Lecture-16

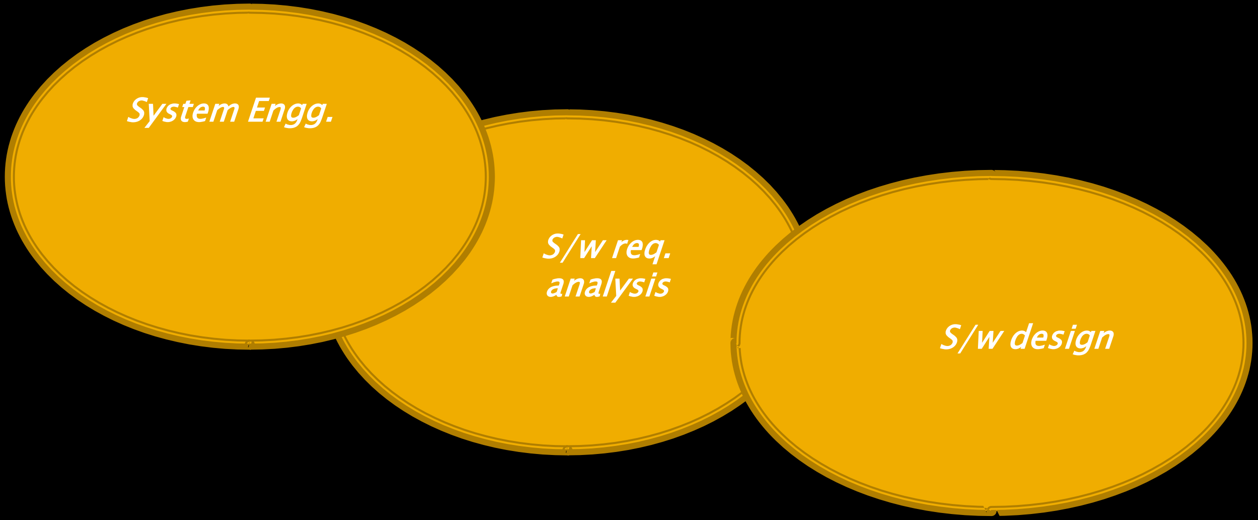
Requirements Analysis Concepts & Principles

(Cont….)

Dronacharya College of Engineering

Analysis Concepts and Principles

Requirement analysis is a software engineering task that bridges the gap between system engineering and software design.



Participated by both the customer and developer

Requirements Analysis Concepts and Principles

- *Requirements Analysis*
- *Communication Techniques*
    - *Initiating the Process*
    - *Facilitated Application Specification Techniques*
- *Analysis Principles*
    - *Information Domain*
    - *Modeling*
    - *Partitioning*
- *Software Prototyping*
    - *Selecting the Prototyping Approach*
    - *Prototyping Methods and Tools*
- *The Software Requirements Specification*
    - *Specification Principles*
    - *Representation*

3.Analysis Principles

Each analysis method has a unique point of view.

All analysis methods are related by a set of operational principles:

- *represent and understand the information domain*
- *define the functions that the software*
- *represent the behavior of the software*
- *use models to depict information, function, and behavior*
- *uncover the details in a layered fashion.*
- *move from essential information toward to details*

A set of guidelines for requirement engineering:

- *understand the problem before beginning to create the analysis model*
- *develop prototypes to help user to understand how human-machine Interactions*
- *record the origin of and the reasons for every requirement*
- *use multiple views of requirements*
- *prioritize requirements*
- *work to eliminate ambiguity*

A-The Information Domain

Software is built to process data, to transform data from one form to another.

The first operational analysis principle requires to exam the information domain.

Information domain contains three different views of the data and control:

- information content and relationship:
information content represent the individual data and control objects there are different relationships between data and objects

- information flow:
represents the manner in which data and control change as each moves through a system. Data and control moves between two transformations (functions)

- information structure:
represent the internal organization of various data and control items
- ***data tree structure***
- ***data table (n-dimension)***

B-Modeling

During software requirements analysis, we create models of the system to be built.

The models focus on:

"what the system must do, not how it does it."

The models usually have a graphic notation to represent:

- information, processing, system behavior, and other features

- ***Functional models***

  Software transforms information. Three generic functions:

  - input, processing, output

- ***Behavior models***

  Most software responds to events from the outside world

  A behavior model creates a representation of the states of the software and events that cause software to change state

Modeling

- Data model
  - shows relationships among system objects
- Functional model
  - description of the functions that enable the transformations of system objects
- Behavioral model
  - manner in which software responds to events from the outside world

Modeling

The models focus on:

"what the system must do, not how it does it."

Important roles of models:

- ❑ ***The model aids the analyst in understanding the information, function, and behavior of a system.***
- ❑ ***The model becomes the focal point for review in the aspects of completeness, consistency, and accuracy of the specification.***
- ❑ ***The model becomes the foundation for design, providing the designer with an essential representation of software.***

C-Partitioning

Process that results in the elaboration of data, function, or behavior.

- *Horizontal partitioning*
  - *breadth-first decomposition of the system function, behavior, or information, one level at a time.*

- *Vertical partitioning*
  - *depth-first elaboration of the system function, behavior, or information, one subsystem at a time.*

