ILLINOIS TECH | College of Computing

ITMD 536 Software Testing & Maintenance

09 Adequate Transition Establishing Infrastructure

ILLINOIS TECH | College of Computing

Objectives

- What are the prerequisites for success?
- What do you need for effective maintenance program?
- What happens when a system does not transition?
- When do you replace rather than repair?
- What is best practice?

Objectives

- What are the roles of Capability Maturity Model (CMM) and Capability Maturity Model Integration (CMMi)?
- What is the role of requirements?
- How can you do budgeting and estimating?
- What is release management?
- What is Reuse, Rejuvenation and Resuscitation?

Objectives

- What Event/Request Management KPA?
- What Management Planning?
- What is Monitoring and Control?
- What is SLA and Supplier Agreement Management? (SMM)

ILLINOIS TECH | College of Computing

Objectives

- What is Predelivery and Transition services?
- What is Operational support services?
- What are the Evolution and correction services?
- What is Verification and a Validation? (SMM)

5. Adequate Transition and Turnover Planning

Transition: The process employed to transfer the software and responsibility for its maintenance and support from the developer to the maintainer.

Turnover: The point in the life cycle when the software and responsibility for its maintenance and support are transfer to occur, all terms and conditions spelled out in the transfer agreement must be satisfied (or waived).

Transition from Implementation to Maintenance

- **Transition Plan – Knowledge Transfer**
  - Project specifications
  - Code documentation
  - Assets transfer
  - Development credentials
  - Deployment procedures
  - Any other technical details

Transition from Implementation to Maintenance

- **Steps for project transition**
  - Identify resources for the maintenance
  - Establish a maintenance status meeting with business partners and IT stakeholders
  - Establish production issues and incidents meeting with product owners & the technical team
  - Establish a change control board
  - Communicate the governance model
  - Provide knowledge transfer between project team and maintenance team

Transition from Implementation to Maintenance

- **Tools to manage Transition**
  - Requirement management
  - Bug tracking
  - Test management

Transition from Implementation to Maintenance

- **Transition Governance–steering team**
  - Transition planning
  - Formulate the transition start date
  - Approve any deviations from the transition schedule
  - Approve change requests raised during migration
  - Monitor and keep a track of the progress of the transition
  - Informed the stakeholders about the transition status
  - Address any escalated issues

5.1 Prerequisites for Success

- Planning for transition and turnover of responsibility from development to maintenance shops should be conducted early in the development life cycle.
- Working groups should be formed to engage all organizations involved as requirements are being formulated.
- Process planning provides the process infrastructures used by the workforce to ensure that both products delivered for software maintenance satisfy the standards. (versions, naming conventions used for configuration baseline and sequences.)

5.1 Prerequisites for Success

- Human resource planning puts the skilled people in place when they are needed during and after the transition.
- Project planning provides the entry/exit checklists required to ensure the developers deliver. (also verifies equipment, facilities, tools and people are ready and trained)

5.2 What you need to Execute an Effective Maintenance Program

- Tools like checklists help because they act as reminders of things that you should not forget to do.
- Track events and follow-up during the move to make sure things go as planned.
- Get ready for transition and turn the software maintenance responsibility and to execute the plans.

5.2.1 During Development

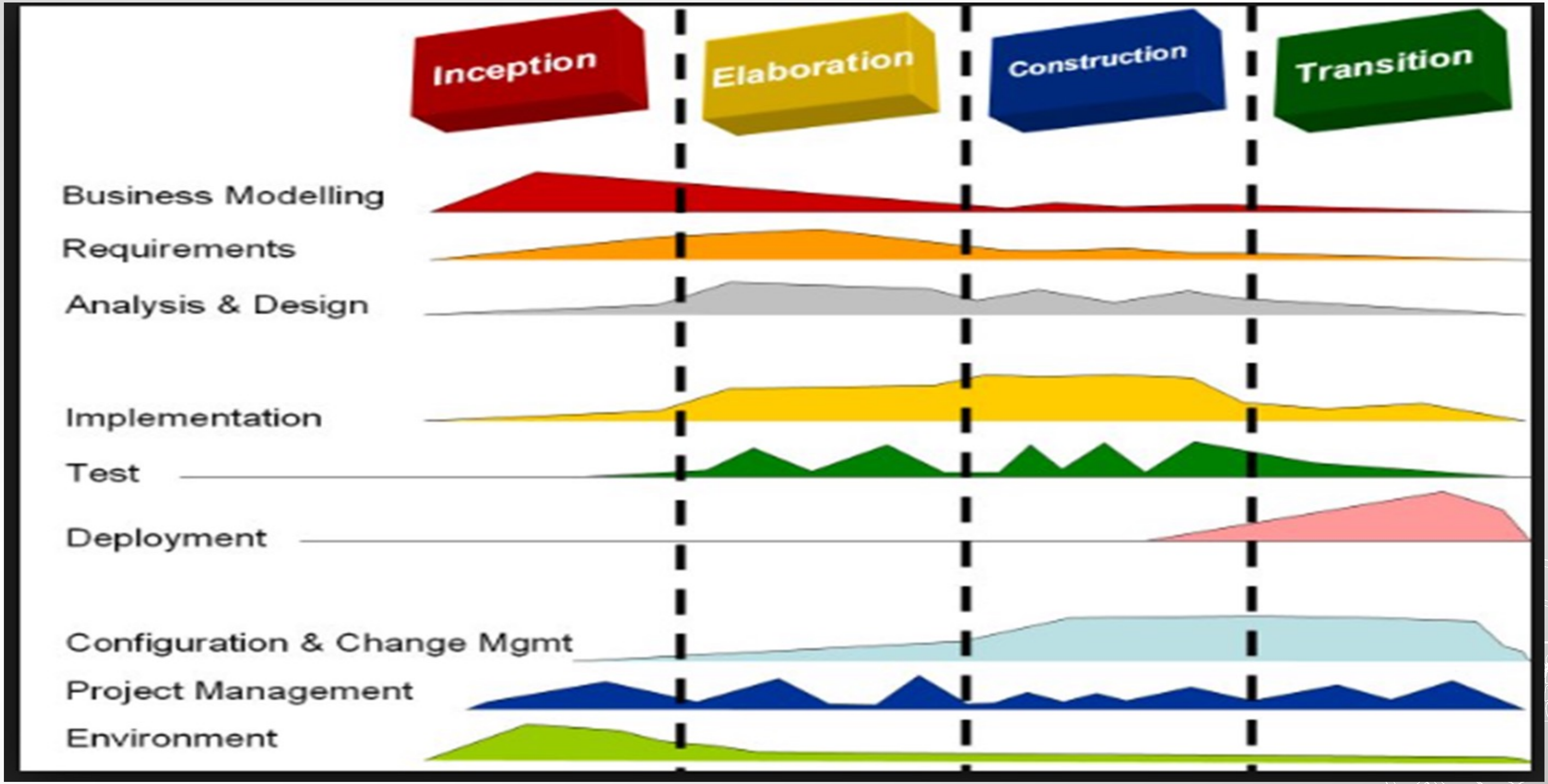
- The primary focus is almost always on development, and maintenance is a secondary consideration at best.
- Regression test baseline and set of tests revalidates the software once it is changed and this makes maintenance job easier.

