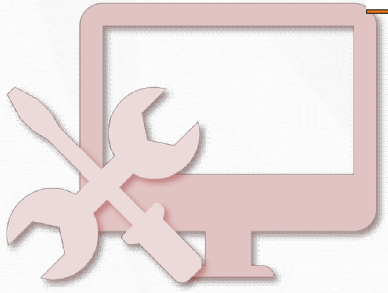
LOVELY PROFESSIONAL UNIVERSITY



Software Maintenance and Configuration Management

Types of Software Maintenance

- In a software lifetime, type of maintenance may vary based on its nature
- It may be just a routine maintenance tasks as some bug discovered by some user or it may be a large event in itself based on maintenance size or nature
- Following are some types of maintenance based on their characteristics
  - Corrective Maintenance
  - Adaptive Maintenance
  - Perfective Maintenance
  - Preventive Maintenance

Types of Software Maintenance Cont.

Corrective Maintenance

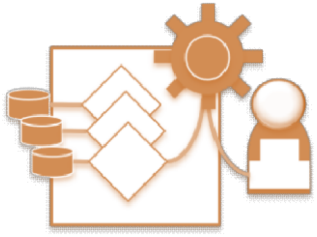
- This includes modifications done in order to fix problems
- Corrective maintenance deals with the repair of defects found in day-to-day system functions

Adaptive Maintenance

- This includes modifications applied to keep the software product up-to-date
- Adaptive maintenance is the implementation of changes in a part of the system, which has been affected by a change that occurred in some other part of the system

Types of Software Maintenance Cont.

Perfective Maintenance



- This includes modifications done in order to keep the software usable over long period of time
- It includes new features, new user requirements for refining the software and improve its reliability and performance.

