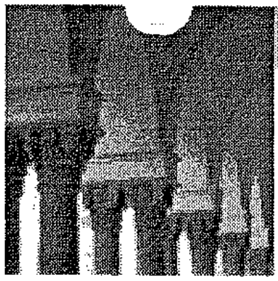
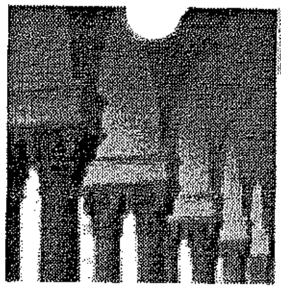


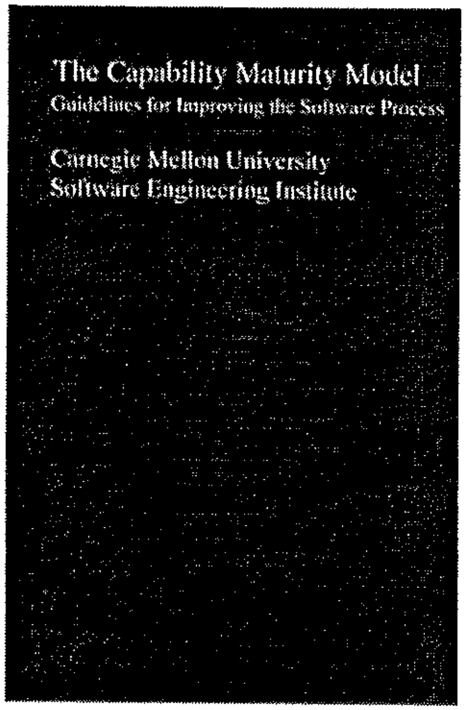
Objectives

- Provide an overview of software process standardization initiatives
- Demonstrate the importance of a software process
- Explain the "generic" software project lifecycle, including the common activities performed in each phase

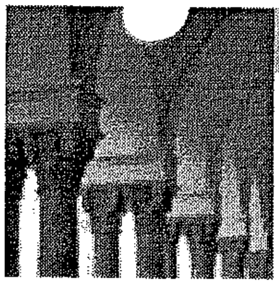


What is the Capability Maturity Model?

- A framework that describes the key elements of an effective software process
- Covers practices for planning, engineering and managing software development and maintenance
- Developed by the Software Engineering Institute of Carnegie Mellon University
  - Provided a response to the needs of the US Defense Department for better techniques for the selection of contractors

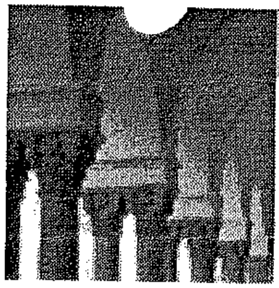

The Capability Maturity Model
Guidelines for Improving the Software Process
Carnegie Mellon University
Software Engineering Institute

SEI, The Capability Maturity Model, pg 4



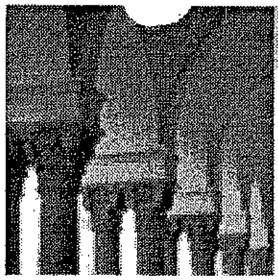
Immature SW organization

- Processes improvised by developers and management
- Organization is reactionary and management focused on solving the immediate crises
- Schedules and budgets exceeded because they are not based on realistic estimates
- Product functionality and quality compromised when hard deadlines are imposed
- No objective basis for judging product quality or solving product/process problems
- Reviews and testing often curtailed/eliminated when projects fall behind



Mature SW organization

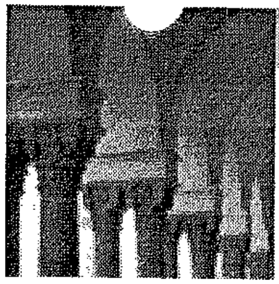
- Communicates software process to existing and new personnel
- Processes are documented, usable and consistent with the way work actually gets done
- Management monitors quality of the product and the processes used to produce them
- Schedules and budgets based upon historical performance
- Expected schedule, cost, functionality and quality are usually achieved



