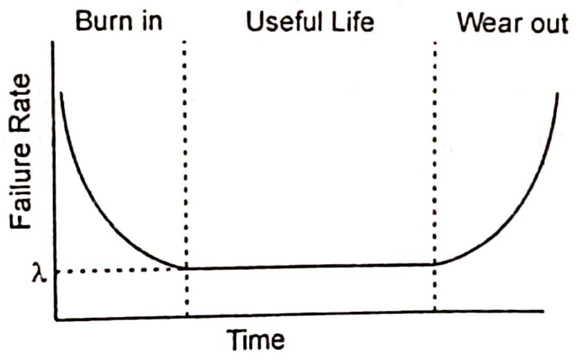
9.1 SOFTWARE RELIABILITY

(Software Reliability is defined as: the probability of failure-free software operation for a specified period of time in a specified environment.) Although Software Reliability is defined as a probabilistic function, and comes with the notion of time, we must note that, different from traditional Hardware Reliability, Software Reliability is not a direct function of time. (Electronic and mechanical parts may become "old" and wear out with time and usage, but software will not rust or wear-out during its life cycle. Software will not change over time unless intentionally changed or upgraded.)

(Software Reliability is an important to attribute of software quality, together with functionality, usability, performance, serviceability, capability, install ability, maintainability, and documentation.) Software Reliability is hard to achieve, because the complexity of software tends to be high. While any system with a high degree of complexity, including software, will be hard to reach a certain level of reliability, system developers tend to push complexity into the software layer, with the rapid growth of system size and ease of doing so by upgrading the software.

9.1.1 The bathtub curve for Software Reliability

Over time, hardware exhibits the failure characteristics shown in Figure, known as the bathtub curve. Period A, B and C stands for burn-in phase, useful life phase and end-of-life phase.



9.1.2 Software Reliability Models

A proliferation of software reliability models have emerged as people try to understand the characteristics of how and why software fails, and try to quantify software reliability. As many models as there are and many more emerging, none of the models can capture a satisfying amount of the complexity of software; constraints and assumptions have to be made for the quantifying process. Therefore, there is no single model that can be used in all situations. No model is complete or even representative. One model may work well for a set of certain software, but may be completely off track for other kinds of problems. Most software models contain the following parts: assumptions, factors, and a mathematical function that relates the reliability with the factors. The mathematical function is usually higher order exponential or logarithmic.

Software modelling techniques can be divided into two subcategories: prediction modelling and estimation modelling. Both kinds of modelling techniques are based on observing and accumulating failure data and analyzing with statistical inference. The major difference of the two models is shown in Table.

Issues	Prediction Models	Estimation Models
Data Reference	Uses historical data	Uses data from the current software development effort
When used In Development Cycle	Usually made prior to development or test phases; can be used as early as concept phase	Usually made later in life cycle(after some data have been collected); not typically used in concept or development phases
Time Frame	Predict reliability at some future time	Estimate reliability at either present or some future time

9.2 SOFTWARE RELIABILITY METRICS

Measurement is commonplace in other engineering field, but not in software engineering. Though frustrating, the quest of quantifying software reliability has never ceased. Until now, we still have no good way of measuring software reliability.

Measuring software reliability remains a difficult problem because we don't have a good understanding of the nature of software. There is no clear definition to what aspects are related to software reliability. We can not find a suitable way to measure software reliability, and most of the aspects related to software reliability. Even the most obvious product metrics such as software size have not uniform definition.

It is tempting to measure something related to reliability to reflect the characteristics, if we can not measure reliability directly. The current practices of software reliability measurement can be divided into four categories: