Faculty of Computers and Artificial Intelligence

Software Maintenance & Evolution

Dr. Lamia Abo Zaid

د. لمياء أبوزيد

COURSE CODE : SCS358

L.ABOZAID@FCI-CU.EDU.EG

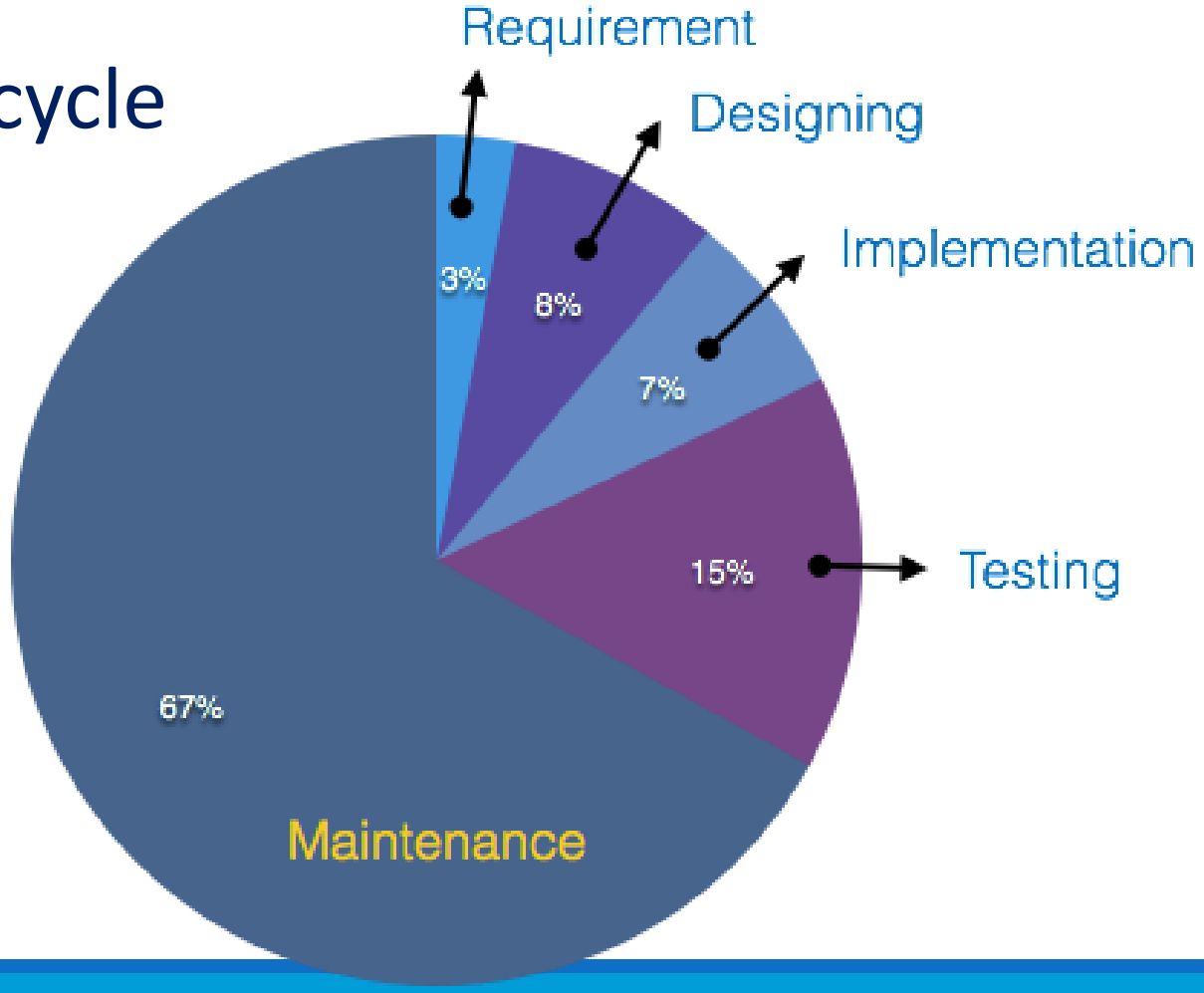
Software Evolution : TOC

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

Software Evolution – Why ?