- This includes changing the functionalities of as per the user's changing needs

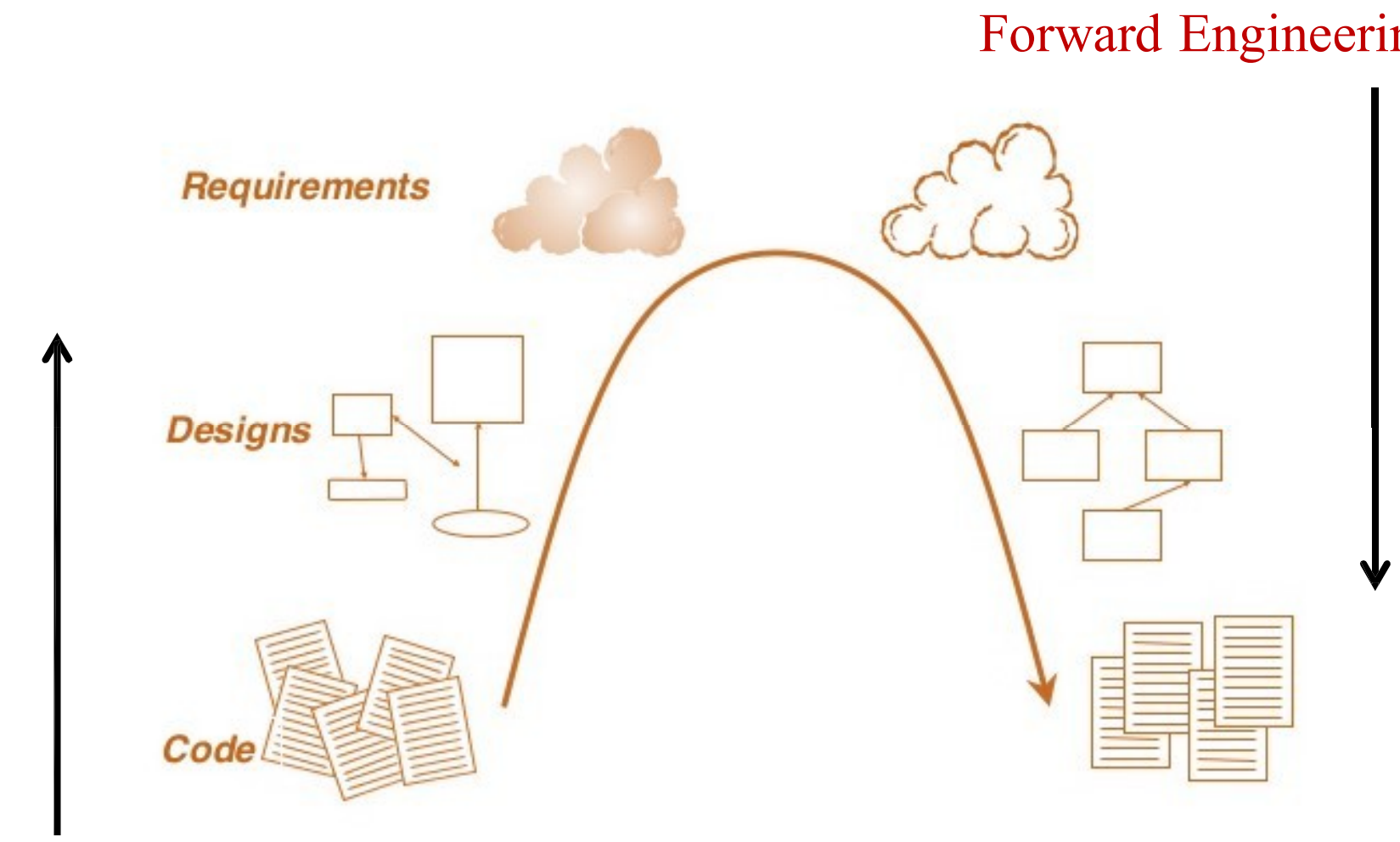
Preventive Maintenance

- Modifications to prevent future problems of software
- It aims to attend problems, which are not significant at this moment but may cause serious issues in future
- It comprises documentation updating, code optimization and code restructuring.

