Software Engineering

Introduction

Outline

- Software and Software Engineering (Software, Software Products, software costs, wear out vs deteriorate, software applications, layered technology, umbrella activities
- Software Process Model (Development Models: Waterfall Model, Incremental Process Model, Rad Model, Evolutionary Process Models)
- Software Process Model (Prototyping Model, Spiral SDLC Model, Concurrent Development Model)
- The Unified Process Model (UP), Different Phases of UP
- Agile Development (An Agile View of Process, Agile Process Models, Extreme Programming-XP)
- Agile Process Models-Scrum Framework, Crystal Methodology
- Software Testing Strategies: Strategy, validation and verification, unit testing, integration testing, validation, system testing, ensuring successful testing, Driver, Stub
- Software Testing Strategies: Different types of Integration testing, sandwich testing, regression testing, smoke testing, testing strategies for object oriented testing, validation testing
- Testing Conventional Applications: alpha testing, beta testing, system testing, debugging
- Testing Conventional Applications: white box testing, black box testing, cyclomatic complexity analysis for finding independent path for testing, gray box testing.
- Estimation for Software Projects: project planning, task for project planning, software scope, resource estimation, categories of resources,
- Estimation for Software Projects: LOC based software estimation, FP count based software estimation and solving math.
- Risk Management

What is Software?

The product that software professionals build and then support over the long term.

Software encompasses:

(1) *instructions (computer programs)* that when executed provide desired features, function, and performance;

(2) *data structures* that enable the programs to adequately store and manipulate information and

(3) *documentation* that describes the operation and use of the programs.

Software Product- Generic Products

Stand-alone systems that are marketed and sold to any customer who wishes to buy them.

Examples –

PC software such as editing, graphics programs, project management tools;

CAD software;

software for specific markets such as appointments systems for dentists.

Software Product- Customized Products

Software that is commissioned by a specific customer to meet their own needs.

Examples –

embedded control systems, air traffic control software, traffic monitoring systems.

Why Software is Important?

- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled (transportation, medical, telecommunications, military, industrial, entertainment,)
- Software engineering is concerned with theories,methods and tools for professional software development.
- Expenditure on software represents a significant fraction of GNP (Gross National Product) in all developed countries