The cost of entire software process cycle

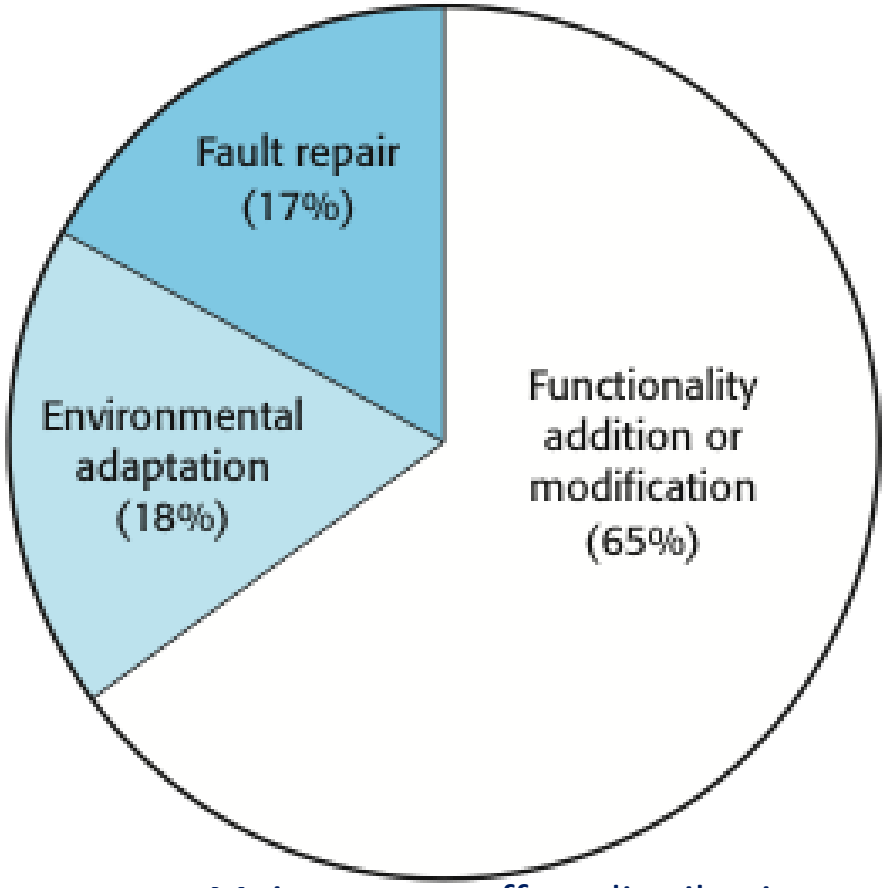
- why modifications are required?
  - **Market Conditions** - Changing Policies
  - **Client Requirements** - Over the time, customer may ask for new features or functions.
  - **Host Modifications** - changes in hardware and/or platform (such as operating system).
  - **Organization Changes** - business level changes



Software Evolution – Why ?

The cost of entire software process cycle

- why modifications are required?
  - **Market Conditions** - Changing Policies
  - **Client Requirements** - Over the time, customer may ask for new features or functions.
  - **Host Modifications** - changes in hardware and/or platform (such as operating system).
  - **Organization Changes** - business level changes