5.2.1.1 Product

- Well structured architecture is software’s building block, it can be added, modified, and removed with the little impact on overall functionality and performance.
- Techniques like object-oriented design, systematic reuse, and product line architectures preserve structure and help maintain the integrity of the product, its components and its databases.

5.2.1.2 Process

- IBM Rational Unified Process (RUP) where transition is identified and formal stage of the development life cycle.
- Other life-cycle models, both waterfall and incremental, have delivery of software to systems testing and acceptance testing as their last phase.


Inception
Elaboration
Construction
Transition
Business Modelling
Requirements
Analysis & Design
Implementation
Test
Deployment
Configuration & Change Mgmt
Project Management
Environment

5.2.1.2 Process

- Deploy plan
- Develop support materials
- Manage acceptance test
- Enable product deployment unit
- Manage acceptance test for custom installation
- Package product
- Provide access to customer site
- Calling out the transition phase with distinct processes and a culminating product release milestone provide a clear path for an orderly transition and turnover to maintenance.

5.2.1.3 People

- To succeed you need a skilled, knowledgeable, able, experienced, and motivated workforce is needed to use these processes and architecture to perform the maintenance work.
- Maintenance workforces are typically better skilled and more experienced than their development counterparts.

ILLINOIS TECH | College of Computing

5.2.1.4 Project

- The software maintenance plan is the primary document that needs to be generated. It is a blueprint for action as you transition responsibility to manage the generation of future software releases and versions from one group to another.
- This document provides strategies, responsibilities, and plans for both transition and maintenance.

5.2.1.4 Project

- Project also identifies the resources and schedules needed for critical tasks and events to succeed.
- It also calls out milestones like the operational readiness review and establishes approaches for dealing with contingencies should bad things happen and delays occur.

Maintenance Plan Outline

1. Introduction
2. Scope
    - 2.1 Product
    - 2.2 Documentation
    - 2.3 Relationships to Other Agencies/Projects
3. Strategies
    - 3.1 Transition and Transfer
    - 3.2 Operational

Maintenance Plan Outline

4. Transition Schedules, Tasks, and Activities
    - 4.1 Installation
    - 4.2 Initial Operations
    - 4.3 Training
    - 4.4 Conversion and Data Migration
    - 4.5 Transition Schedule
    - 4.6 Outstanding Issues

ILLINOIS TECH | College of Computing

Maintenance Plan Outline

5. Maintenance Schedules, Tasks and Activities
    - 5.1 Release Process
    - 5.2 Performance Measures and Reporting
    - 5.3 Governance and Management Approach
    - 5.4 Problem Resolution
    - 5.5 Documentation Strategies
    - 5.6 Training
    - 5.7 Sustainability Measures
    - 5.8 Maintenance Schedule
    - 5.9 Outstanding Issues

ILLINOIS TECH | College of Computing

Maintenance Plan Outline

6. Resource Requirements
    - 6.1 Software Resources
    - 6.2 Hardware Resources
    - 6.3 Facilities
    - 6.4 Personnel
    - 6.5 Other
7. Acceptance Criteria
8. Management Controls

ILLINOIS TECH | College of Computing

Maintenance Plan Outline

9. Reporting Procedures
10. Risks and Contingencies
11. Team Information
12. Review Process
13. Configuration and Distribution Control
14. Plan Approval

Software Operational Readiness Checklist

- The installation has been coordinated with the system owner, operations staff, support staff, and other affected organization.
- All necessary modifications to the physical installation environment are complete.
- The hardware has been inventoried and tested.
- User training has been completed.

ILLINOIS TECH | College of Computing

Software Operational Readiness Checklist

- Software has been installed on the hardware and acceptance testing has been successful repeated to your staff.
- A copy of installation tests has been placed in the project file.
- All acceptance tests have been coordinated with the system owner, users, operations staff, support staff, and other affected organizations.

Software Operational Readiness Checklist

- The test environment and regression test baseline have been placed under configuration management.
- All tests have been executed correctly.
- Any tests that failed have been documented, corrected, and retested.
- A copy of all acceptance test materials have been conducted and a physical configuration audit successfully completed.

Software Operational Readiness Checklist

- Complete operating documentation describing the release has been approved and delivered.
- The software release has transitioned to full operational status and has been transferred to the life-cycle support staff for further maintenance actions.
- All training and certification activities have been successfully completed.

Software Operational Readiness Checklist

- For major software systems involving multiple organization and interfaces with other systems, a formal announcement of transition and transfer has been made.
- A list of open issues and planned enhancements has been provided to the maintenance staff.

Software Operational Readiness Checklist

- Access rules have been modified to provide access to the maintenance staff and to remove the project team from access to the release.
- Software, files and other support software have been placed in the production library and deleted from the test library, as appropriate.
- All files, operating documents, and other pertinent records have been turned over to the maintenance staff.

5.2.2 After Transition and Turnover

- During the first year of operations the maintenance shop tries to stabilize the product and the software environment that it will use to update it.
- Key differences between development and update environment and root cause are listed here:

Development versus Update Environment

1. **Development Environment:**

Emphasis for use is placed on design and development tasks (coding, unit checkout, and test)

- **Update Environment:**

  Emphasis for a use is placed on execution, update, and performance enhancement.

- **Root Cause of Differences**

  Different work gets accomplished during two phases: development implements product and maintenance updates/refines it

ILLINOIS TECH | College of Computing

Development versus Update Environment

2. Development Environment:

Current operational platform (operating system, database managers, etc.)

- **Update Environment:**

Current operational platform plus variants.

- **Root Cause of Differences:**

Multiple platform configurations are maintained for different field sites (different versions of software etc.)

Development versus Update Environment

3. Development Environment:

Pseudo-operational equipment.

- **Update Environment:**

  Actual operational equipment.

- **Root Cause of Differences:**

  Operational equipment may not be available to developers because it is either being developed or needed in the field (limited availability).

Development versus Update Environment

4. Development Environment:

Rich and capable development toolset.

- **Update Environment:**

Not so rich and capable toolset.

- **Root Cause of Differences:**

Tools used during development may be either expensive or proprietary. In addition, tools needed differ between phases because work done differs.

Development versus Update Environment

5. Development Environment:

User representatives.

- **Update Environment:**

  Actual users.

- **Root Cause of Differences:**

  Timing and availability may be such that users are not available during development.

Development versus Update Environment

6. Development Environment:

Field support requirement.

- **Update Environment:**

  Not applicable.

- **Root Cause of Differences:**

  May have to go to field sites in order to make repairs and install updates because users do not have this ability.

Development versus Update Environment

7. Development Environment:

Self-contained configuration.

- **Update Environment:**

  Links to sites worldwide if permissible (security may constrain this option).

- **Root Cause of Differences:**

  Need to download operational configurations automatically and install them in the field.

Development versus Update Environment

8. Development Environment:

Highly available for development and testing.

- **Update Environment:**

  Limited availability during operations.

- **Root Cause of Differences:**

  Due to operational and user demands for raining, the update environment may have limited availability during prime shifts for checkout and testing.

5.2.2 After Transition and Turnover

- Release cycles during software maintenance are frequently tied to business cycles (i.e., annual or biannual).