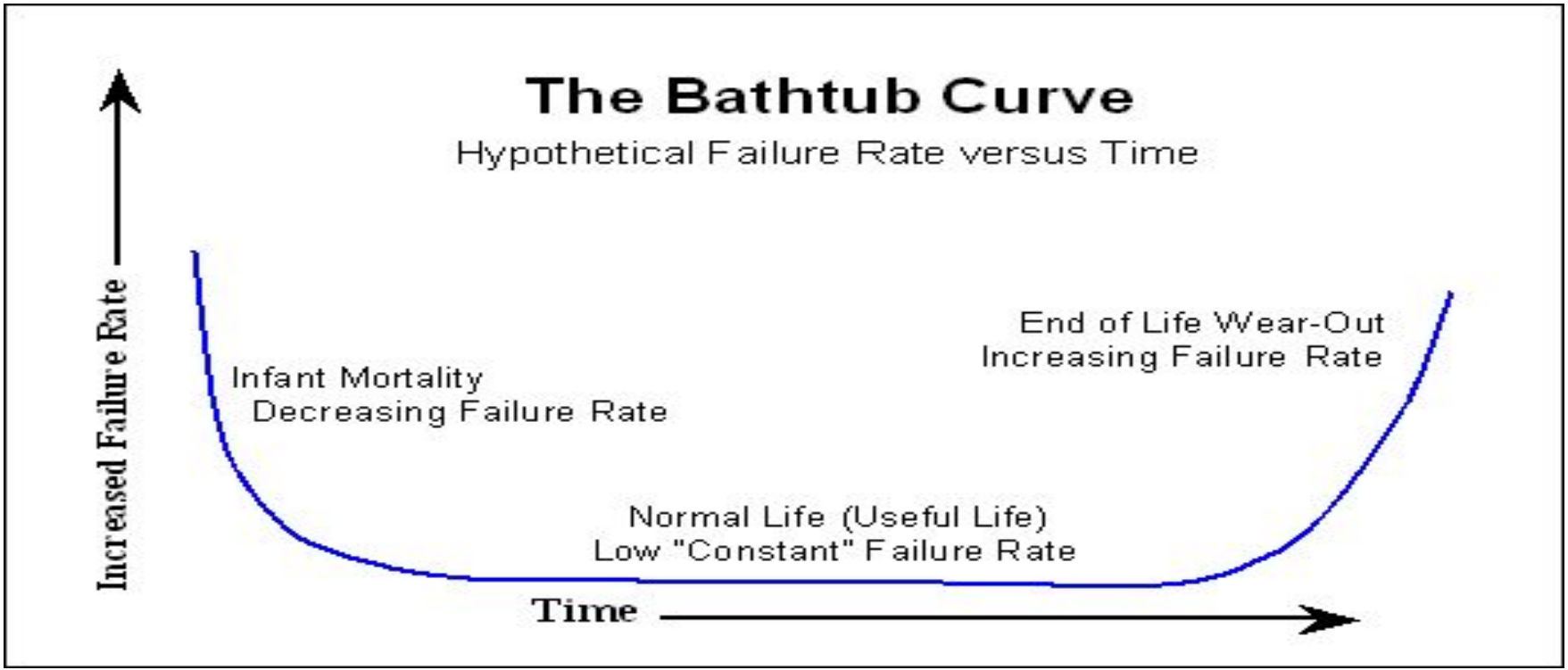
Software Cost

- Software costs often dominate computer system costs. The costs of software on a PC are often greater than the hardware cost.
- Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs.
- Software engineering is concerned with cost effective software development.

