Taxonomy

Idaho State University | Computer Science

Isaac Griffith

CS 4423 and CS 5523
Department of Computer Science
Idaho State University

ROAR

Outcomes

After today's lecture you will:

- Have an understanding of different approaches to classifying software maintenance
- Have an understanding of the processes used during software maintenance

SME Taxonomies

CS 4423/5523

ROAR

General Idea

- In circa 1972, R.G. Canning in his landmark article “The Maintenance ‘Iceberg’,” discussed the problems of software maintenance
- Practitioners took a narrow view of maintenance as
  - correcting errors, and
  - enhancing the functionalities of the software
- The ISO/IEC 14764 standard defines software maintenance as
  - “... the totality of activities required to provide cost-effective support to a software system. Activities are performed during the pre-delivery stage as well as the post-delivery stage.”
    - Post-delivery activities includes changing software, providing training, and operating a help desk.
    - Pre-delivery activities include planning for post-delivery operations.

Intention-based Classification

Maintenance activities are divided into four groups:

- **Corrective maintenance:** The purpose of corrective maintenance is to correct failures: processing failures and performance failures.
- **Adaptive maintenance:** The purpose of adaptive maintenance is to enable the system to adapt to changes in its data environment or processing environment.
- **Perfective maintenance:** The purpose of perfective maintenance is to make a variety of improvements, namely, user experience, processing efficiency, and maintainability.
- **Preventive maintenance:** The purpose of preventive maintenance is to prevent problems from occurring by modifying software products.

Intention-based Classification

- **Corrective maintenance:** The purpose of corrective maintenance is to correct failures: processing failures and performance failures.
- Examples of corrective maintenance:
  - A program producing a wrong output is an example of processing failure.
  - Similarly, a program not being able to meet real-time requirements is an example of performance failure.
- The process of corrective maintenance includes isolation and correction of defective elements in the software.
- There is a variety of situations that can be described as corrective maintenance such as correcting a program that aborts or produces incorrect results.
- Basically, corrective maintenance is a reactive process, which means that corrective maintenance is performed after detecting defects with the system.

Intention-based Classification

- **Adaptive maintenance:** The purpose of adaptive maintenance is to enable the system to adapt to changes in its data environment or processing environment.
- This process modifies the software to properly interface with a changing or changed environment.
- Adaptive maintenance includes system changes, additions, deletions, modifications, extensions, and enhancements to meet the evolving needs of the environment in which the system must operate.
- Examples of Adaptive maintenance are:
  - changing the system to support new hardware configuration;
  - converting the system from batch to on-line operation; and
  - changing the system to be compatible with other applications.

Intention-based Classification

- **Perfective maintenance:** The purpose of perfective maintenance is to make a variety of improvements, namely, user experience, processing efficiency, and maintainability.
- Examples of perfective maintenance are:
  - the program outputs can be made more readable for better user experience;
  - the program can be modified to make it faster, thereby increasing the processing efficiency;
  - and the program can be restructured to improve its readability, thereby increasing its maintainability.
- Activities for perfective maintenance include restructuring of the code, creating and updating documentations, and tuning the system to improve performance.
- It is also called “reengineering”.

Intention-based Classification

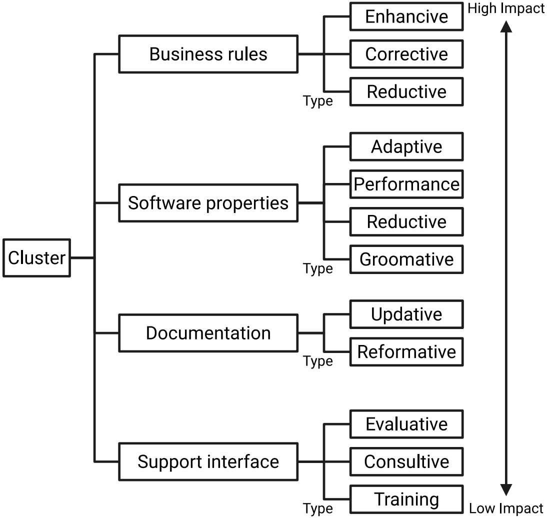
- **Preventive maintenance:** The purpose of preventive maintenance is to prevent problems from occurring by modifying software products.
- Basically, one should look ahead, identify future risks and unknown problems, and take actions so that those problems do not occur.
- Preventive maintenance is very often performed on safety critical and high availability software systems.
- The concept of "**software rejuvenation**" is a preventive maintenance measure to prevent, or at least postpone, the occurrences of failures (crash) due to continuously running the software system.
- It involves occasionally terminating an application or a system, cleaning its internal state, and restarting it.

