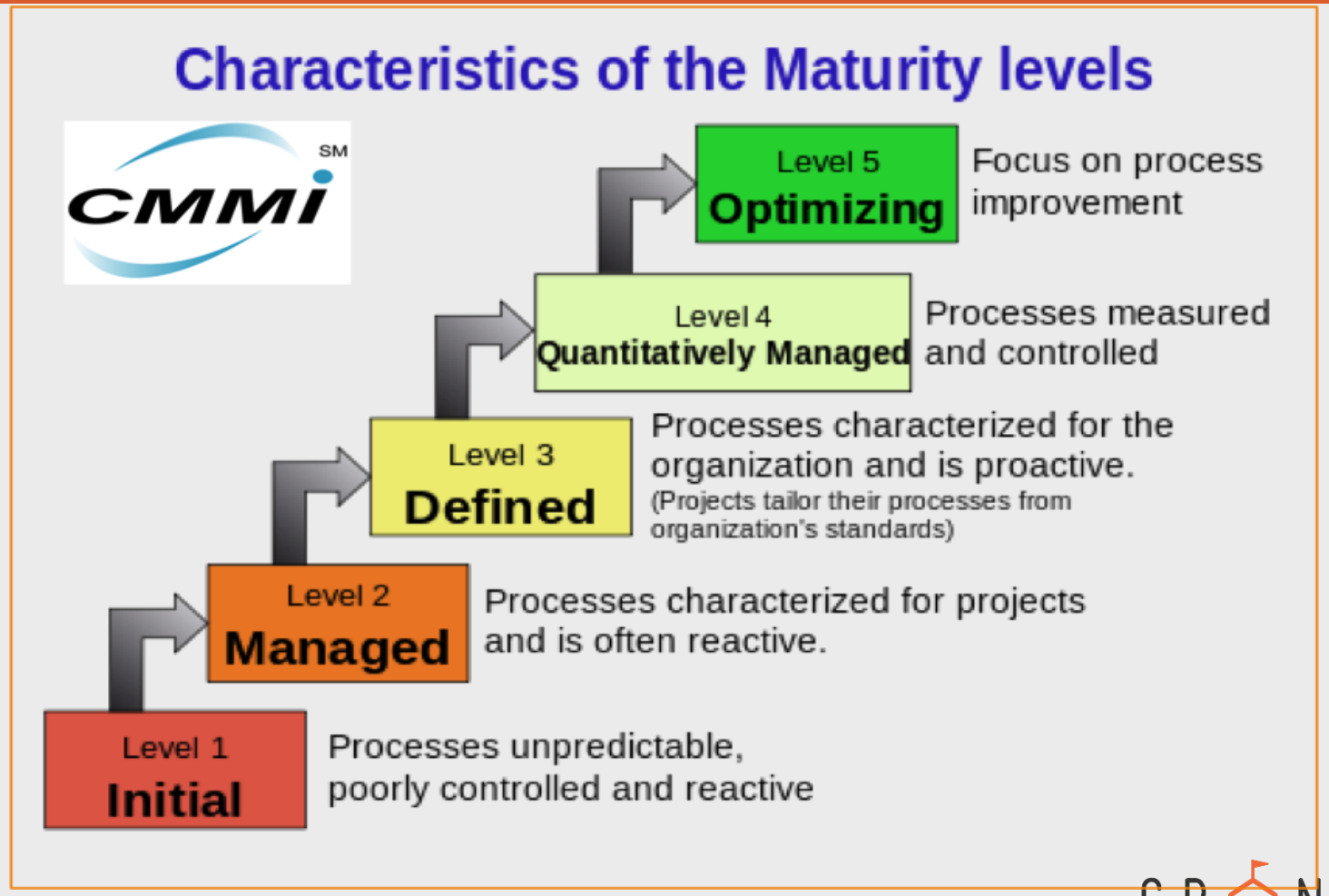
ISO 12207 : System & Software Engineering Standard – Software Lifecycle Processes

- Defines the tasks for developing and maintaining software
- Developed for buyers, developers, maintainers, operators, managers and technicians involved with software development
- Contains 23 processes, 95 activities, 325 tasks and 224 outcomes

