Maintenance Process Models

SADAF FARHAN

Software Maintenance

Process of changing, modifying, and updating software to keep up with customer needs.

Done after the product has launched for several reasons including:

- improving the software overall,
- correcting issues or bugs,
- to boost performance,
- and more.

Maintenance Process Models

The need for maintenance-conscious models has been recognized for some time.

But the maintenance models are neither so well developed nor so well understood as models for software development.

However, understanding of the maintenance process, just like our understanding of the development process, moved on and maintenance process and lifecycle models emerged.

Maintenance Process Models

Quick-Fix Model.

Boehm’s Model.

Osborne’s Model.

Iterative Enhancement Model.

Reuse oriented Model.

Quick-Fix Model

Adhoc approach to software maintenance.

‘Firefighting' approach, waiting for the problem to occur and then trying to fix it as quickly as possible.

Quick-Fix Model

Limitations

Fixes would be done without detailed analysis of the long-term effects, for example ripple effects through the software

There would be little if any documentation

Unreliable model

Why used then?

Best Possible Use

Useful in case if system is developed and maintained by a single person, he or she can come to learn the system well enough to be able to manage without detailed documentation, to be able to make instinctive judgements about how and how not to implement change.

Incorporate the techniques of quick fix into another, more sophisticated model.

Case Study - Storage of Chronological Clinical Data

When the ACME Health Clinic system was originally developed, it catered only for a single recording per patient for things such as blood pressure, weight, medication and so on. This was because of a misunderstanding during requirements analysis which did not come to light until the system was in use. In fact, the system needed to store chronological series of recordings. At that stage, the need for storage of chronological data was immediate.

Case Study - Storage of Chronological Clinical Data

The maintenance programmer assigned to the task drew up a mental model of data held in small arrays to allow speedy retrieval and proceeded to implement the change.

This quick-fix method identified:

- the need for the arrays,
- to amend the data structures to allow for linking of the chronological data,
- for a small restructuring program to modify the existing data.

There was no update of documentation.

Problems Identified Later

Array overflow were not considered.

In fact, once enough information was stored, data was going to 'drop off the end' of the arrays and disappear.

Leading to data corruption in that the chronological links would be broken.

Forced adoption of further fix before the clinic stored sufficient data.

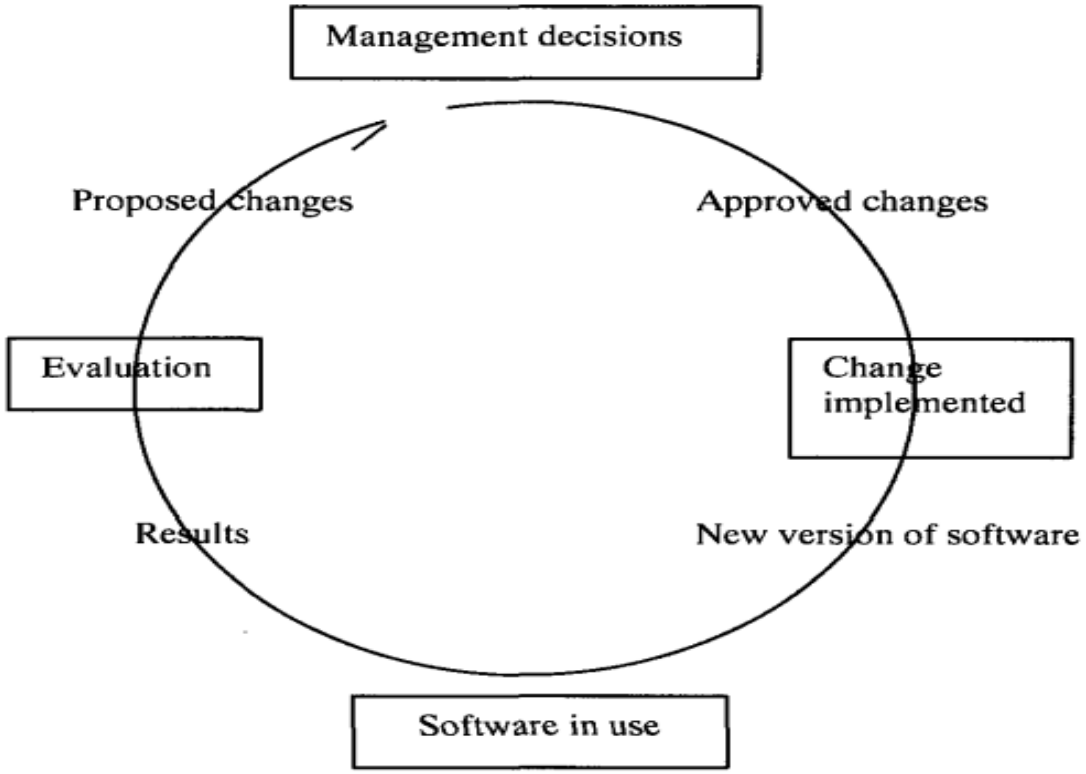
No documentation update, thus decreasing the chances of the error being successfully retrieved.