Fundamental Concepts

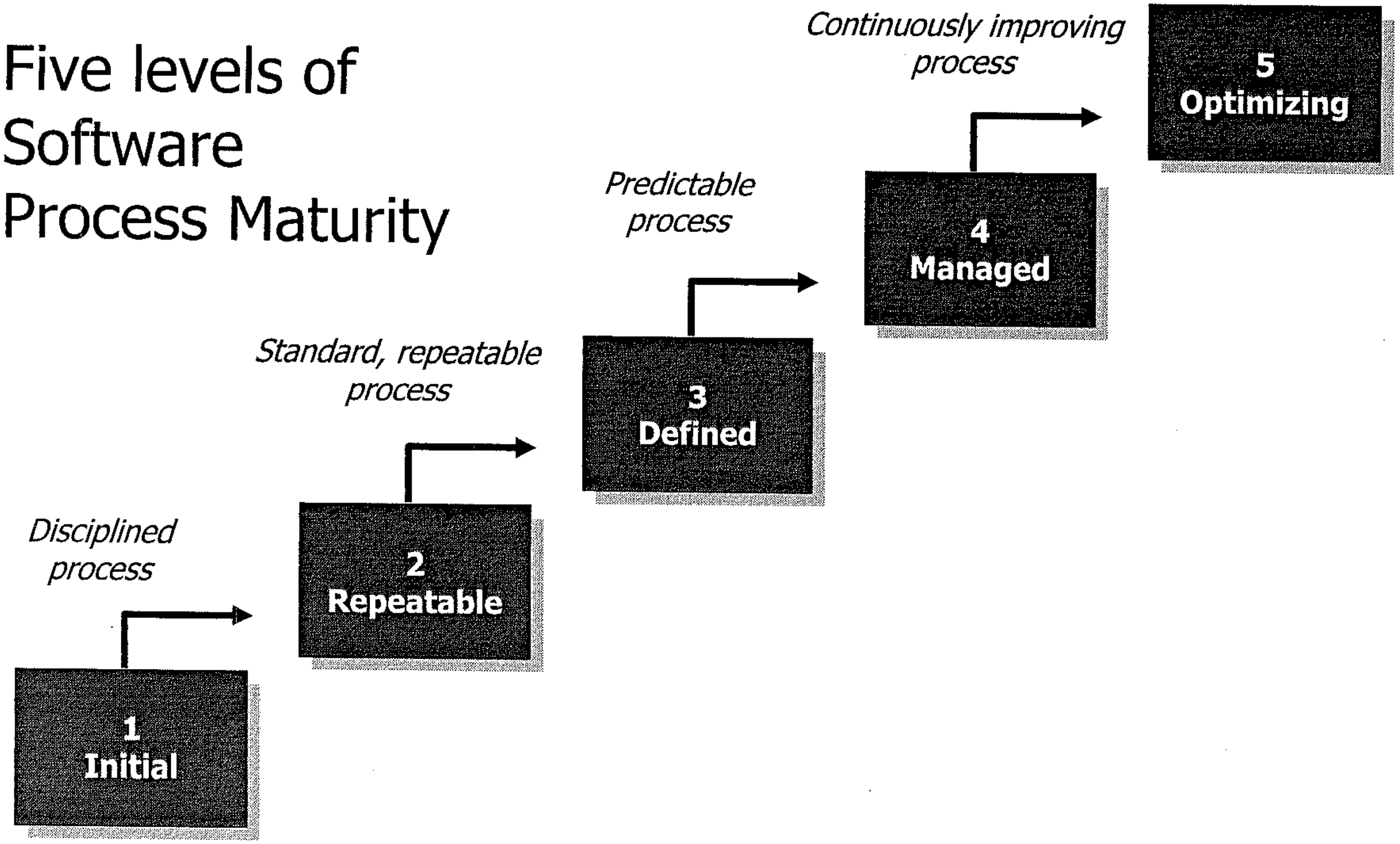
- Process – a sequence of steps performed for a given purpose. Process integrates people, tools, and procedures to produce a product that is valuable to a customer.

- Software Process – set of activities, methods, practices that people employ to develop and maintain software and the associated products (e.g. project plan, code, test plans, user manuals, etc.)



Evolutionary Process

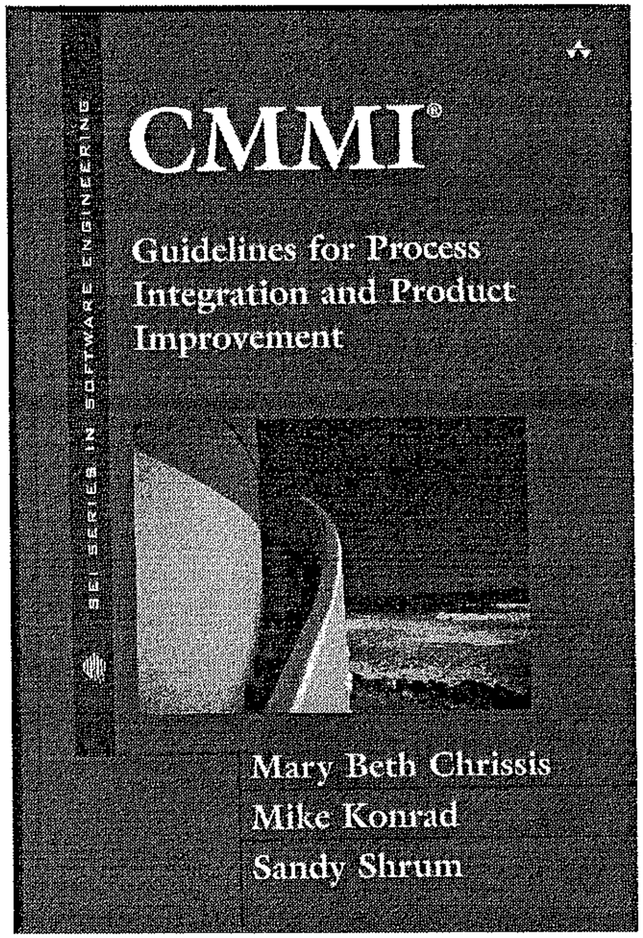
Five levels of Software Process Maturity

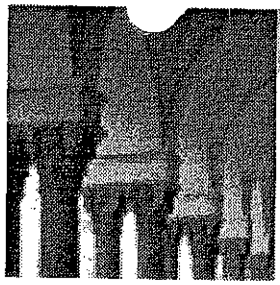


SEI, The Capability Maturity Model, pg 16

What is the CMMI?

