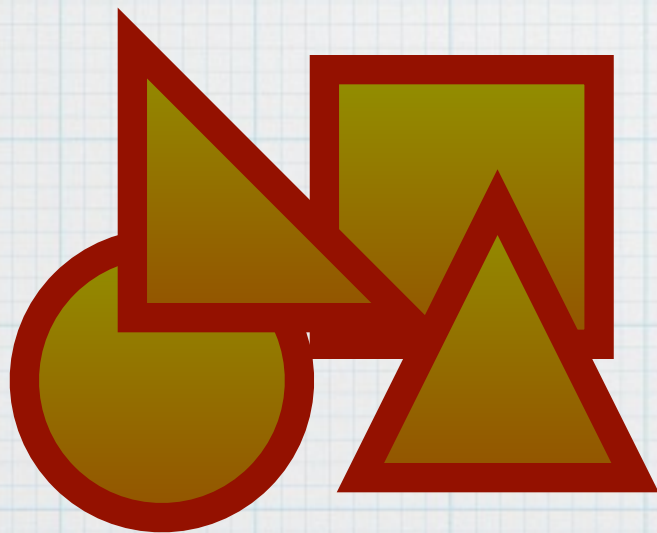


Requirements Engineering Fundamentals

Les Waguespack, Ph.D.



Slides One

Copyright and References

The arrangement, presentation, original illustrations and organization of the materials are copyrighted by Leslie J. Waguespack, Ph.D. with all rights reserved (©2006). Derivations and excerpts in these materials are referenced as follows:

- * Requirements Engineering, Kotonya & Sommerville, Wiley, Chichester, West Sussex, England, ISBN 0-471-97208-8
- * Software Requirements Engineering, Second Edition, Richard H. Thayer and Merlin Dorfman, eds., pp. 7-22. Los Alamitos, Calif.: IEEE Computer Society Press, 1997.
- * Use Case Modeling, Bittner & Spence, Addison-Wesley / Pearson Education, Inc., Boston, MA, ISBN 0-201-70913-9
- * Writing Effective Use Cases, Cockburn, Addison-Wesley, Boston, MA, ISBN 0-201-70225-8
- * UML and the Unified Process - Practical Object-Oriented Analysis and Design, Arlow & Neustadt, Addison-Wesley / Pearson Education, Inc., Boston, MA, ISBN 0-201-77060-1
- * Business Modeling With UML, Eriksson & Penker, Wiley, Indianapolis, IN, ISBN 0-471-29551-5
- * UML 2 Toolkit, Eriksson, Penker, Lyons & Fado, Wiley, Indianapolis, IN, ISBN 0-471-46361-2
- * Enterprise Modeling With UML Designing Successful Software Through Business Analysis, Addison-Wesley, Reading, MA, ISBN 0-201-43313-3
- * Object Oriented Systems Engineering, Waguespack, course notes CS390, CS460, CS630, CS771, Computer Information Systems Department, Bentley College, Waltham, MA.

Outline

1. Introduction

2. Requirement Abstraction Level

3. The Requirements Document

1. Introduction

- * What is Requirement Engineering?
- * Why do we need it?
- * What does it do for us?
- * What can it do TO US?
- * How do we control it?

Requirements Engineering

Kotonya & Sommerville

- * Requirements Engineering** is the development and maintenance of specifications suitable for directing the implementation of a system.

System Requirements

Kontonya & Sommerville

- * **System Requirements** are specifications of the client's problem and the system characteristics that will solve that problem:
 - * 1) static structure,
 - * 2) dynamic behavior,
 - * 3) performance / capacity constraints,
 - * 4) standards for quality assessment