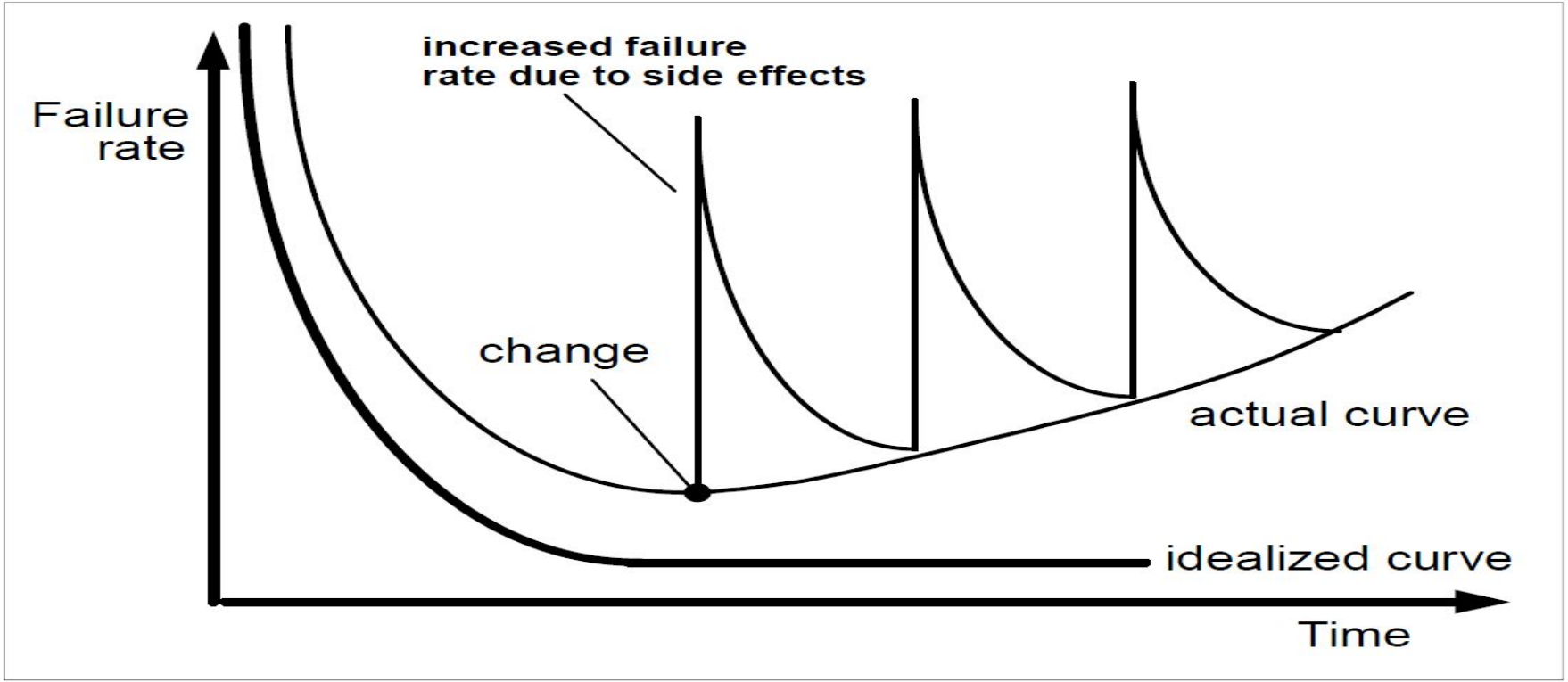
Features of Software

- Software is developed or engineered, it is not manufactured in the classical sense which has quality problem.
- Software doesn't "wear out.” but it deteriorates (due to change). Hardware has bathtub curve of failure rate (high failure rate in the beginning, then drop to steady state, then cumulative effects of dust, vibration, abuse occurs).
- Although the industry is moving toward component-based construction, most software continues to be custom-built. Modern reusable components encapsulate data and processing into software parts to be reused by different programs. E.g. graphical user interface, window, pull-down menus in library etc.

The Bathtub Curve : Failure curve of Hardware



Wear vs. Deterioration



Software Applications

System software: such as compilers, editors, file management utilities.

Application software: stand-alone programs for specific needs.

Engineering/scientific software: Characterized by “number crunching”algorithms. such as automotive stress analysis, molecular biology, orbital dynamics etc

Embedded software resides within a product or system. (key pad control of a microwave oven, digital function of dashboard display in a car)

Software Applications

Product-line software focus on a limited marketplace to address mass consumer market. (word processing, graphics, database management)

WebApps (Web applications) network centric software. As web 2.0 emerges, more sophisticated computing environments is supported integrated with remote database and business applications.

AI software uses non-numerical algorithm to solve complex problem. Robotics, expert system, pattern recognition game playing