The **CMMI** (Capability Maturity Model Integration) combines best practices that address the development and maintenance of products and services covering the product life cycle from conception through delivery and maintenance

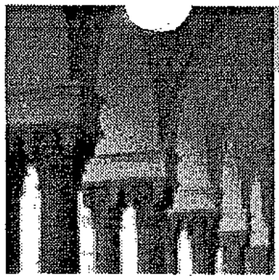

SEI SERIES IN SOFTWARE ENGINEERING
CMMI®
Guidelines for Process Integration and Product Improvement
Mary Beth Chrissis
Mike Konrad
Sandy Shrum



CMMI

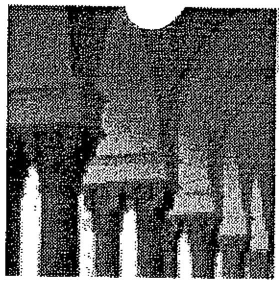
Four bodies of knowledge are incorporated into the CMMI:

- System Engineering (SECM)
- Software Engineering (SW-CMM)
- Integrated product and process development (IPD-CMM)
- Supplier Sourcing



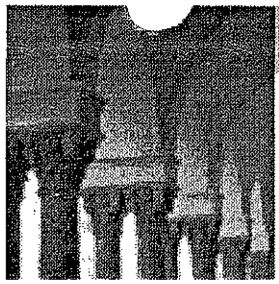
CMMI: Understanding Levels

- Levels are used to describe a path for an organization to improve the processes it uses to develop and maintain products
- Two possible improvement paths:
  - Continuous representation or **capability** level
  - Staged representation or **maturity** level



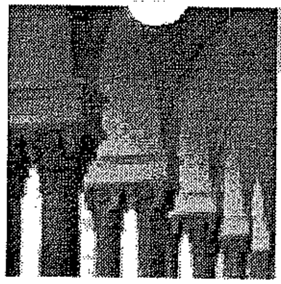
CMMI Process Areas

- Process Management
  - Fundamental
  - Progressive
- Project Management
  - Fundamental
  - Progressive
- Engineering
- Support
  - Fundamental
  - Progressive



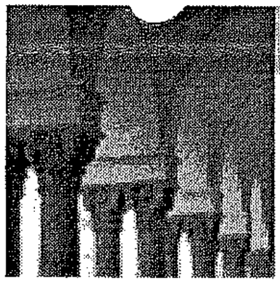
CMMI Capability Levels

- Level 0 – Incomplete
- Level 1 – Performed
- Level 2 – Managed
- Level 3 – Defined
- Level 4 – Quantitatively managed
- Level 5 – Optimizing



CMMI Maturity Levels

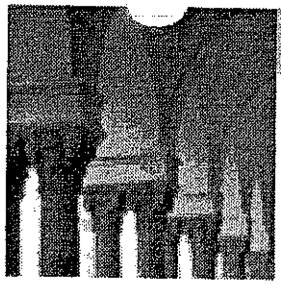
- Level 1 – Initial
- Level 2 – Managed
- Level 3 – Defined
- Level 4 – Quantitatively managed
- Level 5 – Optimizing



What is SPICE?

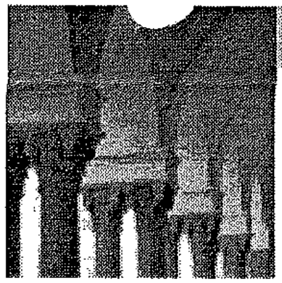
- SPICE is a major international initiative to support the development of an International Standard for Software Process Assessment

- **S**oftware
  Process
  Improvement and
  Capability
  d**E**termination



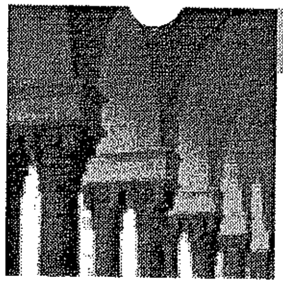
SPICE

- The project has three principal goals
  - Develop a working draft for a standard for software process assessment
  - Conduct industry trials of the emerging standard
  - Promote the technology transfer of software process assessment into the software industry world-wide

