Faculty of Computers and Information
BSc. Software Engineering Program

Dr. Lamia Abo Zaid

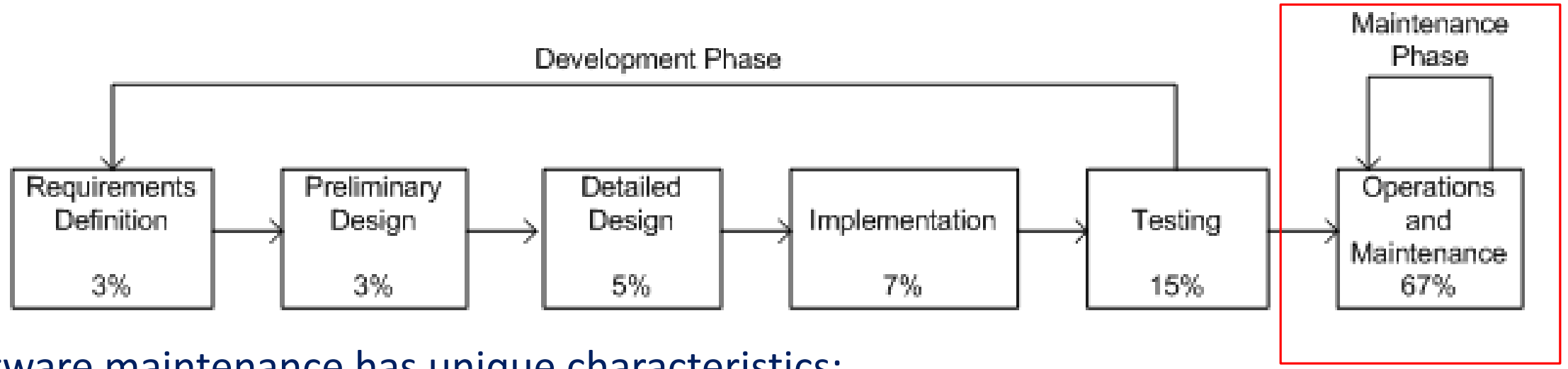
د. لمياء أبوزيد

L.ABOZAID@FCI-CU.EDU.EG

Software Evolution : TOC

1. Introduction to Software Maintenance & Evolution
2. Taxonomy of Software Maintenance and Evolution
3. Evolution and Maintenance Models
4. Program Comprehension
5. Impact Analysis
6. Refactoring
7. Reengineering
8. Legacy Information Systems
9. Reuse and Domain Engineering

Maintenance as part of the SDLC

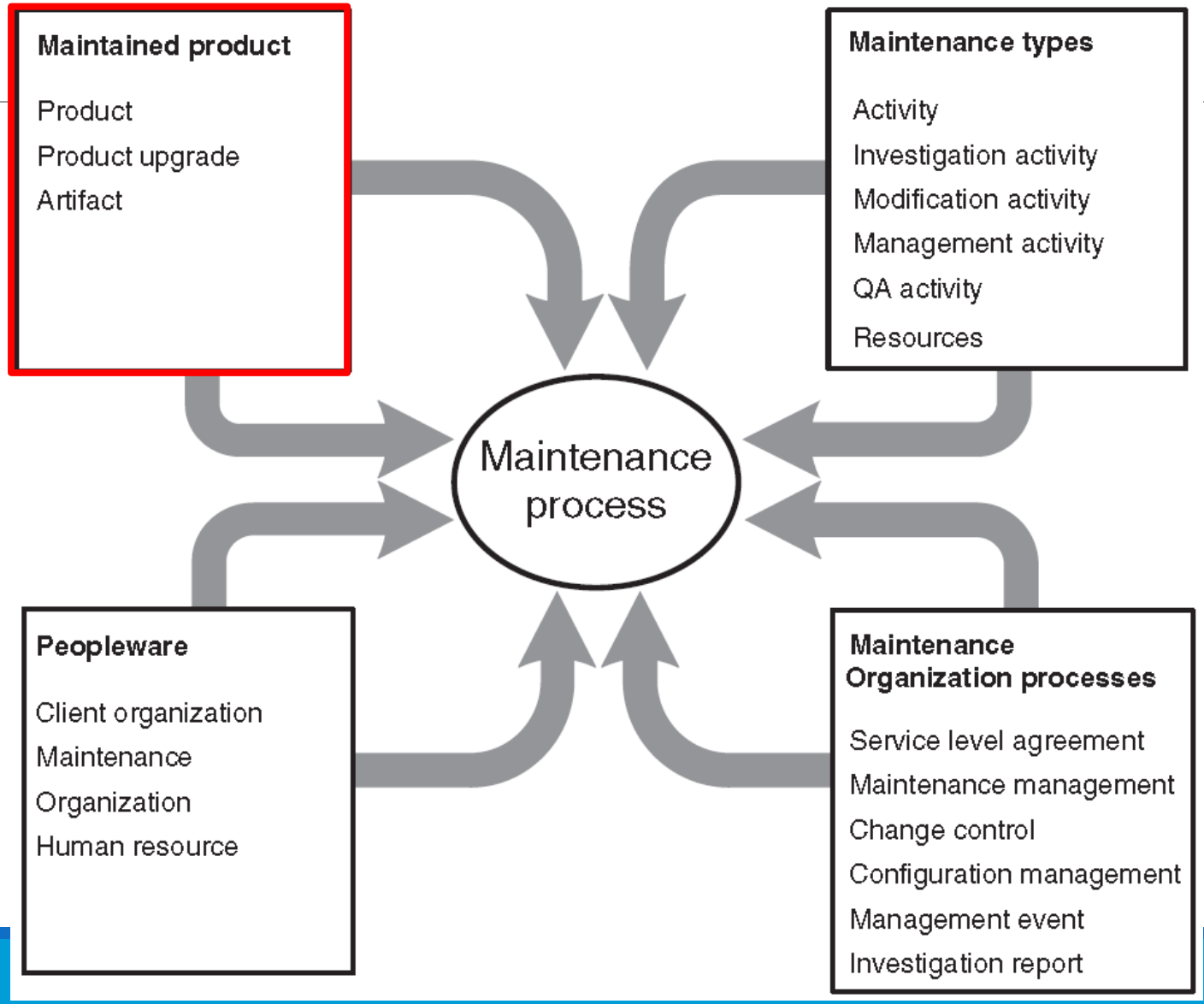


Software Maintenance Life Cycle (SMLC)