Maintenance effort distribution

Software Evolution: Laws of Lehman

Applies to closed source systems - E Type Systems

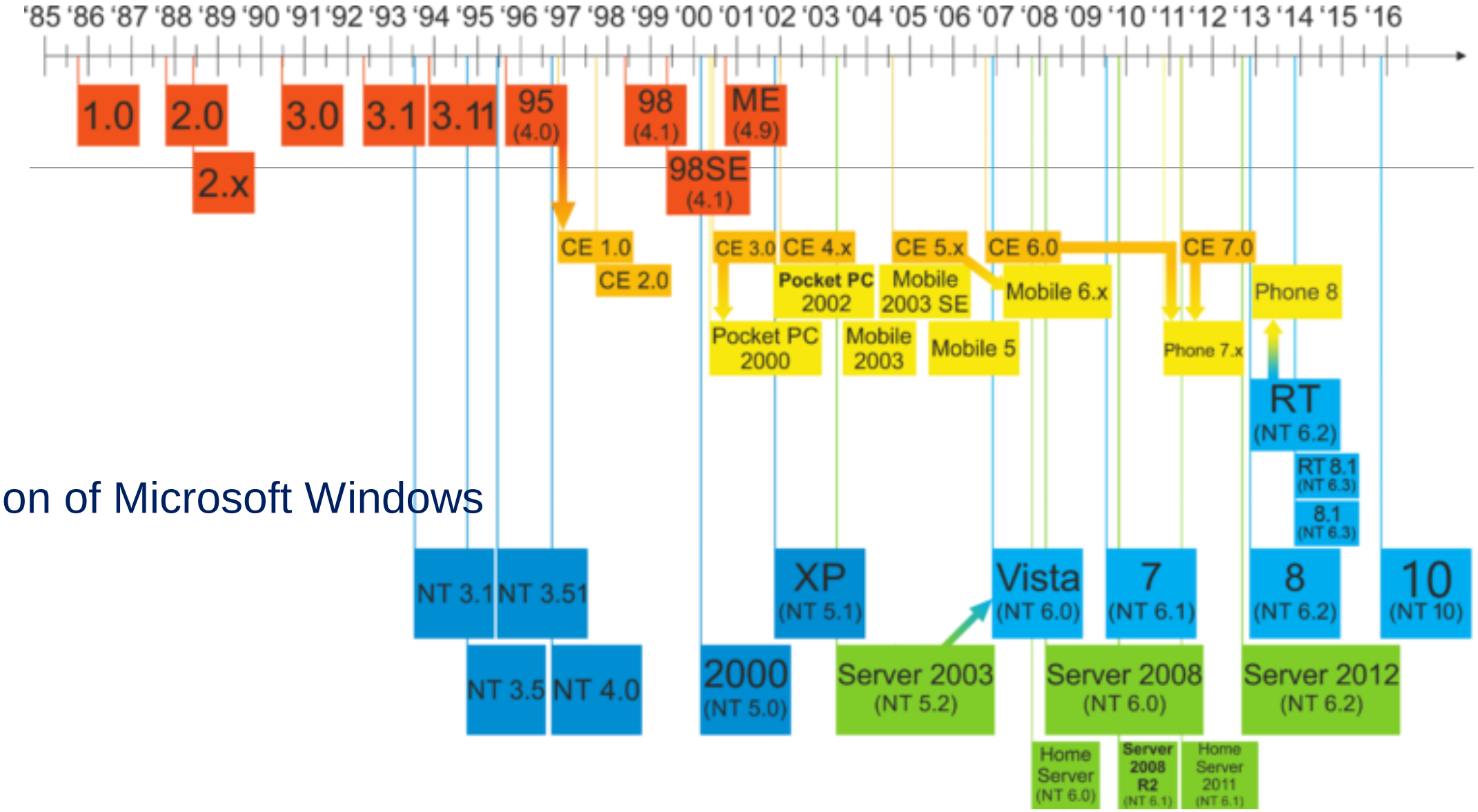
- *Continuing change* (1st) – A system will become progressively less satisfying to its user over time, unless it is continually adapted to meet new needs.
- *Increasing complexity* (2nd) – A system will become progressively more complex, unless work is done to explicitly reduce the complexity.
- *Self-regulation* (3rd) – The process of software evolution is self-regulating with respect to the distributions of the products and process artifacts that are produced.
- *Conservation of organizational stability* (4th) – The average effective global activity rate on an evolving system does not change over time, that is the average amount of work that goes into each release is about the same.