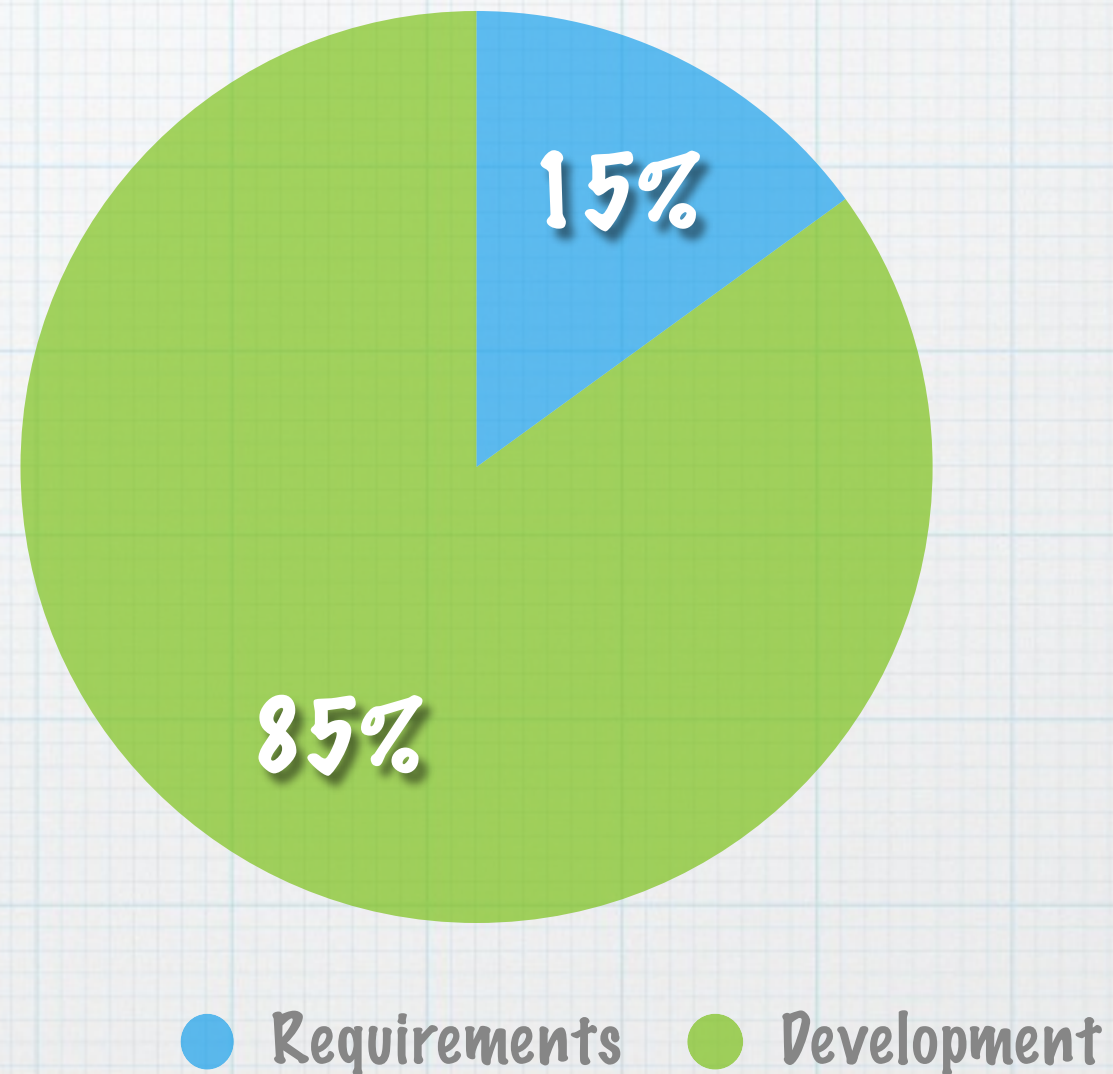
Software Requirements

*** Software Requirements are specifications of the software components of a system intended to guide the design of specific software components:**

- * 1) static structure,**
- * 2) dynamic behavior,**
- * 3) performance / capacity constraints,**
- * 4) standards for quality assessment**

Systems Development Costs

- * As much as 15% of total development costs will be expended in requirements gathering and documentation



What is the purpose of Requirements Engineering?

- * The primary purpose of RE is to control **RISK!**
- * The primary product of RE is the **Requirements Document.**
- * The Requirements Document gathers and records requirements according to the **Stakeholders**

RE is a Systems Process

- * RE as Systems Engineering

- * gathering (people, hardware, software, networking, facilities) to serve a designated goal
- * special risks require special resources
 - * experts in the problem domain, market, country, etc.
- * RE as a process must be “managed for quality”
- * The success of RE depends on how well RE is project managed!


RE: Risk Management

- * Development success is delivering the **Right** solution to the **Right** problem!


RE: Risk Management

* Development success is delivering the **Right solution** to the **Right problem!**

