

MOLACH: The Model For Language Change

Dan Strickland

Dynetics, INC.

daniel.strickland@dynetics.com

Abstract: Any large software project relies on the choice of software language as a key factor in justifying cost and schedule estimates. Whether the effort lies in trying to find enough Ada programmers to hire and train for a large government program or in determining the need for portability exchanged for speed with Java for a new online banking program, language choice impacts the entire software program. The Model for Language Change (MOLACH) was developed to attempt to quantify the relative effort saved or expended in making a change in development language in a project. In a large project, a change in language can be a positive or negative yield in effort depending on several factors over the entire life of the program. MOLACH attempts to quantify those factors and develop a comparative method for decisions on the benefits and detriments of changing software language.

MOLACH examines the relative difference between the status quo (the current language choice) and the target (the proposed language change). MOLACH is based on Dr. Barry Boehm's COCOMO II model, specifically in the areas of Effective SLOC, reuse, and the environmental factors for Language and Tool Experience (LTEX) and Use of Software Tools (TOOL). MOLACH also adds factors for Vendor Support (VSUP), Development Future (DEVF) and Portability (PORT) to the two listed above. The status quo and target languages carry weighted effort based on their performance in the five environmental factors. In MOLACH, the status quo will get the benefit of reuse and less code development, where the target will carry the onus of developing new code to match any previous functionality **plus** the new development. In MOLACH, developed code carries a Language Tax based on the Programming Languages Backfiring Tables (in SLOC per Function Point) developed by Capers Jones. MOLACH produces output in the form of Language Source Lines of Code (L-SLOC). L-SLOC is calculated using the following formula:

$$\text{Language SLOC} = ([\text{Reuse}] + (\text{Estimated New SLOC} * [\text{Language Tax}]]) * [\text{LEF}]$$

where

$$[\text{LEF}] = \Pi (\text{LTEX}, \text{TOOL}, \text{VSUP}, \text{DEVF}, \text{PORT})$$

and

$$[\text{Reuse}] = (\text{Reuse SLOC} * [\text{Modification \%}]) * [\text{Language Tax}]$$

In the case of the status quo language:

[Language Tax] = 1.0 *as there is no change of language if the current is kept*

[Modification %] = ((40% * Design Mod) + (30% * Code Mod) + (30% * Retest Required))

where with the target language:

[Language Tax] = (target/status quo) *from Capers Jones' backfiring tables*

[Modification %] = ((40% * Design Mod) + (30% * 100%) + (30% * 100%)) *accounts for new code development; same reuse on design*

MOLACH was used in conjunction with the Software Language Transition Plan for a large Department of Defense software program. When the DoD Ada mandate was relaxed, several developers proposed a change in software language to C, C++, Visual Basic, or a mixture of languages. The government customer requested a model for quantifying the overall effort and long-term viability of language conversion to the software item level of fidelity. MOLACH was developed in an effort to quantify and validate the requested language changes to the software item level. MOLACH provided the government customer and the software developer with a tool for evaluating the life cycle cost delta based on language change alone.

The values MOLACH develops from an equal model based on the entire lifecycle of a program can quantify and qualify a decision in language change. MOLACH aids in the complicated process of deciding whether the "pain to gain" ratio is in a program's favor when considering a software language change.