CMMi

Maturity Levels

- Initial
- Managed
- Defined
- Quantitatively Managed
- Optimizing



GRAND CIRCUS DETROIT

Chapter 3: Requirements

- **Software Requirements Specification**
- **IEEE 830**

GRAND CIRCUS
·DETROIT·

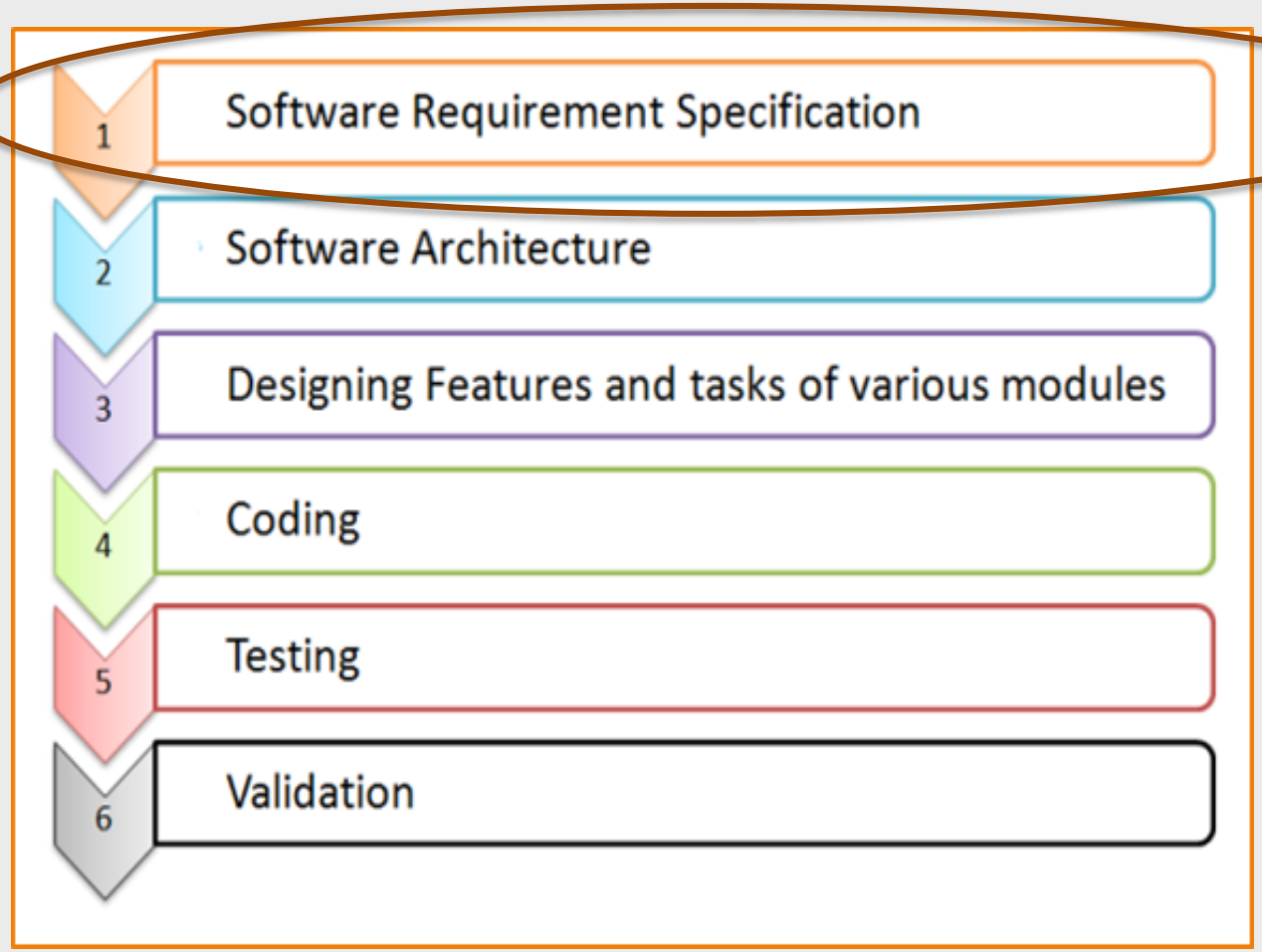
SRS

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

