Lecture Slides for
Managing and Leading Software Projects

Chapter 3: Establishing the Project Foundations

developed by

Richard E. (Dick) Fairley, Ph.D.

to accompany the text

Managing and Leading Software Projects

published by Wiley, 2009

Chapter 3 Topics

- Software Acquisition
- Requirements Engineering
- Requirements Development
- Requirements Analysis
- Technical Specifications
- Requirements Verification
- Requirements Management
- Process Foundations
- Specifying the Scope of Your Project
- The Contractual Agreement

Additional Information

- Appendix 3A to Chapter 3 of the text presents the relevant elements of CMMI-DEV-v1.2, ISO/IEC and IEEE/EIA Standards 12207, IEEE/EIA Standard 1058, and the PMI Body of Knowledge.
- Terms used in this chapter and throughout the text are defined in Appendix A text.
- These presentation slides and other supporting material are available at the URL listed in the Preface to the text.

Objectives for Chapter 3

- After reading this chapter and completing the exercises, you should understand:
  - the nature of requirements engineering,
  - documenting the scope of a project, and
  - establishing a contractual agreement.

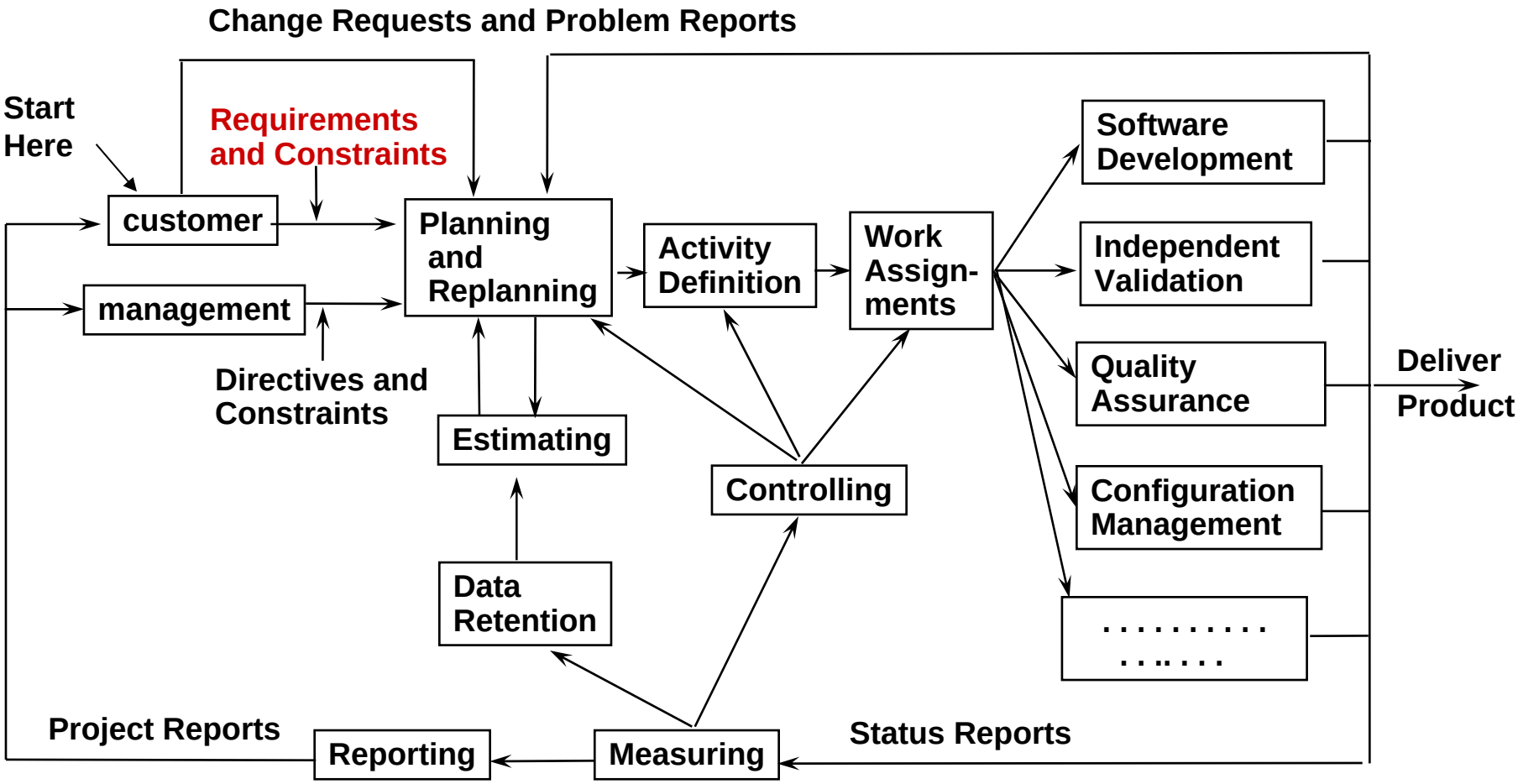
Foundation Elements for Software Projects

