
Abstract
EM: Enhanced model
Existing <software cost accounting> models are deficient in dealing with <the challenge of providing management adequate visibility into costs and benefits> of <reusable components within software products>. An enhanced <cost accounting model based on tracking reuses> is described, capable of providing more accurate analyses of <costs and benefits> in <reusable component> designs. The model has been tested by comparing analyses with empirically measured values of <costs, as analyzed through the traditional model>.

Why EM?
Existing cost accounting models for software development usually attributed the cost of reuse to specific projects. This is inaccurate, since the resulting components often generate value to other projects. The authors adapt an existing cost accounting method (ABC) to the realm of software development by qualitatively describing how to connect the method to our domain.

Question - [Method for analysis]
<model-type>
What cost accounting model shall we use in software development?

<solution strategy>
The solution being analyzed is a software product, potentially including multiple reusable components.

<properties>
The problem with previous models is that they errantly ascribed the full cost of components to the project wherein those components originated, thereby discouraging investments into reusability. The property of interest is therefore to give adequate visibility of costs and benefits to management.

Results - [Qualitative or descr. model]
<model-type>
The authors propose to adapt activity based costing (ABC) to software development by identifying the activity units that contribute to reusable components, and then distributing the cost of those activities over all projects which utilize those components. They focus on the need to weight costs by the number of uses that the components see in each project.

Validation – [Experience]
<properties>
The authors apply their new model to data from a software development corporation, comparing the resulting cost allocations with comparable allocations generated through the traditional method. The new method does indeed produce different breakdowns, but the authors fail to prove by a convincing “higher criterion” that their new breakdowns are “better” than the old ones, except to point and say, “These look more reasonable.” Ideally, future papers will attempt to validate the approach by applying the new and old models to a broader array of software projects, and then presenting the results to managers who can judge which, indeed, provides better guidance and visibility from a management standpoint.