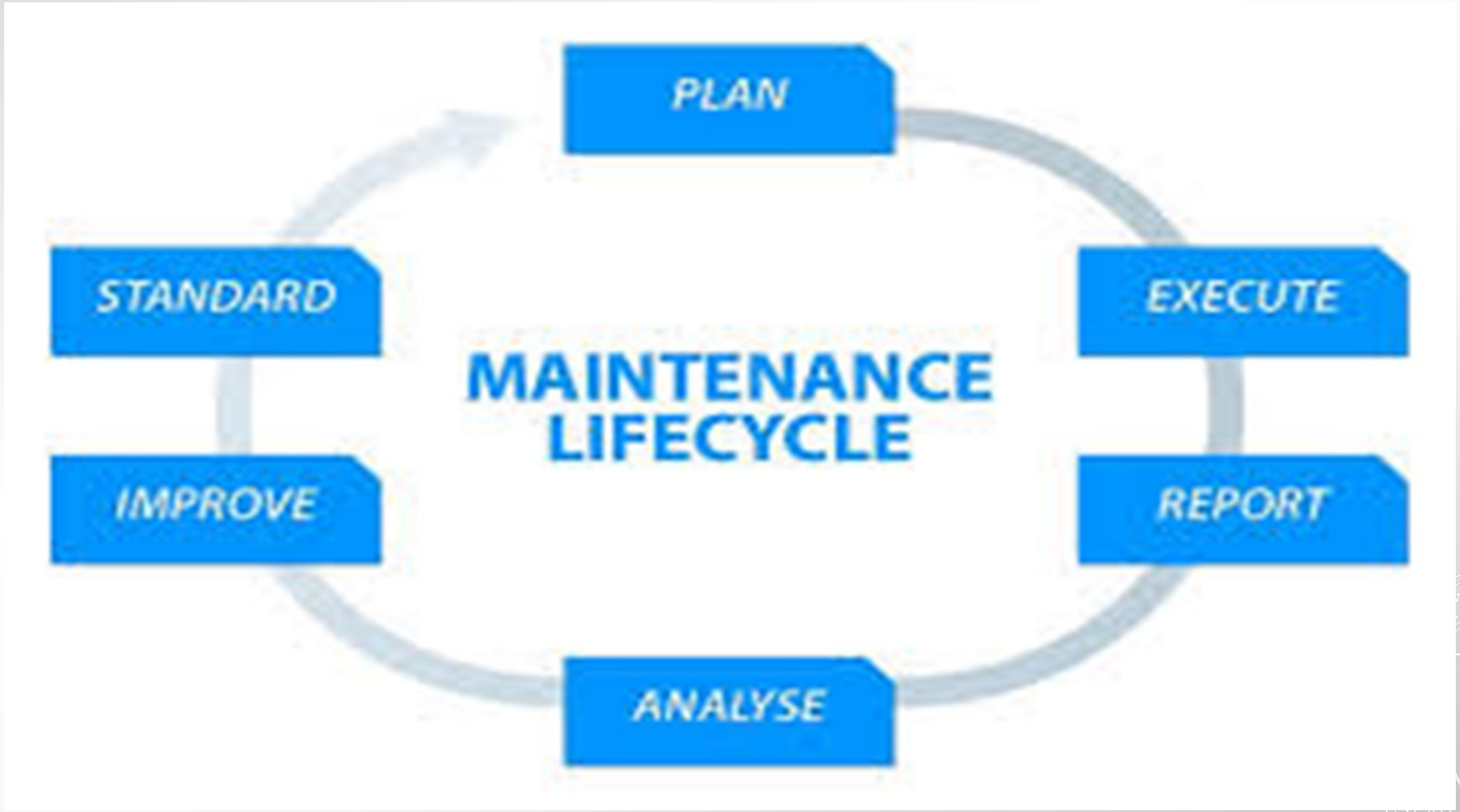
5.3 What Happens When a System Does Not Transition

- Those who develop the software have to maintain it, they often do a better job during development.
- The total life-cycle cost will be less if it is maintained by the same development group.
- There are many reasons for transition to a third party.
- Do not have staff available to work both development and maintenance tasks.
- The software maintenance work is outsourced because rates are cheaper due mainly to lower burden rates.
- Wages in China, India, and other countries compete for maintenance businesses.

5.4 When To Replace Rather Than Repair

- The cost of maintaining obsolete equipment and platforms may be so high that replacement is the only option.
- Replacing hardware may force replacing software.
- Windows 10, XP, Windows Vista etc.
- Retiring software under any circumstance is hard because people are averse to getting rid of anything that they are comfortable with and that works.
- Replacing hardware and software replacements can be viewed as opportunities rather than challenges.

I need a more USER-FRIENDLY computer.
I NEED A SMARTER HUMAN.


PLAN
EXECUTE
REPORT
ANALYSE
IMPROVE
STANDARD
MAINTENANCE LIFECYCLE

5. Event/Request Management Domain

- This process domain covers four KPAs:
  - Event/Request Management
  - Maintenance Planning
  - Maintenance Request/Software Monitoring and Control
  - SLA and Supplier Agreement Management (SMM)

Event/Request Management Domain

1. Event/Request Management:

- This is the entry point, through the help desk, for daily communication with the users.
- The goal of this KPA is to ensure that the software maintenance process/services meet the agreed upon SLA quality/service objectives.
- The maintenance work performed is centered on customer/user priorities.

Expected Results from Event/Request Management KPA

- The interruptions in ongoing work are justified according to the terms of the SLA and agreed on with the customers.
- The maintenance organizational unit meets its levels of service.
- An active monitoring approach, for rapid reactions, is being implemented.
- Information on failure and maintenance activities that affect the customer is quickly circulated.

2. Maintenance Planning

- The maintenance planning KPA helps to plan the allocation of resources to individual requests and to handle rapidly changing priorities.
- It handles requests from:
  - Customers and users
  - Development teams
  - Subcontractors, and
  - Computer operations

Expected Results from Maintenance Planning

- A sound forecast is available on needed resources in software maintenance (strategic plan, tactical plan, and operational plan).
- SLA, agreement, license, and contract renewals are planned.
- The failure/recovery tests are planned for each software in maintenance.

Expected Results from Maintenance Planning

- Requests are allocated to future versions of the software.
- Software upgrades are planned.
- Predelivery and transition services are planned.
- There is an optimal allocation of maintenance resources to work items.
- Software maintenance has a plan, and this plan is communicated and approved.
- The stakeholders are informed about the plans.

Maintenance Request Software Monitoring and Control

- This identifies items of such that need to be controlled and the individual work items assigned to the resources that are currently in progress.
- Maintainers must proactively monitor the gaps in service levels, make needed adjustments, and receive/establish new agreements for delivery or service levels when needed.

Expected Results from Maintenance Request/Software Monitoring and Control

- Maintenance activities are performed according to forecasts from the various levels of maintenance planning.
- Maintenance work items, costs, and respect for commitments are tracked.
- Activities are reviewed and replanned when they progress differently from what was originally planned.
- Responsibility for communication of rectifications, commitments, and promises is known by the resources.
- Reviews are conducted with the customers at main checkpoints to examine the results and plans, and also the progress of work.
- Corrective actions are taken if it is determined that there is a different between results and the forecasting accepted by customers.

