Software Characteristics or Hardware Vs Software

To gain an understanding of software (and ultimately an understanding of software engineering), it is important to examine the characteristics of software that make it different from hardware.

- When hardware is built, the human creative process (analysis, design, construction, testing) is ultimately translated into a physical form.
- If we build a new computer, our initial sketches, formal design drawings, and bread boarded prototype evolve into a physical product (chips, circuit boards, power supplies, etc.).
- Software is a logical rather than a physical system element.
- Therefore, software has characteristics that are considerably different than those of hardware:

1. Software is developed or engineered; it is not manufactured in the classical sense:

- Although some similarities exist between software development and hardware manufacture, the two activities are fundamentally different.
- In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems that are nonexistent (or easily corrected) for software.

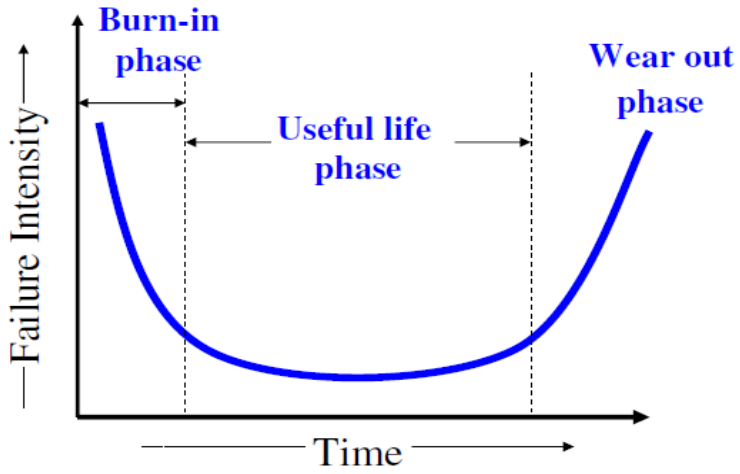

Figure 1: Bath tub curve

- Both activities are dependent on people, but the relationship between people applied and work accomplished is entirely different.
- Both activities require the construction of a "product" but the approaches are different. Software costs are concentrated in engineering.
- This means that software projects cannot be managed as if they were manufacturing projects.

2. Software does not wear out (software doesn't wear out, but it does deteriorate):

In case of Hardware

- The above figure depicts failure rate as a function of time for hardware.
- The relationship, often called the "bathtub curve," indicates that hardware exhibits relatively high failure rates early in its life (these failures are often attributable to design or manufacturing defects); defects are corrected and the failure rate drops to a steady-state level (ideally, quite low) for some period of time.
- As time passes, however, the failure rate rises again as hardware components suffer from the cumulative effects of dust, vibration, abuse, temperature extremes, and many other environmental maladies.
- Stated simply, the hardware begins to wear out.

In case of Software

- Software is not susceptible to the environmental maladies that cause hardware to wear out. In theory, therefore, the failure rate curve for software should take the form of the "idealized curve" shown in above figure.
- Undiscovered defects will cause high failure rates early in the life of a program. However, these are corrected (ideally, without introducing other errors) and the curve flattens as shown.

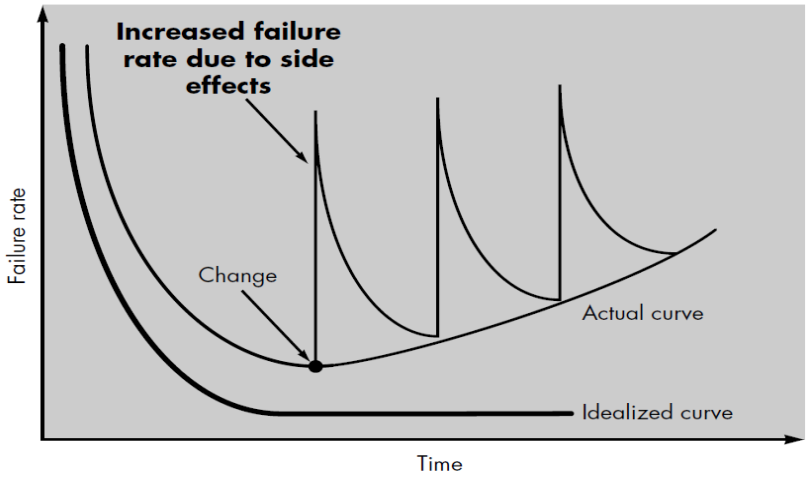

Figure 2: Ideal curve
Source: Roger S. Pressman

- The idealized curve is a gross oversimplification of actual failure models for software. However, the implication is clear—software doesn't wear out. But it does **deteriorate**.
- During its life, software will undergo change (maintenance).
- As changes are made, it is likely that some new defects will be introduced, causing the failure rate curve to spike as shown in Figure.
- Before the curve can return to the original steady-state failure rate, another change is requested, causing the curve to spike again. Slowly, the minimum failure rate level begins to rise—the software is deteriorating due to change.

3. There are no Software Spare parts:

Another aspect of wear illustrates the difference between hardware and software.

- When a hardware component wears out, it is replaced by a spare part.
- There are no software spare parts.
- Every software failure indicates an error in design or in the process through which design was translated into machine executable code.
- Therefore, software maintenance involves considerably more complexity than hardware maintenance.

4. Although the industry is moving toward component-based assembly, most software continues to be custom built.

- Consider the manner in which the control hardware for a computer-based product is designed and built.
- The design engineer draws a simple schematic of the digital circuitry, does some fundamental analysis to assure that proper function will be achieved, and then goes to the shelf where catalogs of digital components exist.
- Each integrated circuit (called an *IC* or a *chip*) has a part number, a defined and validated function, a well-defined interface, and a standard set of integration guidelines.
- After each component is selected, it can be ordered off the shelf.
- As an engineering discipline evolves, a collection of standard design components is created.

- Standard screws and off-the-shelf integrated circuits are only two of thousands of standard components that are used by mechanical and electrical engineers as they design new systems.
- The reusable components have been created so that the engineer can concentrate on the truly innovative elements of a design, that is, the parts of the design that represent something new.
- In the hardware world, component reuse is a natural part of the engineering process.
- In the software world, it is something that has only begun to be achieved on a broad scale.