**Systems Analysis
and Design**

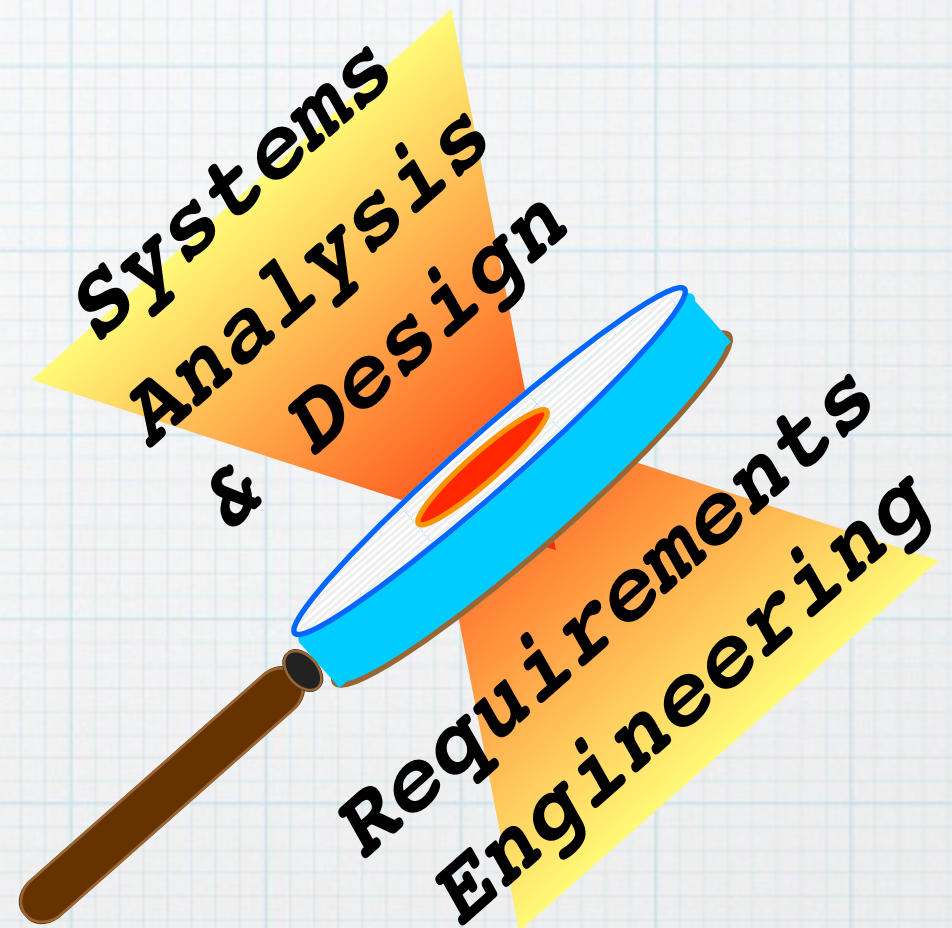


**Requirements
Engineering**



“Through a mirror darkly?”

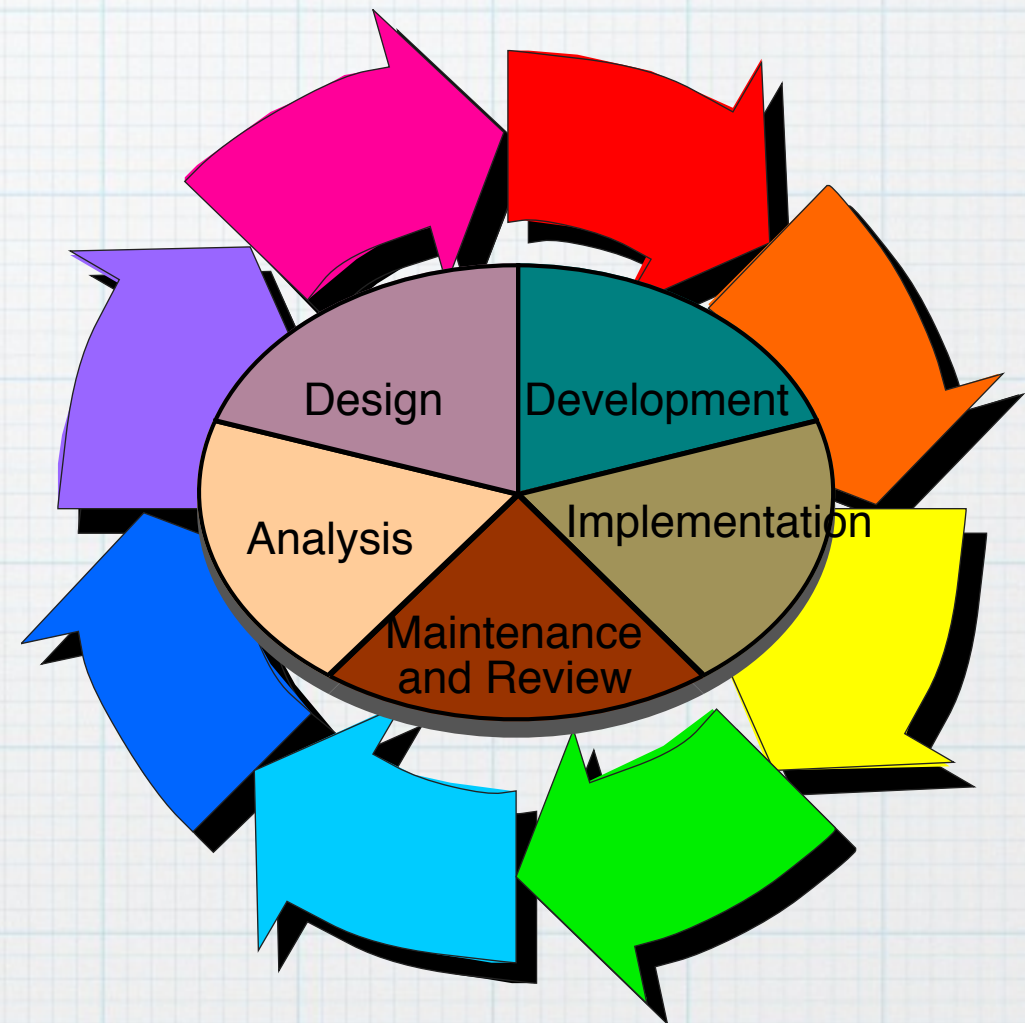
- * Although distinct, their relationship is highly inter-dependent
- * Compatible terminology, glossary, model syntax saves!