SLA and Supplier Agreement Management

- The SLA and Supplier Agreement Management KPA identifies the levels of services and resource requirements, and negotiates prices with customers and service partners.

  A. The number of maintenance services that have to be delivered.

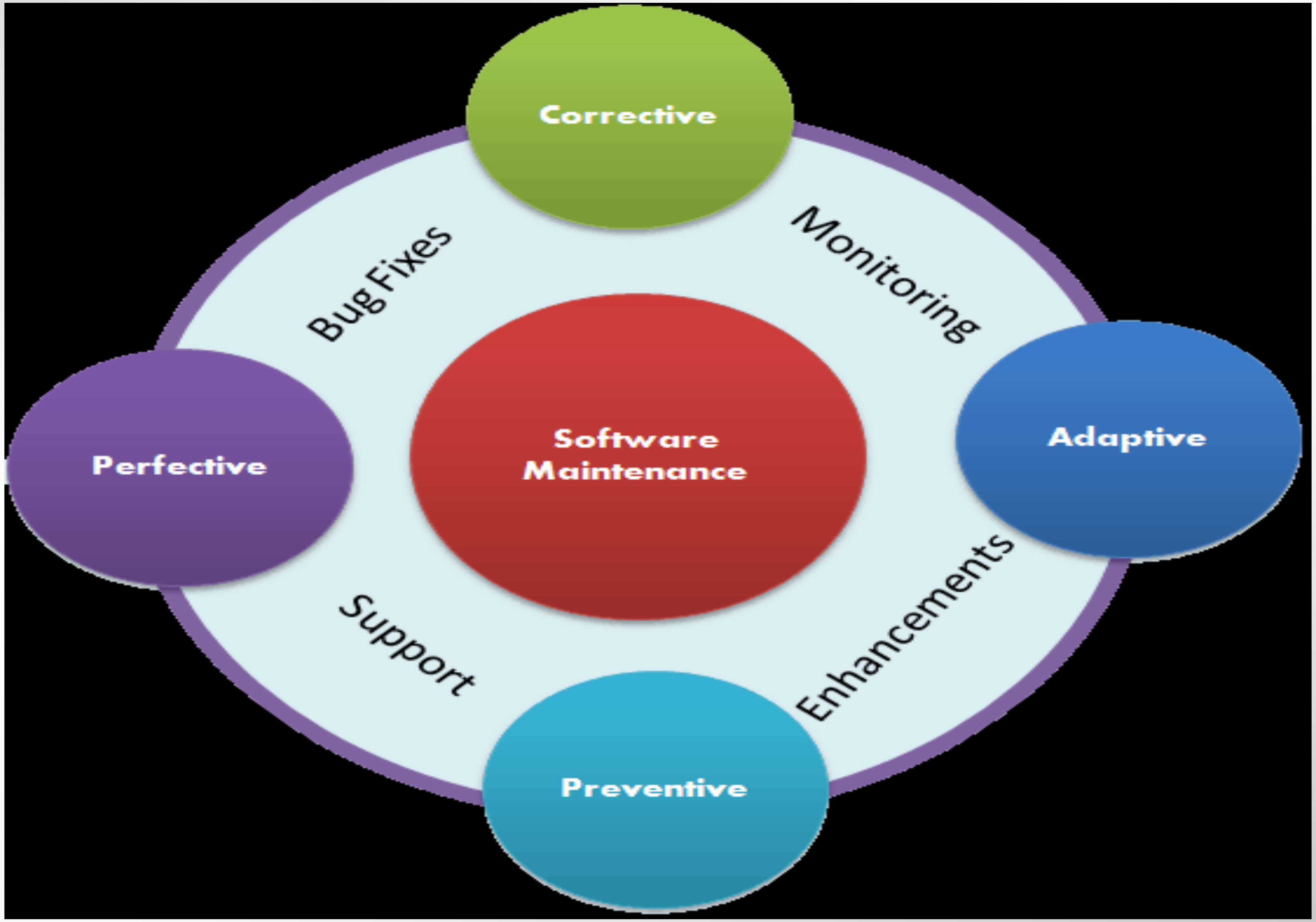
  B. The cost of each service.

  C. The quality and efficiency of the products and services, and

  D. The establishment of standardize measures for performance measurement and reporting customers.

Expected Results from SLA and Supplier Agreement

- The agreements are defined, formalized, and negotiated.
- The agreements and responsibilities are defined.
- The SLAs are expressed in words that are easily understood by the customer.
- A communication channel exists, is used and works to ensure agreements are performing well.


Corrective
Monitoring
Adaptive
Enhancements
Preventive
Support
Perfective
Bug Fixes
Software Maintenance

6 Establishing a Solid Management Infrastructure

Management: Getting things done through the work of other people.

Infrastructure: Underlying framework of an organization or system, including organizational structures, policies, standards, training, facilities, and tools, which supports its ongoing performance.

6.1 Best Practices

- Maintenance Characteristics:
  - They invest in standard processes for work tasks that every project, large or small, has to perform.
  - They provide the needed amplifying procedures, practices, instructions, guidelines, and work-related aides to enact the processes.
  - Senior management in these shops commits to using these processes and acts as champions for their use and improvement.
  - Those doing the work in these shops approve of these processes and are trained to use them prior to their enactment.

6.1 Best Practices

- Processes are enforced, but not in an overbearing and capacious manner (e.g., the process police come after you).
- Process is viewed as important, but so were product and people. All the three receive the attention they deserve, as does the implementation of tried and true project management principles and techniques.

6.2. Role of Capability Maturity Model (CMM) and Capability Maturity Model Integration (CMMi)

- ISO 9000 and CMMi can help simplify the job of software maintenance.
- The CMMi includes three constellations, in recognition of the need for different practices to meet the needs of organizations development, services and acquisition.

ILLINOIS TECH | College of Computing

Best Maintenance Practices by Key Process Area

1. **Category:** Engineering
   1. **Process Area:** *Product Integration*
   - **Best Practices for Software Maintenance:**
   - Practice 1: Build sustainable release during development by focusing on evolutionary architectures and design.

   2. **Process Area:** *Requirement Development*
   - **Best Practices for Software Maintenance:**
   - Practice 2: Generate requirements for maintenance release using some form of engineering change requests or trouble reports as your basis, not specifications.

ILLINOIS TECH | College of Computing

Best Maintenance Practices by Key Process Area

3. Process Area: *Technical Solution*

- **Best Practices for Software Maintenance:**
- Practice 3: Capture implementation knowledge as new maintenance releases are developed, qualified, and fielded.
- Practice 4: Reengineering poor-quality and poorly performing portions of the software as new maintenance releases are developed, qualified and fielded.

Best Maintenance Practices by Key Process Area

4. Process Area: *Validation*

- **Best Practices for Software Maintenance:**
- Practice 5: Conduct an operational readiness review and physical configuration audit to ensure that you can generate the release in the maintenance facility and it will work as intended in its operational environment.

5. Process Area: *Verification*

- **Best Practices for Software Maintenance:**
- Practice 6: Conduct an operational acceptance review and audit on-site in the field to ensure that the release satisfies its requirements and satisfies customer expectations.