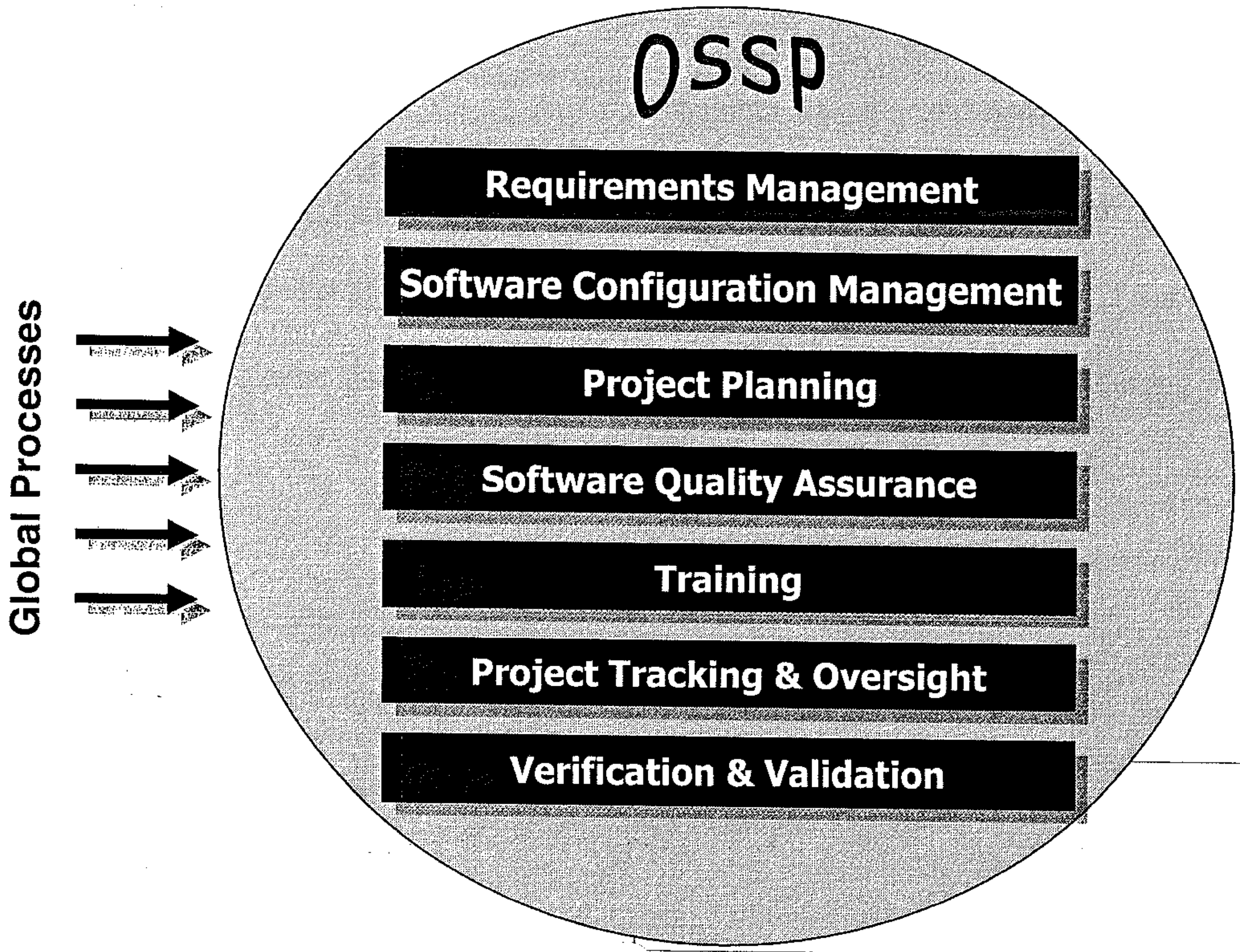


Organization's Standard Software Process (OSSP)

- Drives the software lifecycle
- Defines the software development and maintenance process
- Defines global processes which are fundamental to the performance of all software development and maintenance processes
- Assumes policies, procedures and standards have been developed, maintained and are in use

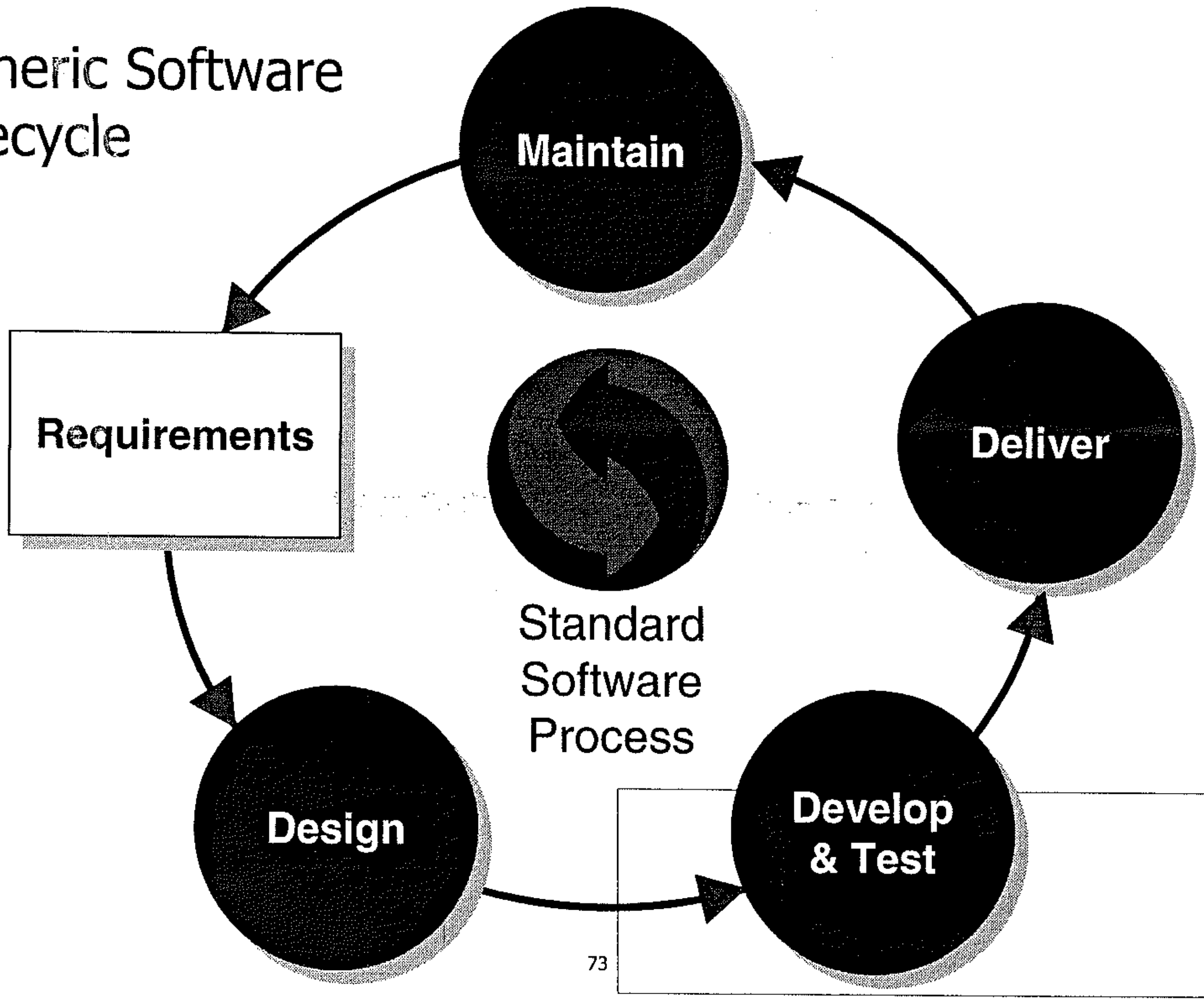


Organization's Standard Software Process (OSSP)