Software maintenance has unique characteristics:

- Constraints of an existing system
- Shorter time frame
- Available test data

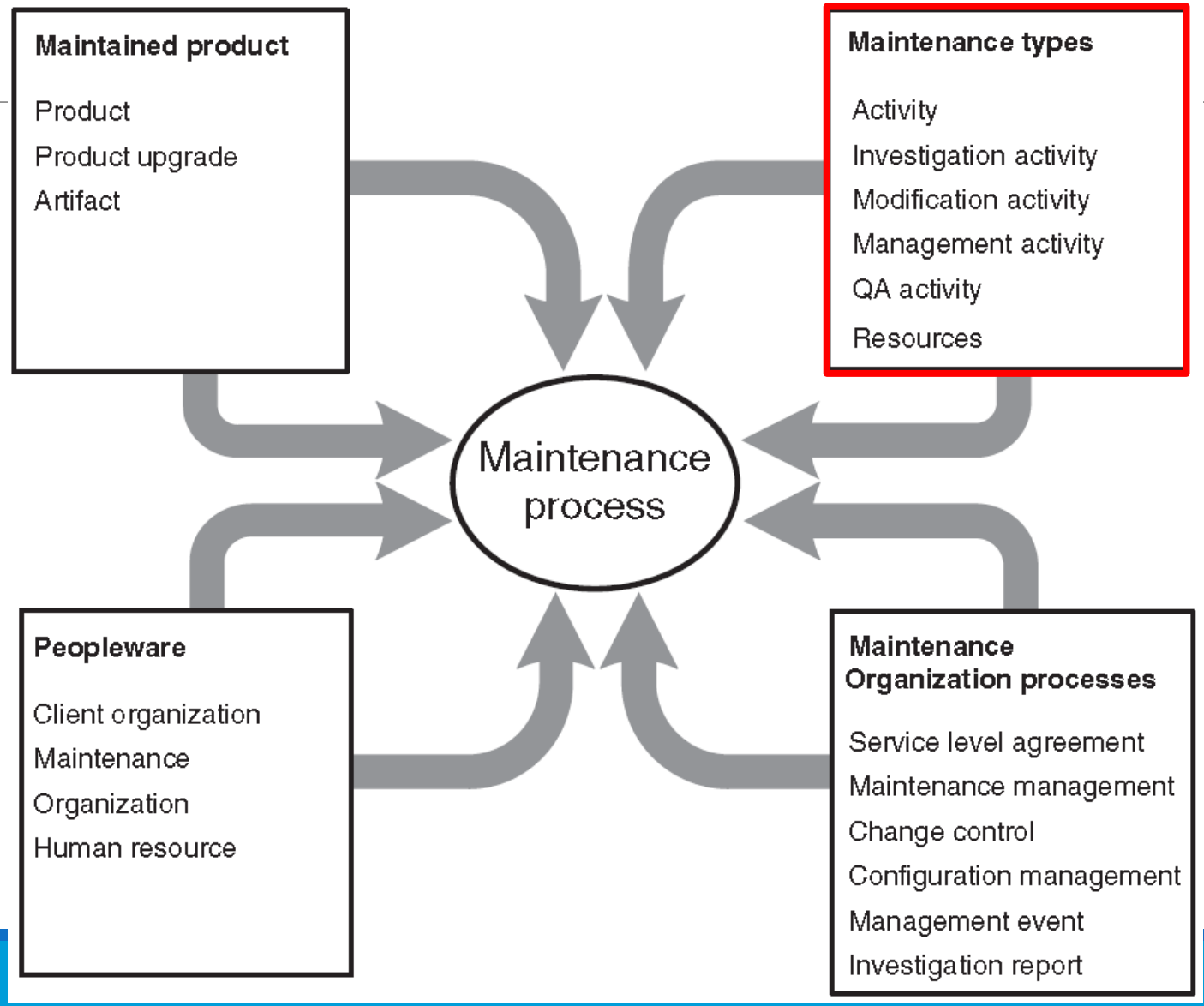
Concepts affecting Software Maintenance



Maintenance Process – Maintained Product

- Product: A product is a coherent collection of several different artifacts. Source code is the key component of a software product.
- Product upgrade: Baseline is an arrangement of related entities that make up a particular software configuration. Any change or upgrade made to a software product relates to a specific baseline.
  - An upgrade can create a new version of the system being maintained, a patch code for an object, or even a notice explaining a restriction on the use of the system.
- Artifacts: A number of different artifacts are used in the design of a software product. One can find the following types of artifacts: textual and graphical documents, component off-the-shelf products, and object code components.
  - The key elements of the maintained product are size, age, application type, composition and quality.