Activity-based Classification

- Kitchenham et al. organize maintenance modification activities into two categories:
  - **Corrections**
    Activities in this category are designed to fix defects in the system, where a defect is a discrepancy between the expected behavior and the actual behavior of the system.
  - **Enhancements**
    Activities in this category are designed to effect changes to the system. It is further divided into three subcategories as follows:
    - enhancements activities that modify some of the existing requirements implemented by the system;
    - enhancement activities that add new system requirements; and
    - enhancement activities that modify the implementation without changing the requirements implemented by the system.

Evidence-based Classification

- Twelve types of maintenance activities were grouped into four clusters.
- Modifications performed, detected, or observed on four aspects of the system being maintained are used as the criteria to cluster the types of maintenance activities:
  - the whole software
  - the external documentation
  - the properties of the program code
  - the system functionality experienced by the customer.



Evidence-based Classification

- Classification of maintenance activities is based on changes in the four kinds of entities
  - the whole software:
  - the external documentation;
  - the properties of the program code; and
  - the system functionality experienced by the customer

- Evidence of changes to those entities is gathered by comparing the appropriate portions of the software before the activity with the appropriate parts after the execution of the activity.

Evidence-based Classification

- **Training:** This means training the stakeholders about the implementation of the system.

- **Consultive:** In this type, cost and length of time are estimated for maintenance work, personnel run a help desk, customers are assisted to prepare maintenance work requests, and personnel make expert knowledge about the available resources and the system to others in the organization to improve efficiency.

- **Evaluative:** In this type, common activities include reviewing the program code and documentations, examining the ripple effect of a proposed change, designing and executing tests, examining the programming support provided by the operating system, and finding the required data and debugging.

Evidence-based Classification

- **Reformative:** Ordinary activities in this type improve the readability of the documentation, make the documentation consistent with other changes in the system, prepare training materials, and add entries to a data dictionary.

- **Updative:** Ordinary activities in this type are substituting out-of-date documentation with up-to-date documentation, making semi-formal, say, in UML to document current program code, and updating the documentation with test plans.

- **Groomative:** Ordinary activities in this type are substituting components and algorithms with more efficient and simpler ones, modifying the conventions for naming data, changing access authorizations, compiling source code, and doing backups.

Evidence-based Classification

- **Preventive:** Ordinary activities in this type perform changes to enhance maintainability, and establish a base for making a future transition to an emerging technology.
- **Performance:** Activities in performance type produce results that impact the user. Those activities improve system up time and replace components and algorithms with faster ones.
- **Adaptive:** Ordinary activities in this type port the software to a different execution platform, and increase the utilization of COTS components.

Evidence-based Classification

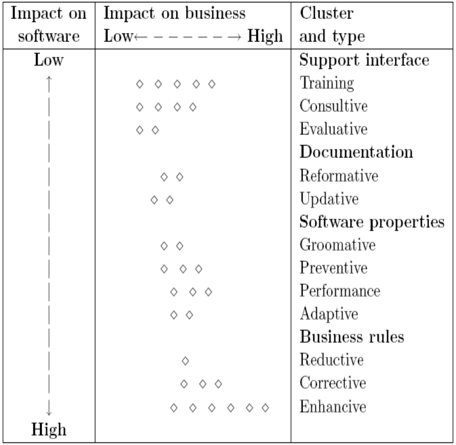
- **Reductive:** Ordinary activities in this type drop some data generated for the customer, decreasing the amount of data input to the system, and decreasing the amount of data produced by the system.

- **Corrective:** Ordinary activities in this type are correcting identified bugs, adding defensive programming strategies, and modifying the ways exceptions are handled.

- **Enhancive:** Ordinary activities in this type are adding and modifying business rules to enhance the system’s functionality available to the customer, and adding new data flows into or out of the software.

Evidence-based Classification

- The impacts of the different types of maintenance activities are summarized in the table.
- The first dimension is the customer's ability to perform its business function while continuing to use the system. This is arranged based on the # of diamonds
- The second dimension is the software. This is arranged from top to bottom.