Boehm’s Model

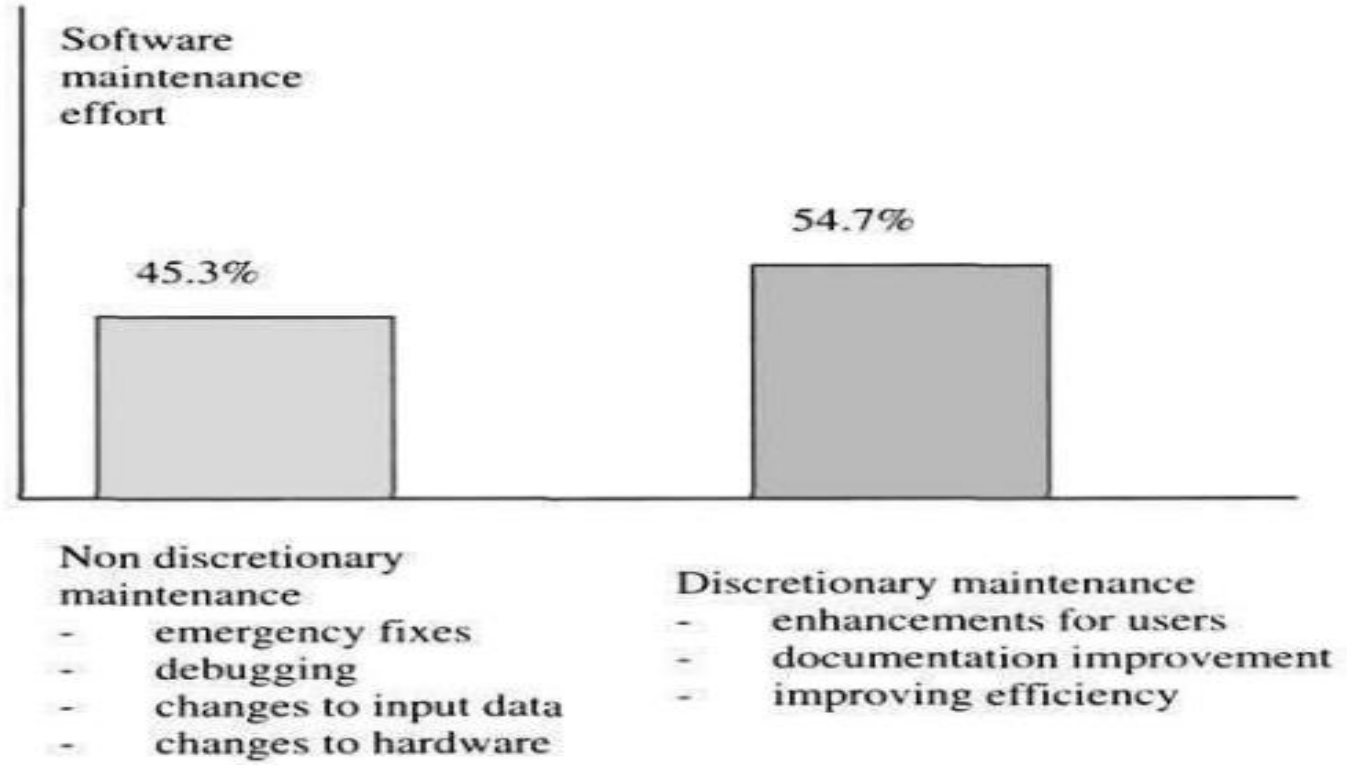
Boehm represents the maintenance process as a closed loop cycle governed by management decisions.

A set of approved changes is determined by applying particular strategies and cost-benefit evaluations to a set of proposed changes.

