Software Portfolio Analysis – Does your Investment perform adequately?

Mary Udeh

Abstract

The objective of this paper is to provide a solution to the problem of escalating Information technology (IT) costs by extending a cost benefit analysis method (CBAM) and portfolio analysis to existing software. This problem of escalating IT software costs is not just a result of an undisciplined approach to software investment decisions, it also highlights a lack of understanding of where and when to invest scarce dollars in software maintenance, enhancements, acquisition or retirement. Though CBAM has been applied to new software projects, this paper introduces a standard method and disciplined approach to making software decisions on existing software portfolios using CBAM. It will also show how we can assess the condition and quality of a software portfolio, to manage the portfolio through disciplined software investment decisions.

Introduction

For large companies with IT departments, many years of building or acquiring cheaper and faster software solutions that meet unique business needs have resulted in an assortment of software for IT to support, track and evaluate. In some cases, multiple versions of the same software are used by different business areas, making it hard to maintain, track, and evaluate the software portfolio. On-going management of the portfolio is important because it affects the cost of doing business but, there is no standard method for conducting an ongoing assessment of the condition of a software portfolio. Most existing methods focus primarily on the development of new software rather than on existing software. Since most software development efforts are irreversible capital investments, they should be expected to add value to the organization as with any other capital expenditure say Butler et al [2]. Thus, the cost of custom-developed or purchased software should not outweigh the returns from that investment.

This paper first discusses methods currently used in software economics: portfolio analysis and architecture-centric solutions, as well as their benefits and shortfalls. It then describes, applies and extends CBAM to an existing software portfolio rather than new software. It illustrates the benefits and challenges of using software portfolio analysis (SPA) to evaluate existing software investments. The conclusions lead to a roadmap for further research that can improve management of software portfolios via disciplined decisions about software acquisition, design, maintenance, and attrition.

Software Economics

Software economics is the field “that seeks to enable significant improvements in software design and engineering through economic reasoning about product, process, program, and portfolio and policy issues” says Boehm [5]. Software economics “seeks to develop technical theories, guidelines and practices of software development based on sound, established and emerging models of value and value-creation” adapted to the domain of software development as necessary according to Erdogmus et al [4].

Limitations of software economics

Software Economics contribute to our understanding of how software adds value to products and services. In a typical economic model, specific IT costs are measured against their potential financial value, risk, and business impact, so budgets can be allocated to the areas with the highest potential returns. According to Hislop [6], this makes it difficult to:

- Evaluate return on prior software investments.
- Identify the strengths and weaknesses of the current software portfolio.
- Prioritize software investment decisions.
- Decide when to overhaul or scrap software systems.

Portfolio Analysis

Portfolio management can enable organizations to manage technology costs better including hardware, software and people. Optimal portfolio theory outlines how to invest in a portfolio of risky assets to maximize its return for a given level of risk.

Financial organizations such as Merrill Lynch have extended the use of optimal portfolio theory to IT investments, making the IT organization smarter about how and when it spends money. Regular measurement of IT investments helps them decide when it is time to retire a piece of software.

Frequently IT has no standard way of showing the business return on software investments, as a portfolio manager would show clients the return on their investments. In times of declining profits, organizations are asked to reduce operating costs. One of the popular key areas to cut is IT costs because IT is not seen by the business as adding value.

How can a portfolio be managed for maximum value creation in specific timeframes? How should the returns on resource investment be evaluated? How should decisions about software acquisition, upgrade or attrition be evaluated for success or failure?
A phased investment approach for example sets up checkpoints for a “go or no-go” decision based on the value the investment brings to the portfolio. When a software component’s cost-benefit ratio continues to rise above 1 (based on a given (CBAM) methodology), then it is time to upgrade, replace or retire software from the portfolio [5].

**Architecture-centric solutions**

Next, we present a number of architecture centric approaches and their value to software portfolio analysis. These methods are used today to evaluate one specific software regardless of domain and they are: Software Architecture Analysis Method (SAAM), Architecture Trade-off Analysis (ATAM), Security attribute evaluation method (SAEM) and Cost Benefit Analysis Method (CBAM).

**SAAM and ATAM**

The Software Architecture Analysis Method (SAAM) and the Architecture Trade-off Analysis Method (ATAM) both base their analysis on how well the architecture was designed with respect to quality attributes (QA) such as modifiability, performance, availability, usability [1]. ATAM goes further by analyzing the consequences of architectural trade-offs for several QA concerns simultaneously.