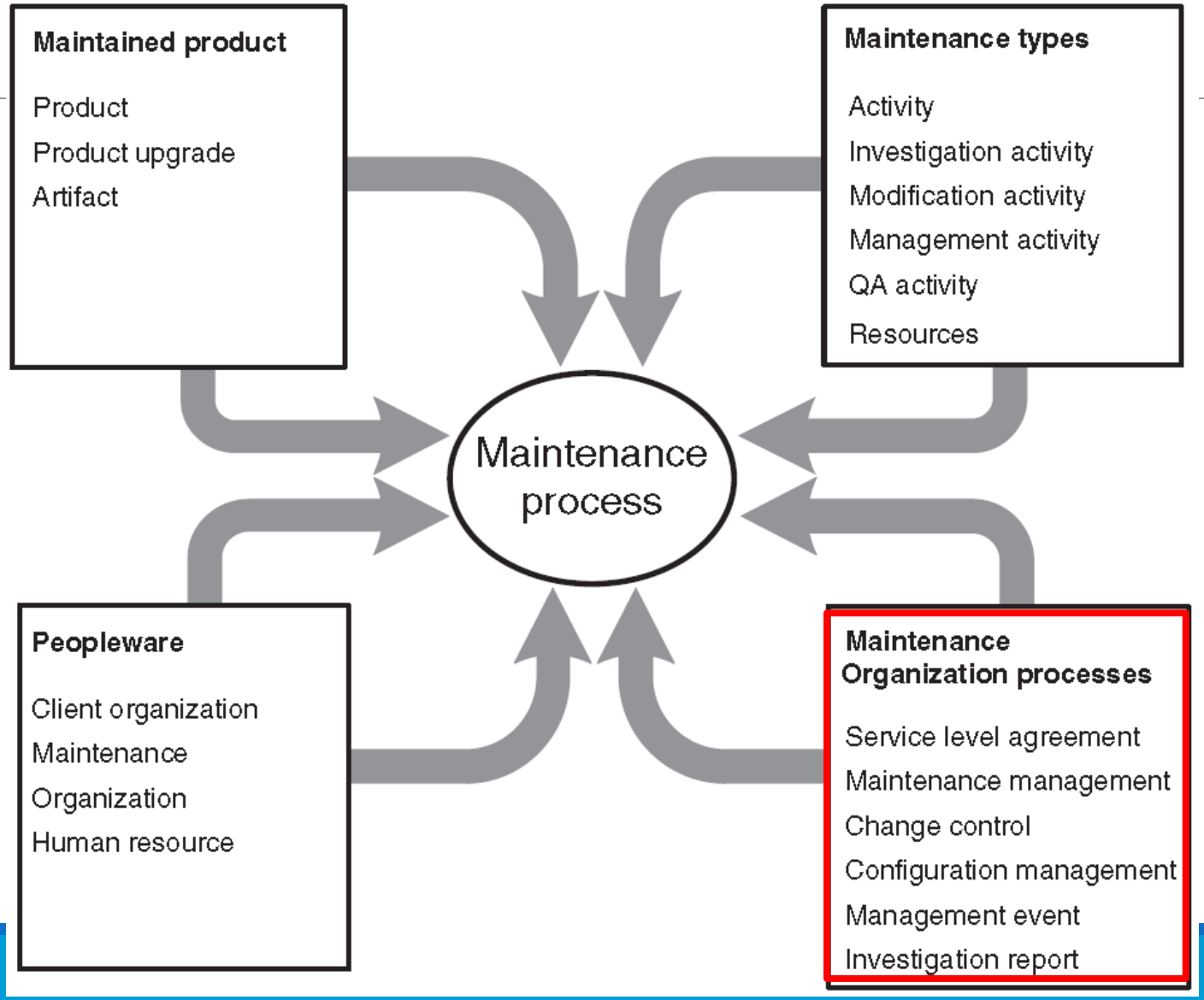
Concepts affecting Software Maintenance



Maintenance Process – Activity Type

- Activity: An activity accepts some artifacts as inputs and produces new or changed artifacts.
- Investigation activity: This kind of activities evaluate the impact of making a certain change to the system.
- Modification activity: This kind of activities change the system's implementation.
- Management activity: This kind of activities relate to the configuration control of the maintained system or the management of the maintenance process.
- Quality assurance activity: This kind of activities ensure that modifications performed on a system do not damage the integrity of the system
  - regression testing is an example of quality assurance activities.