Requirement Management

“Life goes on!”

- * Requirements exist in a “living” environment
- * “Old” requirements are “Wrong” requirements



“Wrong” Requirements Cause Problems

- * “Wrong” as in:**
 - * missed client needs**
 - * misunderstood client needs**
 - * inconsistent or incomplete client needs**
 - * convoluted or obscure need descriptions**
 - * developer requirements vs. client requirements**
- * All of which result in added project cost
(or project failure!?)**

Enough vs. Too Much

(requirement engineering forever!)

- * Client satisfaction hinges on cost / benefit

- * costs

- * time

- * money

- * opportunity

- * benefits

- * function

- * productivity

- * opportunity



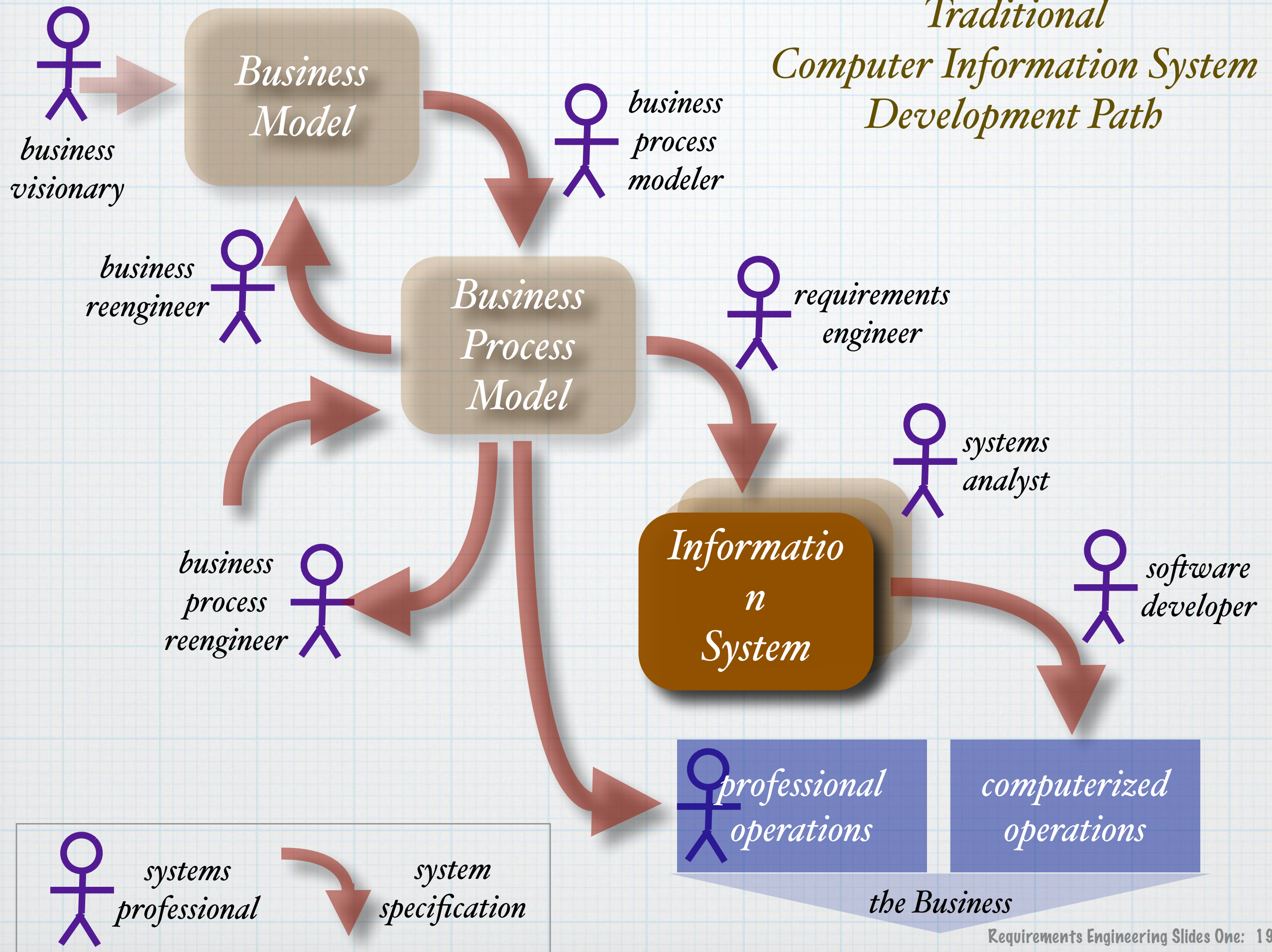
Risk Driven RE Process

- * Matching the scope and rigor of the RE process to the identified risks for the project determines cost/benefit success
- * Effective Risk assessment up front is key to effective requirement engineering down the line