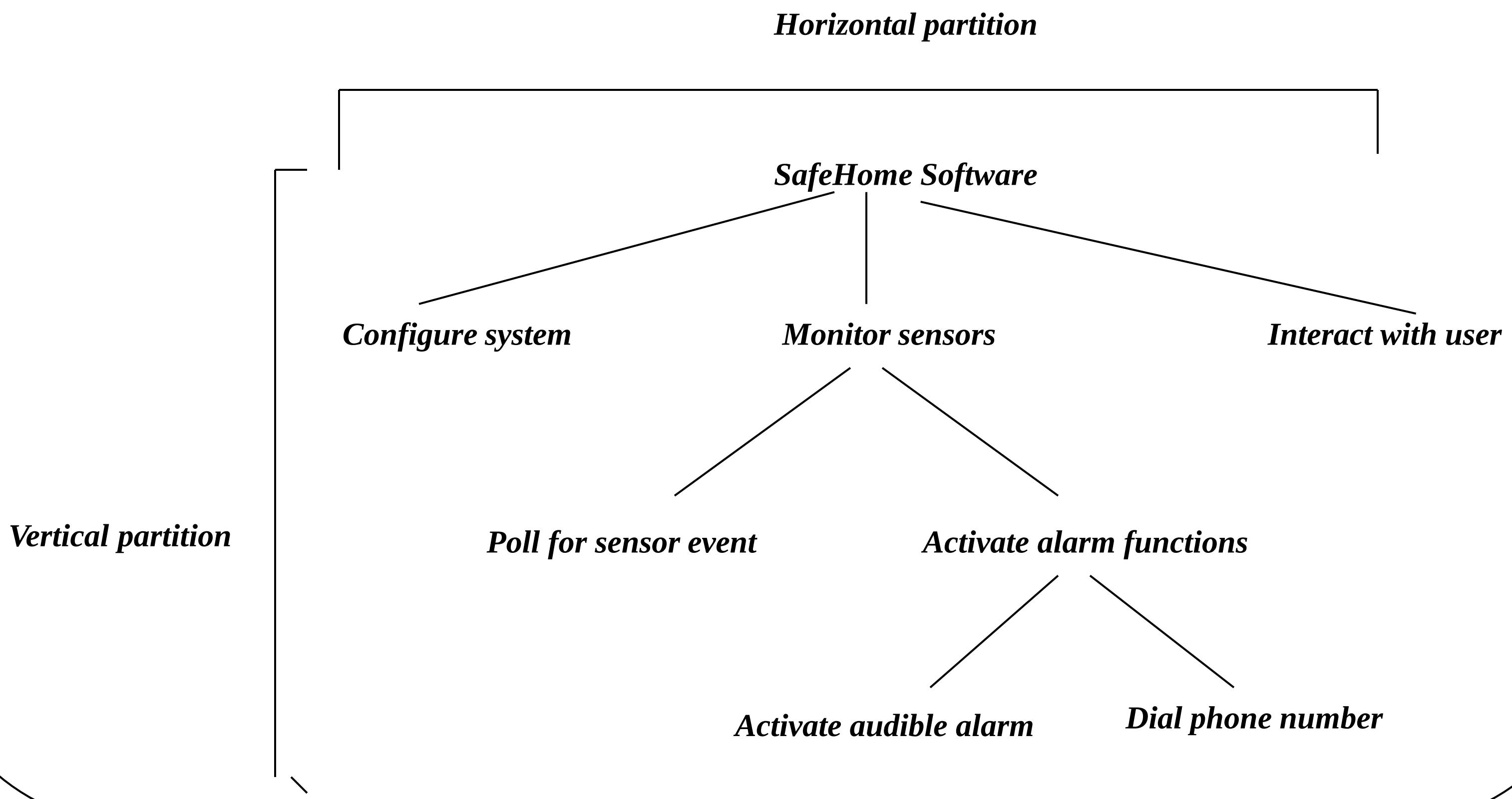
Partitioning

Partitioning decomposes a problem into its constituent parts.

Establish a hierarchical representation of information (or function):

- exposing increasing detail by moving vertically in the hierarchy

- decomposing the problem by moving horizontally in the hierarchy.



4.Software Prototyping

In some cases, it is possible to apply operational analysis principles and derive a model of software from which a design can be developed.

Selecting the prototyping approach:

1. Throwaway (Closed ended)

Here the prototype serves only to demonstrate requirements and is discarded after the desired knowledge is gained.

Since the prototype is ultimately discarded it need not be maintainable or use efficient algorithms.

2. Evolutionary (Open ended)

Once the prototype has been used and requisite knowledge has been gained it is eventually incorporated into the final system.

Must exhibit all quality attributes of the final product.

Software Prototyping

Prototyping Methods and Tools:

- ***Fourth Generation Techniques***

- ***Reusable Software Components***

- ***Formal Specification and Prototyping Environments***

1. *Fourth generation techniques :*

- ***4GLs reduce programming effort, the time it takes to develop software, and the cost of software development. Report generators (Oracle reprots, QUEST, GEM base)Screen generators (Oracle forms etc)Database Query Languages (SQL, Ingres etc)***

- ***These tools generate an executable code quickly and hence are ideal for rapid prototyping***

2. *Reusable software components :*

•Assemble rather than build the prototype by using a set of existing s/w components

3. Formal Specification languages:

•Replace natural language specification
•Eg. Set notation , algebraic notation.
•There are tools which convert these formal language specifications into executable code.

5.Software Specification

Specification principles:

- ***Separate functionality from implementation***
- ***Develop a model of the desired behavior of a system***
- ***Establish the context in which software operates***
- ***Define the environment in which the system operates***
- ***Create a cognitive model rather than a design or implementation model***
- ***Specification is an abstract model of a real system***
- ***Establish the content and structure of a specification (easy to be changed)***

Guidelines for representation:

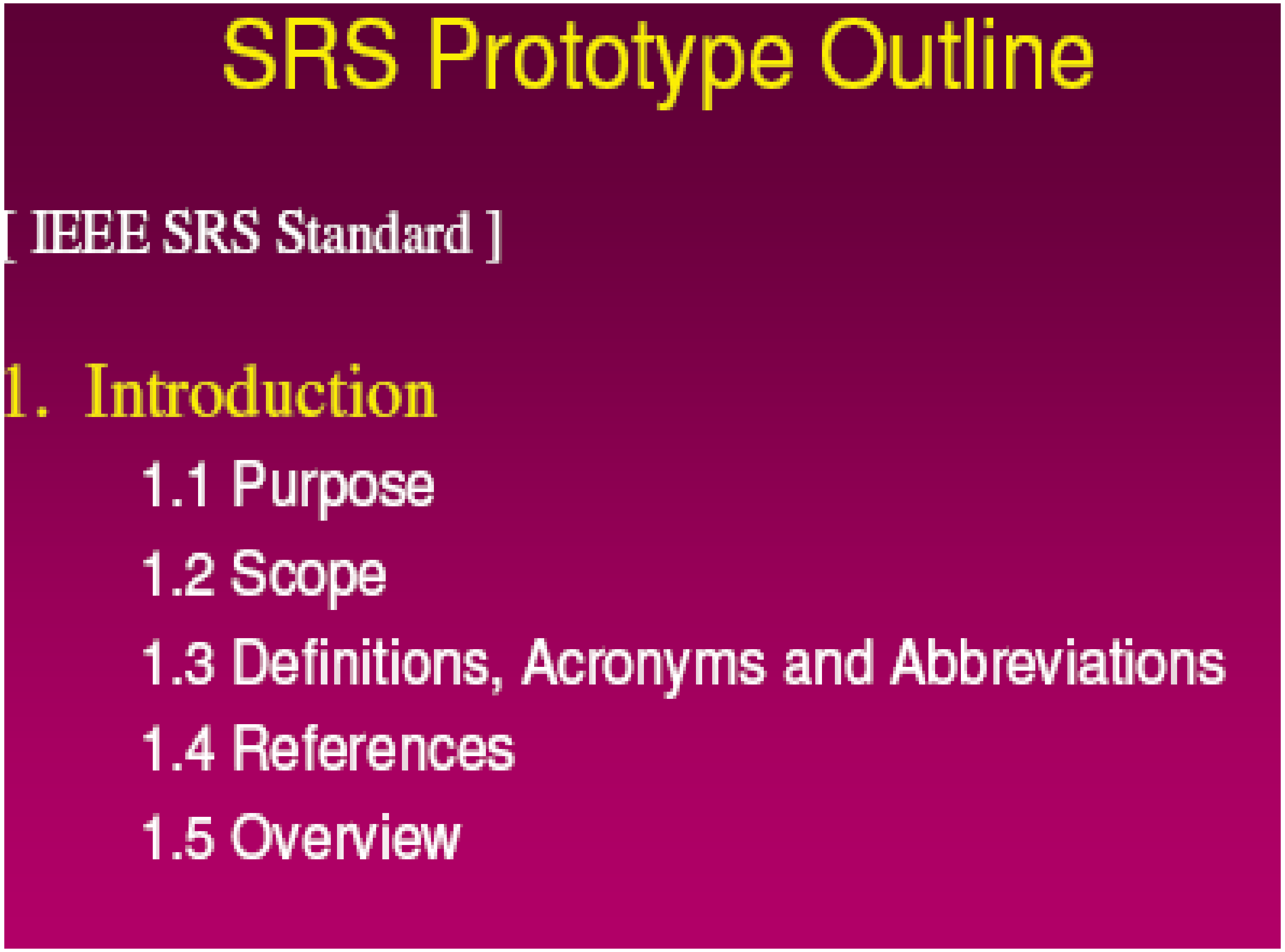
- ***Representation format and content should be relevant to the problem***
- ***Information contained within the specification should be nested***
- ***Diagrams and other notational forms should be restricted in number and consistent in use.***
- ***Representations should be revisable***

Software requirements specification standard:

IEEE (standard No. 830-1984) and U.S. Department of Defense

In many cases, a preliminary user's manual should be provided to presents the software as a black box.

SRS

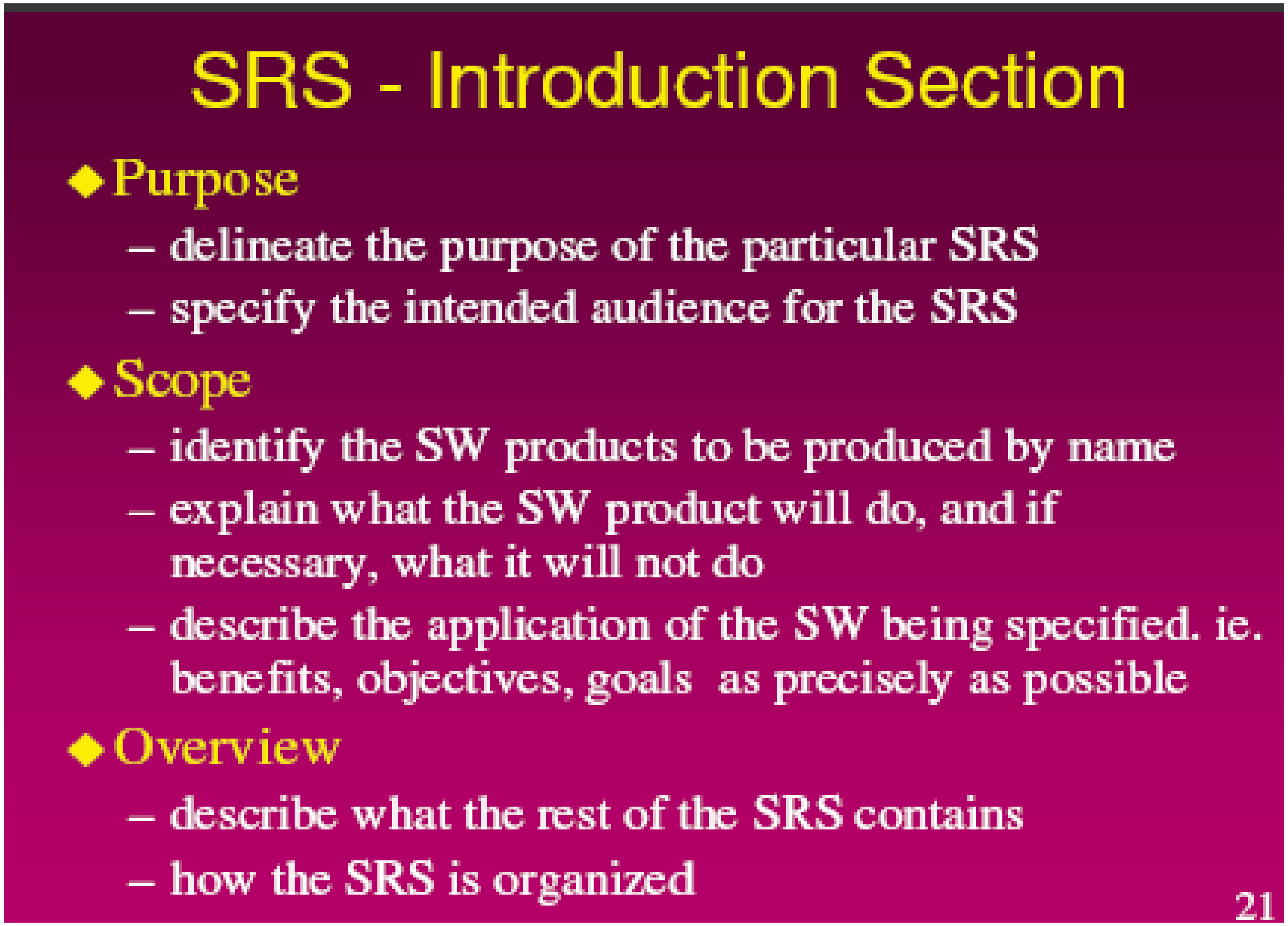


SRS Prototype Outline

[IEEE SRS Standard]

1. Introduction
   1.1 Purpose
   1.2 Scope
   1.3 Definitions, Acronyms and Abbreviations
   1.4 References
   1.5 Overview