Best Maintenance Practices by Key Process Area

6. **Process Area:** *Emergency Solution*

- **Best Practices for Software Maintenance:**
- Practice 7: Develop an emergency procedure under which solutions can be developed, tested, and distributed to the field quickly while preserving their configuration integrity.
- Practice 8: Develop an acceptable procedure for installing and managing patches made in the field to implement emergency repairs.

7. **Process Area:** Test Management

- **Best Practices for Software Maintenance:**
- Practice 9: Develop a regression test baseline for use in revalidating the release once changes have been made to it.

Best Maintenance Practices by Key Process Area

2. **Category:** Project Management
   1. **Process Area:** *Integrated Product Management*

- **Best Practices for Software Maintenance:**
- Practice 10: Establish a working group to engage all stakeholders, including the maintainer, customer, and user, in developmental decisions impacting operations and maintenance of the software in the field.

Best Maintenance Practices by Key Process Area

2. Process Area: *Project Monitoring and Control*

- **Best Practices for Software Maintenance:**
- Practice 11: Establish procedures to monitor technical work being done on the release so that corrective actions can be taken when discovered in a timely manner.

3. Process Area: *Project Planning*

- **Best Practices for Software Maintenance:**
- Practice 12: Ensure that project plans address transition and turnover of responsibility for the software from software development to maintenance.
- Practice 13: Ensure that project plan templates address the release process during maintenance and related concerns.

Best Maintenance Practices by Key Process Area

4. Process Area: *Requirements Management*

- **Best Practices for Software Maintenance:**
- Practice 14: Maintain traceability of approved maintenance changes and fixes to release requirements.
- Practice 15: Maintain traceability of release requirements to regression and qualification tests

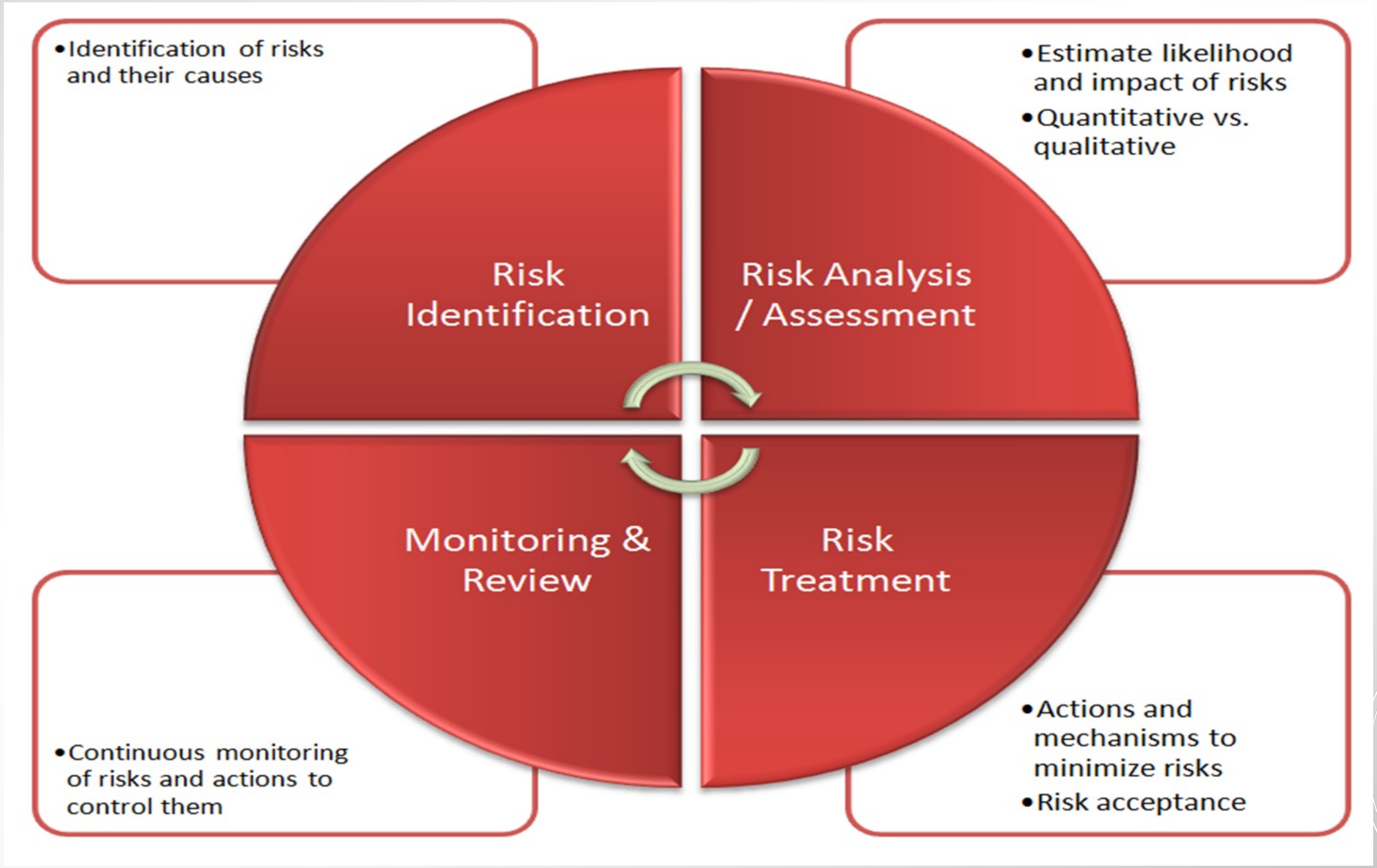
5. Process Area: *Quantitative Project Management*

- **Best Practices for Software Maintenance:**
- Practice 16: Capture measurement data on software cost, schedule, productivity, quality, and process performance as the release is being generated, and use the data to manage the project quantitatively.

Best Maintenance Practices by Key Process Area

6. Process Area: *Risk Management*

- **Best Practices for Software Maintenance:**
- Practice 17: Institute risk management procedures to proactively identify, prioritize, and mitigate risks in timely manner using information provided during software maintenance by projects developed for that purpose.



- Identification of risks and their causes

- Estimate likelihood and impact of risks
- Quantitative vs. qualitative

- Continuous monitoring of risks and actions to control them

- Actions and mechanisms to minimize risks
- Risk acceptance

Best Maintenance Practices by Key Process Area

7. Process Area: *Supplier Agreement Management*

- **Best Practices for Software Maintenance:**
- Practice 18: Establish a market watch function for commercial-off the shelf (COTS) to keep current on available alternatives for packages used as part of maintenance releases being distributed to the filed.
- Practice 19: Use relationship managers to influence the direction key suppliers take in the future with products and services used in maintenance releases.
- Practice 20: Tailor development processes so they can be used during maintenance to manage suppliers developing software for releases.

Best Maintenance Practices by Key Process Area

8. Process Area: *Facilities Management*

- **Best Practices for Software Maintenance:**
- Practice 21: Readying facilities for software maintenance is a long-lead item that requires actions to be planned and taken in advance of the transition and transfer milestones.
- Practice 22: Put in place measurement and management procedures for keeping facilities, equipment, and software used for maintenance operating at their peak efficiency as releases are generated and related work progresses.