Software—New Categories

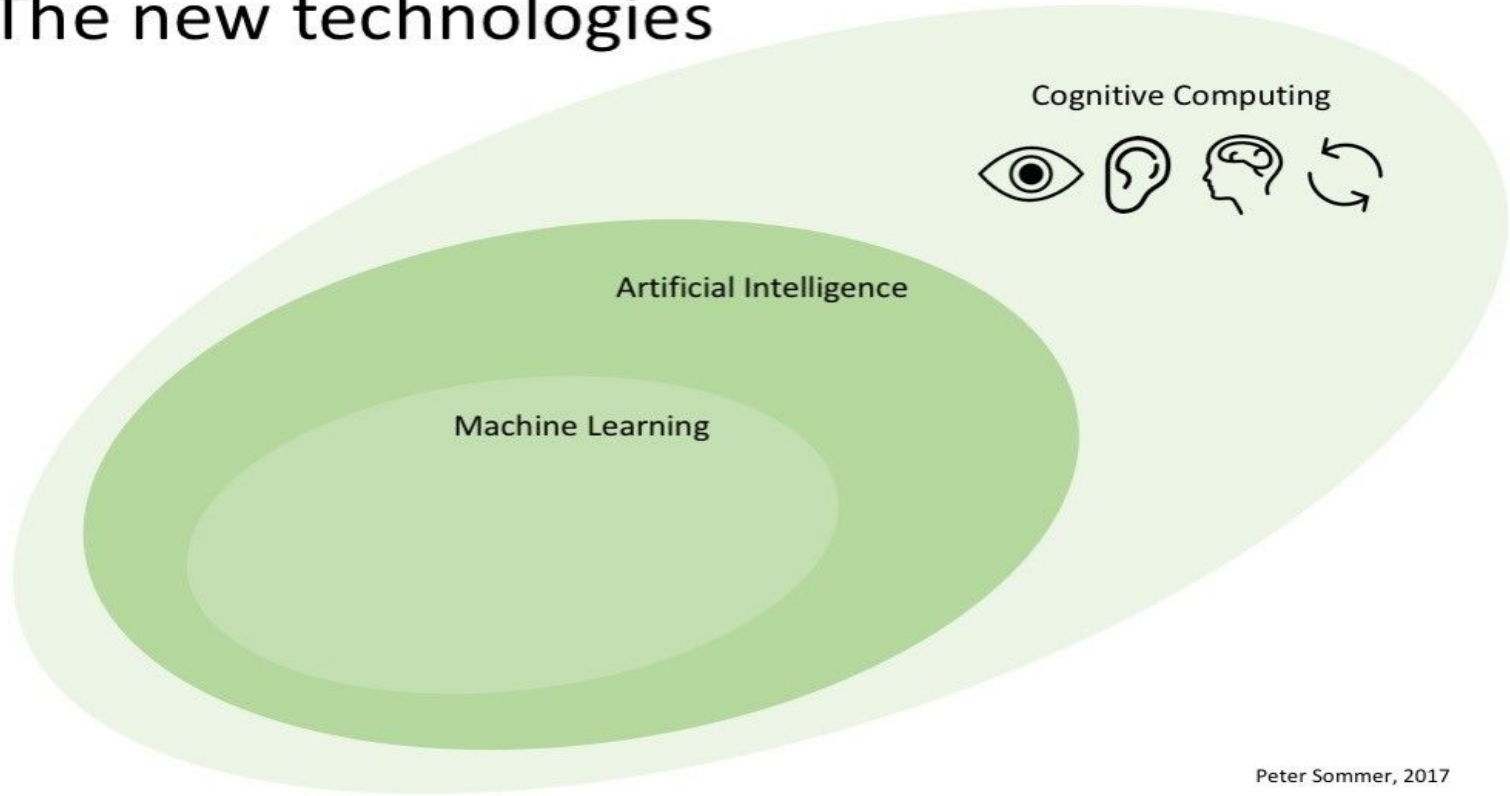
Cognitive machine is an intelligent system that is aware of its surrounding environment or outside world.

Pervasive computing is a growing trend towards embedding up is everyday objects so that they can communicate info.

Grid Computing is the collection of computer resources from multiple locations to reach a common goal. Can be considered distributed systems with non interactive workloads that involves large number of files.

Example: Weather Forecast

The new technologies



Peter Sommer, 2017

What is Software Engineering?

Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

The IEEE definition:

Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).

Importance of SE

- Individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.

- It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of system, the majority of costs are the costs of changing the software after it has gone into use.