Software Evolution: Laws of Lehman

Applies to closed source systems - E Type Systems

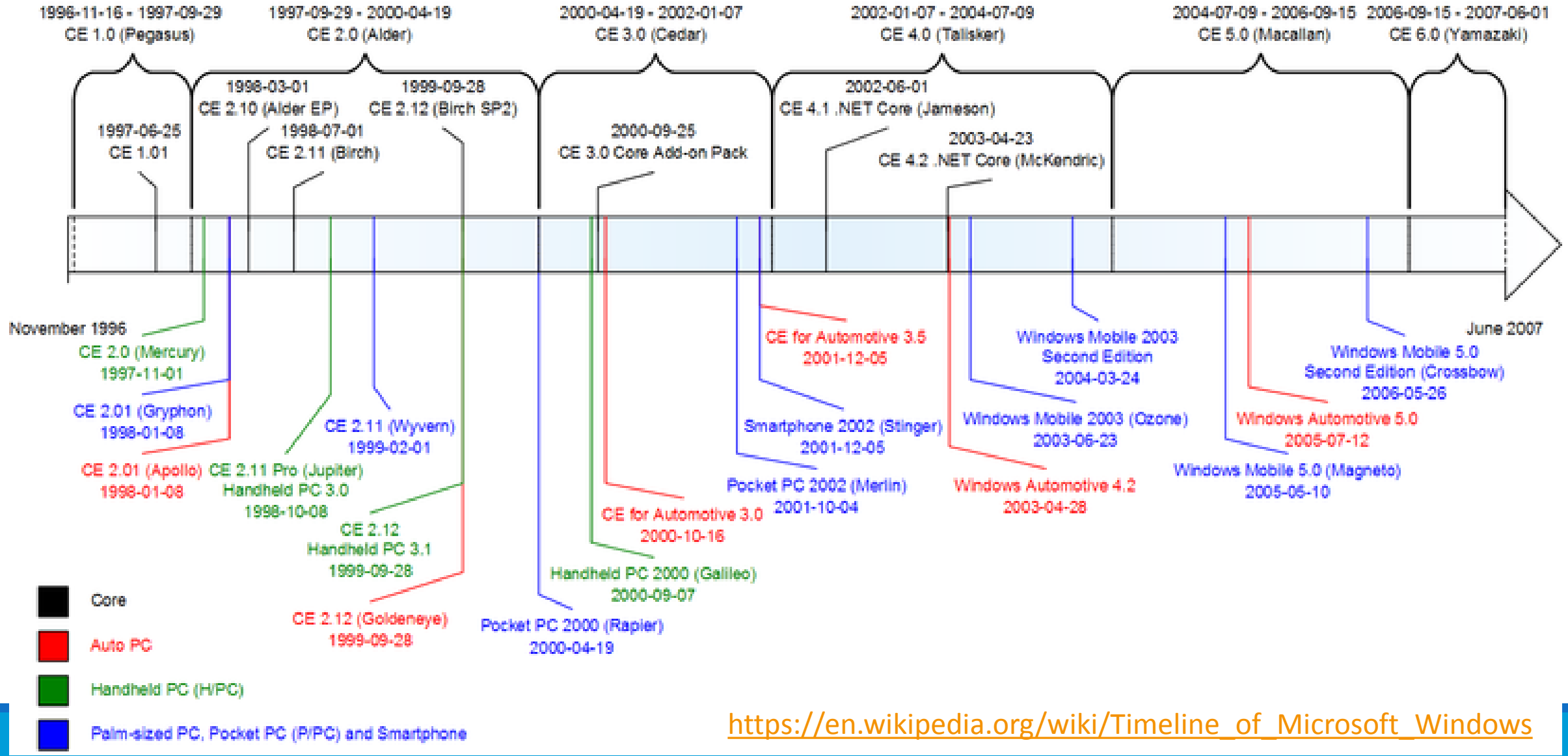
- Conservation of familiarity (5th) – The amount of new content in each successive release of a system tends to stay constant or decrease over time.
- Continuing growth (6th) – The amount of functionality in a system will increase over time, in order to please its users.
- Declining quality (7th) – A system will be perceived as loosing quality over time, unless its design is carefully maintained and adapted to new operational constraints.
- Feedback system (8th) – Successfully evolving a software system requires recognition that the development process is a multi-loop, multi-agent, multi-level feedback system.