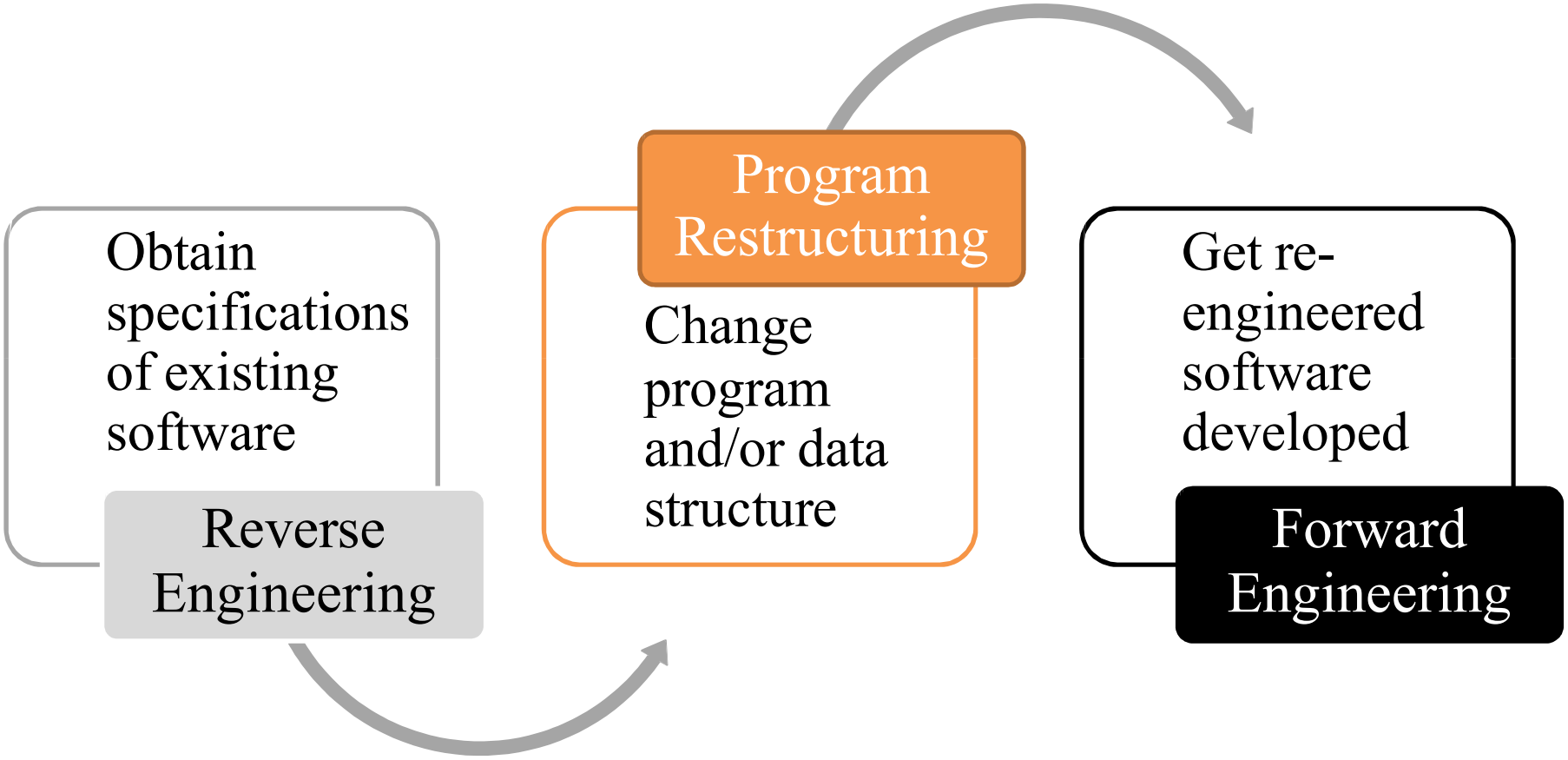
Re-Engineering

- When we need to update the software to keep it to the current market, without impacting its functionality, it is called software re-engineering
- It is a process where the design of software is changed and programs are re-written
- Legacy software cannot keep tuning with the latest technology available in the market
  - For example, initially UNIX was developed in assembly language. When language C came into existence, UNIX was re-engineered in C, because working in assembly language was difficult.
- Other than this, sometimes programmers notice that few parts of software need more maintenance than others and they also need re-engineering

Re-Engineering Cont.



Re-Engineering Cont.



Re-Engineering Process

- Decide what to re-engineer.
  - Is it whole software or a part of it?
- Perform Reverse Engineering, in order to obtain specifications of existing software
- Restructure Program if required
  - For example, changing function-oriented programs into object-oriented programs and re-structure data as required
- Apply Forward engineering concepts in order to get re-engineered software

Reverse Engineering

- Reverse engineering can extract design information from source code
- The abstraction level of a reverse engineering process refers to the sophistication of the design information that can be extracted from source code
- Ideally, the abstraction level should be as high as possible
- The reverse engineering process should be capable of
  - Deriving procedural design representations (a low-level abstraction)
  - Program and data structure information (a somewhat higher level of abstraction)
  - Object models, data flow models (a relatively high level of abstraction)
  - Entity relationship models (a high level of abstraction).

Reverse Engineering Cont.

- As the abstraction level increases, information will allow easier understanding of the program
- Interactivity refers to the degree to which the human is "integrated" with automated tools to create an effective reverse engineering process
- In most cases, as the abstraction level increases, interactivity must increase
- The directionality of the reverse engineering process is one-way, all information extracted from the source code is provided to the software engineer

Forward engineering

- Forward engineering is a process of obtaining desired software from the specifications, which were brought by reverse engineering
- Forward engineering is same as software engineering process with only one difference it is carried out always after reverse engineering
- In most cases, forward engineering does not simply create a modern equivalent of an older program
- Rather, new user and technology requirements are integrated into the reengineering effort
- The redeveloped program extends the capabilities of the older application