**SAEM**

An example of domain-specific cost-benefit analysis is Butler’s security attribute evaluation method (SAEM) which helps IT stakeholders decide whether their security investment is consistent with the expected risks [3]. Despite its formal approach, SAEM depends ultimately on a risk assessment that results in a prioritized list of risks based on best-guess estimation of software effectiveness. The result is then used to determine the potential risk mitigation benefit.

The relationship between architecture and economics involves value implications of both technical and managerial decisions not addressed by many architecture-centric methods. This paper focuses on how the cost benefit analysis method (CBAM) can be used to evaluate the existing software portfolio on an on-going basis.

**A Cost Benefit Analysis Method (CBAM)**

CBAM is a formal methodology for analyzing and illustrating how the costs and benefits of architectural decisions relate to business goals in terms that make technical and managerial decisions easier. Since the biggest trade-off in complex systems is economic, CBAM has extended ATAM by including analysis of the costs (long-term and development) and benefits of architectural decisions. Therefore, CBAM offers an economic model of software which takes into account costs, benefits and risks. According to Kazman et al [1], CBAM adds dollars to ATAM and QA responses, as additional attributes to be traded off. Figure A is a visual representation of CBAM.

The shaded costs and benefit circles illustrate how CBAM is different from ATAM.

**Figure 1: CBAM (adapted from Kazman et al [1])**

**Applying the six steps of CBAM**

This section illustrates the application of CBAM to a software portfolio. The six Steps of CBAM from Kazman et al [1] focus on specific new software to be acquired. Since the objective of this paper is to extend CBAM to existing software portfolio, the examples used are a subset of existing software from company A. The software portfolio was selected from diverse business domains such as accounting, sales, asset management using criteria such as vendor products and custom-built business software. The choice of diverse software was made so its effects on the results could be observed. This section describes the CBAM step and lists their inputs, expected outputs and applicable results for the selected software portfolio.

**Step 1:** Choosing scenarios and architectural strategies
- **Input –** a set of high importance scenarios for which improvement is desired, their quality attribute (QA) response goals and any associated architectural decisions.
- **Output –** a set of desired improvements to the systems, the affected parts of the existing architecture and one or more architectural strategies for realizing the improvement.

Examples of QA goals are: Ease of re-use and less than 5 seconds response time. There can be multiple quality attributes associated with one QA goal.

**Step 2:** Assessing quality attributes (QA) benefits – relate architectural strategies (AS) to QA goals back to business goals
- **Input -** QA and QA goals
- **Output -** QA score assigned by stakeholders to each QA goal (such as performance, availability, modifiability, security).

The AS descriptions were defined as part of this project. QA attributes were selected to achieve the goals of each strategy. Five stakeholders (project managers, business experts, operations support, etc.) were selected and asked to decide on the average weighted score (QA score). The QA score were assigned based on the relative
importance of each goal. QA scores must total 100. Each stakeholder was then asked to rate the contribution of each AS in meeting the QA goal. Figure 5 shows the average contribution and the average QA score.

Step 3: Quantifying the architectural strategies’ benefit – evaluate each architectural strategy (AS) based on the score of their dependent quality attributes.

- Input – AS, QA scores, Contribution (Cont) - a ranking of each AS in terms of how much each AS contributes to each QA on a scale of -1 to +1. For example, a +0.5 for AS1 means it has a positive effect on security.

- Output – The benefit score for each AS is calculated for each stakeholder Using the formula: Benefit Score = Avg Contribution multiplied by Avg QA score [1]. Variations in stakeholders’ judgment in QA scores and contribution can be seen in Figure 5.

The yellow line shows the variability among the stakeholders. The resulting benefit score incorporates the variable stakeholder feedback. We selected seven architectural strategies that were deemed important for this evaluation cycle. Figure 5 shows the selected architectural strategies.

Step 4: Quantifying the architectural strategies’ costs and schedule implications

- Input - resource costs and each AS under consideration.

- Output – Implementation costs (High, Medium, Low) for each AS as well as schedule implications and any contention for shared resources (hardware, software, personnel).

(The costs used here are estimates). This is the first time we have two values for the same AS illustrating the difference between total costs when software is new, versus support costs for existing software. Software integration AS has a cost of zero dollars for existing software because it already fits into the environment and no integration is needed.