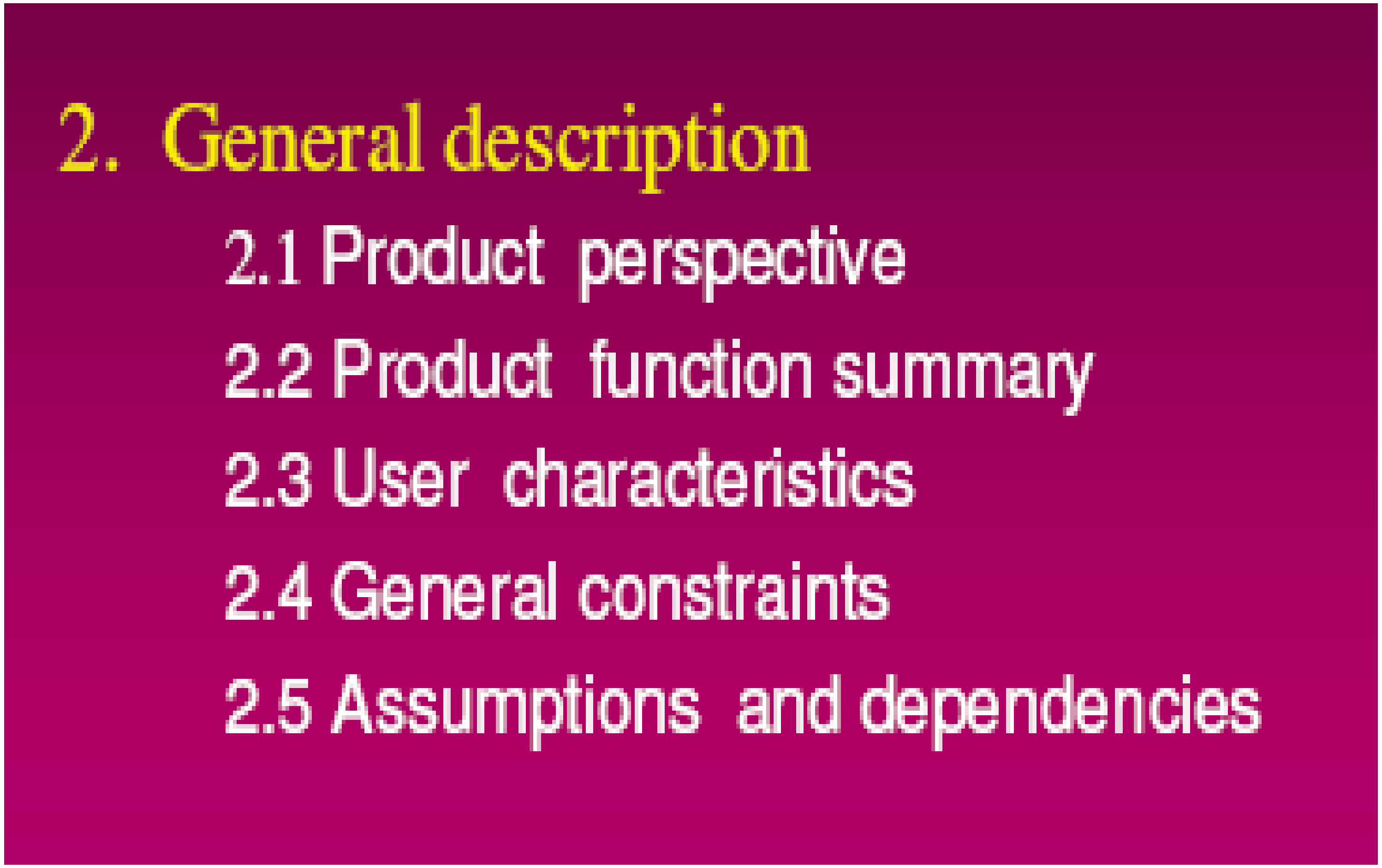
SRS



SRS - Introduction Section

- Purpose
  - delineate the purpose of the particular SRS
  - specify the intended audience for the SRS
- Scope
  - identify the SW products to be produced by name
  - explain what the SW product will do, and if necessary, what it will not do
  - describe the application of the SW being specified. ie. benefits, objectives, goals as precisely as possible
- Overview
  - describe what the rest of the SRS contains
  - how the SRS is organized