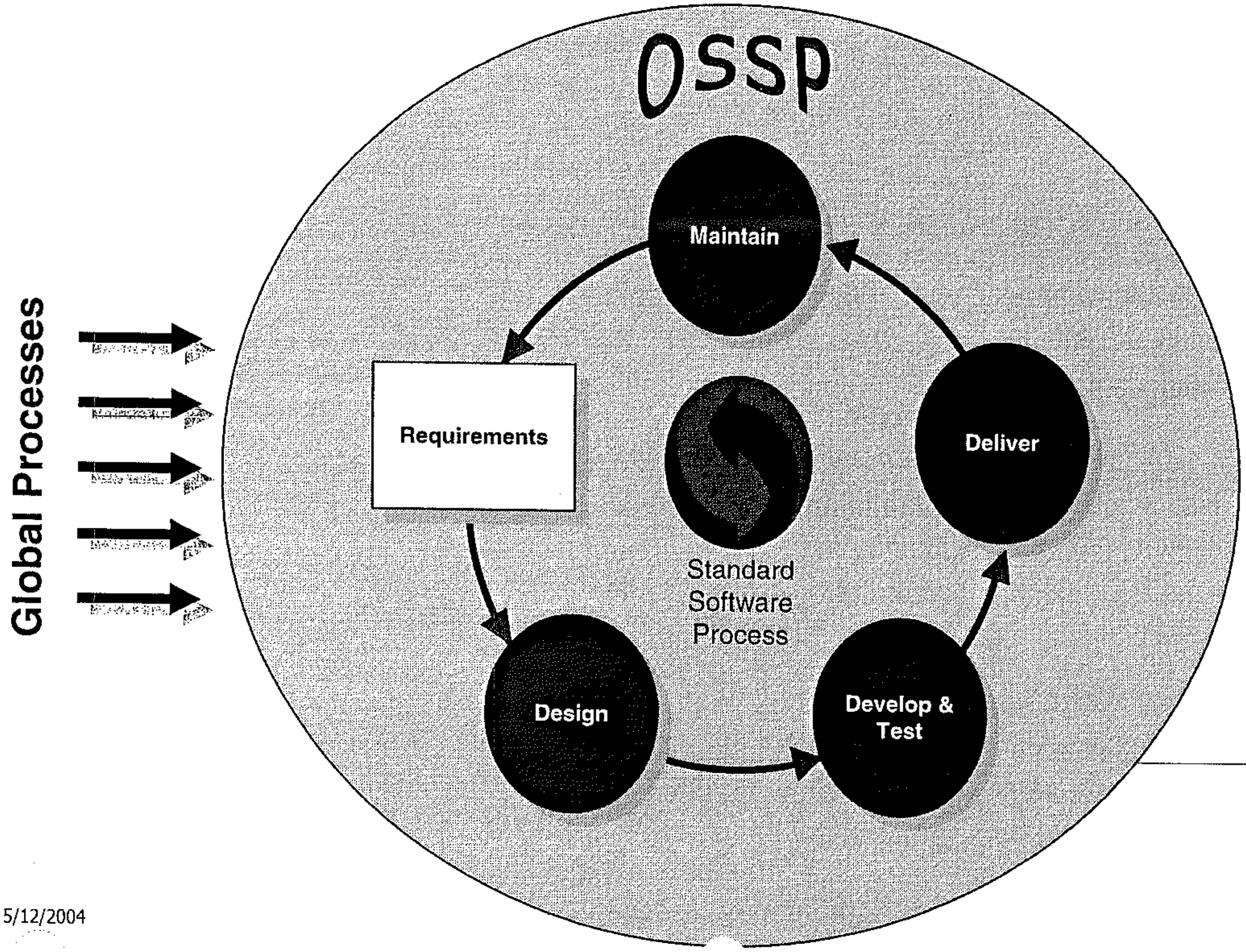

OSSP
Requirements Management
Software Configuration Management
Project Planning
Software Quality Assurance
Training
Project Tracking & Oversight
Verification & Validation
Global Processes