2. Requirement Abstraction Level

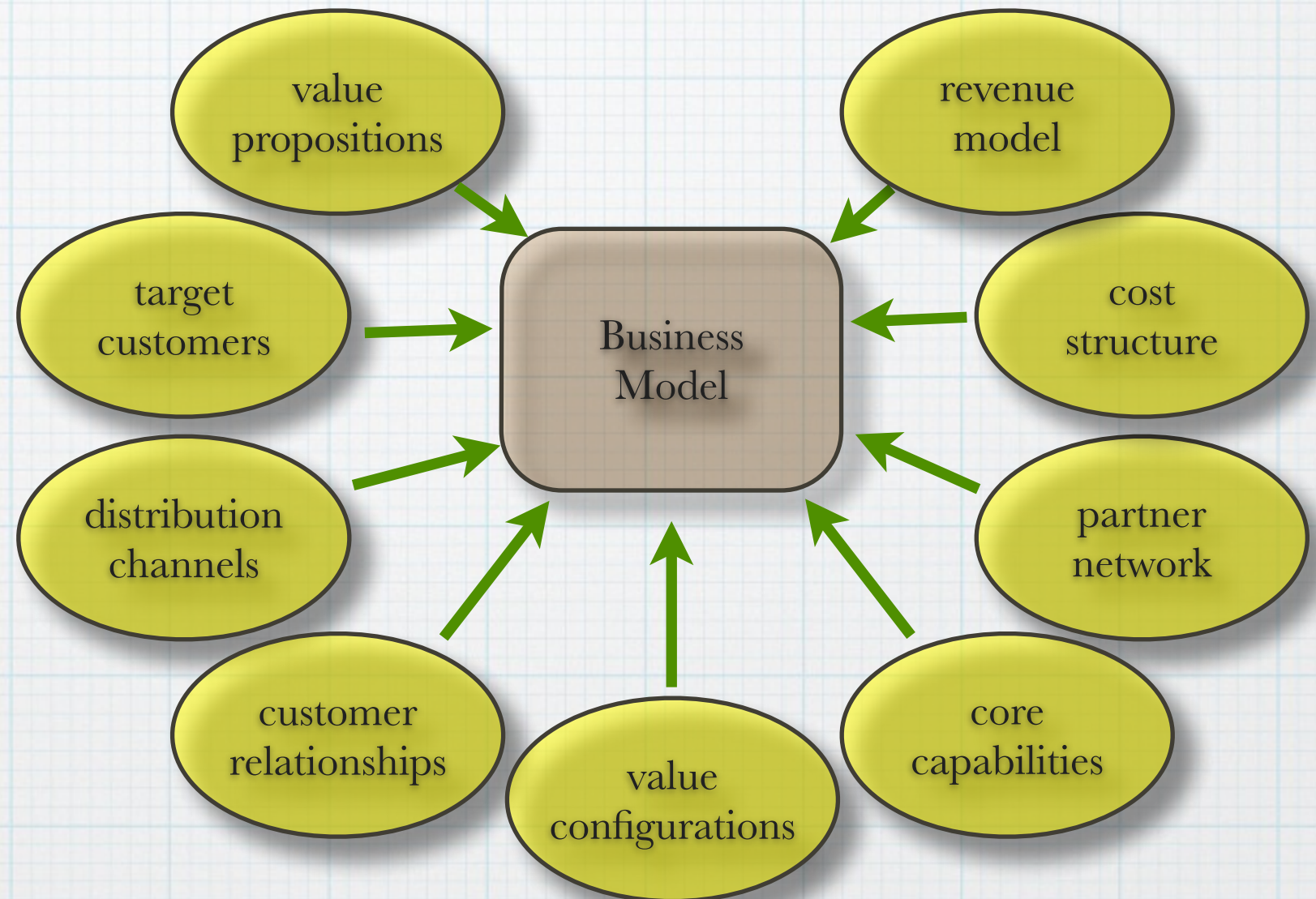
- * Business Model Level
- * Business Process Level
- * Information System Level