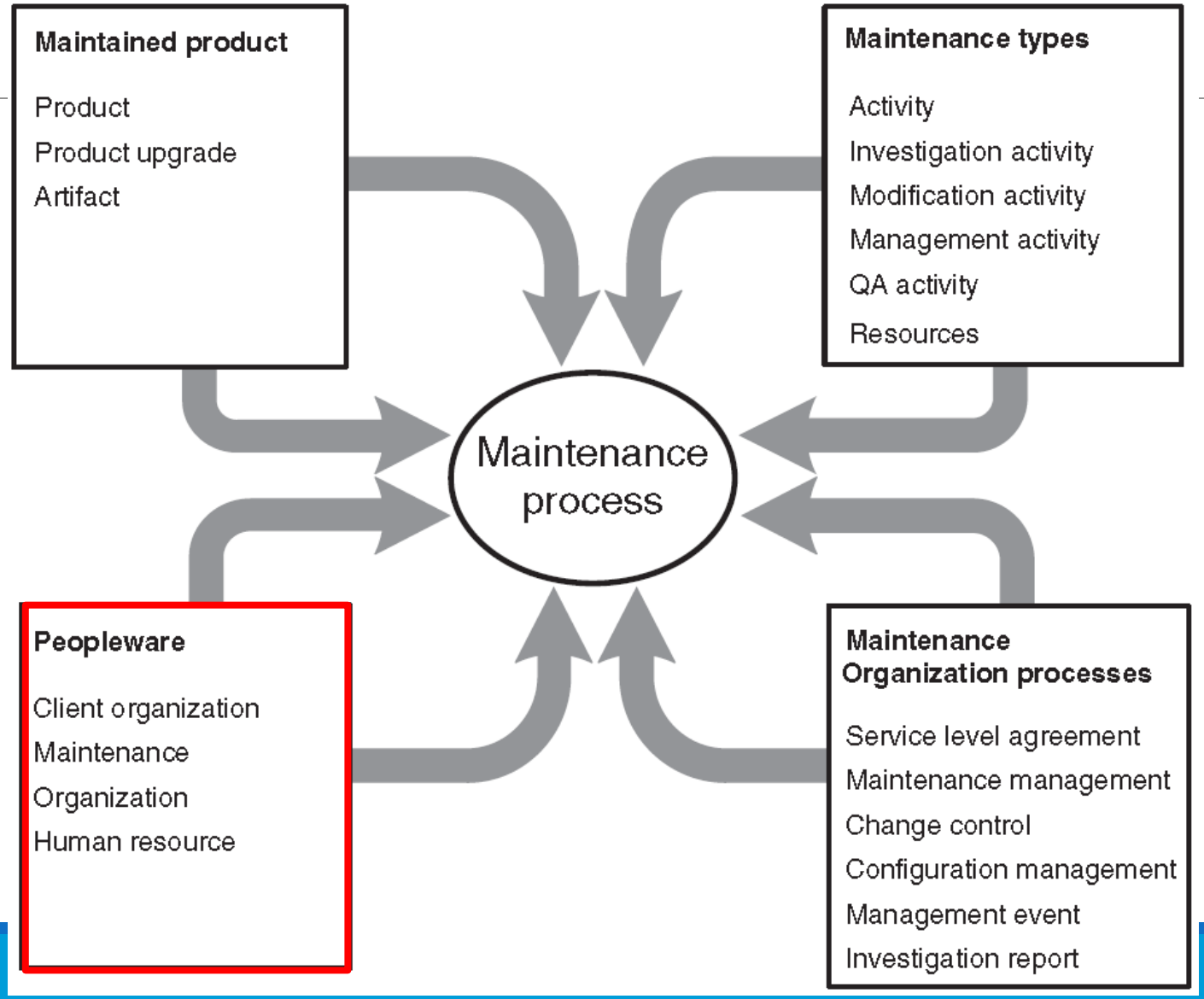
Concepts affecting Software Maintenance



Maintenance Process – Maintenance Organization Processes

- Three major elements of a maintenance organization:
  - Event management: The stream of events, namely, all the CRs (or MRs) from various sources, received by the maintenance organization, is handled in an event management process.
  - Change control: Evaluation of results of investigations of maintenance events is performed in a process called change control. Based on the evaluation, the organization approves a system change.
  - Configuration management: A system's integrity is maintained by means of a configuration management process. Integrity of a product is maintained in terms of its modification status and version number.

Concepts affecting Software Maintenance

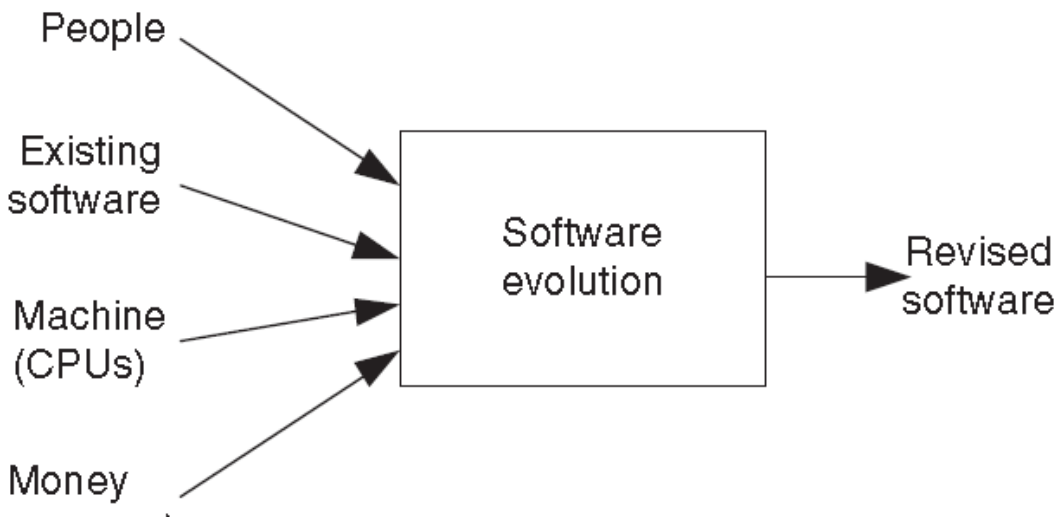


Maintenance Process – Maintenance Organization Processes

- Maintenance organization. This is the organization that maintains the product(s).
- Client organization. A client organization uses the maintained system and it has a clear relationship with the maintenance organization.
- Human resource. Human resource includes personnel from the maintenance and client organizations.
  - Maintenance organization personnel include managers and engineers, whereas client organization personnel include customers and users.
  - The management negotiates with the customers to find out the SLA, scheduling of requirement enhancements, and cost.

Software Evolution – Revisited

- Maintenance means preserving software from decline or failure.
- Evolution means a continuously changing software from a worse state to a better state.
- Chapin defines software evolution as “the applications of software maintenance activities and processes that generate a new operational software version with a changed customer-experienced functionality or properties from a prior operational version, where the time period between versions may last from less than a minute to decades, together with the associated quality assurance activities and processes, and with the management of the activities and processes”



Why Study Software Evolution ?

❑Software evolution is studied with two broad, complementary approaches:

1. Explanatory (What/Why):

- This approach attempts to explain the causes of software evolution , the processes used, and the effects of software evolution.
- The explanatory approach studies evolution from a scientific view point; one strives to understand its driving factors and impacts.