Software Project Phases

Generic Software Lifecycle

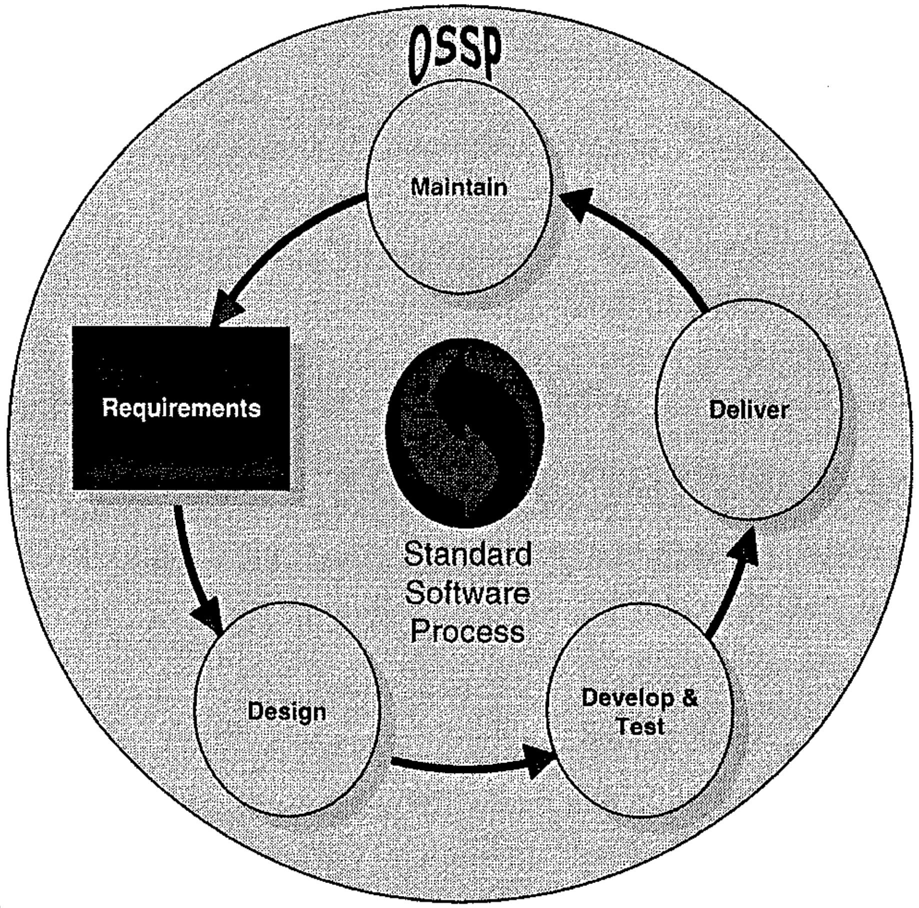


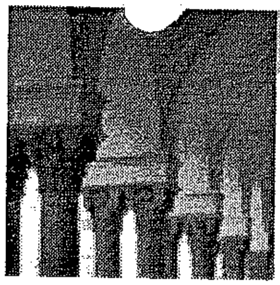
Phases and Processes



Requirements

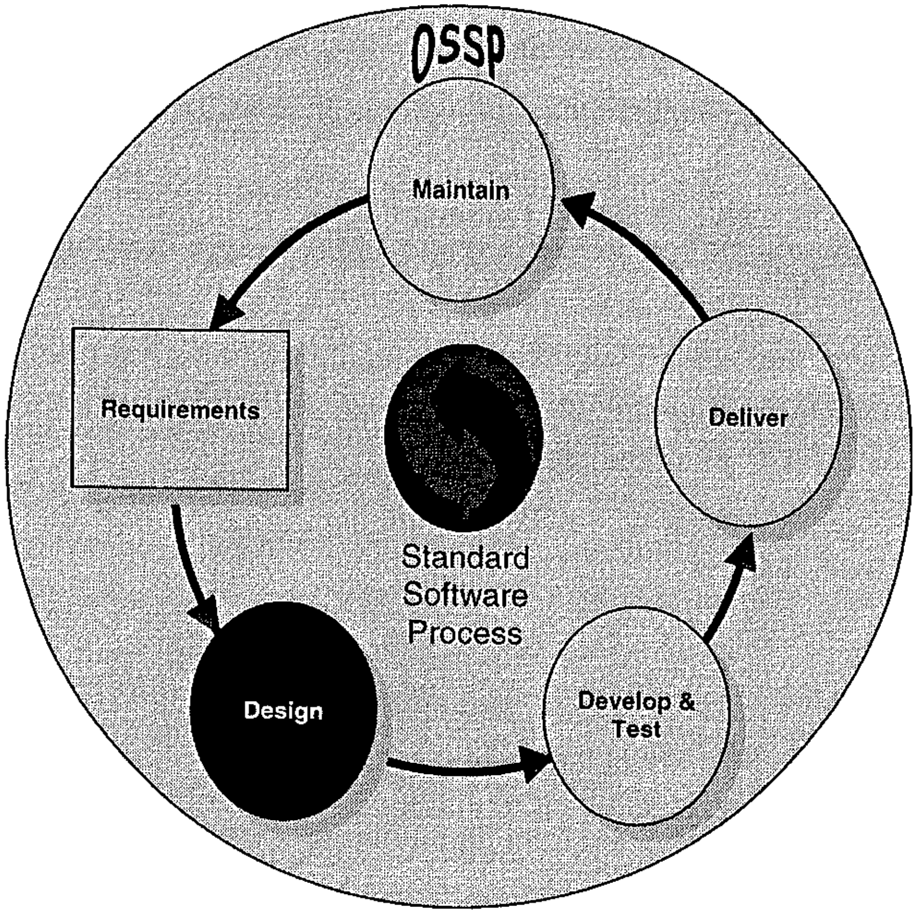
- Objective
  - Work with the customer to define the required features and functions that the software must perform
  - Document requirements to the level of detail required to support system design
  - Determine system requirements allocated to software
- Traps
  - Avoid rushing to judgment - not all requirements are allocated to software