Evolution of Microsoft Windows



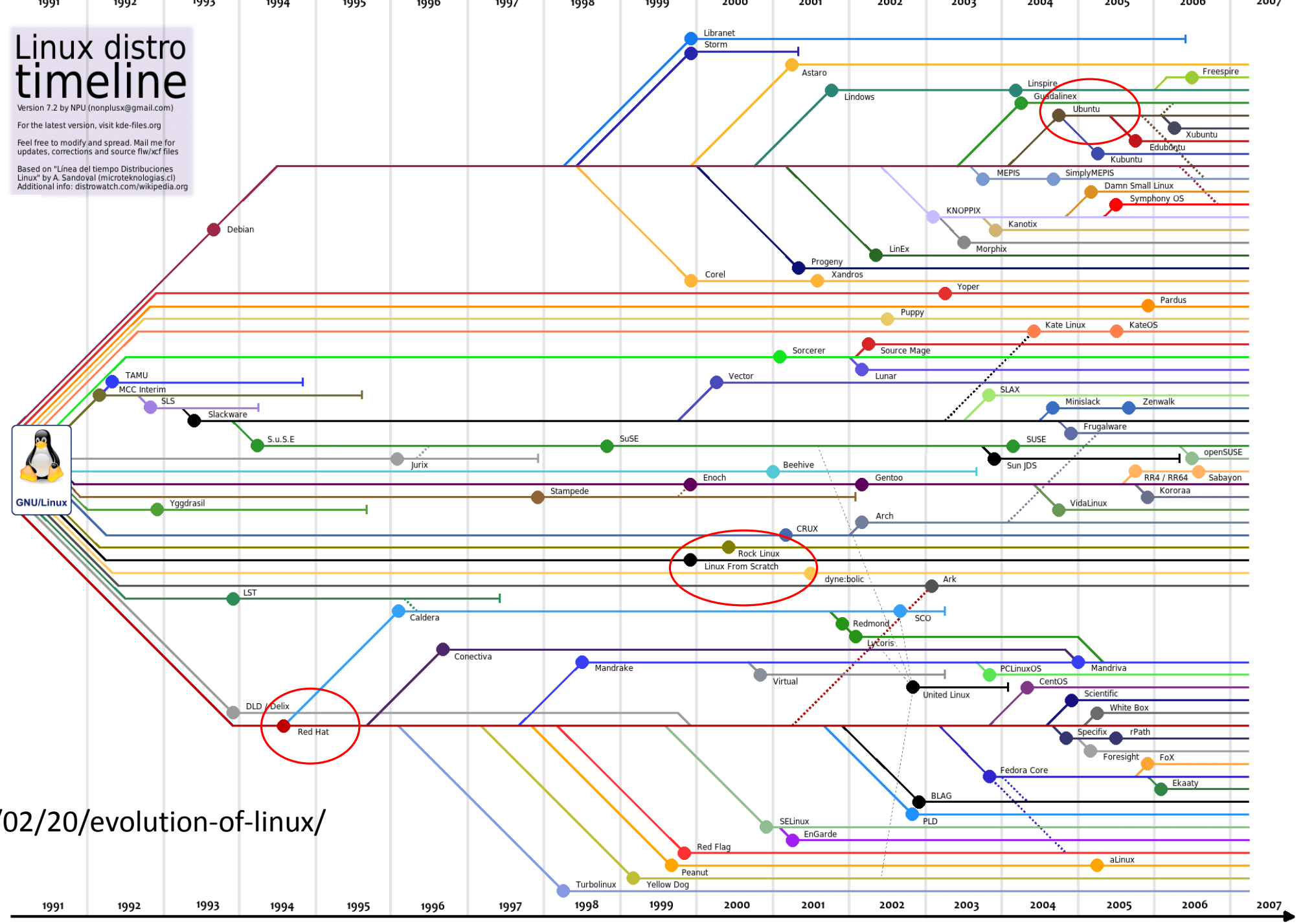
Windows CE Timeline

Source: "A Brief History of Windows CE" (http://www.hpcfactor.com/support/windowsce/), *HPC:Factor*, retrieved May 21, 2007