Traditional Computer Information System Development Path



Business Model Level

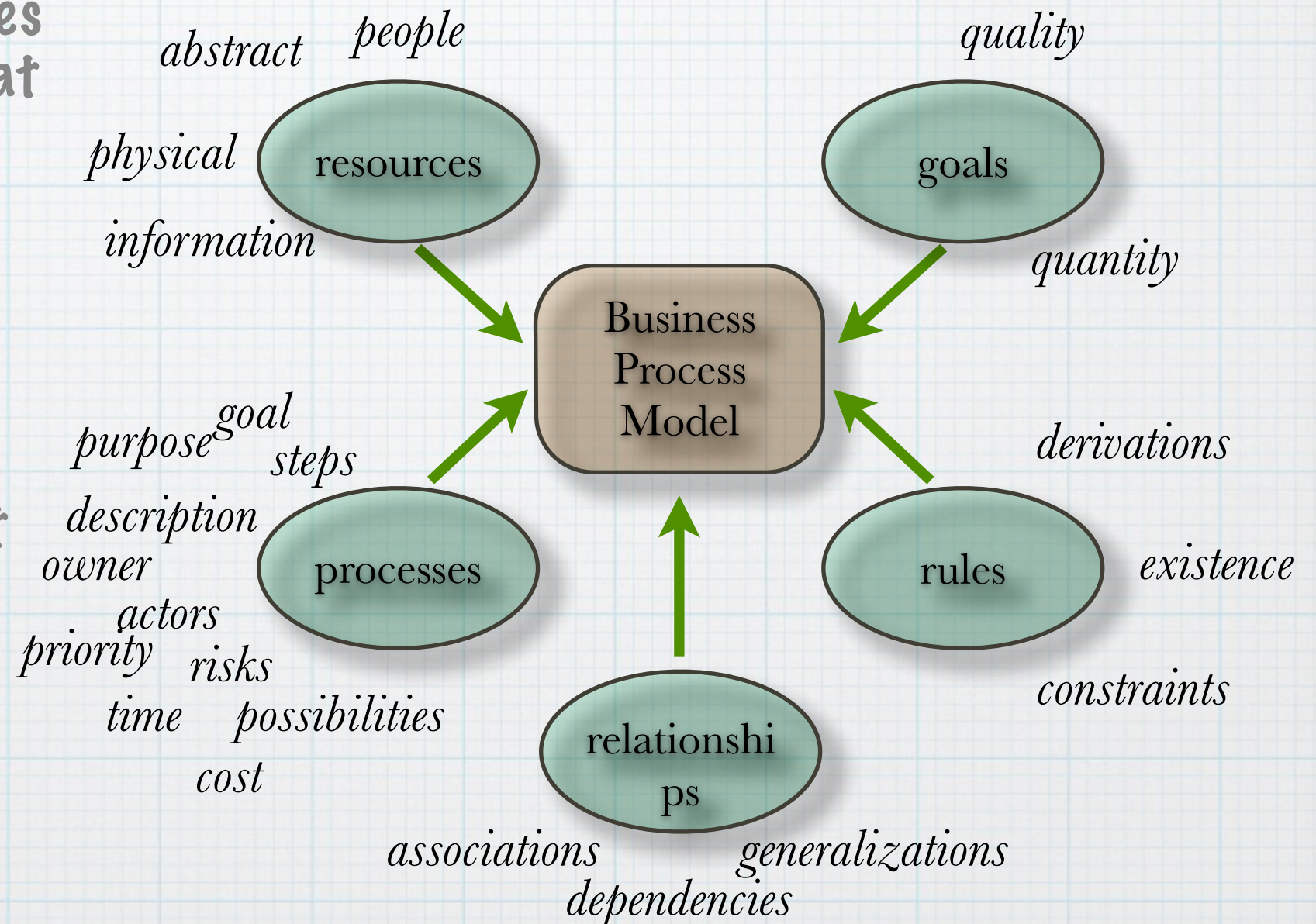
- * This level describes the purpose and function of the business entity irrespective of technology and implementation
- * The business model establishes the core values of the enterprise



Process Model Level

- * This level describes the activities that define each operation of the business