Step 5: Calculate desirability

- Input – Benefit (B) for each AS, Costs (C) for each AS.

- Output – Desirability (D) for each AS is calculated using the formula, \( D = \frac{B}{C} \) [1]

An AS can be more desirable when the product is new versus when it already exists. For example, security architectural strategy was ranked number 2 for new software but ranked 4 when software exists. An average DESIRABILITY is calculated for each piece of software in the portfolio being analyzed.

Step 6: Make decisions

- Input – Desirability ranking, schedule constraints and stakeholder uncertainty

- Output – decisions

<table>
<thead>
<tr>
<th>Software Name (S)</th>
<th>Desirability (New)</th>
<th>Ranking (new)</th>
<th>Desirability (Monthly/Quarterly)</th>
<th>Ranking (Monthly/Quarterly)</th>
<th>% Change in Desirability</th>
<th>Decision (Buy/retire/upgrade/Keep)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>0.00074650</td>
<td>6</td>
<td>0.00806049</td>
<td>9</td>
<td>979.761938</td>
<td>Keep</td>
</tr>
<tr>
<td>S2</td>
<td>0.02</td>
<td>8</td>
<td>0.025</td>
<td>7</td>
<td>25</td>
<td>Keep</td>
</tr>
<tr>
<td>S3</td>
<td>0.5</td>
<td>6</td>
<td>0.005</td>
<td>10</td>
<td>-99</td>
<td>Retire</td>
</tr>
<tr>
<td>S4</td>
<td>2.4</td>
<td>4</td>
<td>1.4</td>
<td>4</td>
<td>-41.6666667</td>
<td>Upgrade</td>
</tr>
<tr>
<td>S5</td>
<td>5.2</td>
<td>1</td>
<td>3.2</td>
<td>2</td>
<td>-38.4615385</td>
<td>Upgrade</td>
</tr>
<tr>
<td>S6</td>
<td>0.002</td>
<td>11</td>
<td>-0.003</td>
<td>11</td>
<td>-250</td>
<td>Retire</td>
</tr>
<tr>
<td>S7</td>
<td>1.09</td>
<td>5</td>
<td>-1.0015</td>
<td>12</td>
<td>-191.880734</td>
<td>Retire</td>
</tr>
<tr>
<td>S8</td>
<td>0.099</td>
<td>9</td>
<td>1.003</td>
<td>5</td>
<td>913.131313</td>
<td>Keep</td>
</tr>
<tr>
<td>S9</td>
<td>0.0083</td>
<td>10</td>
<td>0.0083</td>
<td>8</td>
<td>0</td>
<td>Keep</td>
</tr>
<tr>
<td>S10</td>
<td>3.32</td>
<td>2</td>
<td>6.32</td>
<td>1</td>
<td>90.3614458</td>
<td>Keep</td>
</tr>
<tr>
<td>S11 (new)</td>
<td>2.43</td>
<td>3</td>
<td>2.43</td>
<td>3</td>
<td>0</td>
<td>Buy</td>
</tr>
<tr>
<td>S12</td>
<td>0.09</td>
<td>7</td>
<td>0.05</td>
<td>6</td>
<td>-44.4444444</td>
<td>Upgrade</td>
</tr>
</tbody>
</table>

** Decision Rule: Over 50% negative change in desirability should be retired
Schedule constraints and business risk prioritization can affect this decision rule.

Figure 2: Applying CBAM to a Software Portfolio
Software Portfolio Desirability Comparison

Figure 3: Side-by-side comparison of desirability for each software in the portfolio

Portfolio evaluation result and recommendations for Company A

Based on the decision rule stated in Figure 2, our analysis results in the following recommendations:

- Keep S1, S2, S8, S9 and S10 in the portfolio.
- Upgrade S4, S5 and S12 to increase their value (desirability) and contribution to the portfolio.
- Retire S3, S6 and S7 because they cost more to keep and support due to significant depreciation in value (desirability)
- Build or buy S11 because it has been shown to be a desirable addition to the software portfolio.

Figure 3 is a side-by-side comparison of desirability for each software in the portfolio, from when it was newly acquired up to the specific quarter or month measured. Figure 4 shows the change in desirability.