SRS



2. General description

2.1 Product perspective

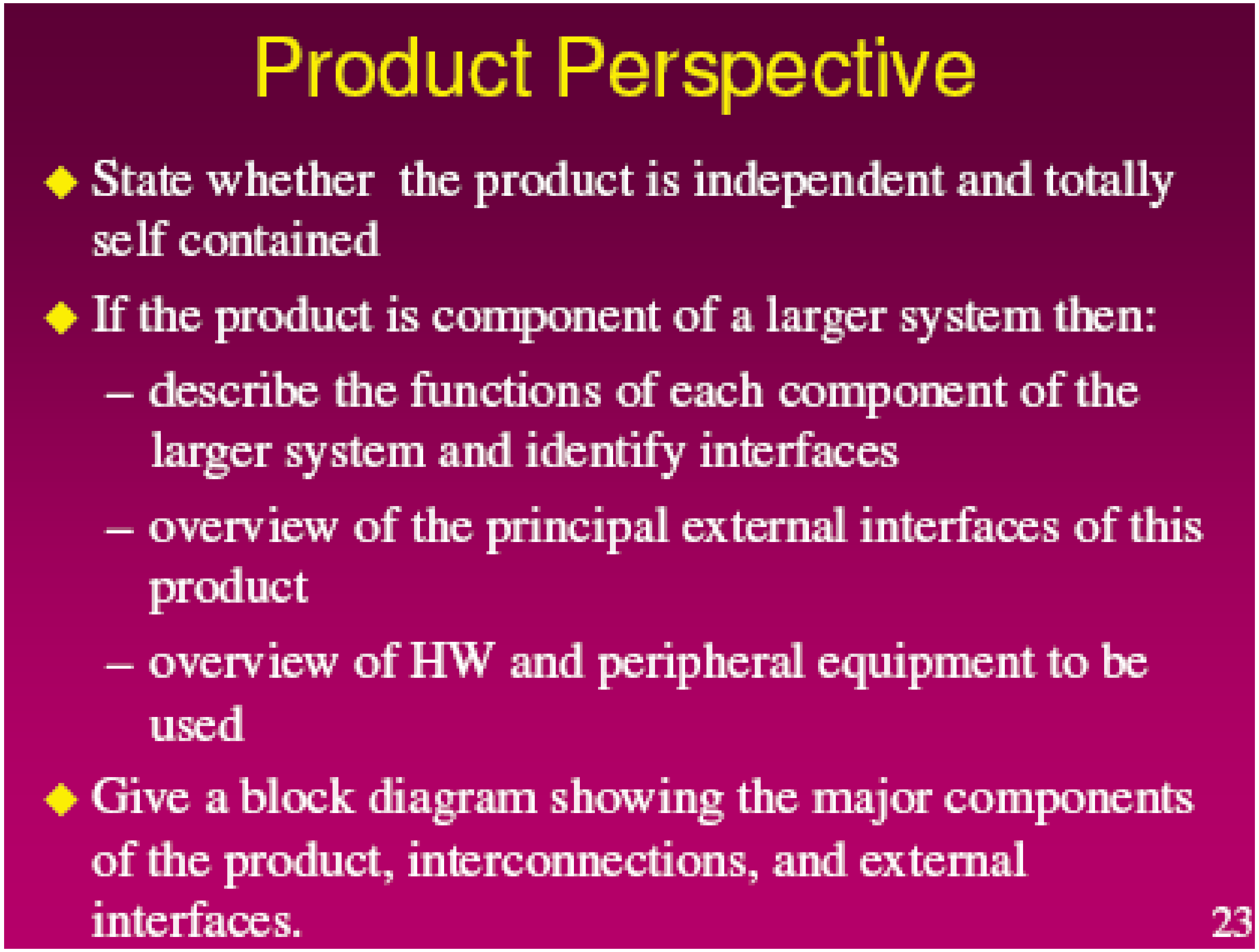
2.2 Product function summary

2.3 User characteristics

2.4 General constraints

2.5 Assumptions and dependencies

SRS

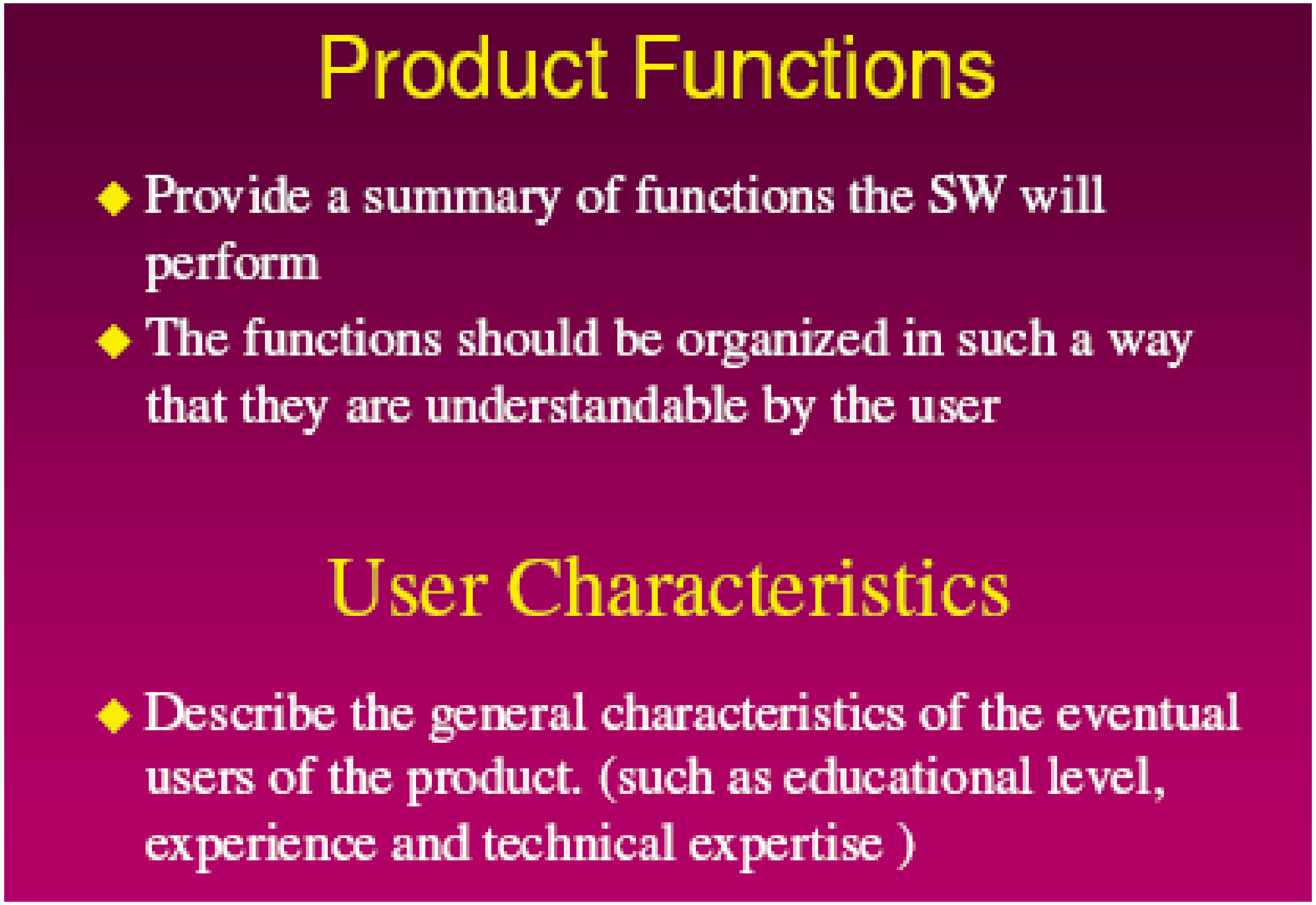


Product Perspective

- State whether the product is independent and totally self contained
- If the product is component of a larger system then:
  - describe the functions of each component of the larger system and identify interfaces
  - overview of the principal external interfaces of this product
  - overview of HW and peripheral equipment to be used
- Give a block diagram showing the major components of the product, interconnections, and external interfaces.