Best Maintenance Practices by Key Process Area

9. Process Area: *Transition Management*

- **Best Practices for Software Maintenance:**
- Practice 23: Engage all stakeholders and plan for transition and turnover of responsibility form development to software maintenance group as early as possible during development.
- Practice 24: Conduct a software operational readiness review prior to transition and transfer taking place ensure that you are ready to turn over responsibility for the software from the development to the maintenance organization.
- Practice 25: Be prepared to rejuvenate or retire software during maintenance based on business trade-offs.

ILLINOIS TECH | College of Computing

Best Maintenance Practices by Key Process Area

3. **Category:** Support

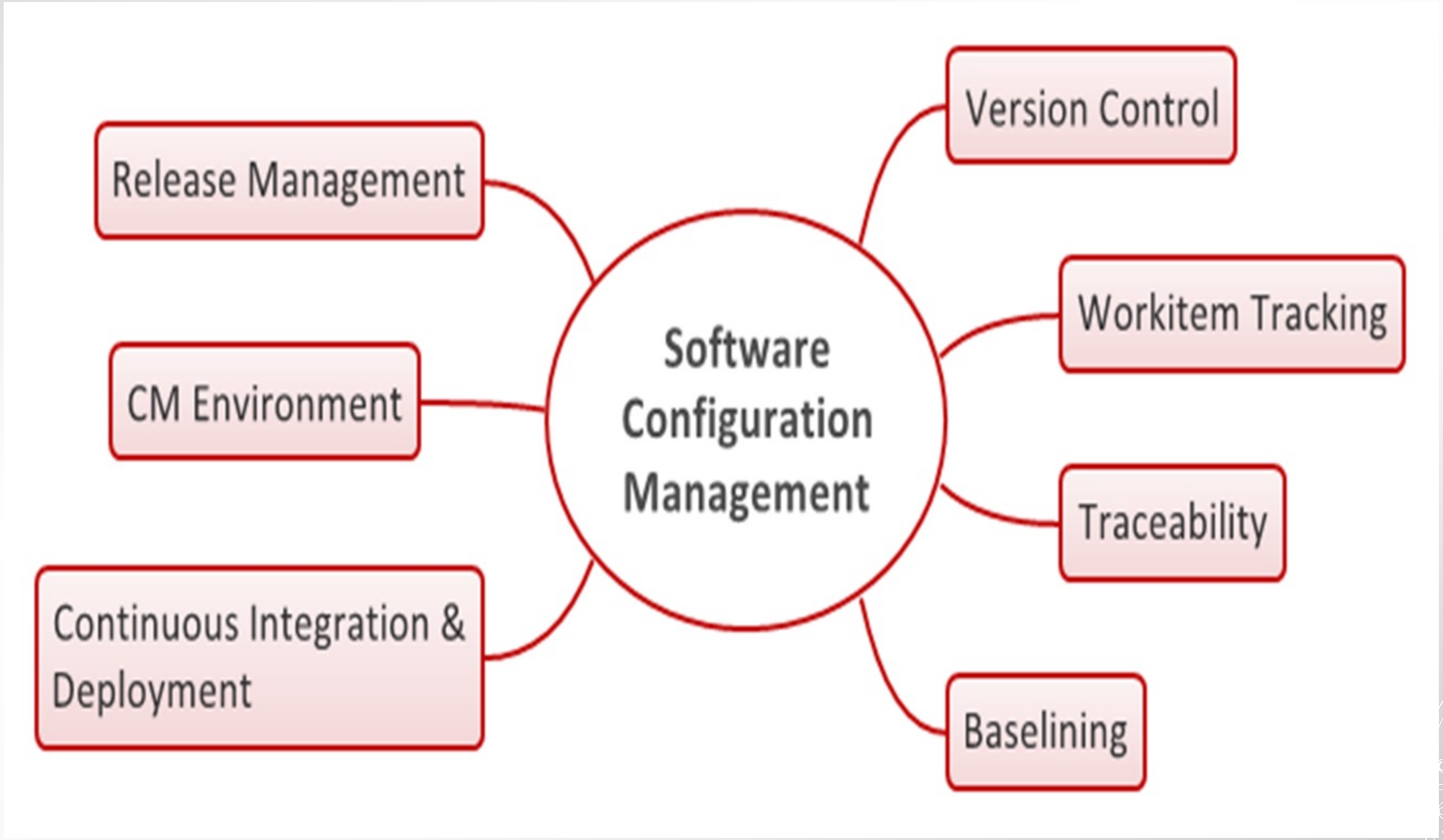
   1. **Process Area:** *Casual Analysis and Resolution*

   - **Best Practices for Software Maintenance:**
   - Practice 26: Analyze defect data to uncover the root causes of defects and put preventive measure in place.

Best Maintenance Practices by Key Process Area

2. Process Area: *Configuration Management*

- **Best Practices for Software Maintenance:**
- Practice 27: Tailor development configuration identification, control, status accounting, and audit procedures to be used to maintain the integrity of releases to be used to maintain the integrity of releases as they are developed and delivered to the field.
- Practice 28: Ensure that the transfer of all approved configuration management baselines between development and maintenance occurs seamlessly.


Software Configuration Management
Release Management
CM Environment
Continuous Integration & Deployment
Version Control
Workitem Tracking
Traceability
Baselining

Best Maintenance Practices by Key Process Area

3. Process Area: *Decision Analysis and Resolution*

- **Best Practices for Software Maintenance:**
- Practice 29: Trade-off alternatives using value-based software engineering principles to help make decisions.
- Practice 30: The software maintenance decisions to business goals and use business cases to weigh alternatives.

ILLINOIS TECH | College of Computing

Best Maintenance Practices by Key Process Area

4. **Process Area:** *Measurement and Analysis*

- **Best Practices for Software Maintenance:**
- Practice 31: Establish a software measurement program whose aim is to gain insight into performance of people, processes, and projects relative to standards and benchmarks and assess the quality of products, processes, and services being generated during software maintenance

Best Maintenance Practices by Key Process Area

5. Process Area: *Process and Product Quality Assurance*

- **Best Practices for Software Maintenance:**
- Practice 32: Maintain the integrity of processes being used during software maintenance.
- Practice 33: Assurance the quality of products being generated during software maintenance, including those associated with both the releases and the environment used to create it.

Best Maintenance Practices by Key Process Area

6. Process Area: *Customer Support*

- **Best Practices for Software Maintenance:**
- Practice 34: Create a website as soon as possible to keep customers/users informed, and keep it updated with current and useful information about releases during maintenance.
- Practice 35: Provide high levels of customer support during operations and maintenance including user training, hand-holding, and query handling.
- Practice 36: Provide prompt response to user requests via help desk staffed by those who can provide answers.

Best Maintenance Practices by Key Process Area

7. Process Area: *Distribution Management*

- **Best Practices for Software Maintenance:**
- Practice 37: Maintenance the integrity of the distribution process by assuring that the products delivered to the field are configured properly and work in customer sites.