Product Foundations	Explanation
Operational requirements	External, user view of the system
System requirements and architecture	Hardware, software, people elements
Design constraints	Predetermined design decisions
Software requirements	Internal technical specifications
Process Foundations	
Workflow model	Managerial work activities and work products
Development model	Technical work activities and work products
Contractual agreement	Statement of understanding between developer and acquirer
Project plan	The project roadmap

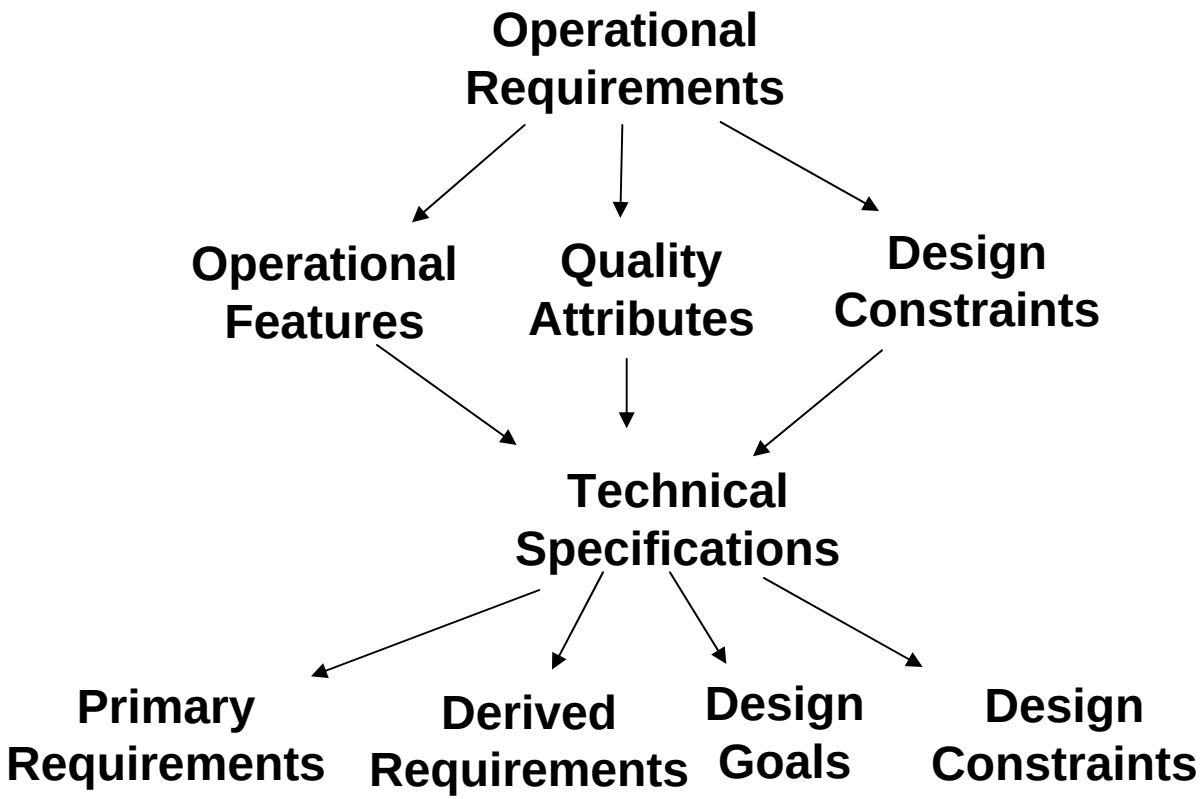
Foundation Elements

- Chapter 3 of the text covers:
  - operational requirements
  - system requirements and design
  - design constraints,
  - software requirements,
  - contractual agreements
- Other foundation elements:
  - the workflow model for software projects is presented in Chapter 1
  - selecting and tailoring of software development models are presented in Chapter 2
  - project planning and the format and contents of project plans are presented in Chapter 4