23

SRS



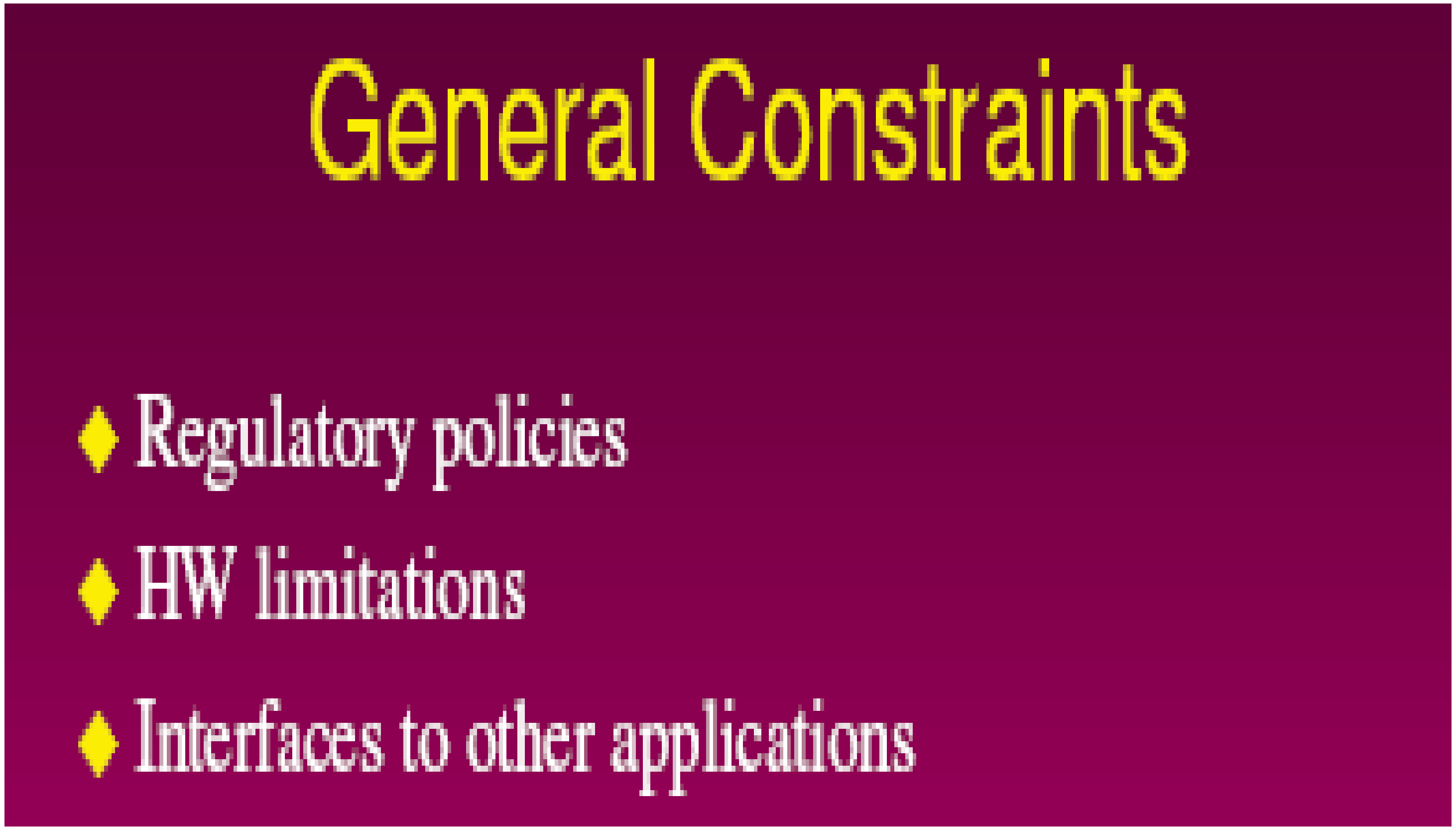
Product Functions

- Provide a summary of functions the SW will perform
- The functions should be organized in such a way that they are understandable by the user

User Characteristics

- Describe the general characteristics of the eventual users of the product. (such as educational level, experience and technical expertise)