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

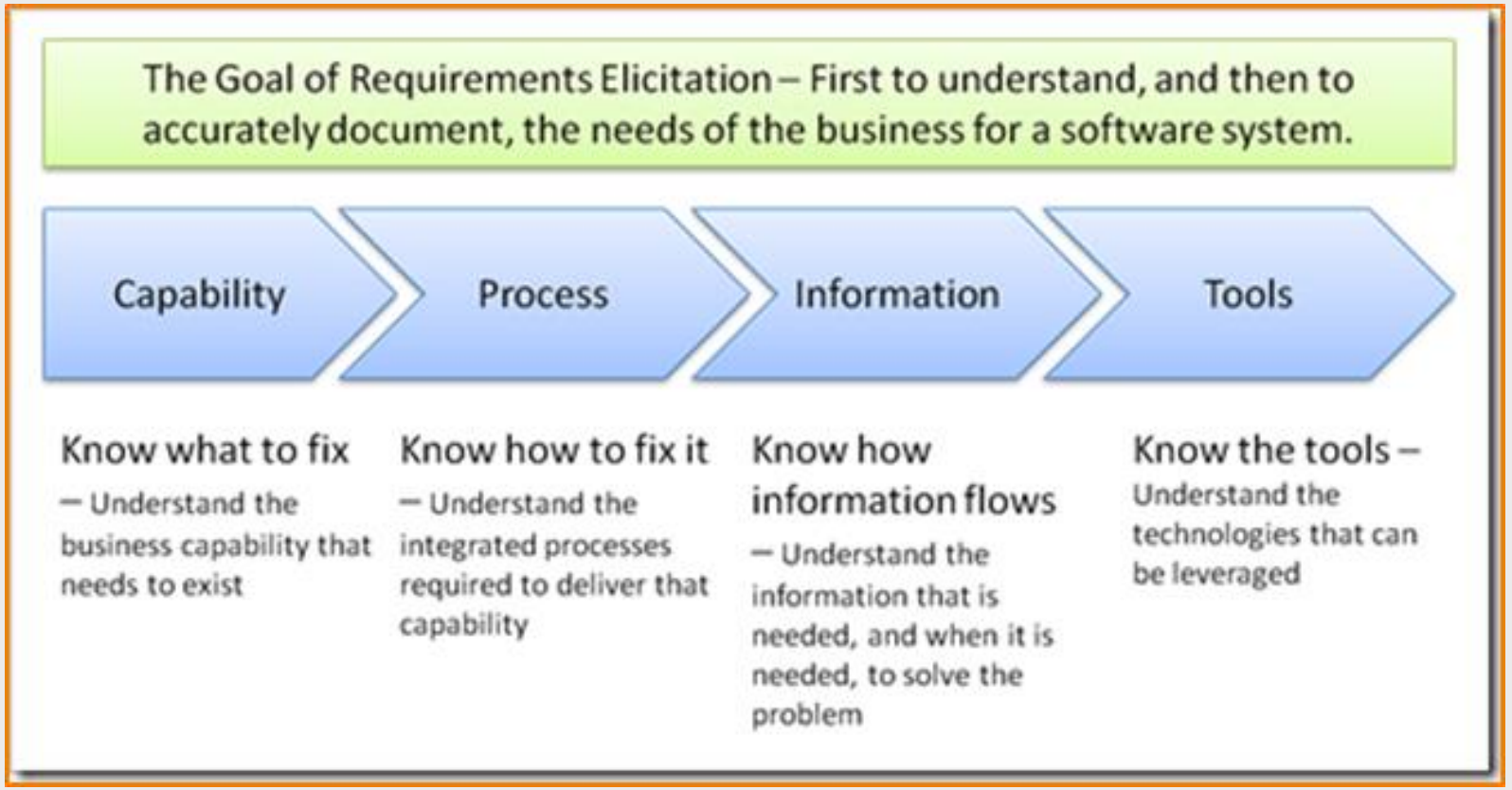
Of those found, 45% originated from a defect in the SRS.

25% of all defects found were introduced during requirements

GRAND CIRCUS
·DETROIT·

Requirements Elicitation

Goal: Understand the needs of the user



Requirements Elicitation (2)