Impact on software	Impact on business Low← – – – – – – → High	Cluster and type
Low		Support interface
↑	◇ ◇ ◇ ◇ ◇	Training
|	◇ ◇ ◇ ◇	Consultive
|	◇ ◇	Evaluative
|		Documentation
|	◇ ◇	Reformative
|	◇ ◇	Updative
|		Software properties
|	◇ ◇	Groomative
|	◇ ◇ ◇	Preventive
|	◇ ◇ ◇	Performance
|	◇ ◇	Adaptive
|		Business rules
|	◇	Reductive
|	◇ ◇ ◇	Corrective
↓	◇ ◇ ◇ ◇ ◇ ◇	Enhancive
High		

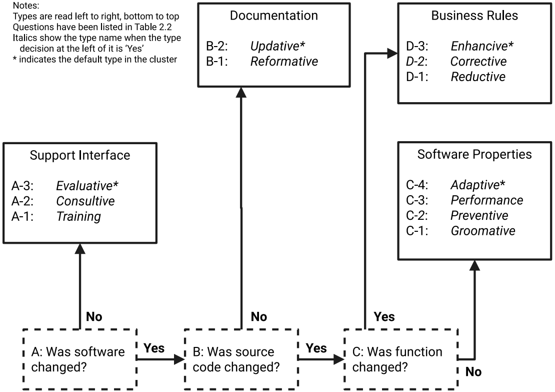
Table 2.2: Impact of the types [15] (©[2001] John Wiley)

Decision Tree Based Criteria

Activities are classified into different types by applying a two-step decision process

❶ Apply criteria based decisions to make the clusters of types.

❷ Apply the type decisions to identify one type within the cluster.



Notes:
Types are read left to right, bottom to top
Questions have been listed in Table 2.2
Italics show the type name when the type decision at the left of it is 'Yes'
* indicates the default type in the cluster

Documentation
B-2: *Updative**
B-1: *Reformative*

Business Rules
D-3: *Enhancive**
D-2: *Corrective*
D-1: *Reductive*

Support Interface
A-3: *Evaluative**
A-2: *Consultive*
A-1: *Training*

Software Properties
C-4: *Adaptive**
C-3: *Performance*
C-2: *Preventive*
C-1: *Groomative*

A: Was software changed?

B: Was source code changed?

C: Was function changed?

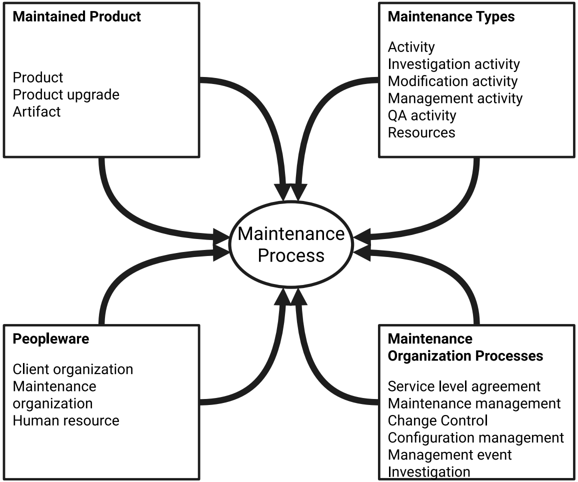
Decision Tree Based Criteria

- Sometimes, an objective evidence may be found to be ambiguous. In that case, clusters have their designated default types for use. The overall default type is evaluative, if there are ambiguities in an activity

Criteria	Type decision question	Type
A-1	To train the stakeholders, did the activities utilize the software as subject?	Training
A-2	As a basis for consultation, did the activities employ the software?	Consultive
A-3	Did the activities evaluate the software?	Evaluative
B-1	To meet stakeholder needs, did the activities modify the non-code documentation?	Reformative
B-2	To conform to implementation, did the activities modify the non-code documentation?	Updative
C-1	Was maintainability or security changed by the activities?	Groomative
C-2	Did the activities constrain the scope of future maintenance activities?	Preventive
C-3	Were performance properties or characteristics modified by the activities?	Performance
C-4	Were different technology or resources used by the activities?	Adaptive
D-1	Did the activities constrain, reduce, or remove some functionalities experienced by the customer?	Reductive
D-2	Did the activities fix bugs in customer-experienced functionality?	Corrective
D-3	Did the activities substitute, add to, or expand some functionalities experienced by the customers?	Enhancive

Categories of Maintenance Concepts

The four categories and the concepts that influence the maintenance process are illustrated below:



Maintained Product

Product
Product upgrade
Artifact

Maintenance Types

Activity
Investigation activity
Modification activity
Management activity
QA activity
Resources

Maintenance Process

Peopleware

Client organization
Maintenance organization
Human resource

Maintenance Organization Processes

Service level agreement
Maintenance management
Change Control
Configuration management
Management event
Investigation

Maintained Product

The maintained product dimension is characterized by three concepts:

- **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 base line.

  An upgrade can create a new version of the system being maintained, a patch code for an object, or event 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, COTS products, and object code components.

The key elements of the maintained product are size, age, application type, composition and quality.

Maintained Product

Life Cycle Stage	Maintenance Task	User Population
Initial development	–	–
Evolution	Corrections, enhancements	Small
Servicing	Corrections	Growing
Phaseout	Corrections	Maximum
Closedown	Corrections	Declining

Maintenance Types

Four types of maintenance activities are defined:

- **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.