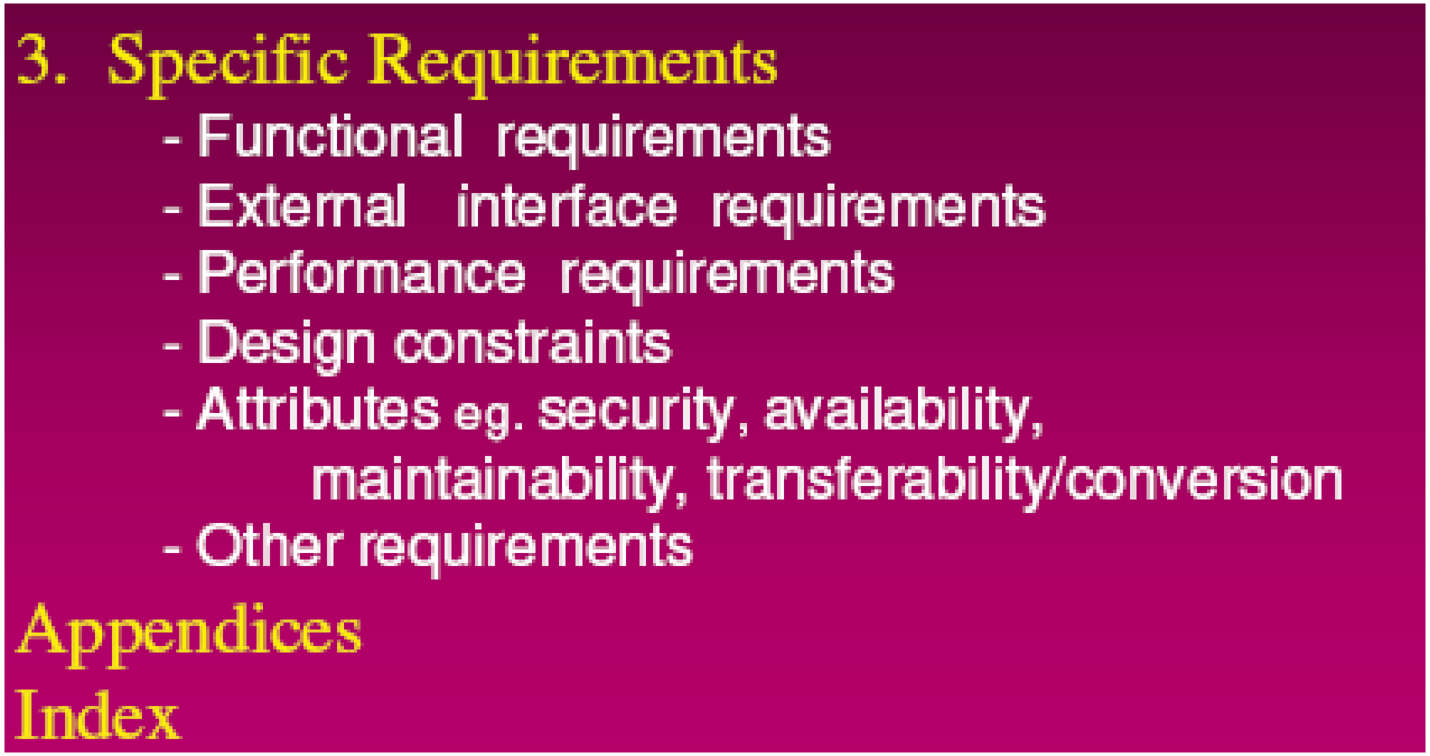
SRS



General Constraints

- Regulatory policies
- HW limitations
- Interfaces to other applications

SRS

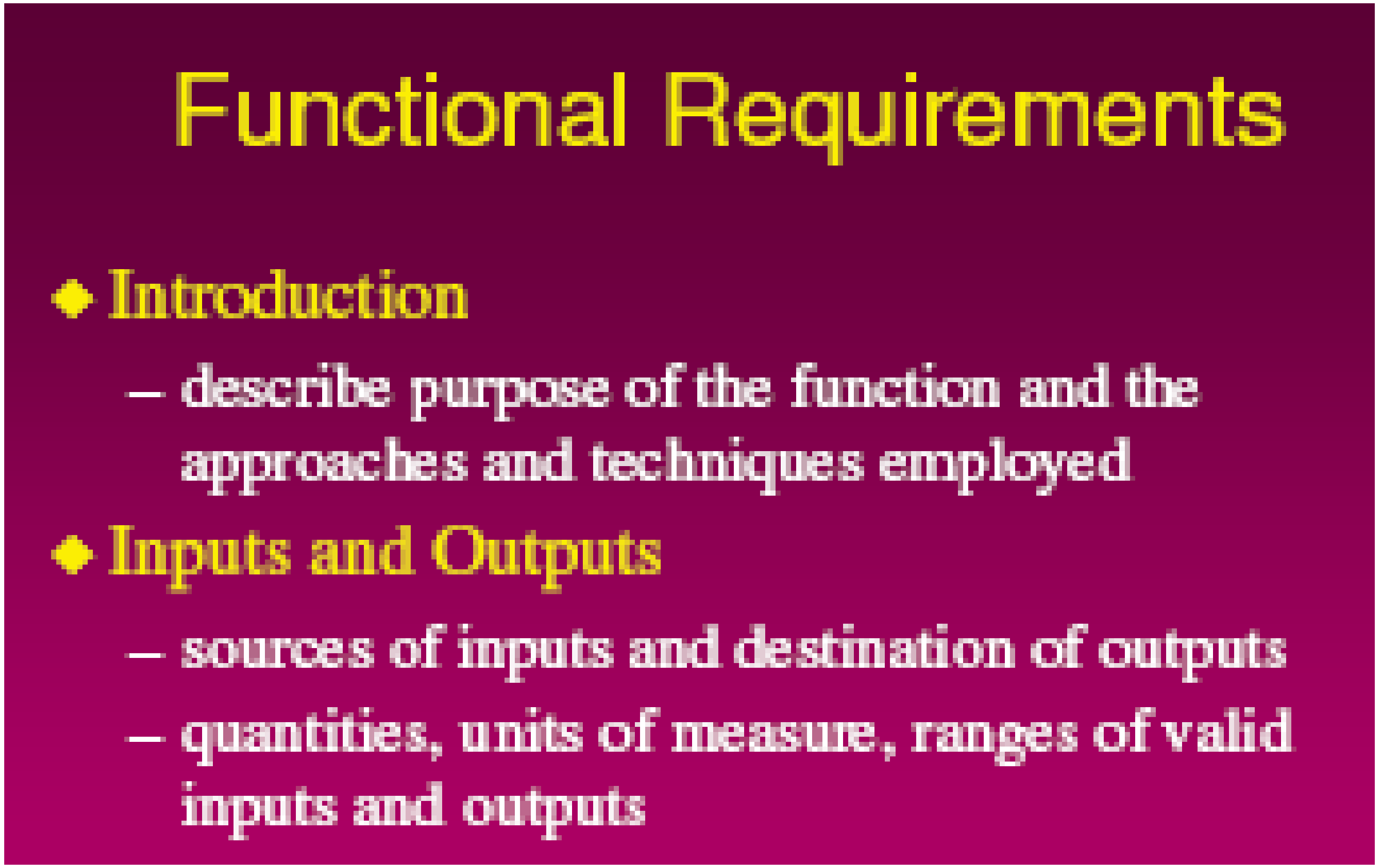


3. Specific Requirements
   - Functional requirements
   - External interface requirements
   - Performance requirements
   - Design constraints
   - Attributes eg. security, availability, maintainability, transferability/conversion
   - Other requirements