Basic Process – Analysis

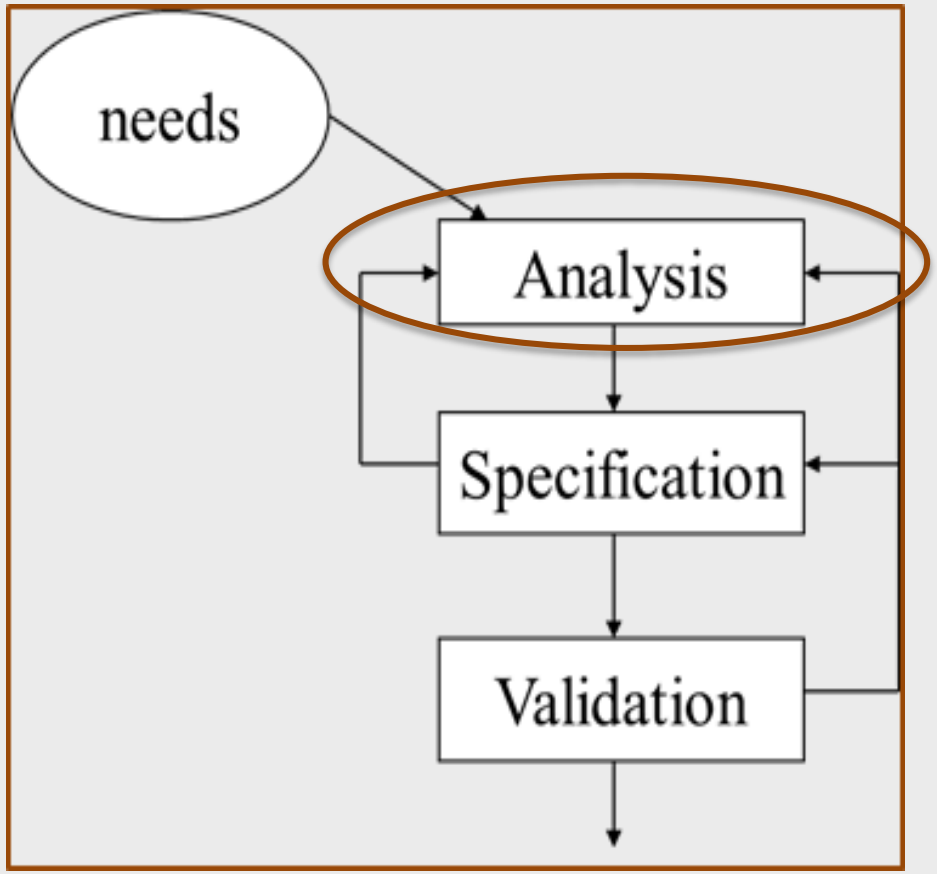
Problem Analysis is the hardest

It is an iterative and parallel process

Gain understanding of the user's needs

Usually conducted by a Business Analyst

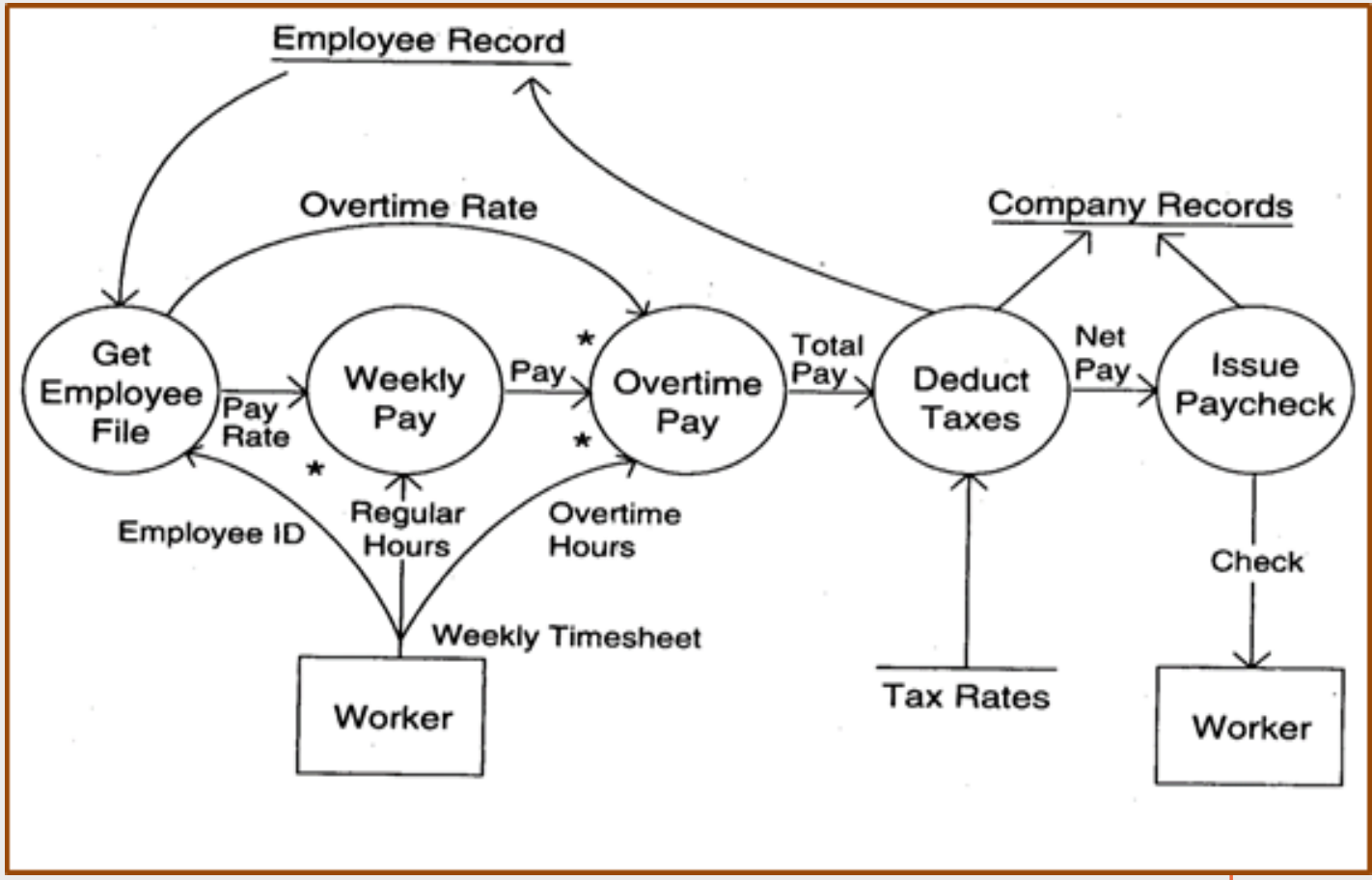
Divide and Conquer Approach used