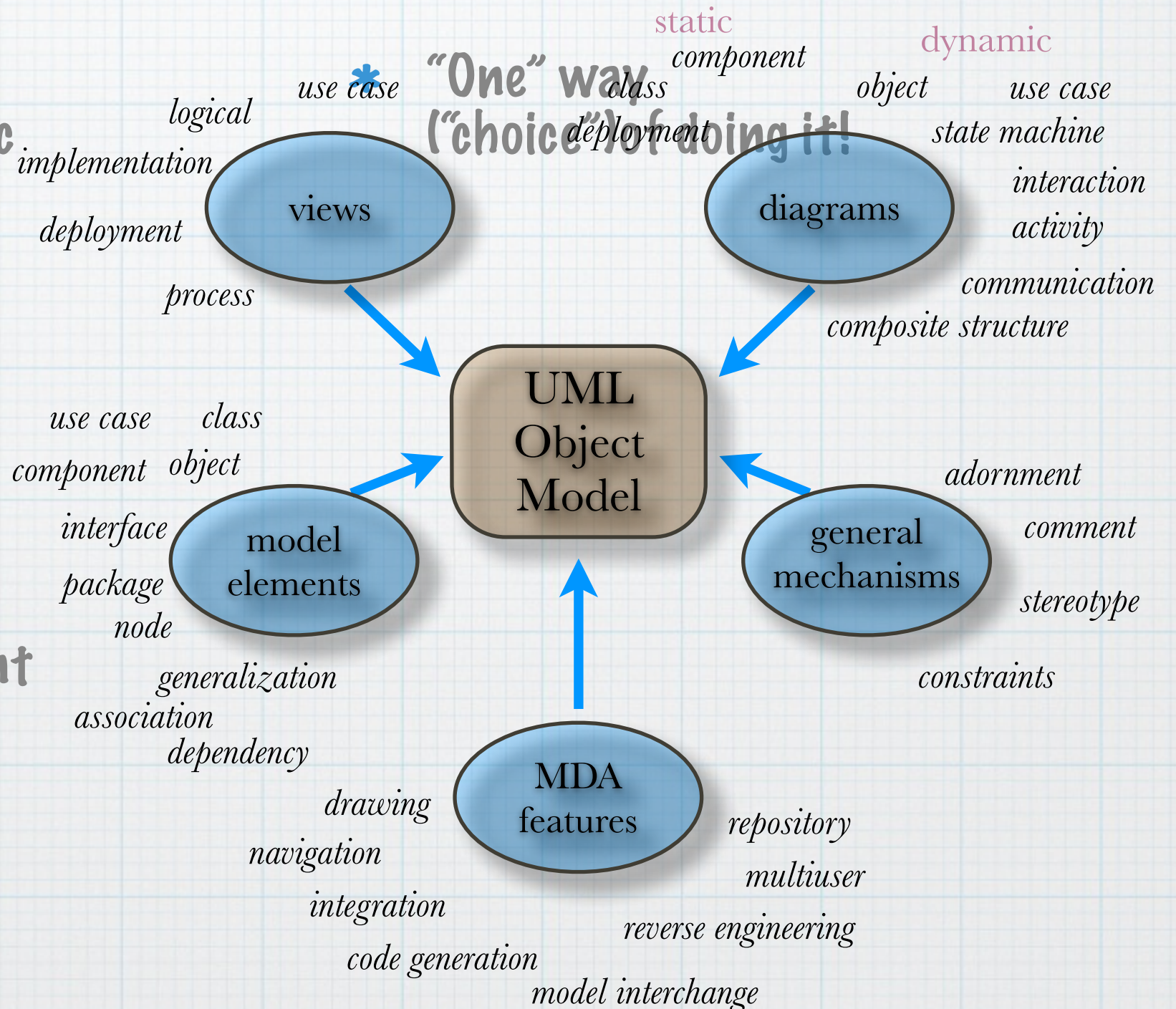
- * There will be several of these descriptions that together explain business operations