2. Process improvement (How):

- This approach attempts to manage the effects of software evolution by developing better methods and tools, namely, design, maintenance, refactoring, and reengineering.
- The process improvement approach studies evolution from an engineering view point. Therefore methods, tools, and activities that provide the means to direct, implement, and control software evolution

Why Do We Need Models to Manage & Execute Software Maintenance Activities?

- Change, Evolution, and system configuration complicate maintenance activities.
- The software product released to a customer is in the form of executable code, whereas the corresponding "product" is source code. Thus, strict control must be kept, otherwise exact source code representation of a particular executable version may not exist.
- Three maintenance models will be explained:

1. Reuse Oriented → old
2. Simple Staged → relatively new
3. Change Mini-cycle → still in research

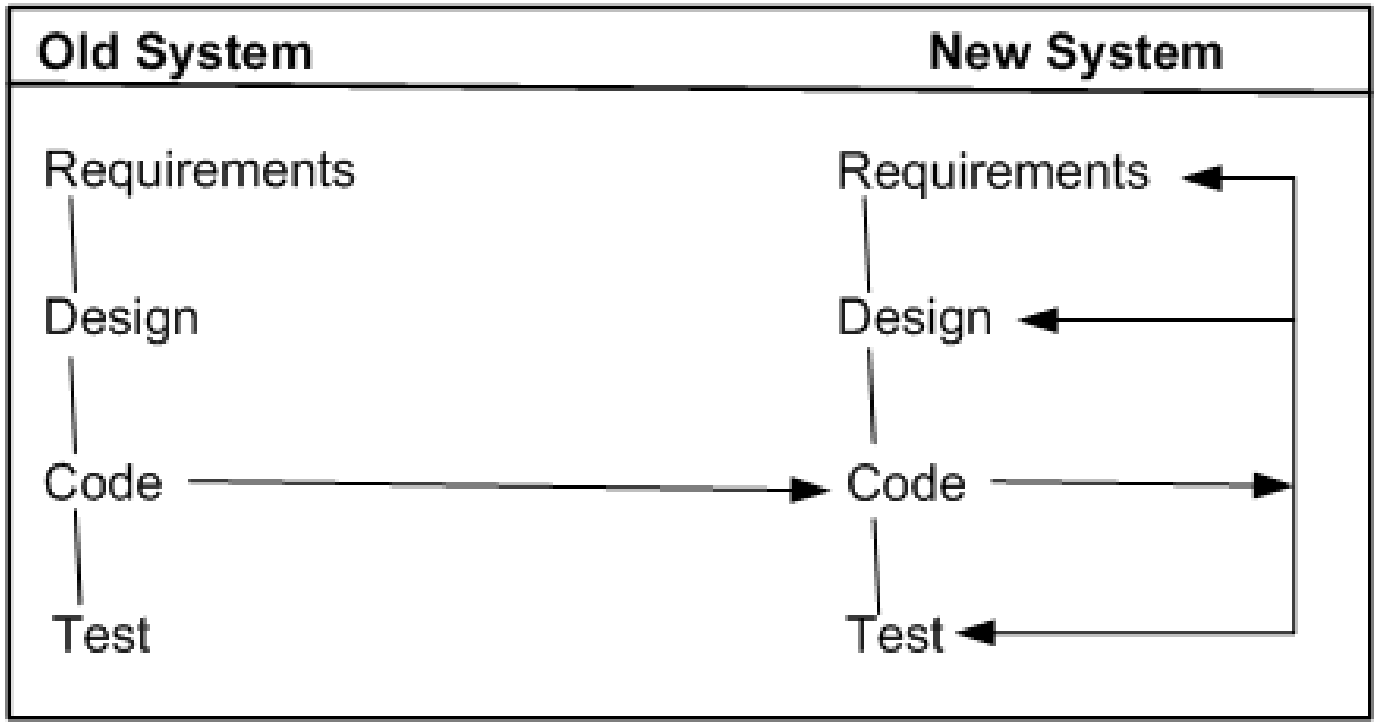
Reuse-oriented Model

- One obtains a new version of an old system by modifying one or several components of the old system and possibly adding new components.
- In Reuse-oriented Model three process models for maintenance have been proposed by Basili (1990):