6.3 The Role of Requirements

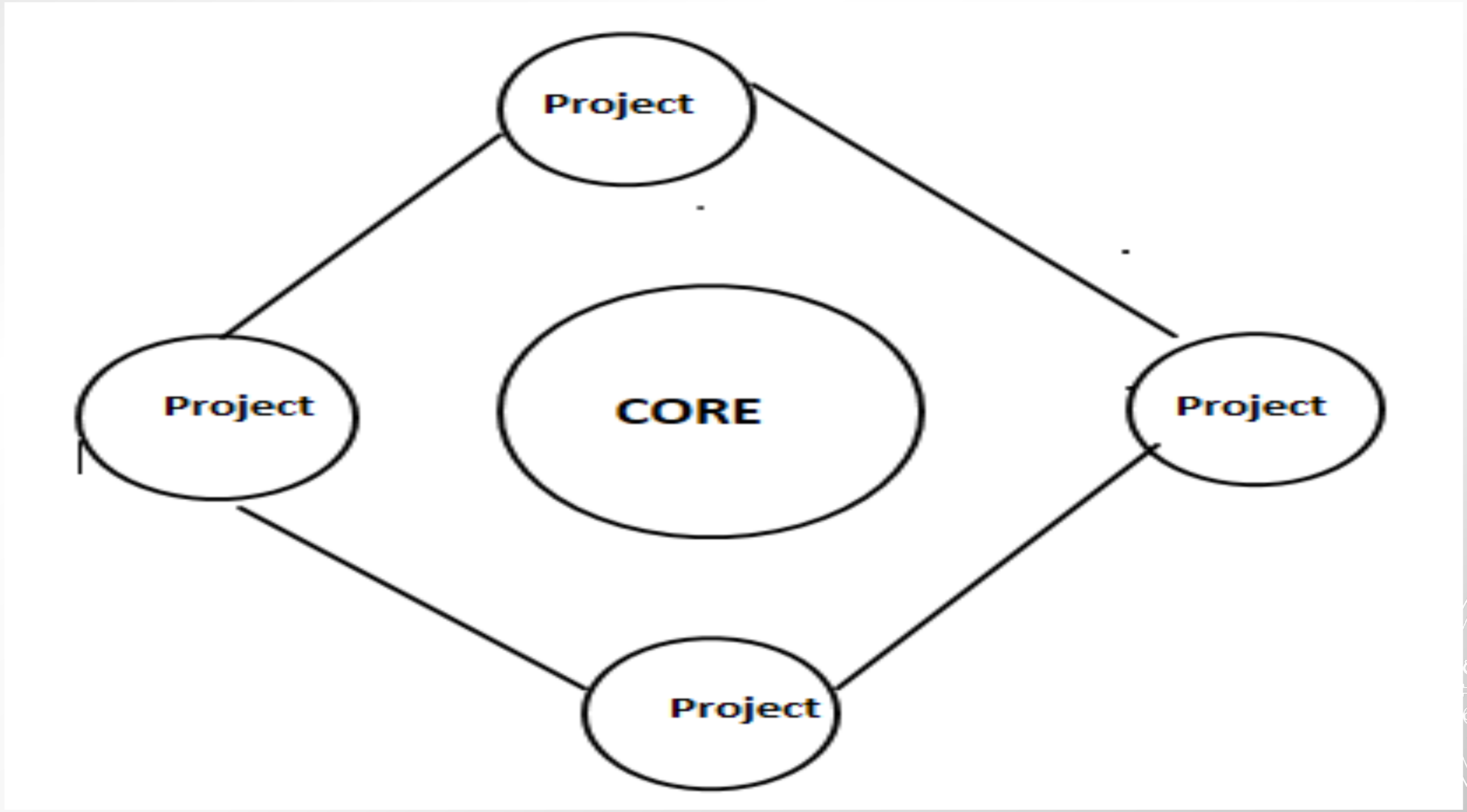
Separation of Work in Maintenance:

- **Project Role:**
  - Release Planning and Production
  - Defect repairs and backlog reduction
  - Enhancements

6.3 The Role of Requirements

- **Core Role:**
  - Sustaining engineering
  - Independent test and verification
  - Product support
  - Information assurance
  - Operations and facility support
  - Field support

Separation of work in Maintenance



6.3 The Role of Requirements

- **Seven reasons for change requests**:

1. Software trouble reports (STRs) that identify defects that must be fixed in the next release.
2. STRs that identify defects in the backlog that should be considered to be fixed in the next release
3. System enhancement requests that add, subtract, or modify features and functionality that users want (new and changed requirements).

6.3 The Role of Requirements

4. System enhancement requests to improve performance (memory usage, disk utilization, central processing unit (CPU) utilization, etc.) and usability in the field.

5. Company change driven by interfaces with other systems, typically external.

6. Compatibility changes with platform (hardware, operating systems, etc.) and field site configurations (specialized drivers, platform variations, etc.)

7. Demands from senior management for improvements (These may take the form of enhancements or changes needed to provide performance.)

ILLINOIS TECH | College of Computing

6.3 The Role of Requirements

Defects are separated by Priority:

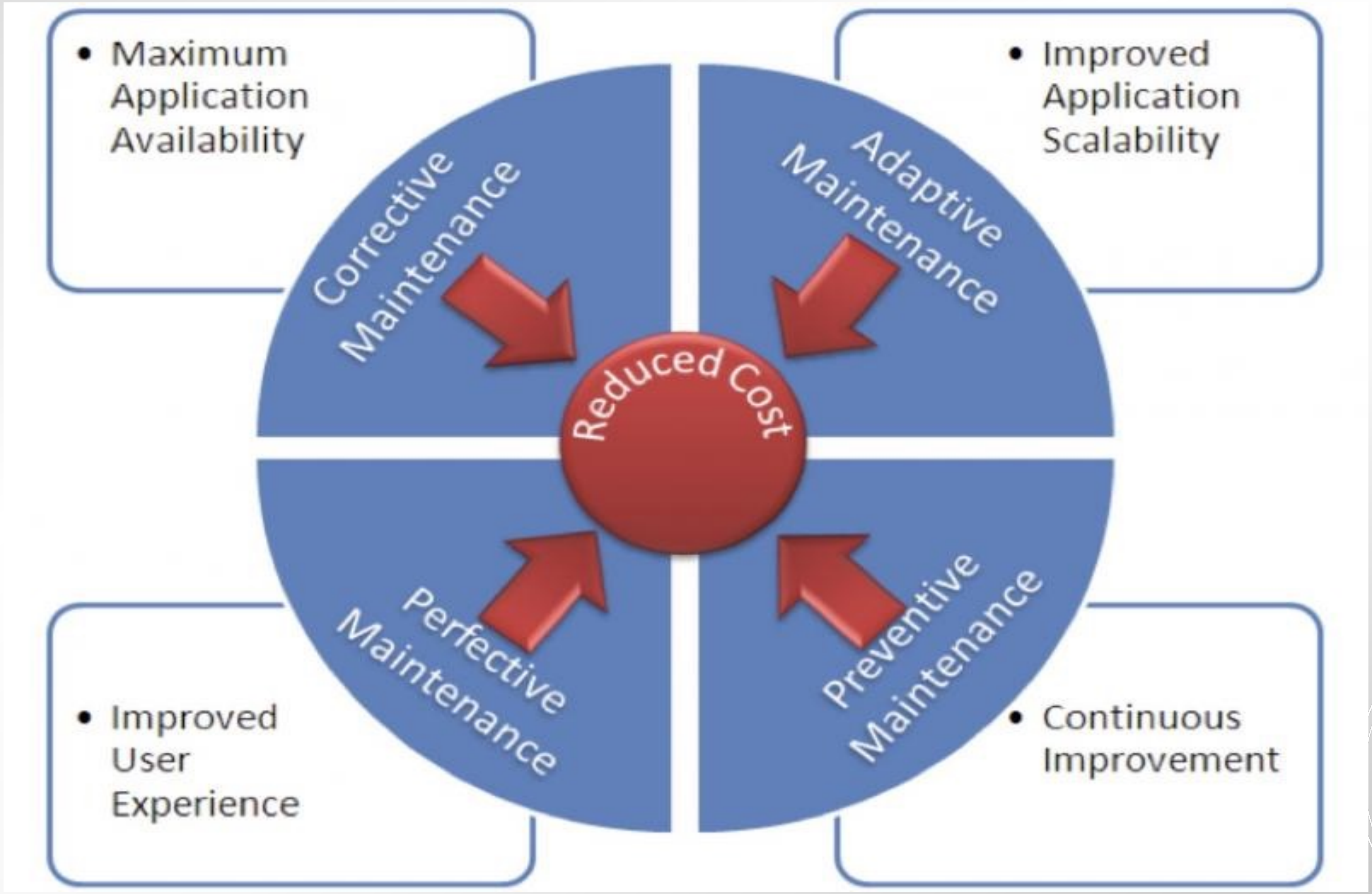
1. Catastrophic (Showstopper)
2. Critical (Major)
3. Serious (High)
4. Annoyance (Medium)
5. Minimal (Minor)