A Workflow Model for Software Projects



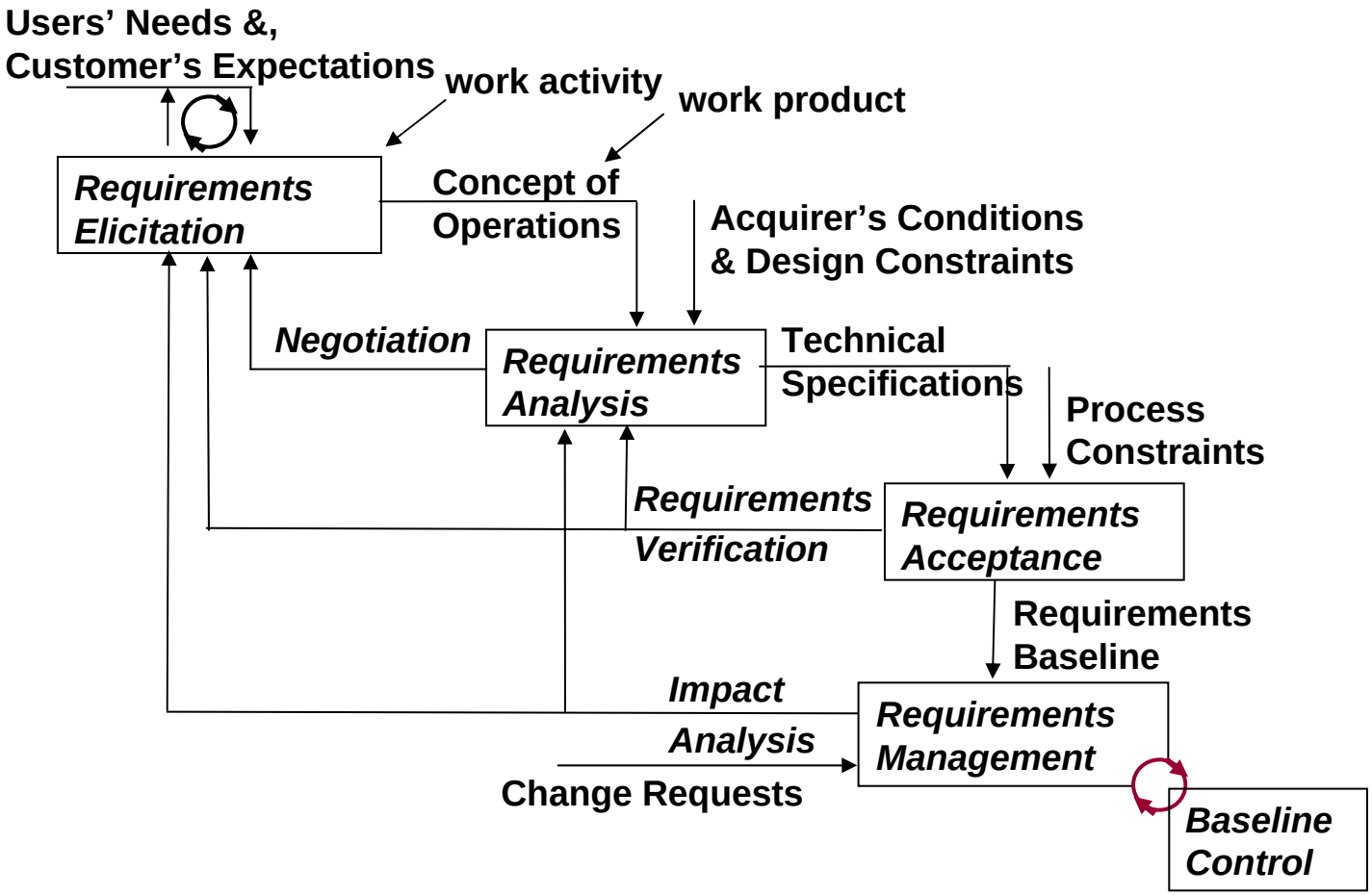
A Taxonomy of Software Requirements



Requirements Engineering

- Requirements engineering consists of:
  - requirements development, and
  - requirements management.
- The primary activities of requirements development are:
  - elicitation of operational requirements, analysis, translation into technical specifications, and acceptance baselining of the technical specifications
- Requirements management is concerned with baseline control of requirements
  - and with maintaining consistency among requirements, effort, schedule, budget, and technology