1. Quick fix model
2. Iterative enhancement model
3. Full reuse model

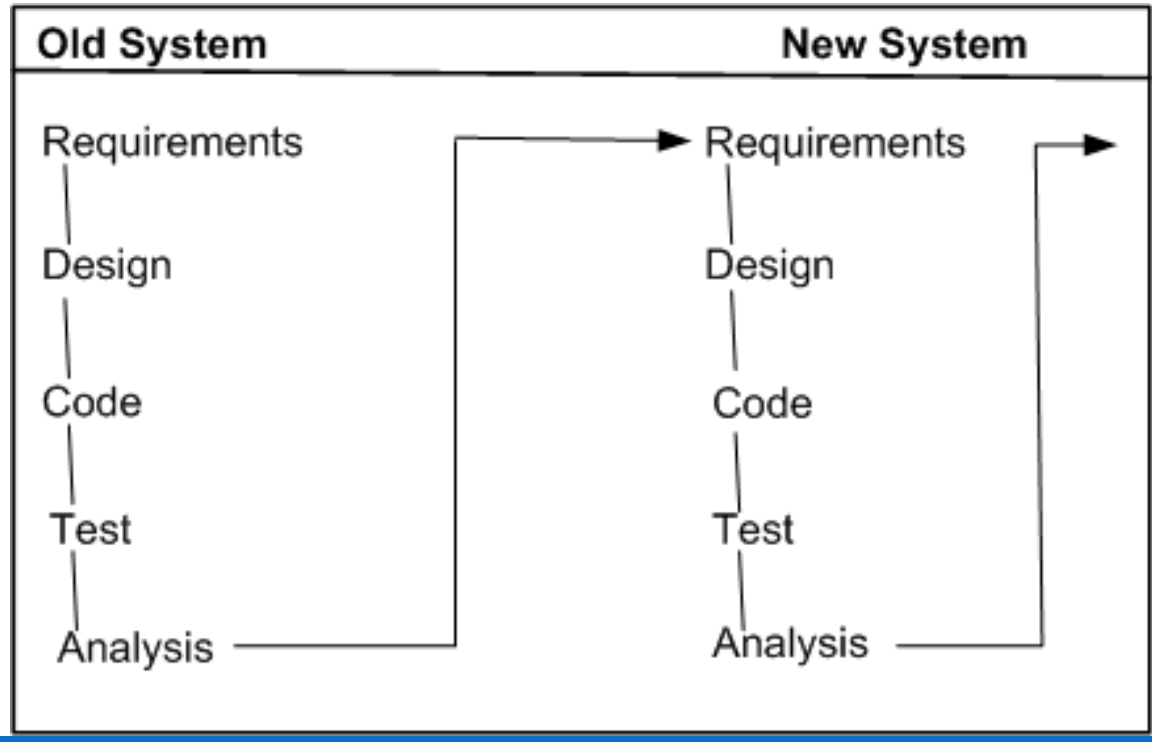
Reuse-oriented -Quick Fix Model

- One obtains a new version of an old system by modifying code of existing one
- Necessary changes are quickly made to the code and then to the accompanying documentation
- Often no impact analysis or ripple effect are studied.



Reuse-oriented - Iterative Enhancement Model

- It begins with the existing system's artifacts, namely, requirements, design, code, test, and analysis documents.
- It revises the highest-level documents affected by the changes and propagates the changes down through the lower-level documents.
- The model allows maintainers to redesign the system, based on the analysis of the existing system.

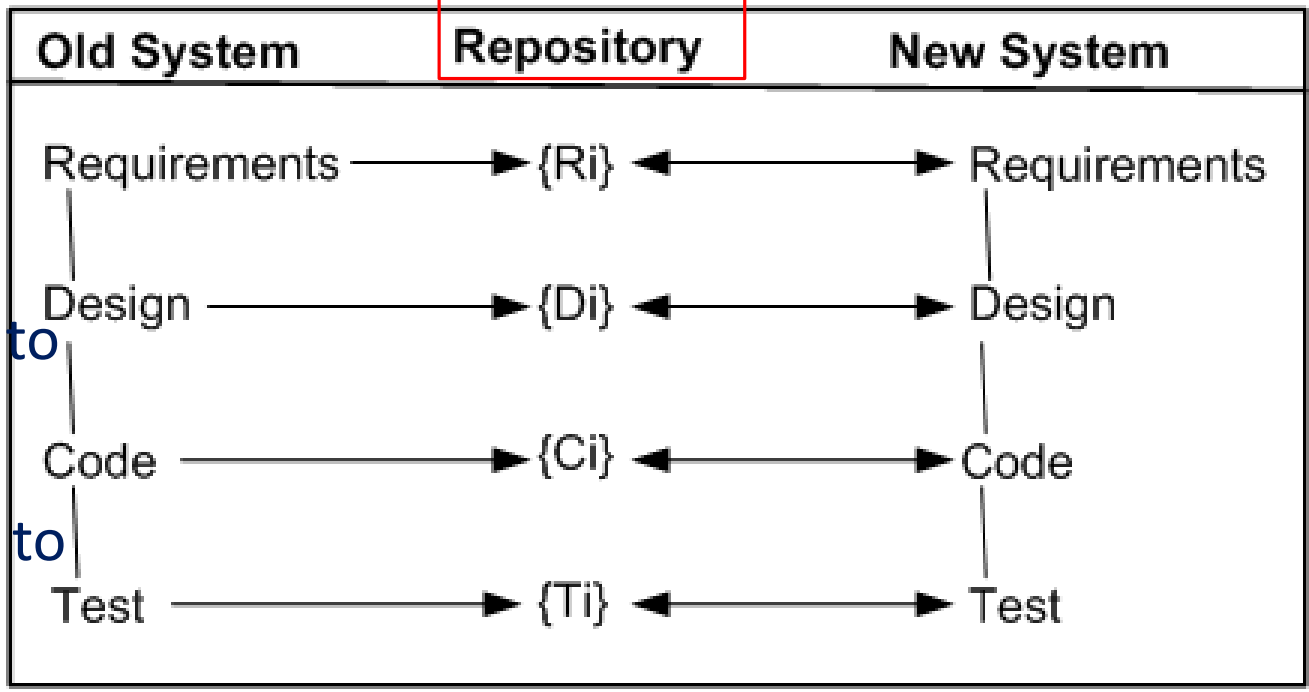


Reuse-oriented - Full Reuse Model

- the following activities are performed:

1. Identify from the old system components candidates for reuse.
2. Understand the identified system components.
3. Modify the old system components to support the new requirements.
4. Integrate the modified components to form the newly developed system.