Software Configuration Management

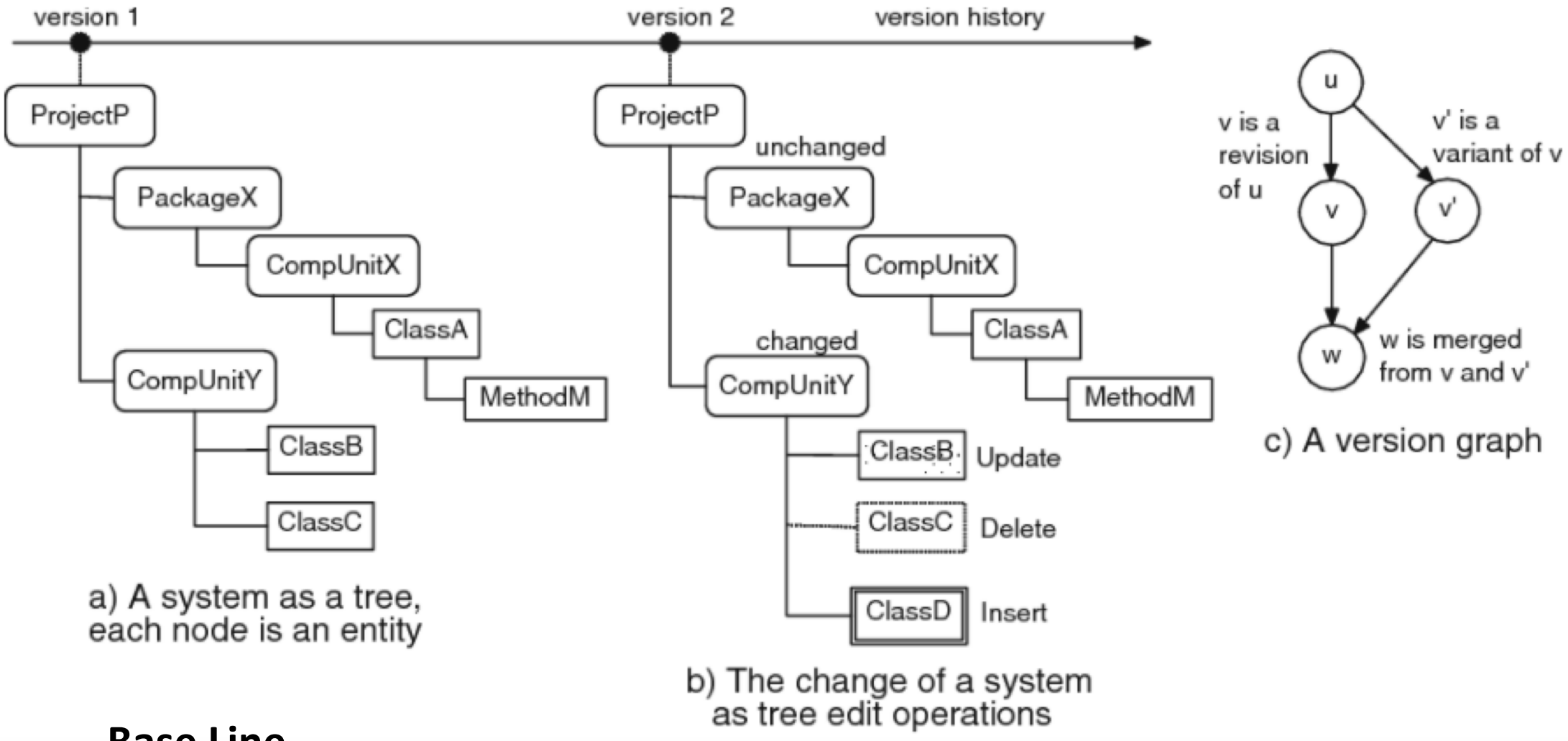
The SCM (Software Configuration Management) is a set of activities that have been developed to manage change throughout the software life cycle

Purpose:

Systematically control changes to the configuration and maintain the integrity and traceability of the configuration throughout the system's life cycle