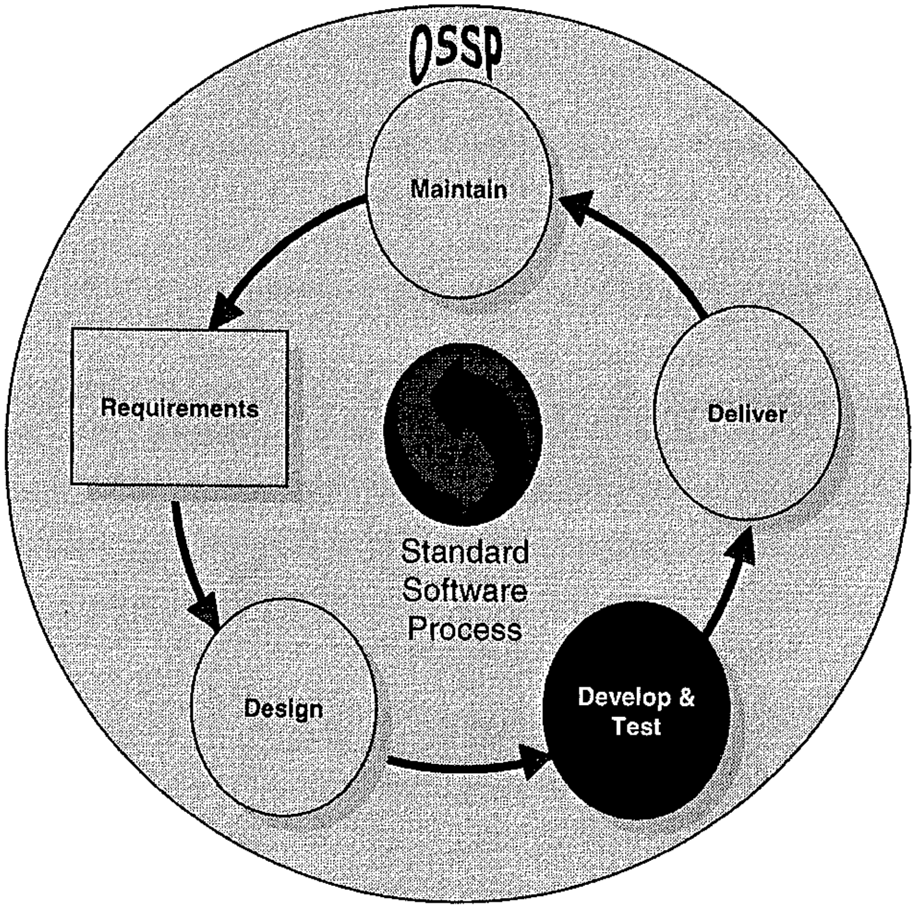
Design

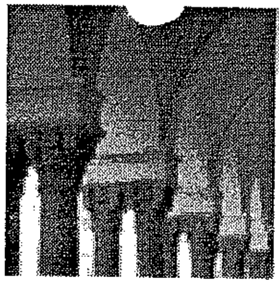
- Objective
  - Produce a detailed specification of the application being developed
  - Translate requirements into concepts that the developer can program
- Traps
  - Project Manager may feel lost
  - Technical staff may get off-track
  - Very programmer/system architect intensive



Develop & Test

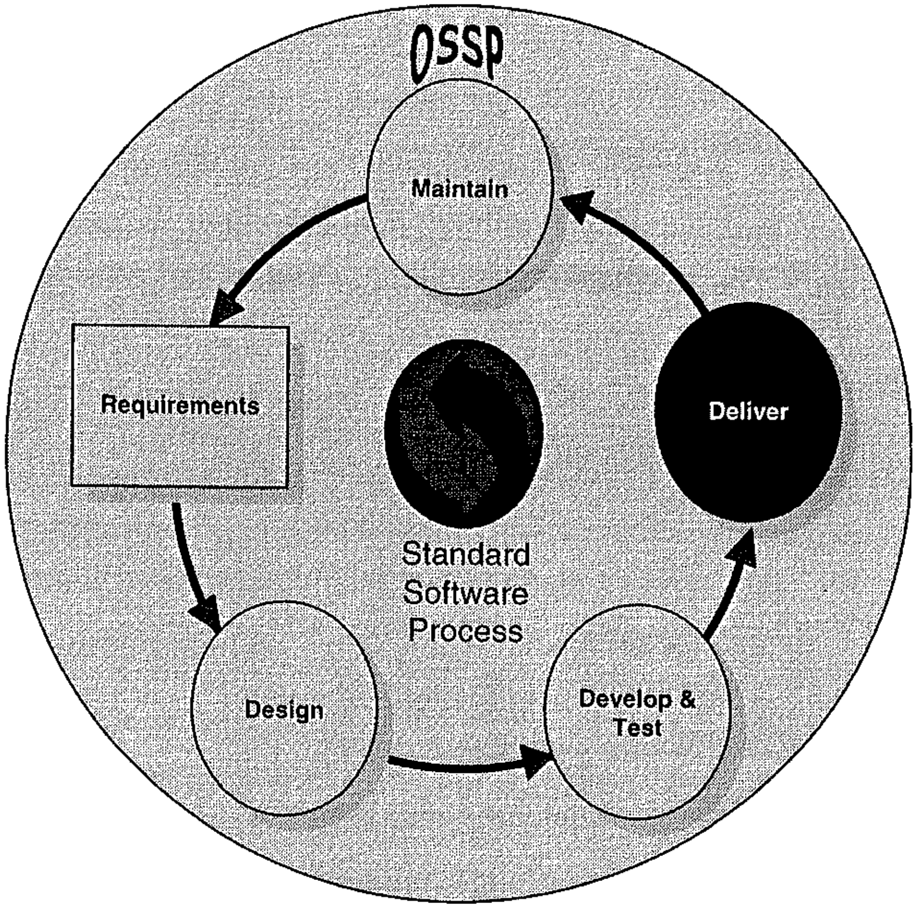
- Objective
  - Develop the code according to the design specification
  - Perform unit testing
  - Perform integration testing
  - Perform system testing
- Traps
  - Avoiding requirements traceability