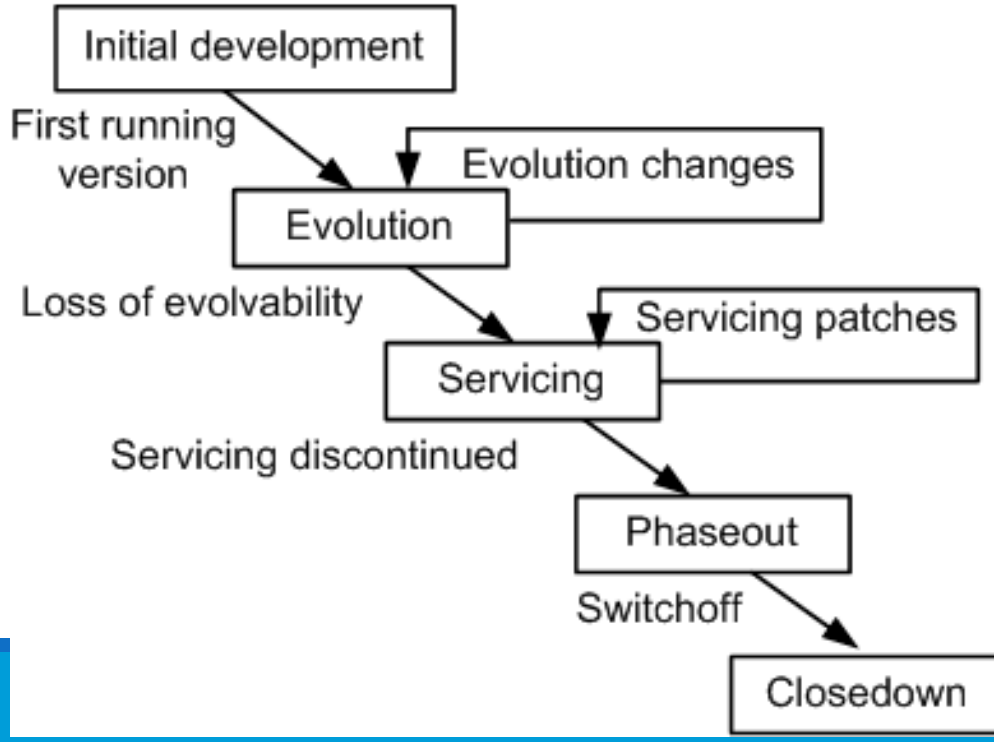
repository of artifacts of earlier versions of the systems.



The Staged Model for CSS

- It is a descriptive model of software evolution presented by Rajlich and Bennett (2000)
- The objective is to improve understanding of how long-lived software evolves, rather than aiding in its management.
- It divides the lifespan of a system into four stages:

1. Initial development
2. Evolution
3. Servicing
4. Phase out

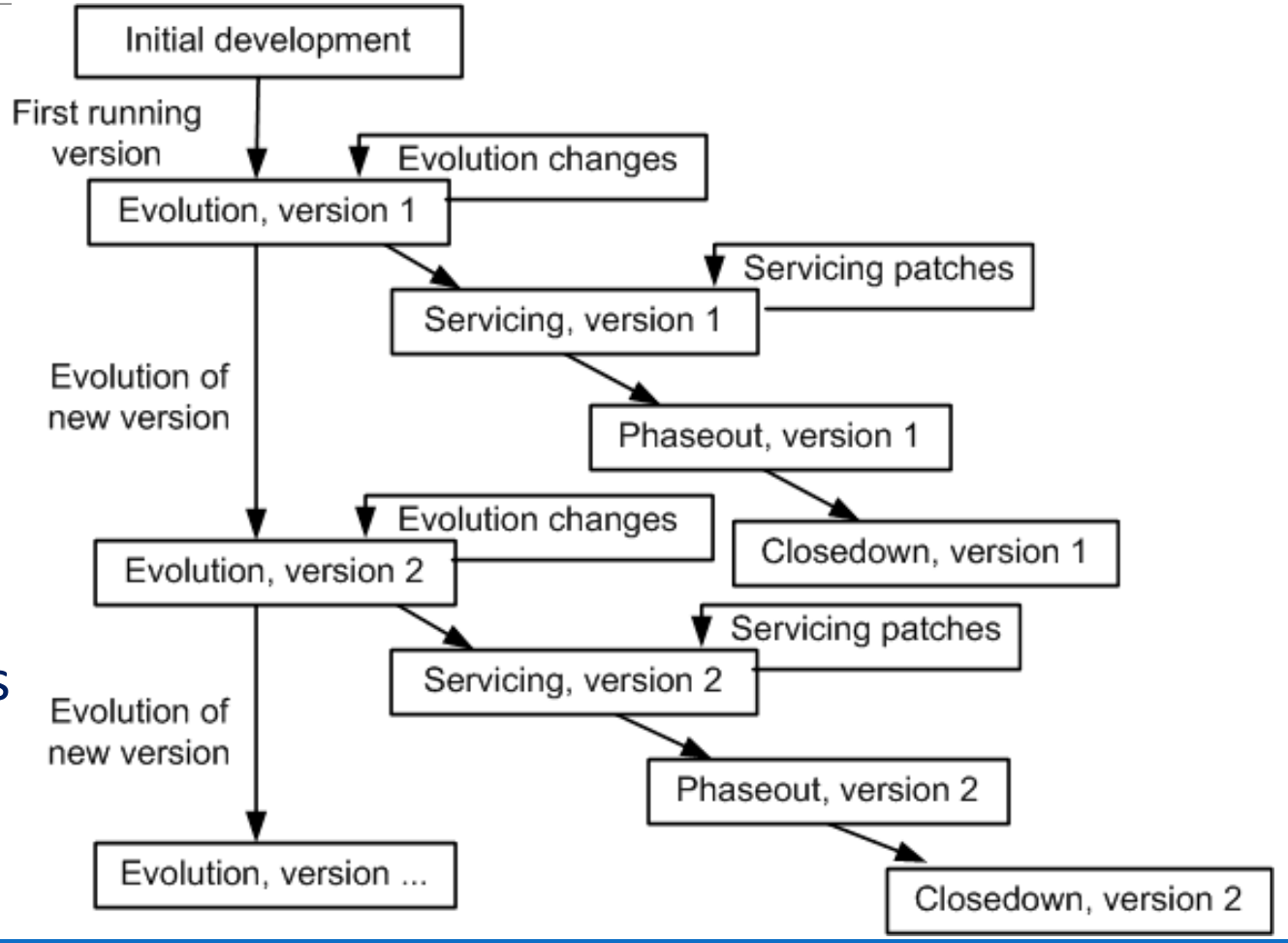


The Staged Model for CSS

1. Initial development: the first delivered version is produced. Knowledge about the system is fresh and constantly changing.
2. Evolution: Simple changes are easily performed and more major changes are possible. The cost and risk are greater than in the previous stage. Knowledge about the system is still good. For many systems, most of its lifespan is spent in this phase.
3. Servicing: The system is no longer a key asset. Concentration is mainly on maintenance tasks rather than architectural or functional change. Knowledge about the system has lessened and the effects of change have become harder to predict. The costs and risks of change have increased significantly.
4. Phase out: It has been decided to replace or eliminate the system entirely, either because the costs of maintaining the old system have become prohibitive or because there is a newer solution waiting in the pipeline. An exit strategy involving techniques such as wrapping and data migration is defined.

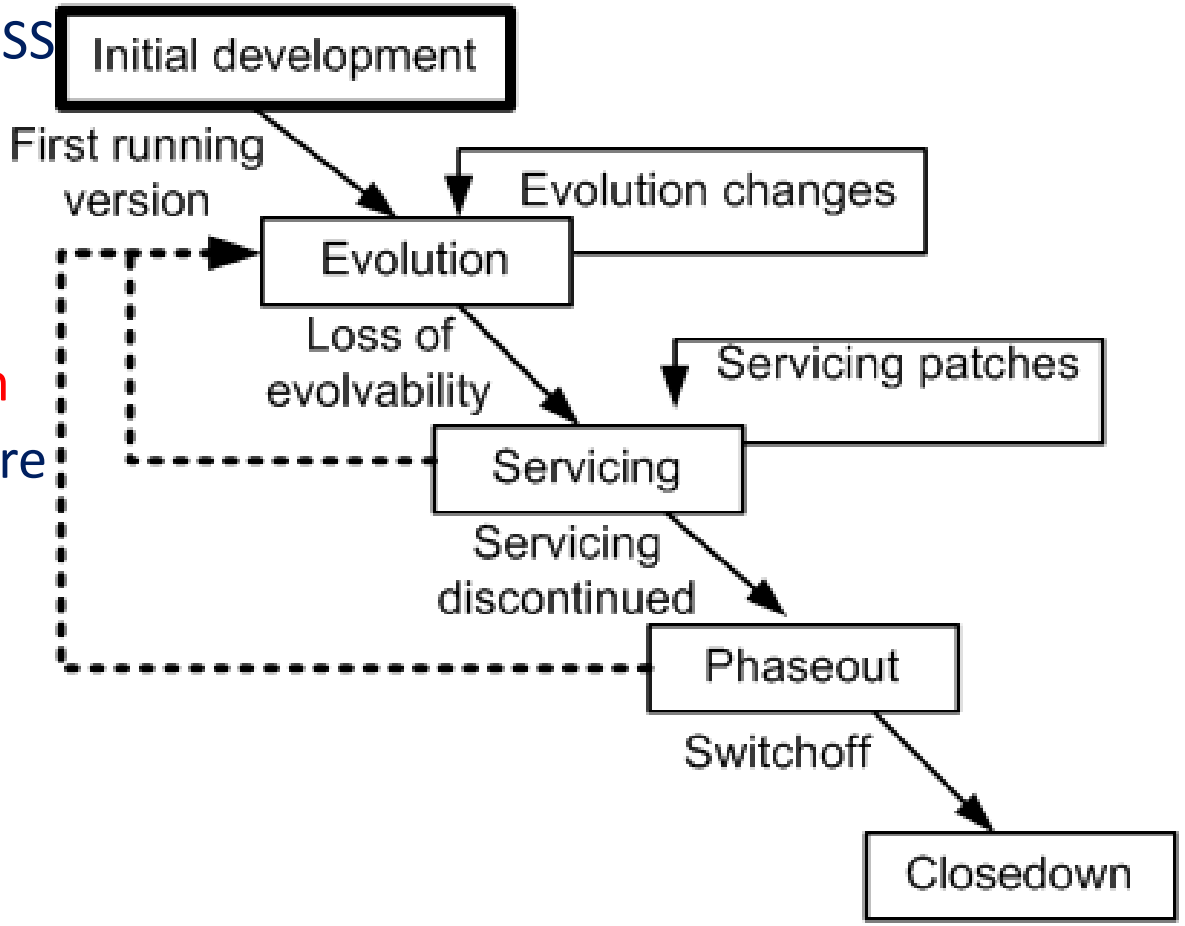
The Staged Model- Versioned Staged Model

- The evolution process is the backbone of the model.
- Each evolution track includes servicing, phaseout, and closedown
- A scheme such as <product><version><release><build>, Is used to reflect the strategic changes made to the system during evolution



The Staged Model- For FOSS

- In 2007 Capiluppi et al studied evolution of FOSS and adapted the Staged Model for Software Evolution
- In FOSS some systems after a transition from Evolution to Servicing, a new period of evolution was observed. This possibility is depicted in Figure as a broken arc from the Servicing stage to the Evolution stage.
- Also possibility of a transition from Phaseout stage to Evolution stage due to change in active developers .



Questions

?

Quiz

In the Intention-based model of Software Maintenance

- What are the causes of Adaptive maintenance?
- What are the causes of Preventative maintenance ?

Reading

- Chapter 2: 2.1,2.2
- Chapter 3: 3.1, 3.2,3.3