6.4 Budgeting and Estimating

- Maintenance organizations pay for their projects using budgets developed based on the number of changes scheduled to be incorporated into the next release.
- Usually last year's budget dictates this year's expenditures.
- If emergency funding for fixes needed they go to the project lead for additional funding on a fee-for-service basis.

6.5 Release Management - Schedule

- Release Planning
- Architecture analysis
- Hardware defect repair
- Software defect repair
- Hardware enhancements
- Software enhancements
- Release integration and test
- Release qualification and delivery


Maximum Application Availability
Improved Application Scalability
Corrective Maintenance
Adaptive Maintenance
Reduced Cost
Perfective Maintenance
Preventive Maintenance
Improved User Experience
Continuous Improvement

Evolution Engineering Domain-(SMM)

Predelivery and Transitions Services:

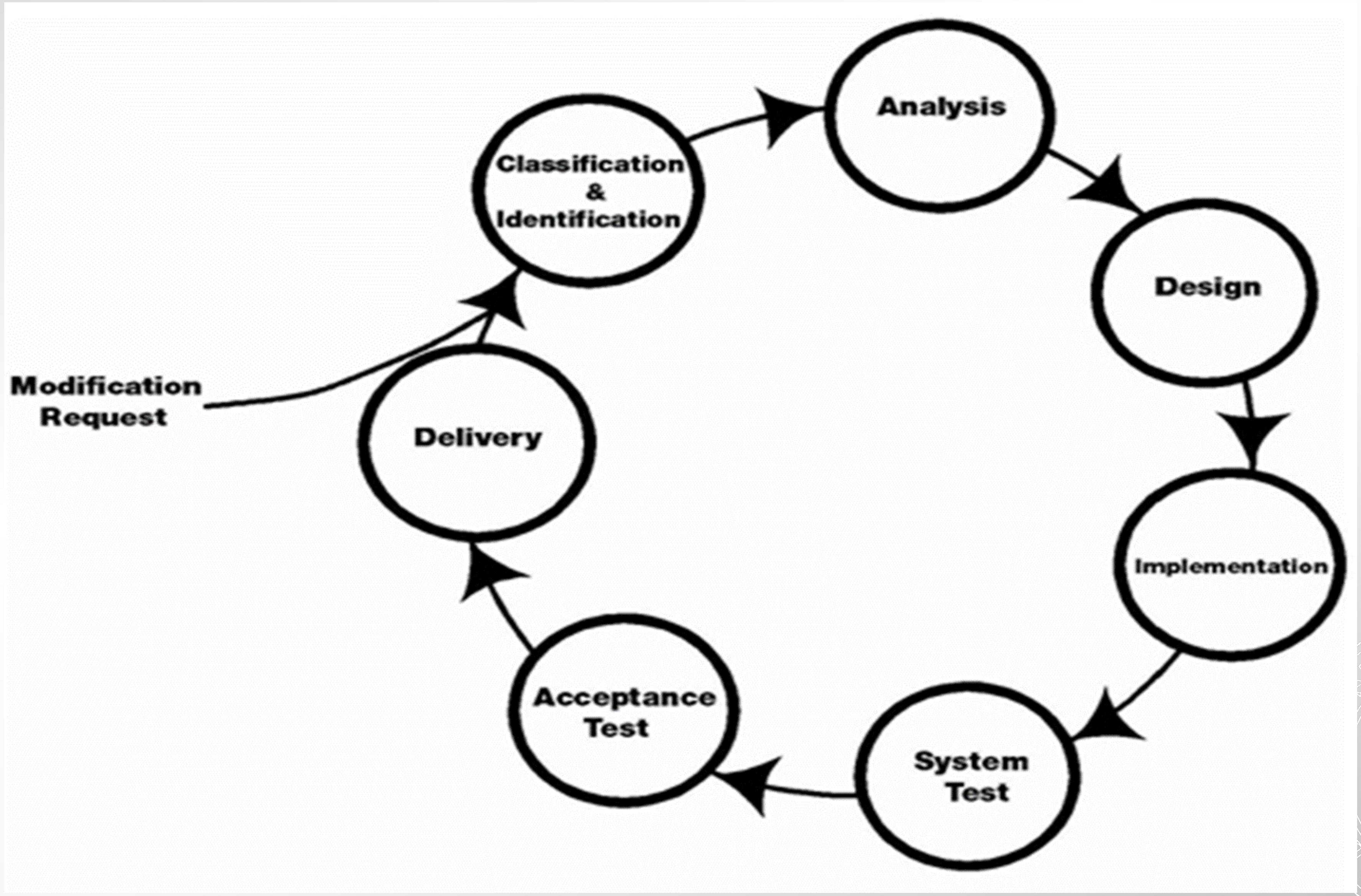
- Maintainers influence during design phase
- An SLA exists for the new software
- Maintenance environment exists
- Record of transition measures exist
- Identify, and complete the documentation
- Communicate status to the stakeholders

Operational Support Services

- Support activities are recognized as value-added activities
- Software Maintenance is organizing its evolution activities around it
- Operations, development and maintenance activities are softened
- Value added support activities are promoted – knowledge transfer activities.

Evolution and Correction Services

- The authorized customer option is implemented
- Requirements are documented any modifications are negotiated
- Modifications are documented (tractability)
- Regression tests are carried out
- Communicate readiness-acceptance testing
- Software is improved each time


Analysis
Design
Implementation
System Test
Acceptance Test
Delivery
Modification Request
Classification & Identification

ILLINOIS TECH | College of Computing

Verification and Validation

- Techniques for V&V are adapted for maintenance
- Follow the V&V procedures in maintenance context and by service type
- Adequate regression testing is carried out for every maintenance release (MR)