Four primary objectives

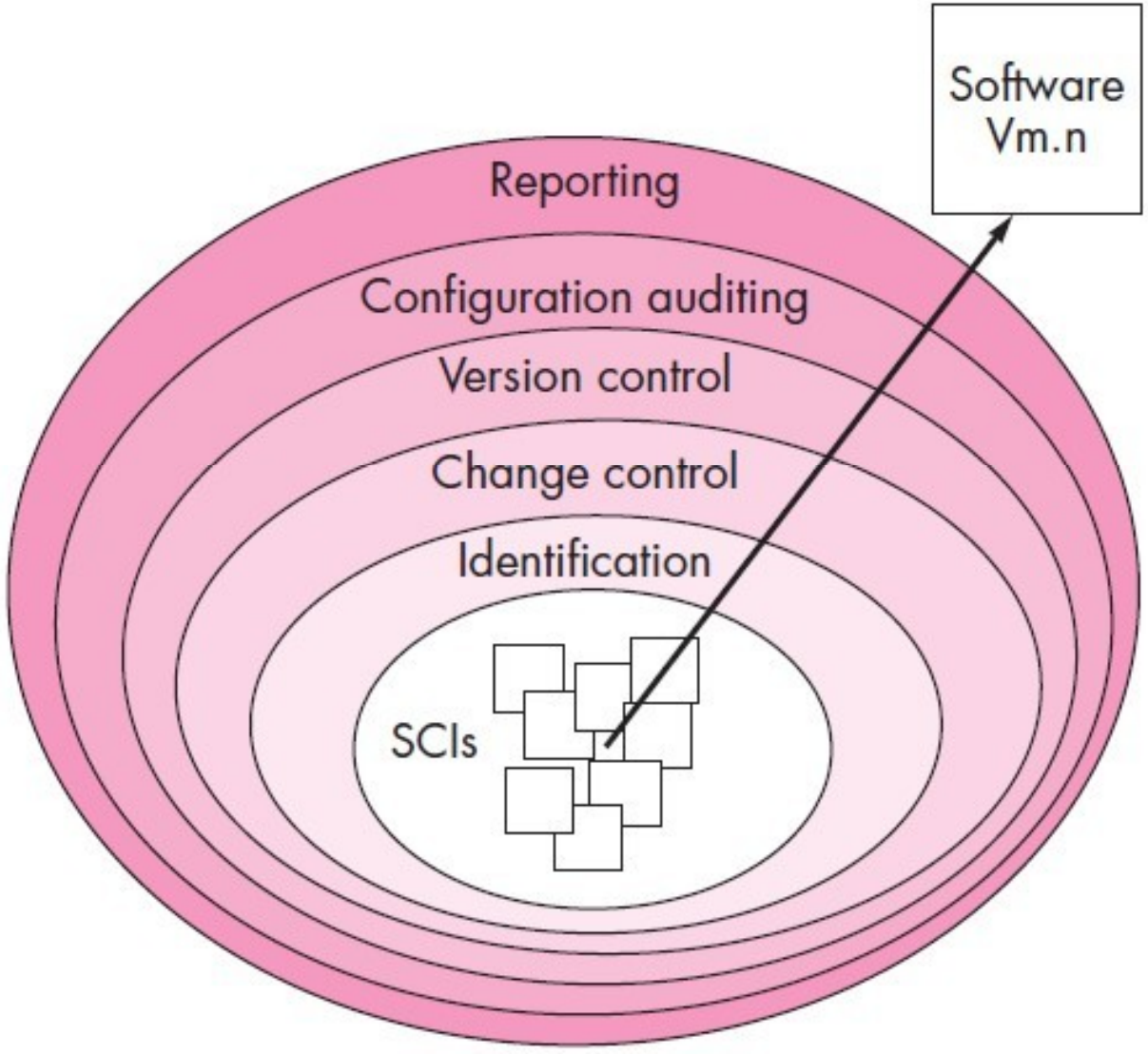
- To identify all items that collectively define the software configuration
- To manage changes to one or more of these items
- To facilitate the construction of different versions of an application
- To ensure that software quality is maintained as the configuration evolves over time



Base Line

Fig. 1. OperV Concepts

Layers of SCM Process



The SCM Process

- Referring to the figure, SCM tasks can viewed as concentric layers
- SCIs (Software Configuration Item) flow outward through these layers throughout their useful life
- As an SCI moves through a layer, the actions implied by each SCM task may or may not be applicable
  - For example, when a new SCI is created, it must be identified.
  - However, if no changes are requested for the SCI, the change control layer does not apply
- The SCI is assigned to a specific version of the software (version control mechanisms come into play)
- A record of the SCI (its name, creation date, version, etc.) is maintained for configuration auditing purposes

Change Control

- Change control is a procedural activity that ensures quality and consistency as changes are made to a configuration object
- A change request is submitted to a configuration control authority, which is usually a change control board (CCB).
  - The request is evaluated for technical merit, potential side effects, overall impact on other configuration objects and system functions, and projected cost in terms of money, time and resources
- An engineering change order (ECO) is issued for each approved change request
  - Describes the change to be made, the constraints to follow and the criteria for review and audit

Change Control Cont.