Activity: an activity accepts some artifacts as inputs and produces new or changed artifacts.

Artifacts: Artifacts include plans, documents, system representations, and source and object code items. Artifacts are created during the software development process and changed during maintenance.

Maintenance Organization Processes

Two different levels of maintenance processes are followed within a maintenance organization:

- **individual-level maintenance processes** followed by maintenance personnel to implement a specific change request, and
- **Organization-level process** followed to manage the change requests from maintenance personnel, users, and customers/clients.

Maintenance Organization Processes

Three major elements of a maintenance organization are:

- **Event management:** The stream of events, namely, all the change requests from various sources, received by the maintenance organization is handled in an event management process.
- **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.
- **Change control system:** 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.

Maintenance Organization Processes

- A maintenance organization handles maintenance change request from:
  1. **Users,**
  2. **Customers, and**
  3. **Maintainers**
- After an initial investigation of a change request, a management process is put in place for approving change activities.
- Approval of a change request is normally the responsibility of a change control board.
- A proposed modification activity is scheduled only after the modification is approved by the board and a Service Level Agreement (SLA) is signed with the client.
- Service level agreement **(SLA)** is a contract between the customers and the providers of a maintenance service. Performance targets for the maintenance services are specified in the **SLA**.

Maintenance Organization Processes

In general, maintenance organizations use three different support levels to organize the staff:

- **Level 1:** This group files problem reports and identifies the technical support person who can best assist the person reporting a problem.
- **Level 2:** This level includes experts who know how to communicate with users and analyze their problems. These people recommend quick fixes and temporary workarounds.
- **Level 3:** This level includes programmers who can perform actual changes to the product software.

Peopleware

- Maintenance activities cannot ignore the human element, because software production and maintenance are human intensive activities.
- The three people-centric concepts related to maintenance are as follows:
  1. **Maintenance organization:** This is the organization that maintains the product(s).
  2. **Client organization:** A client organization uses the maintained system.
  3. **Human resource:** Human resources includes personnel from the maintenance and client organizations.

Peopleware

Finally, the following user and customer issues affect maintenance:

- **Size:** The size of the customer base and the number of licenses they hold affect the amount of effort needed to support a system.
- **Variability:** High variability in the customer base impacts the scope of maintenance tasks.
- **Common goals:** The extent to which the users and the customer have common goals affect the SLAs.

Ultimately, customer fund maintenance activities. If the customers do not have a good understanding of the requirements of the actual users, some SLAs may not be appropriate to the end users.

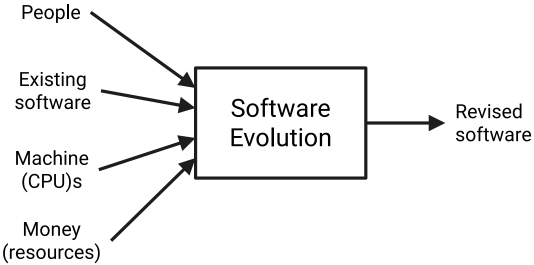
Evolution of Software Systems

- The term **evolution** was used by Mark I. Halpern in circa 1965 to define the dynamic growth of software.
- It attracted more attention in the 1980s after Meir M. Lehman proposed eight broad principles about how certain types of software systems evolve.
- Bennett and Rajlich researched the term “software evolution,” but found no widely accepted definition of the term.
- Some researchers and practitioners used the term **software evolution** as a substitute for the term **software maintenance**.

Evolution of Software Systems

Lowell Jay Arthur distinguished the two terms as follows:

- **Maintenance** means preserving software from decline or failure.
- **Evolution** means a continuously changing software from a worse state to a better state.



- Software evolution is like a computer program, with inputs, processes, and outputs.
- Keith H. Bennett and Jie Xu use **maintenance** for all post-delivery support, whereas they use **evolution** to refer to perfective modifications - modifications triggered by changes in requirements.

Evolution of Software Systems

Software evolution is studied with two broad, complementary approaches:

- **Explanatory (What/Why):**
  1. This approach attempts to explain the causes of software evolution, the processes used, and the effects of software evolution.
  2. The explanatory approach studies evolution from a scientific view point.
- **Process Improvement (How):**
  1. This approach attempts to manage the effects of software evolution by developing better methods and tools, namely, design, maintenance, refactoring, and reengineering
  2. The process improvement approach studies evolution from an engineering view point.

ROAR

For Next Time

- Review EVO Chapter 2.1 - 2.3.1
- Read EVO Chapter 2.3.1 - 2.5
- Watch the Lecture 03

Are there any questions?