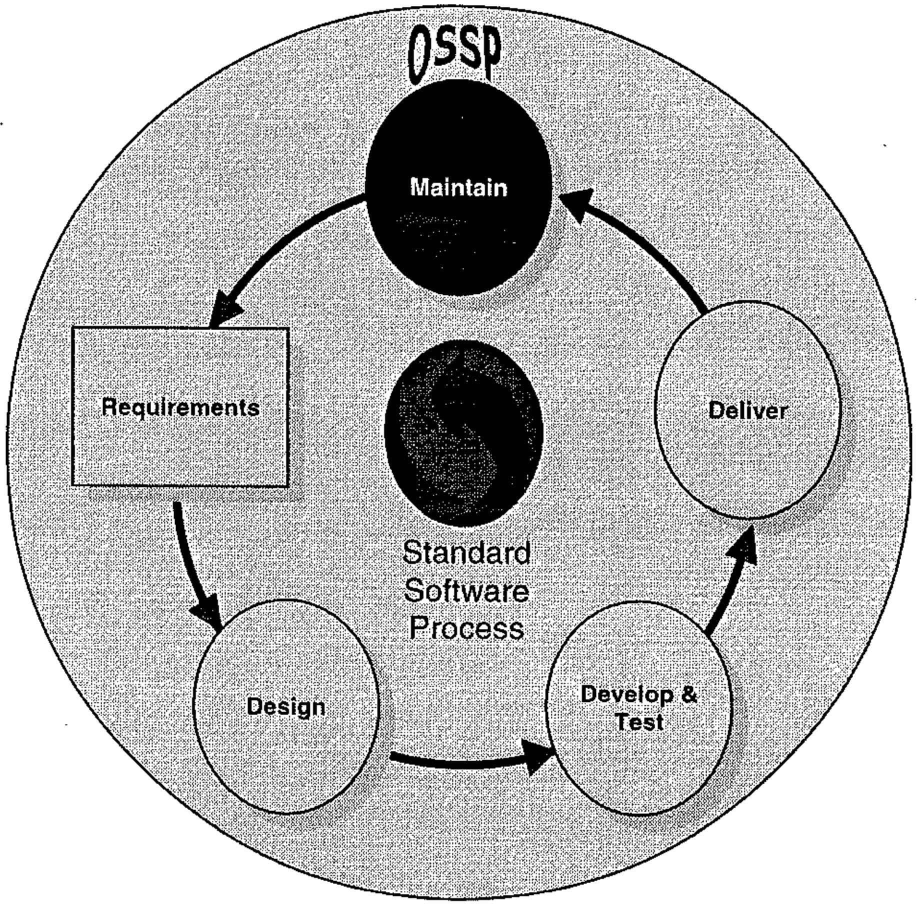
Deliver

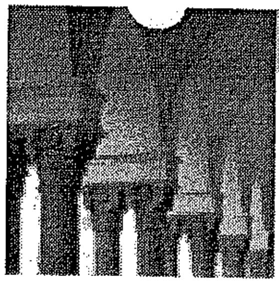
- Objective
  - Install the software
  - Implement the system
  - Deliver user manuals
  - Perform training
  - Provide technical assistance
- Traps
  - Skimping on training and documentation
  - Not planning for support



Maintain

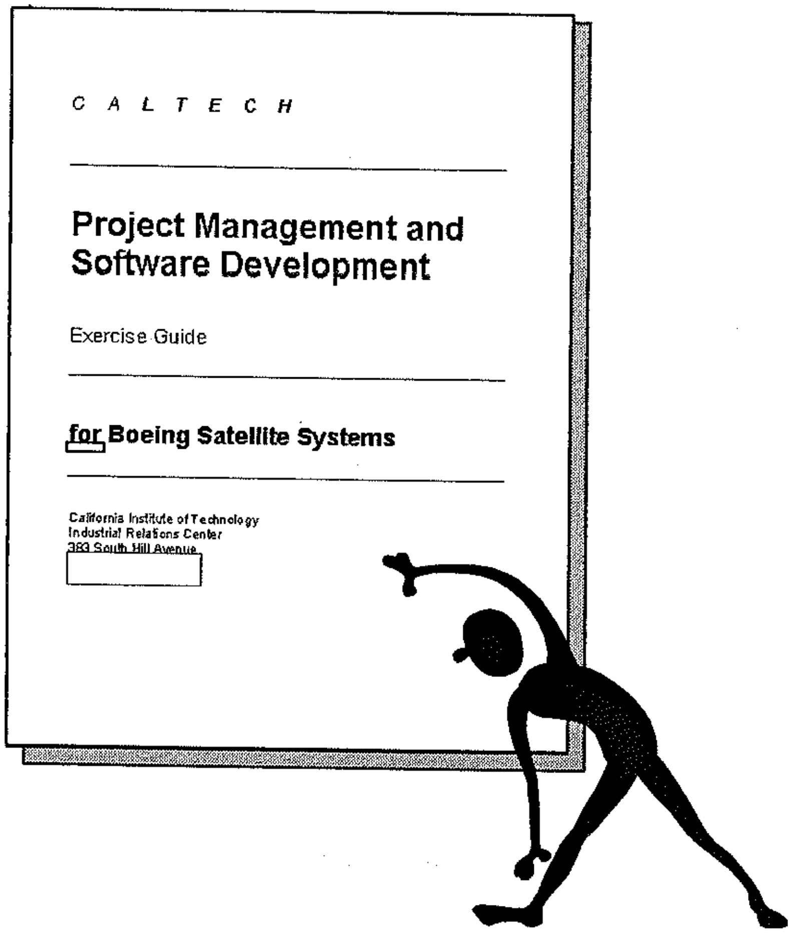
- Objective
  - Provide ongoing support
  - Obtain feedback
  - Evaluate change requests
  - Publish release schedules
- Traps
  - Failing to plan for support resources





Exercise

- 3.1 – Your Role in the Software Lifecycle



C A L T E C H

Project Management and Software Development

Exercise Guide

for Boeing Satellite Systems

California Institute of Technology
Industrial Relations Center
383 South Hill Avenue