Requirements Elicitation (3)

Data Flow Diagrams (DFD)

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



Issue Pay Check

GRAND CIRCUS DETROIT

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

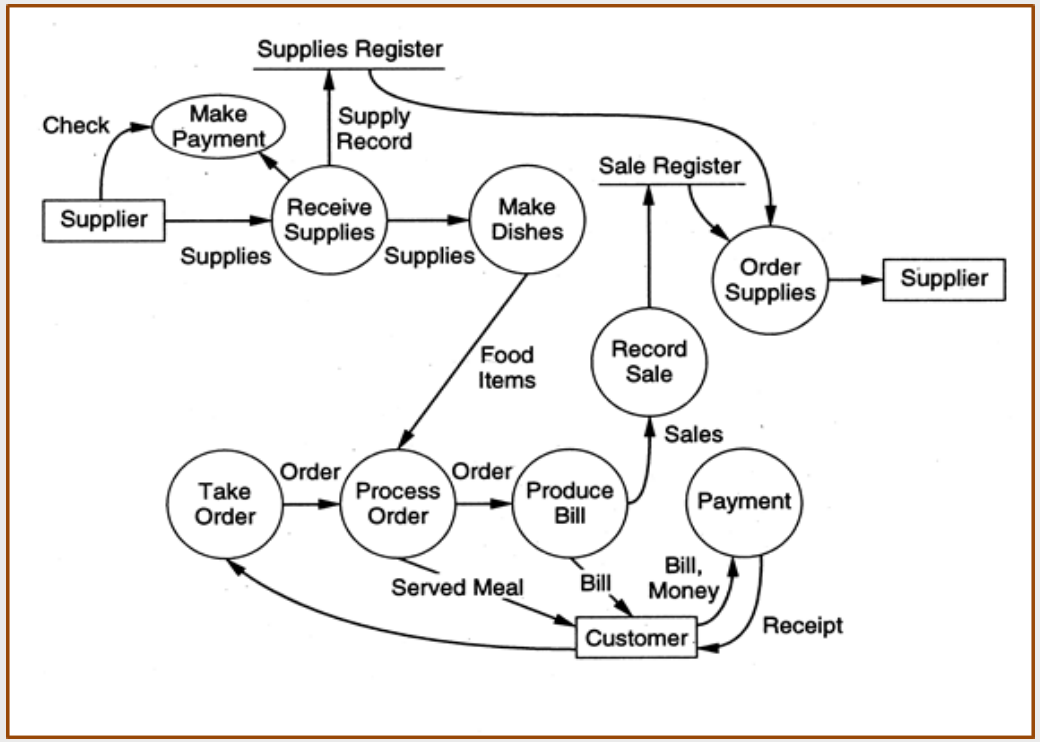
GRAND CIRCUS
·DETROIT·

Requirements Elicitation (5)

