Estimating Software Maintenance

Arun Mukhija
Contents

- What is Software Maintenance?
- Facts and Figures
- Maintenance Activities and Costs
- Maintenance Estimation Models
- Conclusion and Discussion
What is Software Maintenance?
“Changes that have to be made to computer programs after they have been delivered to the customer or user.” *

Software maintenance includes:

- Corrective maintenance
- Adaptive maintenance
- Perfective maintenance
- Enhancements (Although technically they are not a part of software maintenance but, being a post-release activity, are often considered a part of it)

Software maintenance costs around 50% of total software life-cycle cost.

But relatively little is known about the software maintenance process and the factors that influence its cost.
Software Development and Maintenance Costs in Large Organizations [Boehm81]

![Bar chart showing present of 10-year life-cycle costs for General Telephone and Electronics, USAF Command and control No.1, USAF command and control No.2, and General Motors.](image)

- Development
- Update and maintenance

Legend:
- General Telephone and Electronics
- USAF Command and control No.1
- USAF command and control No.2
- General Motors

21.01.2002
Software Development and Maintenance Costs in 487 Organizations [Boehm81]

Percent of total software effort

- Maintenance: 49%
- Development: 43%
- Other: 8%
Software Maintenance Production Function
[Boehm81]
Distribution of Software Maintenance Effort
[Boehm81]

- Emergency program fixes: 12.4%
- Routine debugging: 9.3%
- Accommodate changing of input data, files: 17.4%
- Accommodate changing of hardware, OS: 6.2%
- Enhancements for users: 41.8%
- Improve documentation: 5.5%
- Improve code efficiency: 4%
- Other: 3.4%

Percent of software maintenance effort
Distribution of User Enhancement Effort
[Boehm81]

- New reports: 40.8%
- Added data for existing reports: 27.1%
- Reformatting existing reports: 10%
- Condensing existing reports: 5.6%
- Consolidating existing reports: 6.4%
- Other: 10.1%
Maintenance Activities and Costs
Defect repairs

- keep software in operational condition
- costs absorbed by software supplier
- low pre-release defect removal efficiency (~85%)
- productivity = 8 defect repairs per month
  (can be higher with experienced personnel and defect-tracking tools etc.)

Factors influencing defect repairs:

- Abeyant defects (10%) - based on unique combination of events
- Invalid defects (15%) - misdiagnosed errors
- Bad fix injection (7%) - derivative errors
- Duplicate defects - multiple complaints about the same error
Error-prone module removal

- concentration of errors in particular modules
- common among large poorly-structured systems
- expensive to maintain, due to high bad fix injection rate
- 500% more expensive than normal modules

Customer support

- interface between clients and defect repair teams
- effort depends on number of users
  - with phone contact, 1 customer support person for 150 users
  - with electronic contact, 1 customer support person for 1000 users
Code restructuring
- done by automated tools to lower complexity levels
- lowering complexity eases maintenance
- precursor to other maintenance activities

Migration across platforms
- from one OS or hardware to another
- with well-documented specifications,
  migration speed = 50 FP per month
- with missing or obsolete specifications,
  migration speed = 5 FP per month
Conversion to new architectures

- changes to interface or file structure of apps.
- quality of specifications affects productivity
- reverse engineering may need to be performed to extract missing design info.

Mandatory changes

- in response to changes in law or policy
- involve high costs and tight schedules
- difficult to predict in advance
Performance optimization

- to minimize delays in transactions
- improving performance at trouble spots

Enhancements

- adding new features as per user request
- funded by user
- annual rate = 7% increase in FP total of an app.
- high integration and testing costs for poorly structured apps.
Maintenance
Estimation Models
COCOMO Maintenance Model
for software maintenance effort estimation

\[(MM)_{AM} = (ACT)(MM)_{DEV}\]

\((MM)_{AM}\) : annual maintenance effort in man-month
\((MM)_{DEV}\) : development effort in man-month
\(ACT\) : annual change traffic (fraction of software that undergoes change during a year)

For intermediate and detailed COCOMO,

\[(MM)_{AM} = (EAF)_M (ACT)(MM)_{NOM}\]

\((EAF)_M\) : maintenance effort adjustment factor
Maintenance/Development Cost Ratio

\[(MM)_M = (M/D)(MM)_{DEV}\]

\((MM)_M\) : overall life-cycle maintenance effort in man-month
\((MM)_{DEV}\) : development effort in man-month
\(M/D\) : maintenance/development cost ratio
Value of \(M/D\) ranges from 0.67 to 4.5, depending on application type.
Cards-per-person ratio

origin: number of cards each software person can maintain

\[(\text{KDSI/FSP})_M : \text{KDSI maintained per full-time software person}\]

\[
\frac{(\text{KDSI})_{DEV}}{(\text{FSP})_M} = \frac{(\text{KDSI/FSP})_M}{(\text{FSP})_M}
\]

\[(\text{FSP})_M : \text{number of software maintenance personnel required}\]

\[(\text{KDSI})_{DEV} : \text{size of software in KDSI}\]

Value of \((\text{KDSI/FSP})_M\) ranges from 3 to 132, depending on application type.

The annual maintenance effort \((\text{MM})_{AM}\) is then simply

\[
(\text{MM})_{AM} = 12 (\text{FSP})_M
\]
**Maintenance Productivity Ratio**

\[
(DSI)_{MOD/YR} = (ACT)(DSI)_{DEV}
\]

\[
(MM)_{AM} = \frac{(DSI)_{MOD/YR}}{(DSI/MM)_{MOD}}
\]

- \((DSI)_{MOD/YR}\): number of source instructions modified per year
- \((DSI)_{DEV}\): size of software in source instructions
- \((MM)_{AM}\): annual maintenance effort in man-month
- ACT: annual change traffic
- \((DSI/MM)_{MOD}\): maintenance productivity ratio (number of source instructions modified per man-month of maintenance effort)

Average value of ACT is 0.092 and of \((DSI/MM)_{MOD}\) is 241, based on a survey.
Conclusion and Discussion
“Software processes must produce software that can be gracefully evolved at reasonable costs. The choice of software architecture significantly influences modifiability and hence maintainability.” *

Estimating maintenance is complex because of the relationship between base application and changes being made. Moreover predicting adaptive maintenance and enhancements in advance is very difficult.

References