Process Flow of Requirements Engineering



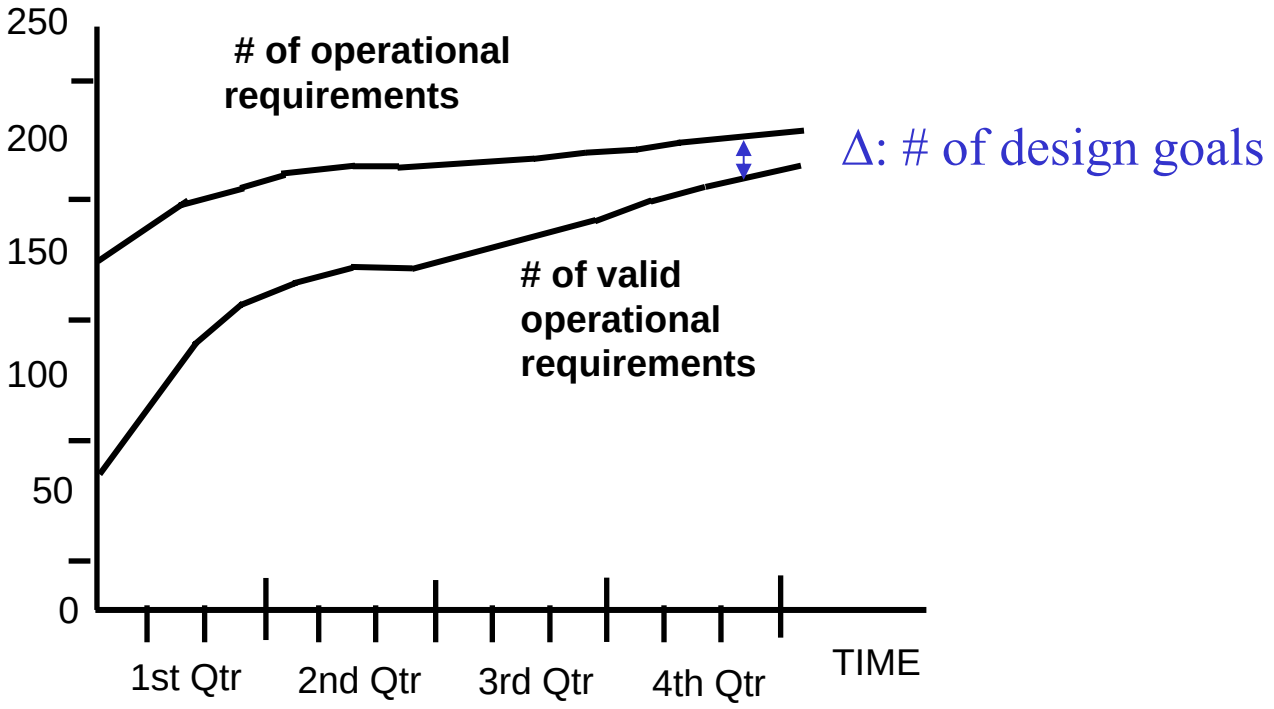
Operational Requirements*

- The external (user) view of the system
- Includes
  1. Functional requirements (what must the system do?)
     - Stimulus / response relationships
     - Prioritized as Essential, Desirable, Optional
       - and prioritized within categories
     - Uniquely allocated to elements of the system
  2. Quality attributes (how well must the system do it?)
     - Safety, security, reliability, performance, throughput, capacities
     - May be prioritized
       - e.g., security versus performance
     - Usually apply to multiple (perhaps all) elements of the system
- Each operational requirement is tagged as “valid” or “design goal”

* See Chapter 3 for examples

Valid Requirements and Design Goals

- Valid requirements have objective validation* criteria
  - design goals do not



* Validation is concerned with determining the degree to which a work product satisfies it intended purpose when placed in its intended environment

Techniques for Eliciting Operational Requirements (1)

- Introspection: what would I want/need/desire if I were a user of the proposed system?
- Brainstorming: free association and generation of ideas for the proposed system
- Post-It notes and white board: create, modify, group, and rearrange statements of needs and desires
- Paper prototypes and storyboards: construct user interfaces and operational scenarios
- Questionnaires: which of the following features do you need/desire?
- Observation: watch people performing their work tasks
- Open-ended interviews: tell me how you would use the proposed system

Techniques for Eliciting Operational Requirements (2)

- Focus groups: please tell us what you would want/need/desire in the proposed system
- Operational walkthroughs: development of scenarios by interacting with users
- Demonstrations: how to you like this interface? What should be added/removed/changed?
- Protocol analysis: document the tasks users perform and the features they would need in the proposed system
- Business case analysis: what features are needed to support the operations of our business?
- JAD (Joint Application Development) sessions: facilitated meetings with users

The Concept of Operations* (ConOps)