- The Base lined SCI is obtained from the SCM repository
  - Access control governs which software engineers have the authority to access and modify a particular configuration object
  - Synchronization control helps to ensure that parallel changes performed by two different people don't overwrite one another.

Version Control

- Version control is a set of procedures and tools for managing the creation and use of multiple occurrences of objects in the SCM repository
- Version Control Capabilities
  - An SCM repository that stores all relevant configuration objects
  - A version management capability that stores all versions of a configuration object
  - A make facility that enables the software engineer to collect all relevant configuration objects and construct a specific version of the software
  - Issues or bug tracking capability that enables the team to record and track the status of all outstanding issues associated with each configuration object

Version Control cont.

- The SCM repository maintains a change set
  - Serves as a collection of all changes made to a baseline configuration
  - Used to create a specific version of the software
  - Captures all changes to all files in the configuration along with the reason for changes and details of who made the changes and when

Few version control systems

Configuration Audit

- Configuration auditing is an SQA activity that helps to ensure that quality is maintained as changes are made.
- It complements the formal technical review and is conducted by the SQA group
- It addresses the following questions
  - Has a formal technical review been conducted to assess technical correctness?
  - Has the software process been followed and have software engineering standards been properly applied?
  - Has the change been "highlighted" and "documented" in the SCI? Have the change data and change author been specified? Do the attributes of the configuration object reflect the change?

Configuration Audit Cont.

- Have SCM procedures for noting the change, recording it and reporting it been followed?
- Have all related SCIs been properly updated?

- A configuration audit ensures that
  - The correct SCI (by version) have been incorporated into a specific build s
  - That all documentation is up-to-date and consistent with the version that has been built

BUSINESS PROCESS REENGINEERING

- *Business process reengineering* (BPR) extends far beyond the scope of information technologies and software engineering.
- Definition: “the search for, and the implementation of, radical change in business process to achieve breakthrough results.”
- **Business Processes:** A business process is “a set of logically related tasks performed to achieve a defined business outcome”.
- Every business process has a defined customer—a person or group that receives the outcome (e.g., an idea, a report, a design, a service, a product).

- *In addition, business processes cross organizational boundaries. They require that different organizational groups participate in the "logically related tasks" that define the process.*
- *Every system is actually a hierarchy of subsystems. A business is no exception.*
- *The overall business is segmented in the following manner:*
- *The business -> business systems -> business processes -> business subprocesses.*
- *Each business system (also called business function) is composed of one or more business processes, and each business process is defined by a set of subprocesses.*
- *BPR can be applied at any level of the hierarchy, but as the scope of BPR broadens the risks associated with BPR grow dramatically.*

A BPR Model

- Like most engineering activities, business process reengineering is iterative.
- Business goals and the processes that achieve them must be adapted to a changing business environment.
- For this reason, there is no start and end to BPR—it is an evolutionary process.

