## Title: A Metrics-Based Software Maintenance Effort Model

## Description

Tactic and strategy applied on software maintenance effort estimation model is greatly influence on accuracy of project planning and resource allocation sector. By estimating maintenance time and effort efficiently, team members can overcome long haul burden of underestimate issue. With this intention, Hayes, Patel and Zhao introduces metrics-based maintenance effort estimation model called Adaptive Maintenance Effort Model (AMEffMo) for managers and maintainers.

## Research Objectives

This paper aimed to formulate maintenance effort estimation model to produce results in person hours.

## Research algorithms and Theory

In the beginning, researchers applied metrics from two previous studied, which outlines the influential factors on maintenance effort, to preform correlation analysis. Then, the top two correlation metrics namely; "%Operator Changed" & LOC Difference Delta-DLOC are selected to design the model. Based on two metrics, below two estimation models are provided in the study.

E = 63 + .1DLOC

E = -124 + 7.5 DNoprtr

## How to apply tool/approach/framework

Project managers and maintainer can apply AMEffMo model by using the two given formula, to estimate maintenance effort hours.

## Research Evaluation

AMEffMo model is formulated and evaluated by applying regression analysis on four sources of software project data including CS 499,CS 616 from University of Kentucky and Industry research data from [12]. The researcher used 70% of project data to design the model and the rest 30% for verification and validation.

## Research Restrictions of Usage

Researcher recommend broad range of industry projects testing is still needed to refine the model as the currently research evaluation is based on small set of domain.

## Title: Observe-mine-adopt (OMA): an agile way to enhance software maintainability

## Description

Software maintenance process improvement is one of the key research areas focused by researchers in past few decades. Even though, Software development model such as ISO 9001, Capability Maturity Model Integrated (CMMI) team and SPICE (now ISO/IEC TR 15504) are providing efficient guidelines, it require a great amount of time, money and effort to adapt in software development organization. In this paper, Hayes, Mohamed and Gao introduce new software maintenance paradigm called Observe-Mine-Adopt(OMA) to enhance the software maintenance practices in an agile way. The approach is defined based on the natural observation practice and knowledges of software teams and, provided a model to measures two maintainability namely; maintainability product and perceived maintainability.

## Research Objectives

Goal of OMA paradigm is aimed at improving the quality, reliability and maintainability of the software development process in an iterative way. To improve the process without weighting a great commitment on development teams, OMA simply used natural human observation skill in an agile way to improve software maintenance process. It can also applied as a subset of any existing software improvement process principal.

## Research algorithms and Theory

Conceptual design of OMA paradigm is emphasized on people-oriented approach rather than process-oriented approach to serve in agile projects. In OMA, tasks are structured in a small iterative modules as overhead of the approach is much lighter and quicker to adapt by development teams.

## How to apply tool/approach/framework

In general, OMA paradigm consists of three basic components; Observe, Mine and Adopt and process in a hierarchal way. Figure 1 describes OMA Paradigm steps.

1. Observe

In observation phase, the components is further sub-level into three steps

1.1 First, project team members must list and rank maintainable comportments

1.2 Second, team members decide and agree on what and when to observe.

1.3 Finally, Document the observation

2. Mine

In data mining phase, it is also sub-divided into four process steps

2.1 First of all, Observe objects should be determined

2.2 Then, Investigate the observe objects and examine criteria. In accordance with code changes objective, criteria such as effort required to make changes, comment ratio, amount of changes can define as object criteria.

2.3 Thirdly, learn the casual relationships between collected data and criteria. If relationship exist, we can learn that the best practice resulted in observed object.

2.4 Lastly, Validate the findings and study whether the best practice is maintainable or not.

3. Adopt

The outcomes gained from prior two steps are used it as input into this adoption step. Similar to research 'publish' phase in an academic area. The success of paradigm is measured by how well project teams adopt the new best practice in their future development and maintenance process.

## Research Evaluation

OMA paradigm was experienced in a few different projects including Perot Health Care System, Persona Digital Assistant Project developed by Graduate and three more others (PA0, PA2 & Spathic).

## Research Restrictions of Usage

OMA paradigm still require to perform further refinement to improve the approach. In addition, there is no support tool to implement the approach easily and repetitively.

# Title: Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods

## Description

In Advancing Candidate Link Generation paper, Jane Hayes & IEEE members introduces an independent RETRO (REquirements TRacing On-target) prototype tool to address the issue relating to improving quality of dynamic candidate links (aka trace link) generation process. Prototype is designed to assist system analyst in tracing requirements for the purpose of Verifying and Validating pairs of high & low sets of requirements to ensure the right processes have been applied to build the right system.

## Research Objectives

The fundamental purpose of this research is to design and implement an effective, efficient requirements tracing tool to minimize the onerous task of analyst. Alternatively, the study intends to improve overall usability and quality of requirement tracing technique.

## Research algorithms and Theory

The tool applied two Information Retrieval Algorithms namely; Tf-Idf vector retrieval and Latent Semantic Indexing

## How to apply tool/approach/framework

In general, requirements tracing process includes various stages ranging from document parsing, candidate link evaluation, Trace Results Verification & Verification and so forth. In this paper, the tracing processes are broken down into the following seven high level steps

a) Define requirements to trace

b) Allocate and assign a unique identifier to each requirement

c) For each high level requirement, locate all matching low level requirement

d) For each low level requirement, locate a parent requirement in the high level document

e) Verify trace results and seen low level requirements satisfied high level requirements

f) Produce Traceability Matrix Report based on the trace results

g) Produce the summary report to display the document pair's level of traceability

## Research Evaluation

In RETRO tool, inaccurate candidate links aka trace links are commonly introduced from two types of errors source:

- Commission error (a false positive link was added to the list)

- Omission error ( a true link was not recognized)

## Research Restrictions of Usage

The tool require analyst input to evaluate candidate links in order to make decision on whether or not the links should be accepted or rejected or look for additional candidate links.

Thus, it does not provide full automation approach.

## References

HAYES, J. H., DEKHTYAR, A. & SUNDARAM, S. K. 2006. Advancing candidate link generation for requirements tracing: The study of methods. IEEE Transactions on Software Engineering, 32, 4-19.

HAYES, J. H., MOHAMED, N. & GAO, T. H. 2003. Observe-mine-adopt (OMA): an agile way to enhance software maintainability. Journal of Software Maintenance and Evolution-Research and Practice, 15, 297-323.

HAYES, J. H., PATEL, S. C., ZHAO, L. M. & IEEE COMPUTER, S. 2004. A metrics-based software maintenance effort model.