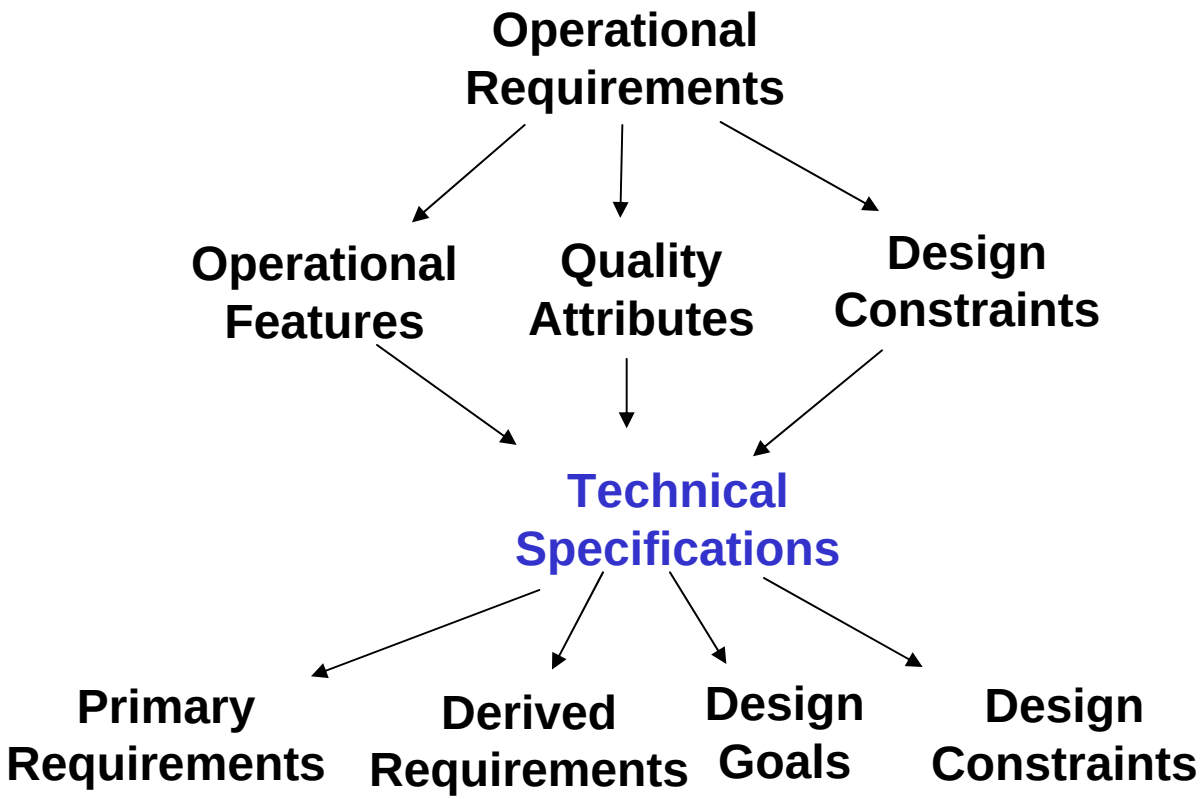
- Contents of a ConOps include:
  - Needs and expectations that motivate development of a new system or modification of an existing system
  - An operational vision for the proposed system
  - Modes of operation for the proposed system
  - User classes and characteristics
  - Kinds of and characteristics of operations personnel
  - Operational requirements
  - Operational scenarios
  - Prioritized system features and quality attributes
  - Impact of the proposed system on the development, operational, and maintenance environments

*see IEEE Std 1362 Concept of Operations document

Use Case Template and Example

- *Use Case ID:* ATM #34
- *Use Case Name:* User Logon
- *Actor that initiates the use case:* Bank customer
- *Other actors, if any:* none
- *Statement of the purpose of this use case:*
  this use case documents the way bank customers log onto an ATM
- *Preconditions* (must be true before this use case can be "executed"):
- customer has a valid bank card and PIN (Personal Identification Number)
- *Primary scenario* (to describe the main action of the use case):
- can be documented using a sequence diagram, state diagram, or narrative
- *Post-conditions* (what *must be true* after this use case "executes"):
- customer is logged on OR customer has received a sorry message
- *Alternative scenarios* (exception handling):
- invalid bank card; incorrect PIN; invalid account number; ATM is off-line
- *Comments:* this use case belongs to Initiate Transaction. The ATM should have good response time during User Logon.

A Taxonomy of Software Requirements



Technical Specifications

- The technical specifications, which are derived from the operational requirements, include:
  - primary requirements,
  - derived requirements,
  - design goals, and
  - design constraints.
- Each of these four kinds of technical specifications should be:
  - separately identified ,
  - listed within their own category, and
  - prioritized with each category.