Appendices

Index

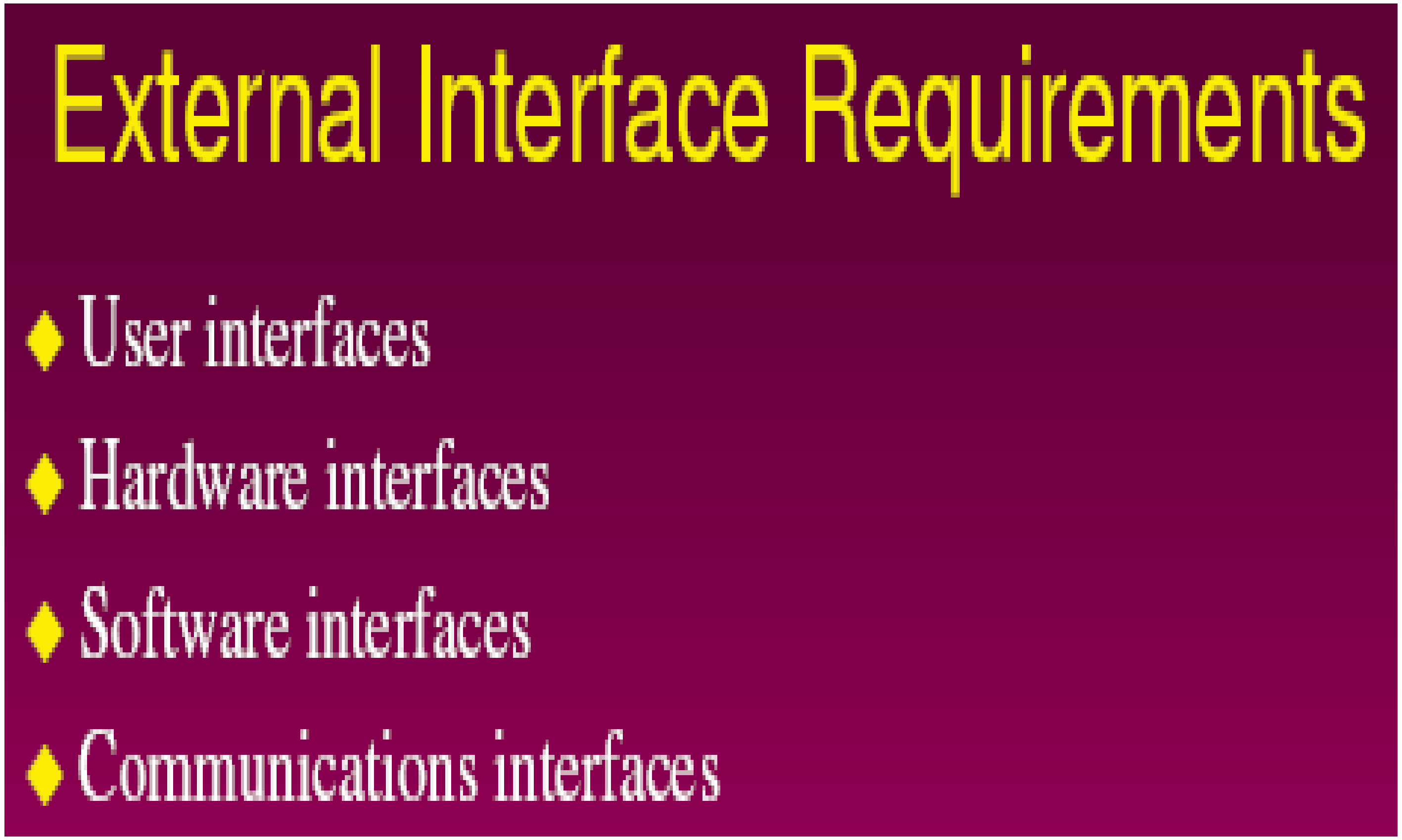
SRS



Functional Requirements

- Introduction
  - describe purpose of the function and the approaches and techniques employed
- Inputs and Outputs
  - sources of inputs and destination of outputs
  - quantities, units of measure, ranges of valid inputs and outputs

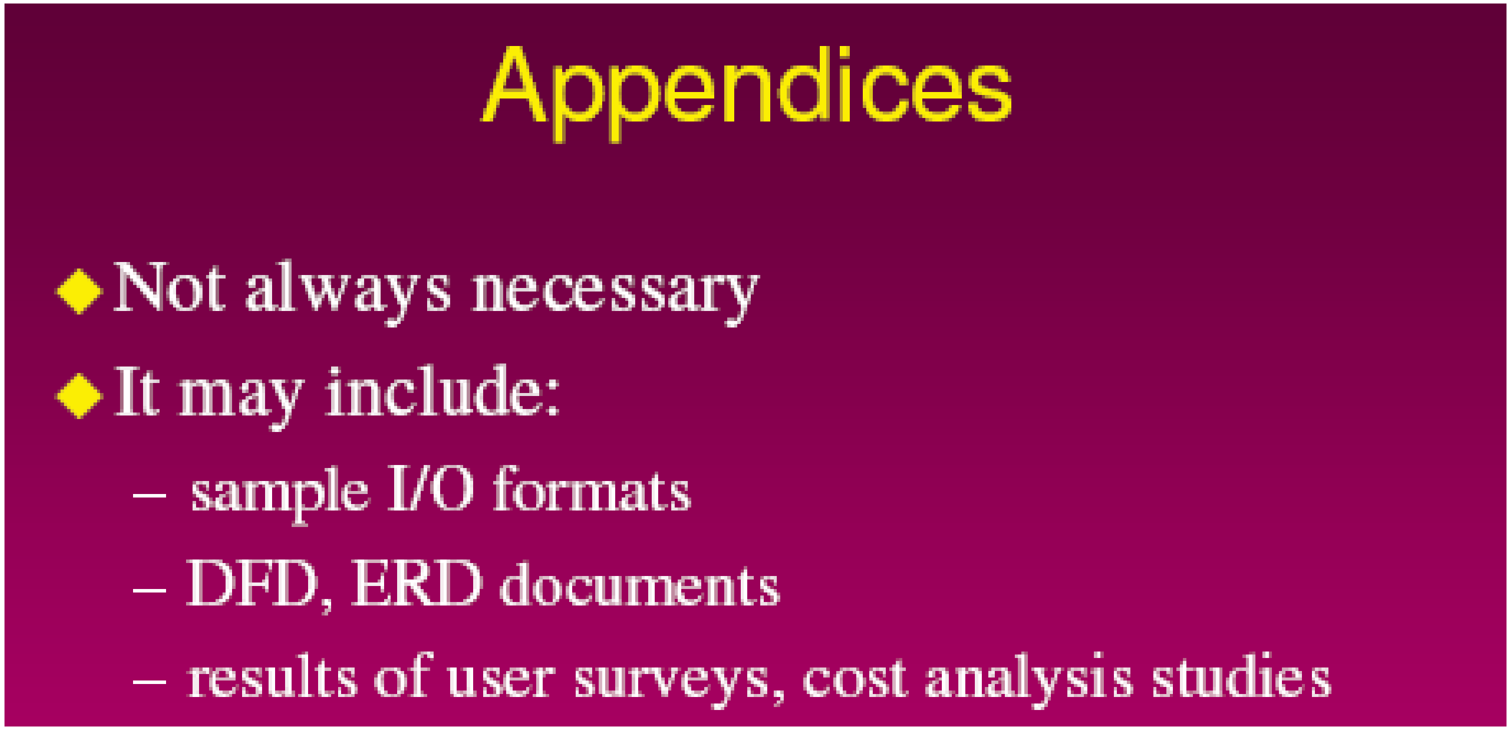
SRS



External Interface Requirements

- User interfaces
- Hardware interfaces
- Software interfaces
- Communications interfaces

SRS



Appendices

- Not always necessary
- It may include:
  - sample I/O formats
  - DFD, ERD documents
  - results of user surveys, cost analysis studies