Software Engineering- a layered Technology

Software Engineering- a layered Technology

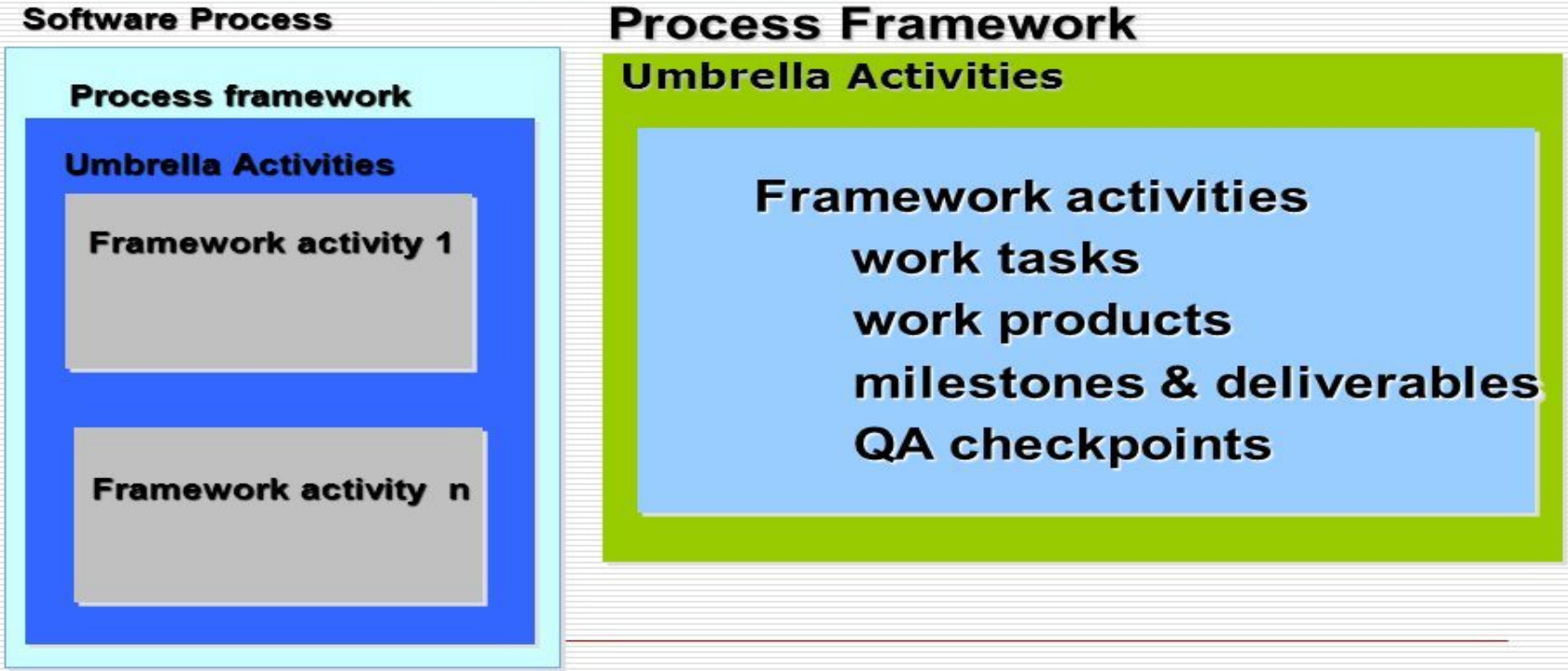
- Any engineering approach must rest on organizational commitment to **quality** which fosters a continuous process improvement culture.
- **Process** layer as the foundation defines a framework with activities for effective delivery of software engineering technology. Establish the context where products (model, data, report, and forms) are produced, milestone are established, quality is ensured and change is managed.
- **Method** provides technical how-to's for building software. It encompasses many tasks including‼ communication, requirement analysis, design modeling, program construction, testing and support.
- **Tools** provide automated or semi-automated support for the process and methods.

Software Process

- A process is a collection of activities, actions and tasks that are performed when some work product is to be created. It is not a rigid prescription for how to build computer software. Rather, it is an adaptable approach that enables the people doing the work to pick and choose the appropriate set of work actions and tasks.

- Purpose of process is to deliver software in a timely manner and with sufficient quality to satisfy those who have sponsored its creation and those who will use it.

Process Framework



Umbrella Activities

Software project tracking and control

- Assessing progress against the project plan.
- Take adequate action to maintain schedule.

Formal technical reviews

- Assessing software work products in an effort to uncover and remove errors before goes into next action or activity.

Software quality assurance

- Define and conducts the activities required to ensure software quality.

Software configuration management

- Manages the effects of change.

Document preparation and production

- Help to create work products such as models, documents, logs, form and list.

Common Process Framework Activities

COMMUNICATION 	Communication with customers/ stackholders to understand project requirements for defining software features.	**PLANNING** 	Software Project Plan which defines workflow that is to follow. It describes technical task, risks, resources, product to be produced & work schedule.
MODELING 	Creating models to understand requirements and shows design of software to achieve requirements.	**CONSTRUCTIONS** 	Code generation (manual or automated) & testing (to uncover errors in the code)
DEPLOYMENT 	Deliver Software to Customer collect feedback from customer based on evaluation Software Support		