Primary and Derived Requirements

- Primary requirements are operational features that have been translated into objectively stated specifications
  - they include both functional requirements and quality requirements
- Derived requirements are requirements for system features and quality attributes that are not visible to users but are necessary to support the operational requirements.
  - derived requirements are included in the technical specifications for two reasons:
    - to decompose high-level operational requirements into detailed technical specifications, and
    - to provide additional capabilities needed to satisfy operational requirements.

Design Goals

- Design goals are operational requirements that have not yet been, or cannot be, translated into objectively stated technical specifications
  - each design goal should be examined for possible translation into one of the other three categories (primary requirement, derived requirement, or design constraint)

Design Constraints

- A design constraint is a design decision stated in the requirements and for which no flexibility in design or implementation is allowed
- Each design constraint restricts the design space available to the software designers
  - and may result in a sub-optimal design of the overall system
  - design constraints should therefore:
    - be identified as such,
    - have their necessity justified,
    - provide flexibility, if any, and
    - be stated in an objective manner

Documenting Technical Specifications*

Specific Requirements	Description
External interfaces	Inputs to and outputs from the software system
Functions	Actions taken in accepting and processing of inputs and generating of outputs
Performance requirements	Static and dynamic numerical requirements
Logical database requirements	Requirements for information to be placed into a database
Design constraints	Constraints imposed by conformance to standards conformance, hardware limitations, etc
Software system attributes	Reliability; availability; security; maintainability; portability

*see IEEE Standard 830

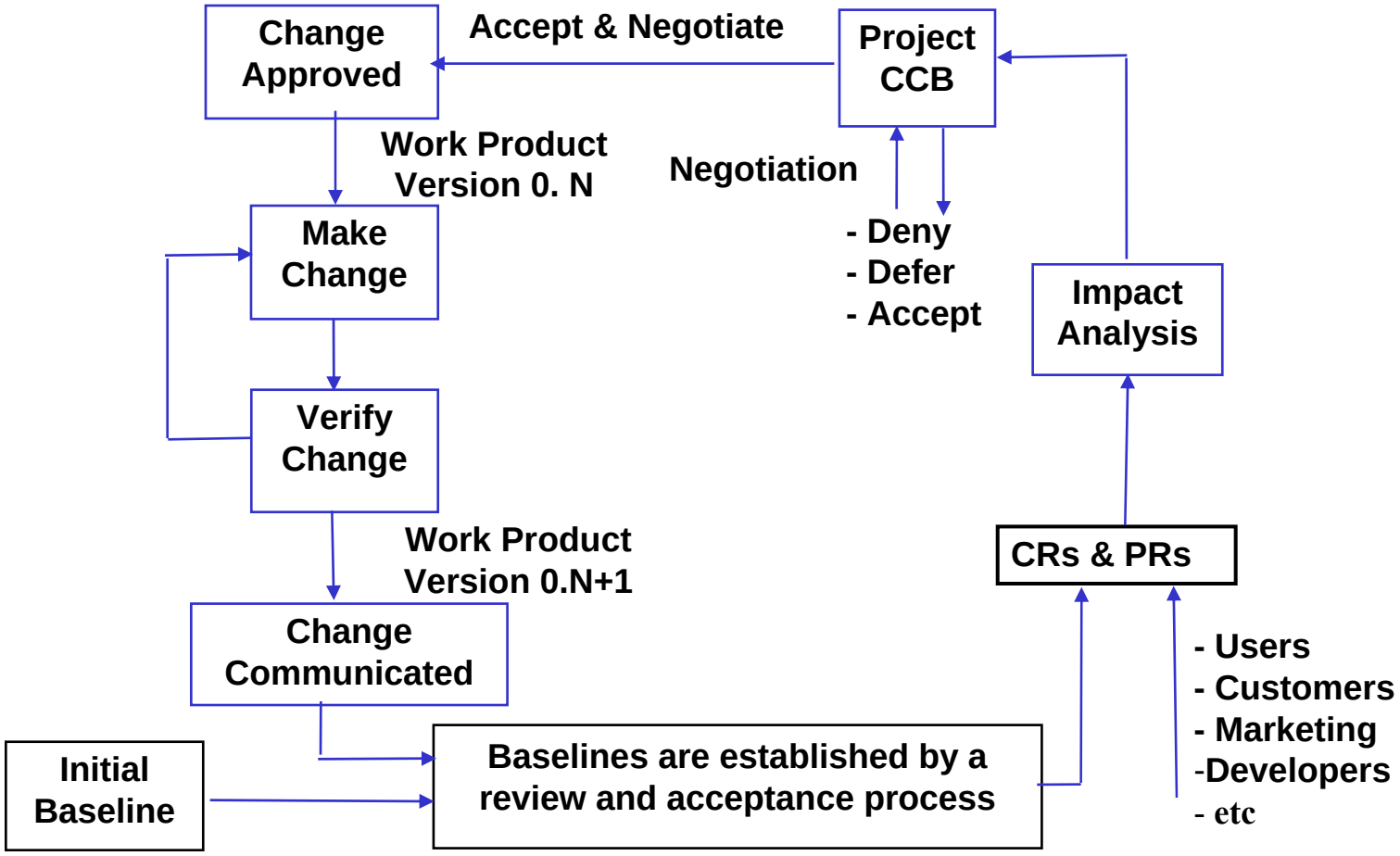
Verification of Technical Specifications

- Verification is concerned with determining the degree of completeness, correctness, and consistency of a work product
- Verification techniques for technical specifications include:
  - analysis
  - reviews
  - walkthroughs
  - traceability

A Traceability Matrix

Primary Requirements →						
Operational Features ↓	[P1]	[P2]	[P3]	[P4]	[P5]	[P6]
[E1]	X					
[E2]			X			
[E3]				X		
[E4]						
[D1]					X	
[D2]		X				
[D3]						
[D4]						X

Requirements Management and Change Control Boards



CR: Change Request
DR: Defect Report

Duties of a CCB

A CCB:

- identifies the work products to be placed under version control,
- approves initial baselines of those work products,
- evaluates proposed changes to baselines, approves, defers, or denies change proposals,
- adjusts schedule, budget, resources, and technology to accommodate approved changes
- schedules and tracks changes to completion,
- analyzes change trends and responds as appropriate, and
- has the authority *and the will* to accomplish these tasks.

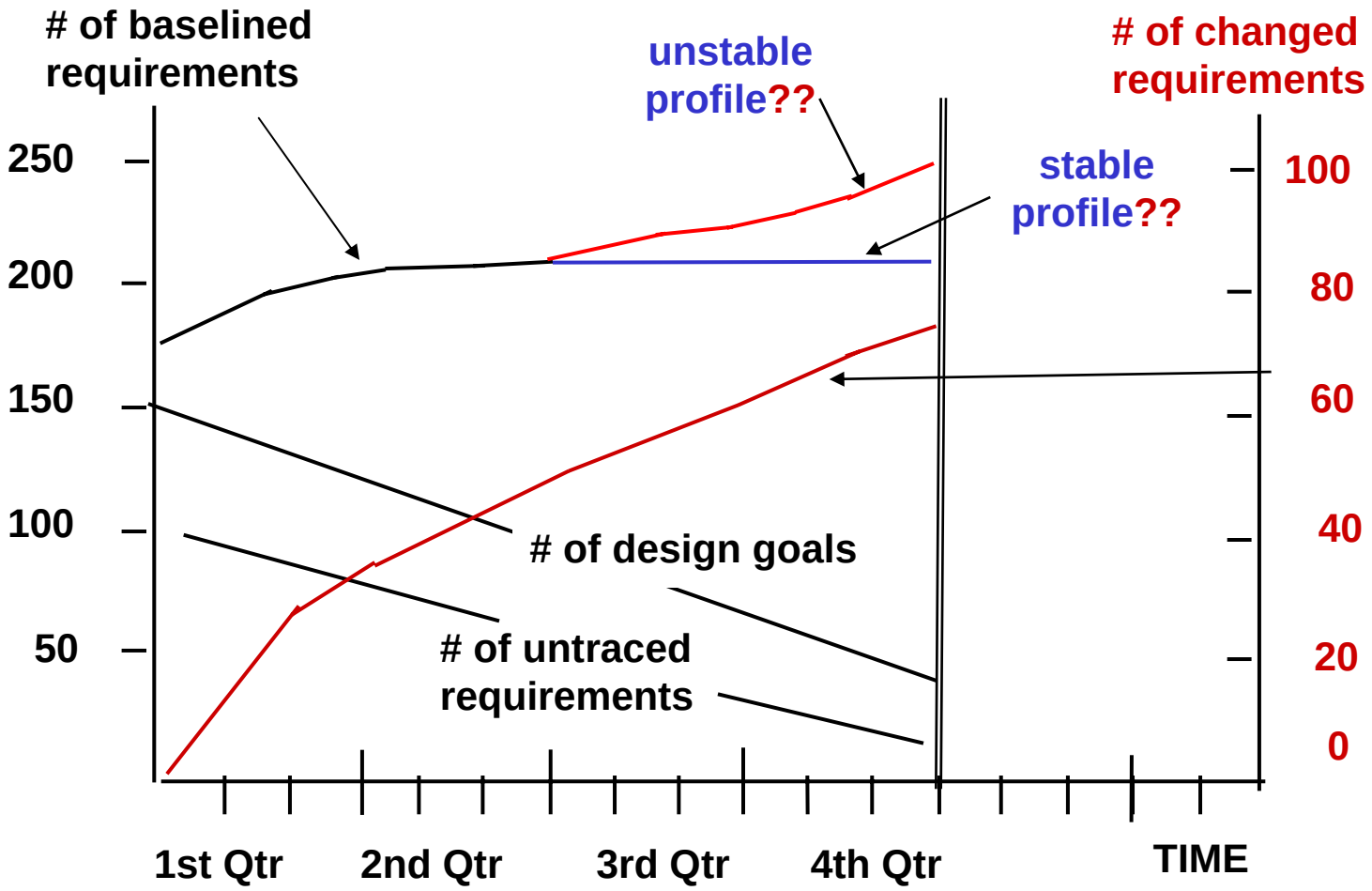
Version control, change control boards, and trend analysis are presented in the context of requirements management; however, the same processes apply to configuration management of all baselined work products.