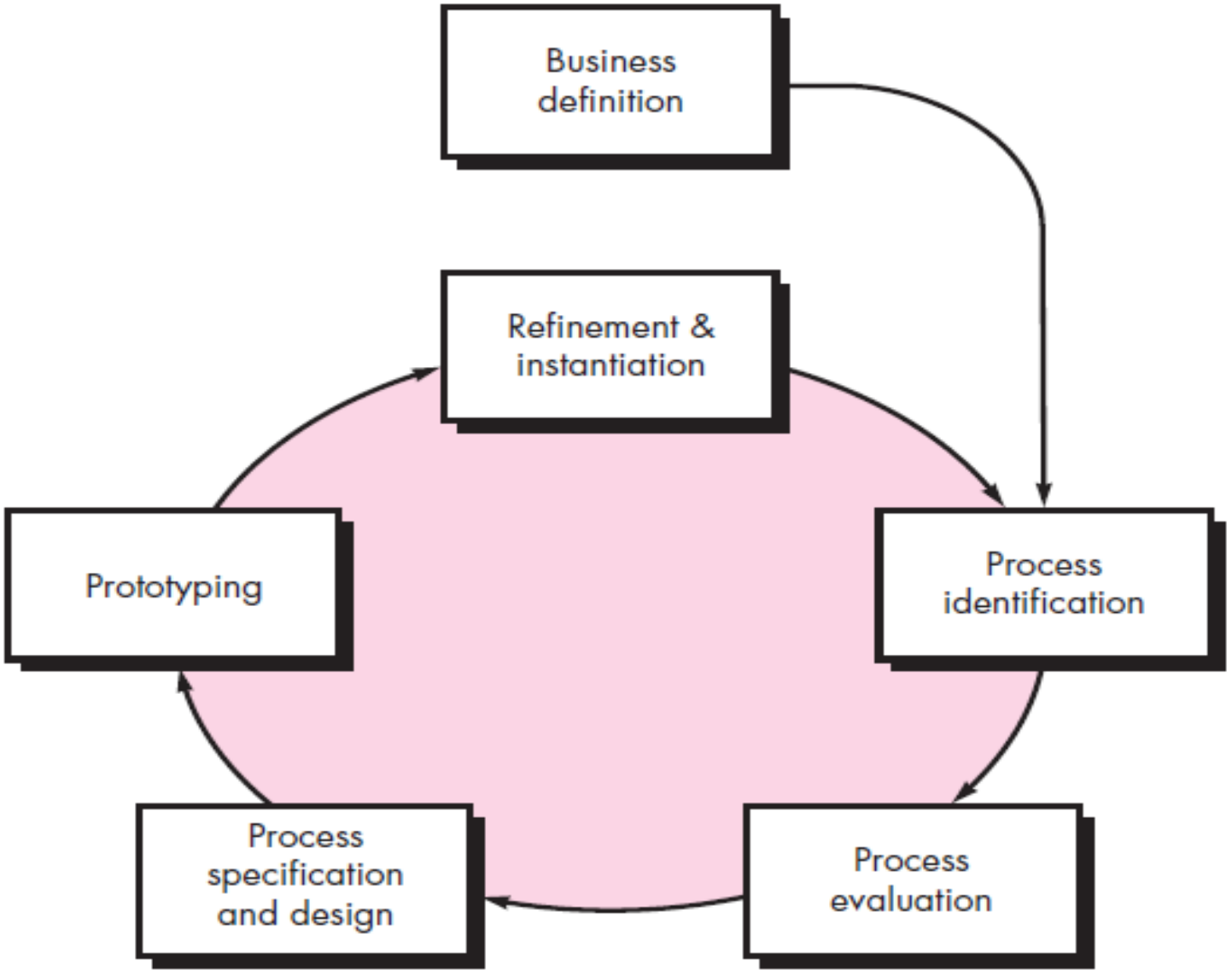
The model defines following six activities:

- **Business definition.** Business goals are identified within the context of four key drivers:
  - cost reduction,
  - time reduction,
  - quality improvement
  - personnel development and empowerment.
- Goals may be defined at the business level or for a specific component of the business.

- **Process identification**. Processes that are critical to achieving the goals defined in the business definition are identified.
- They may then be ranked by
  - importance
  - by need for change
  - or in any other way that is appropriate for the reengineering activity.

- **Process evaluation.** The existing process is thoroughly analyzed and measured. Process tasks are identified; the costs and time consumed by process tasks are noted; and quality/performance problems are isolated.

- **Process specification and design.** Based on information obtained during the first three BPR activities, use cases are prepared for each process that is to be redesigned. Within the context of BPR, use cases identify a scenario that delivers some outcome to a customer. With the use case as the specification of the process, a new set of tasks are designed for the process.
- **Prototyping.** A redesigned business process must be prototyped before it is fully integrated into the business. This activity "tests" the process so that refinements can be made.
- **Refinement and instantiation.** Based on feedback from the prototype, the business process is refined and then instantiated within a business system.


Business definition
Refinement & instantiation
Process identification
Process evaluation
Process specification and design
Prototyping