https://en.wikipedia.org/wiki/Timeline_of_Microsoft_Windows

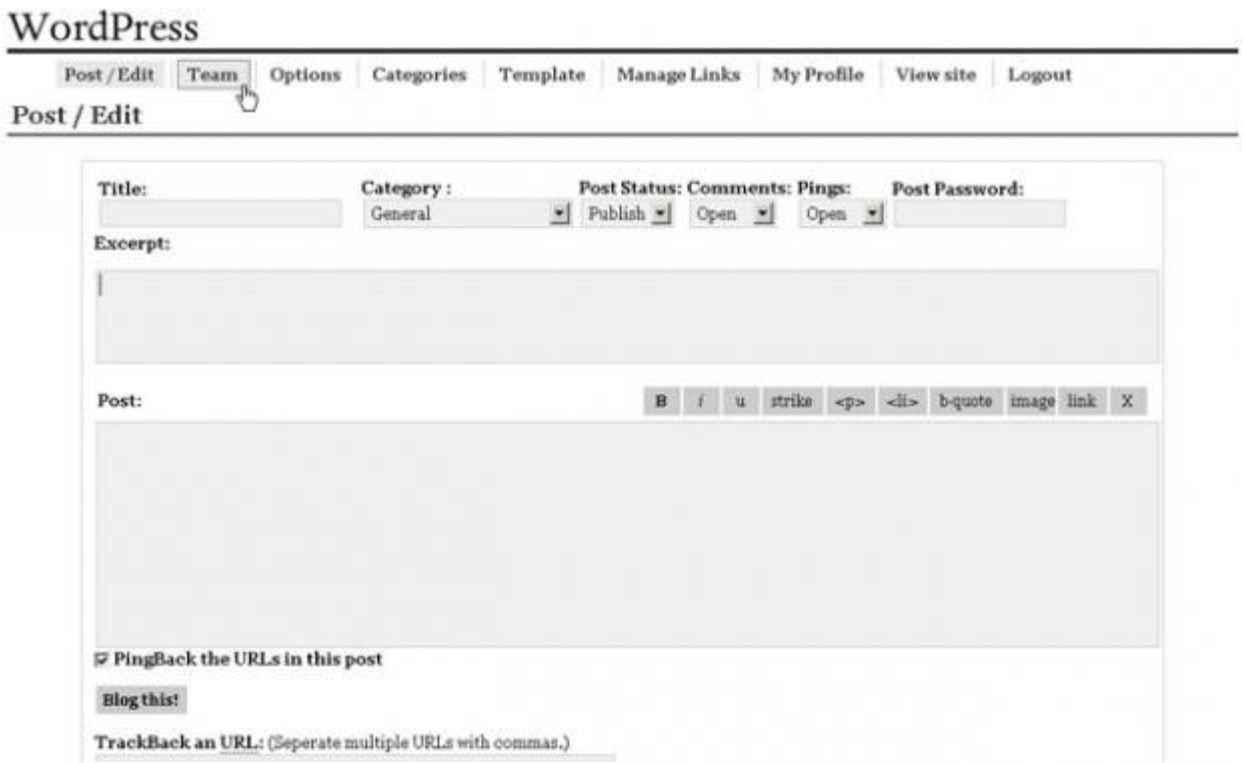
Evolution of Linux



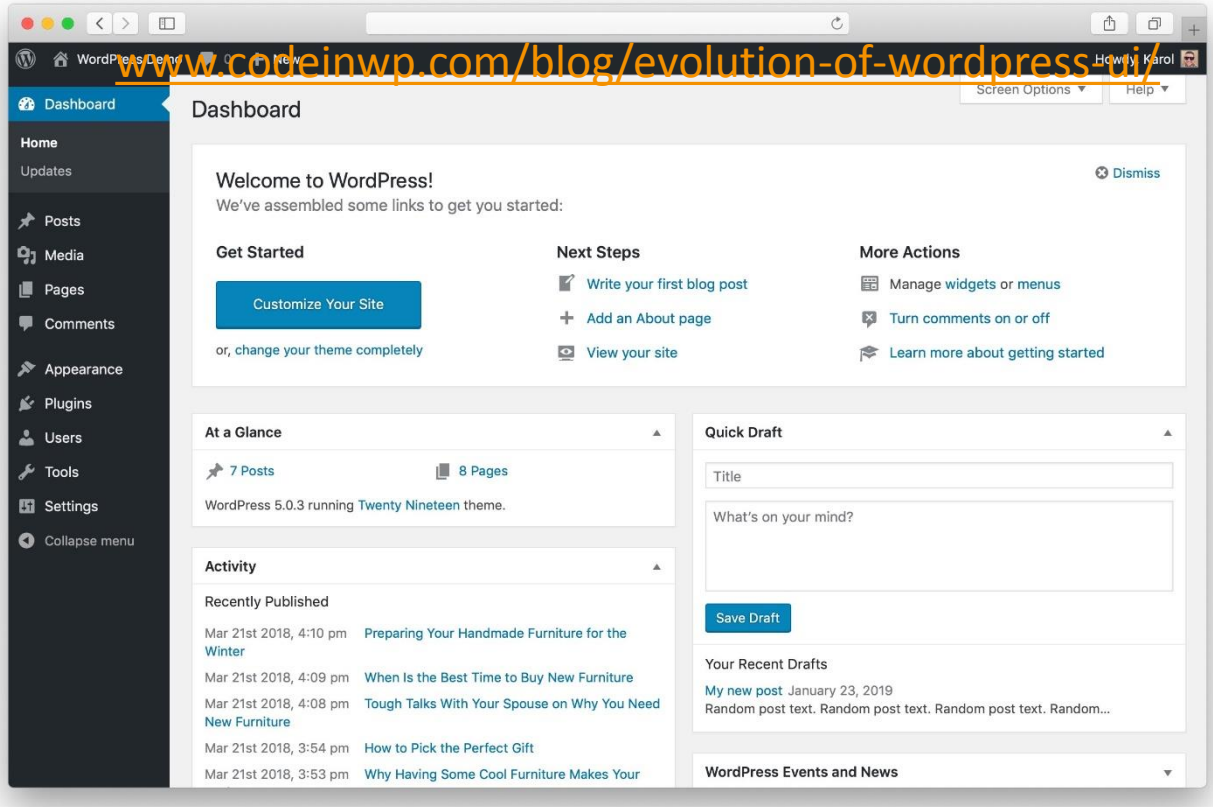
http://techpp.com/2013/02/20/evolution-of-linux/