CCBs

- A Change Control Board (CCB) includes those individuals who have the authority and the will to:
  - accept,
  - deny, or
  - defer a CR or DR
- Small, simple projects have small CCBs
- Large, complex projects have large CCBs

the importance of version control and CCBs is not limited to requirements management

A Trend Report for Baselined Requirements



Foundation Elements for Software Projects

Product Foundations	Explanation
Operational requirements	External, user view of the system
System requirements and architecture	Hardware, software, people elements
Design constraints	Predetermined design decisions
Software requirements	Internal technical specifications
Process Foundations	
Workflow model	Managerial work activities and work products
Development model	Technical work activities and work products
Contractual agreement	Statement of understanding between supplier and acquirer
Project plan	The project roadmap

Documenting Contractual Agreements

A contractual agreement for a project should specify:

- scope of work to be performed
- deliverable work products
- delivery date(s)
- customer-user-developer review schedule
- change request procedures
- design constraints
- development constraints
- product acceptance criteria
- training schedule (as appropriate)
- price and payment schedule (as appropriate)

Contractual Documents

- Formal contractual models are called Statements of Work (SOW)
  - they are legally binding documents
  - they are typically used with customers (acquireres) external to an organization
- Informal contracts are called Memos of Understanding (MOUs)
  - they are also call project authorizations, project charters, and project work orders
  - they are typically used for projects internal to an organization

Foundation Elements

- Chapter 3 of the text covers:
  - operational requirements
  - system requirements and design
  - design constraints,
  - software requirements,
  - contractual agreements
- Other foundation elements:
  - the workflow model for software projects is presented in Chapter 1
  - selecting and tailoring of software development models are presented in Chapter 2
  - project planning and the format and contents of project plans are presented in Chapter 4

The Main Points of Chapter 3 (1)

- Software projects have four kinds of product foundations and four kinds of process foundations
- Software requirements include operational requirements and technical specifications
- Operational requirements include operational features, quality attributes, design constraints
- Technical specifications include primary requirements, derived requirements, design goals, and quantified design constraints
- Design goals are operational requirements that have not been, or cannot be, translated into technical specifications
- The primary activities of requirements development are elicitation of operational requirements, analysis, translation into technical specifications, and acceptance baselining of the technical specifications
- Requirements management is concerned with baseline control of requirements and with maintaining consistency among requirements, effort, schedule, budget, and technology

The Main Points of Chapter 3 (2)

- Configuration management, which includes version control, a change control board (CCB), and status reporting is the primary mechanism of requirements management
- Technical specifications can be categorized in various ways, including interfaces, functions, performance, behavior, capacities, design constraints, and quality attributes
- A contractual agreement includes items such as the scope of work to be accomplished, deliverable work products, development constraints, and product acceptance criteria
- Every project must have a contractual agreement. Informal contractual agreements are called Memos of Understanding (MOUs). Formal contractual agreements are called Statements of Work (SOWs)
- SEI, ISO, IEEE, and PMI provide frameworks, standards, and guidelines relevant to establishing project foundations (see Appendix 3A to Chapter 3)