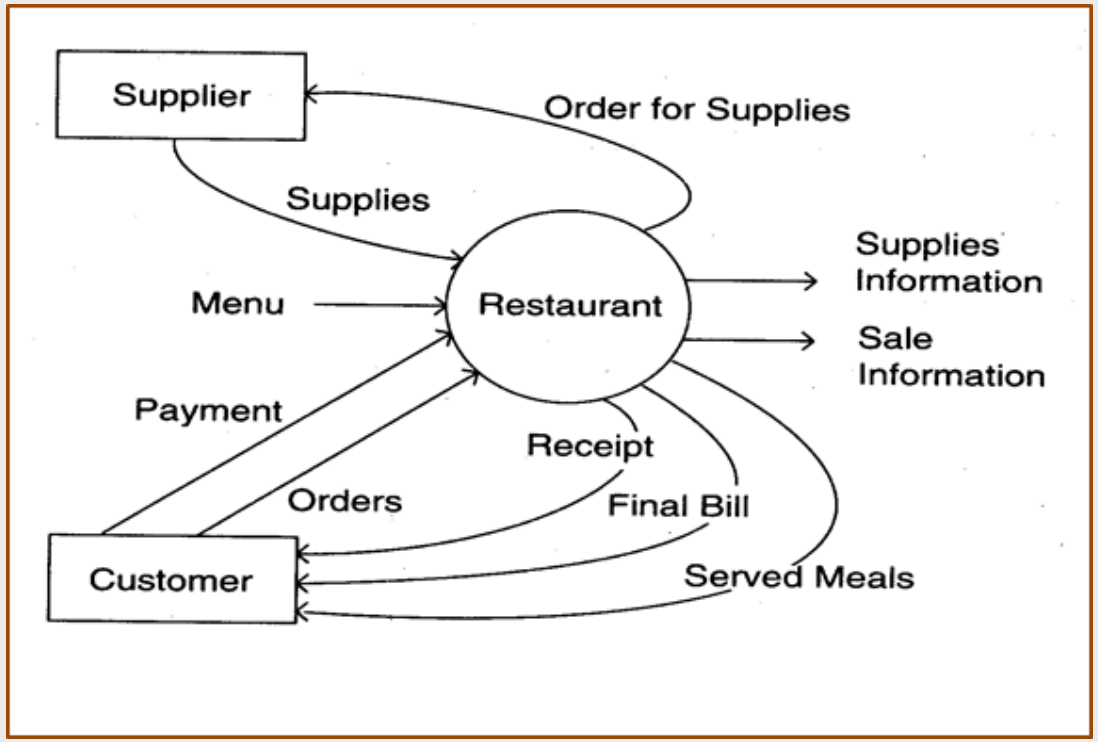
Context Diagram

- 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·

Requirements Elicitation

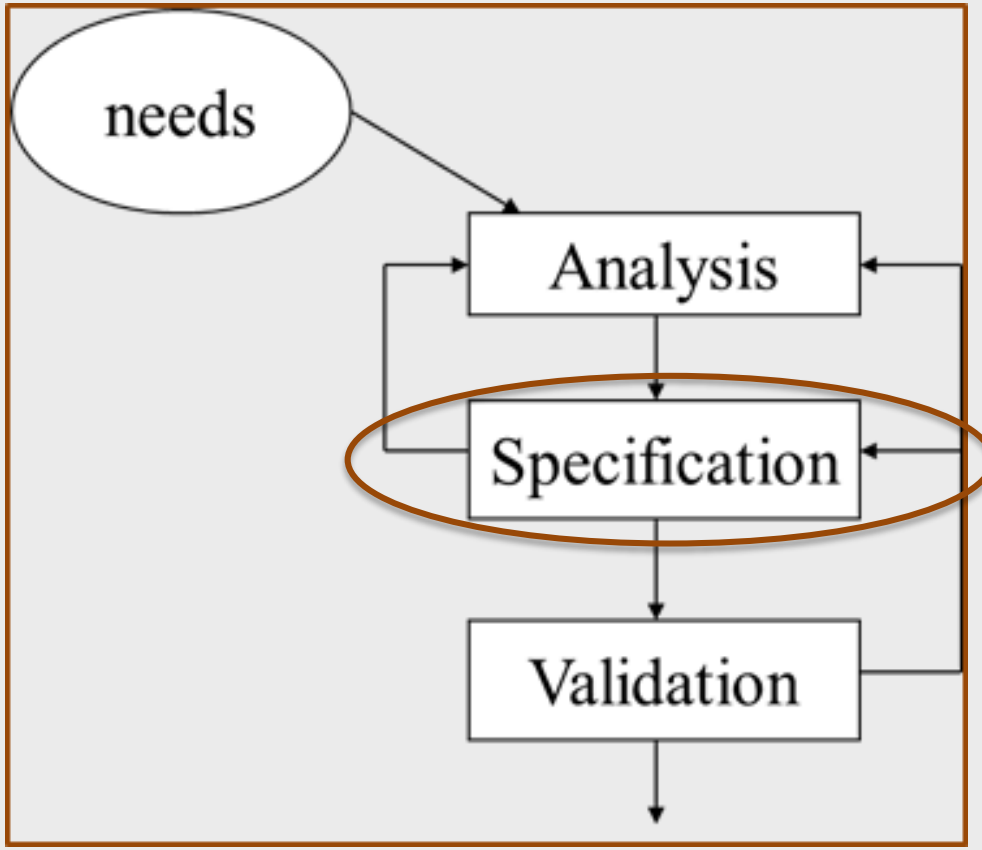
Basic Process – SRS

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

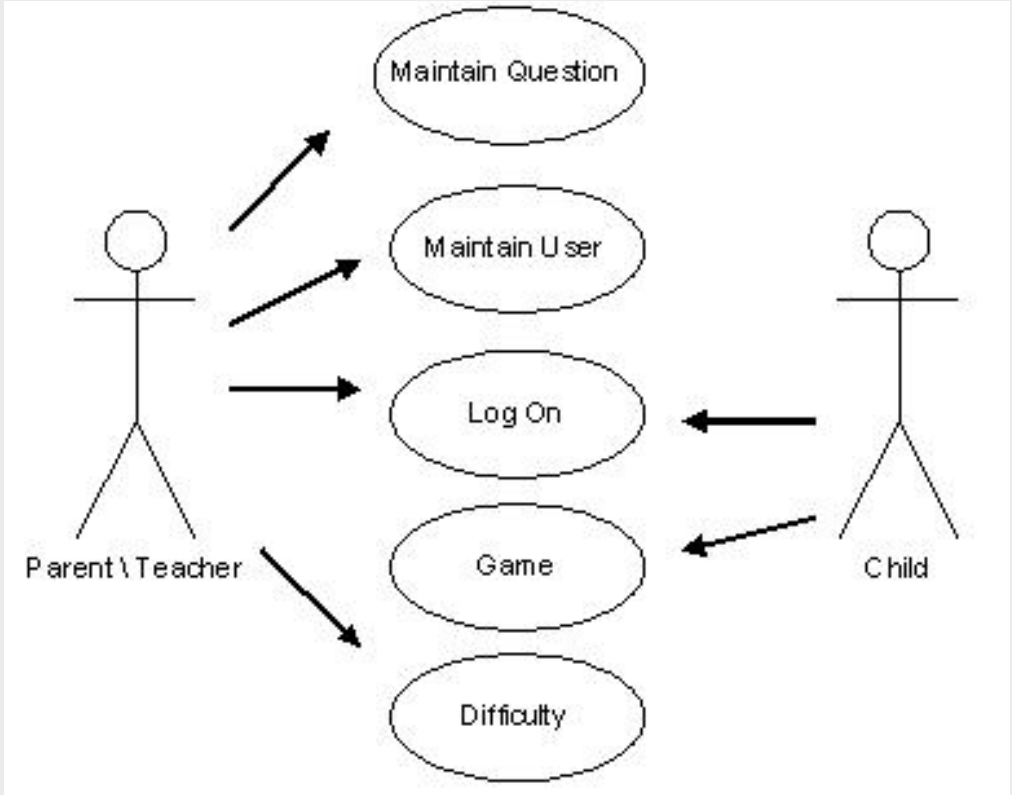
Software Requirements Specification

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.

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



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 How can you gather requirements?

5. What is SRS?

6. What are functional requirements? What are non-functional requirements?