Boehm's Model



Limitations



Osborne’s Model

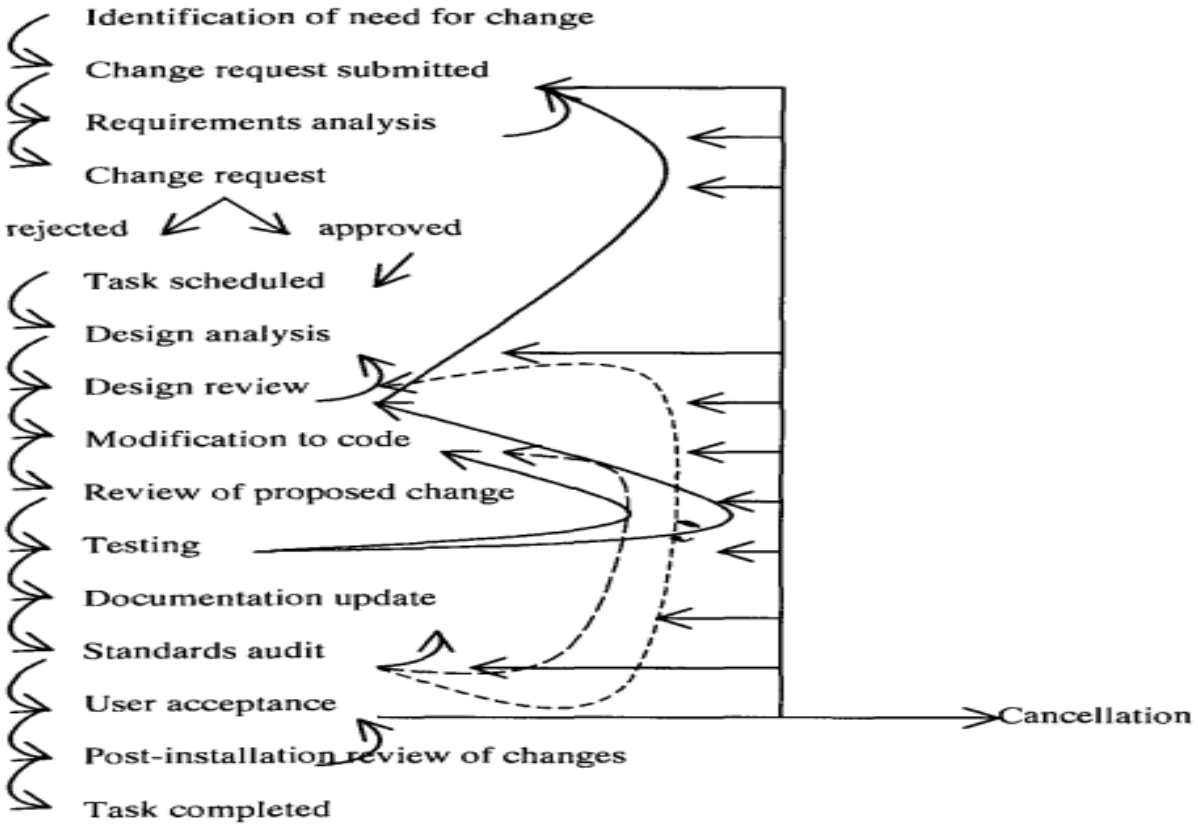
The maintenance model is treated as continuous iterations of the software life-cycle with, at each stage, provision made for maintainability to be built in.

Deals directly with the reality of the maintenance environment.

Other models tend to assume some facet of an ideal situation - the existence of full documentation, for example.

Osborne's model makes allowance for how things are rather than how we would like them to be.

Osborne's Model

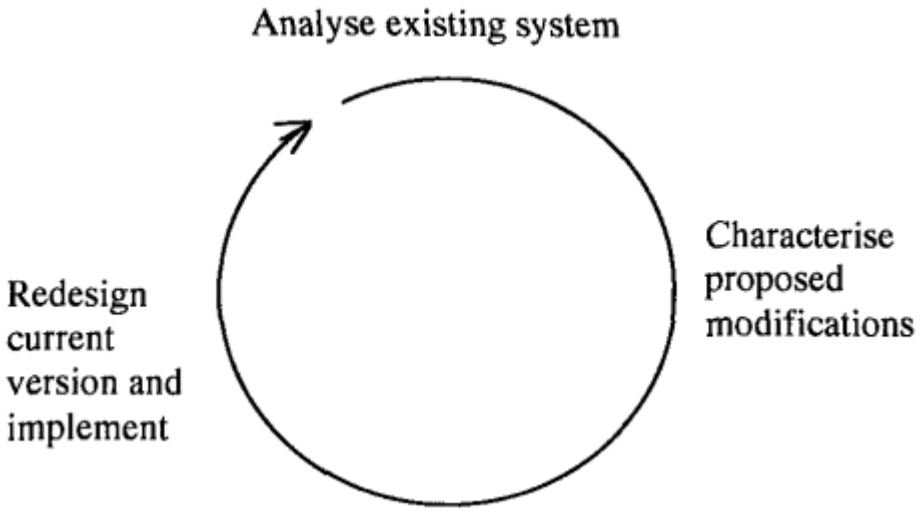


Iterative Enhancement Model

Proposed based on the tenet that the implementation of changes to a software system throughout its lifetime is an iterative process and involves enhancing such a system in an iterative way

Motivation for this was the environment where requirements were not fully understood and a full system could not be built.

Iterative Enhancement Model



Pros and Cons

Pros:

Accommodates other models, for example the quick-fix model. A quick fix may be carried out, problem areas identified, and the next iteration would specifically address them.

Cons:

Assumptions made about the existence of full documentation and the ability of the maintenance team to analyze the existing product in full.

Reuse-Oriented Model

Based on the principle that maintenance could be viewed as an activity involving the reuse of existing program components.

Four step model:

Identification of the parts of the old system that are candidates for reuse

Understanding these system parts,

Modification of the old system parts appropriate to the new requirements

Integration of the modified parts into the new system.

The End