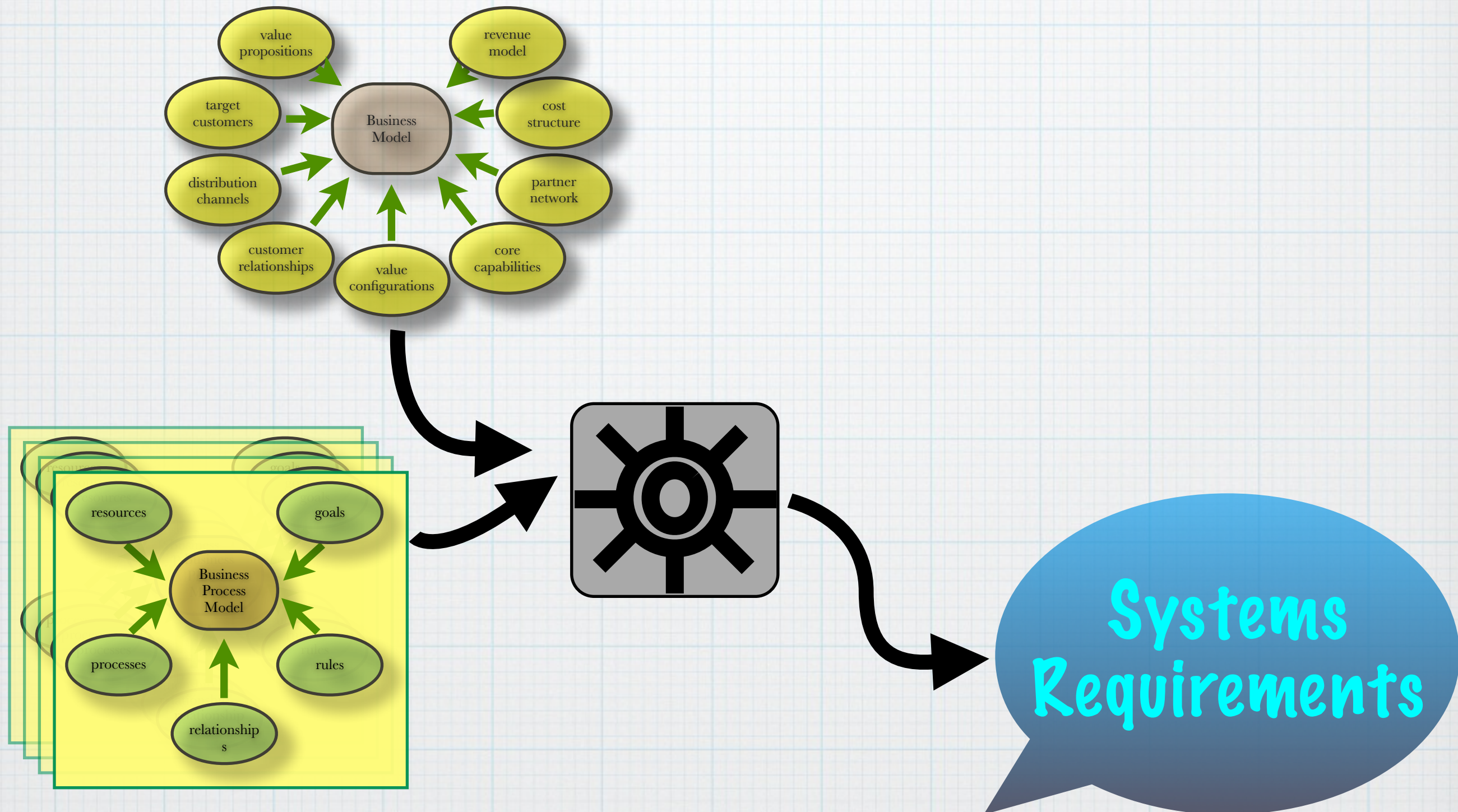
Information System Level

- * This level focuses on IS specific static and dynamic structures leading to programming and implementation

- * This example is based on the OO paradigm; different paradigms would focus on different characteristics



Requirements Definition



3. The Requirements Document

- * Requirement contents
- * Document standards
- * Document users

Requirement Document Contents

- * system services and functions
- * operating constraints
- * quality standards
- * system interfaces / interconnections
- * problem domain specific characteristics
- * development process constraints

services & functions

- * specific services provided to
 - * owner, management, users, customers, partners (all stakeholders)
- * specific functions required to be visible in the interfaces
 - * transactions, reports, controls
- * specific support to the business & process models

operating constraints

- * operating environment description
- * fault tolerance / recovery
 - * “user friendliness,” fault diagnosis, mtbf - mean time between failures, mttr - mean time to recovery
- * performance parameters
 - * capacity, frequency, response time, accuracy

quality standards

- * application domain standards

- * FASB, ANSI, ISO, DOD, IEEE, Sarbanes Oxley

- * standard of user expectation

- * companion system standards HCI, Look and Feel, adaptability, extensibility

system interfaces / interconnections

- * Human-Computer-Interface

- * language, color, consistency, intuition, configurability, localization (language, currency, time, culture)

- * Supply chain

- * suppliers, customers, partners, government

- * Data interchange

- * internet, financial, international

problem domain specific characteristics

- * application specific expert knowledge
 - * client confidential, proprietary, experimental
- * market idiosyncrasies
- * non-standard, exceptional, innovative business practice

development process constraints

- * technological compatibility
 - * hardware, software, tools, CASE, legacy, “opportunity”
- * skill level compatibility
- * client project management standards

Documentation Standards

- * Financial Accounting Standards Board
- * Department of Defense
- * American National Standards Institute
- * Institute of Electrical and Electronic Engineers

Typical Requirements TOC

IEEE/ANSI 830-1993

1. Introduction
2. General Description
3. Specific Requirements
4. Appendices
5. Index

1. Introduction

1. Purpose of the requirement document
2. Scope of the product
3. Glossary: Definitions, acronyms, abbreviations
4. References
5. Overview

2. General Description

2.1. Product perspective

2.2. Product functions

2.3. User characteristics

2.4. General constraints

2.5. Assumptions and dependencies

3. Specific Requirements

3.1. functional

3.2. non-functional

3.3. interface

- * internal

- * external

3.4. performance

3.5. logical database

3.6. design constraints

3.7. system attributes and quality

Write Once - Read Never?!

- * All stakeholders need to read and understand it
- * Unreadable documentations isn't worth writing!
- * use audience-aware style and vocabulary
- * use formal syntax for complex information
- * CASE friendly documentation will remain current in the long run
- * technical writing expertise is not a luxury - it is a necessity!

Wrap Up

1. Introduction

- * what, why, how ???

2. Requirement Abstraction Level

- * business, business Process, information system

3. The Requirements Document

- * contents, standards, users
- * legibility