Evolution of Wordpress

- Check out https://www.wpexplorer.com/history-wordpress-ui/



May 27, 2003 – WordPress 0.7 "Gold"



WordPress 5.0 "Bebo" – December 6, 2018

Evolution in CSS Vs. FOSS

- "FOSS is made available to the general public with either relaxed or non-existent intellectual property restrictions. The free emphasizes the freedom to modify and redistribute under the terms of the original license while open emphasizes the accessibility to the source code."
- Pirzada was the first one to study the differences between the evolution of the Unix operating system developed by Bell Laboratories and the CSS systems studied by Lehman.
- Pirzada argued that the differences in academic and industrial software development could lead to differences in the evolutionary pattern

Evolution in CSS Vs. FOSS

- In 2000, empirical study of free and open source software (FOSS) evolution was conducted by Godfrey and Tu.
  - They found that the growth trends from 1994-1999 for the evolution of FOSS Lunix OS to be super-linear, that is greater than linear.

- Robles and his collaborators concluded that Lehman's laws, 3, 4, and 5 are not fitted to large scale FOSS system such as Linux.

Software Maintenance

- There will always be defects in the delivered software application because software defect removal and quality control are not perfect.
- Therefore, software maintenance is needed to repair these defects in the released software.
- E. Burton Swanson defined three type of software maintenance:

  Corrective, Adaptive & Perfective.

- It is based on the intent of the developer towards the system. The intention of an activity depends upon the motivations for the change.

Software Maintenance

- Swanson definition was later updated in the standard for software engineering – ISO/IEC 14764.
- Introduced a fourth category called preventive maintenance.
- Some researchers and developers view preventive maintenance as a subset of perfective maintenance.

Software Maintenance

- Kitchenham described maintenance modifications in a hierarchical way based on the concept of activity:
  - Activities to make **corrections**: If there are discrepancies between the expected behavior and the actual behavior, then some activities are performed to eliminate or reduce the discrepancies.
  - Activities to make **enhancements**: A number of activities are performed to implement change to the system, thereby changing the behavior or implementation of the system.

    This category is subdivided into three types:

    - enhancements that change existing requirement
    - enhancements that add new system requirements
    - enhancements that change the implementation but not the requirements.

Maintenance of COTS Based Systems

- The motivations for performing software maintenance in component-based software systems (CBS) or commercial off-the-shelf (COTS) and custom-built software systems are the same. however major differences between the maintenance approach used exist.

This is due to the following differences :

- Skills of system maintenance teams. - Infrastructure and organization.
- COTS maintenance cost.
- Larger user community.
- Modernization.
- Split maintenance function.
- More complex planning.

Software Maintenance Life Cycle (SMLC)

- Software maintenance has its own software maintenance life cycle (SMLC) model, tasks carried are: understanding the code; modifying the code; and revalidating the code.
- Three popular SMLC models exist:

1. Iterative models.
   - systems are constructed in builds, each of which is a refinement of requirements (by considering feedback from users) of the previous build.
2. Change mini-cycle models.
   - These models consist of five major phases: **change request**, **analyze and plan change, implement change**, **verify and validate**, and **documentation change**.
   - In this process model, several important activities were identified, such as program comprehension, impact analysis, refactoring, and change propagation.

Software Maintenance Life Cycle (SMLC)

3. Staged model of maintenance and evolution.

- is descriptive in nature, and its primary objective is to improve the understanding of how long-lived software evolve
- considers four distinct, sequential stages of the lifetime of a system: Initial development, Evolution, Servicing , Phaseout

Readings

- Book Chapter 2
  - Section 2.3

Questions

?