Observed benefits of CBAM

The benefits of applying CBAM to Company A’s software portfolio are:

1. Decisions made through CBAM allow a better mix of risk levels by incorporating a variety of stakeholder scoring data.
2. CBAM’s decision methodology takes less time to re-evaluate a portfolio by using a simple cost model and calculations which can be modeled easily and inexpensively in Microsoft Excel. Depending on the size of the portfolio, a custom built MS Access application, or some more robust vendor application is equally suitable.
3. CBAM is easier to automate for continuous monitoring leaving more time to focus on new software acquisitions.
4. In CBAM, a software’s fit with the current portfolio is determined not just based on costs and business value but also on the overall risks (technical, schedule and business).
5. CBAM enables and promotes software optimization to increase the business value and shelf-life of existing software, while decreasing its technical risks and maintenance costs.
6. CBAM merges both business and technical concerns around architectures that have been shown to cost money. For example, a business area acquires a product such as Crystal Reports for its desktop reporting with no consideration on
General Recommendations

To facilitate value assessment and IT portfolio oversight for operational effectiveness (which includes gathering product data), and based on the portfolio evaluation results using CBAM, this paper recommends:

1. The establishment and maintenance of domain portfolios that allow flexible control and optimal utilization of IT resources.
2. Choose a specific number of AS for analysis e.g. 7 out of 12. This means using the same sample size for consistency and manageability in approach and comparison for a specific analysis. The organization can create different templates with subsets of AS definitions.
3. The organization can calculate monthly desirability or quarterly desirability depending on how often the information is used to make a financial decision.
4. Since a portfolio must provide access to dynamic real-time portfolio asset information, it must also support software (& hardware) reusability in order to minimize rework and redundancy.
5. The assessment process should identify the strengths and weaknesses of each asset proposed for the portfolio, and also provide a level of assurance that the software meets application requirements.
6. Portfolios should also include new software that is in-progress or in the acquisition phase because it’s assessment process provides valuable opportunities to expand the scope and capabilities of existing and planned applications.
7. Few companies put themselves in the position to measure their IT investments regularly in order to decide when to retire given software. On-going measurement is recommended using CBAM and SPA. Then a product strategy that will allow for continuous monitoring of existing software should be put in place to decide when to retire, upgrade or replace software.
8. The secret to portfolio management is to use it to drive and sustain real business value as determined by both business and IT.

Conclusion:

This research paper has addressed an important problem which causes businesses to lose money either by acquiring redundant software, incompatible software or software that may not be easily adaptable to re-use. Sometimes the cost of implementing software is enormous compared to purchase costs due to hardware issues or incompatibility with existing IT resources. A good methodology for software portfolio analysis used early in the acquisition process avoids these problems.

Though much has been published in recent years about software economics, the value that this paper adds is its extension of an existing model (CBAM) to evaluate the quality of a software portfolio, rather than a single project. This method will be flexible, quick and cost-effective for on-going portfolio management.

We showed how the results of analysis can be used to make decisions while incorporating cost-benefit analysis and architectural attributes. The key is tying architecture to cost-benefits and business value which affects investment decisions and portfolio management. Rather than just focusing on new development efforts and the software development life cycle, we can make investment decisions early, even before robust detail requirements have been completed.

This paper has answered the question: Can CBAM be applied to a software portfolio analysis for existing applications? Yes, it can as shown by the examples. Yes, we can determine whether our chosen IT investments are performing adequately using a simple tool such as CBAM.

Challenges and Suggestions

The challenges presented by this paper, along with suggestions on how to address those challenges are as follows:

a. Is Portfolio management a silver bullet? No most organizations are using portfolio management to manage IT costs in isolation but not with value. Value is seen in terms of return on investment (ROI) of IT. Applying CBAM might facilitate the integration of value, cost and risk concerns.

b. Cultural resistance is a challenge in most organizations. IT decision-making can be simplified with a tool such as CBAM, so it becomes more fact-based, repeatable and less experiential.

Opportunities for Future Work

In applying CBAM to existing software portfolio, some questions could not be addressed. They are presented here as opportunities for future work.

- Can we use different strategies of creating (sub-setting) a portfolio, by platform, language, business domain, etc., so it is easier to track implementation costs, resource costs (hardware, people and software) and implementation schedule?
- Should CBAM be focused on just the architectural strategies versus portfolio theory of reducing risks and costs?
- Which offers better value and risk reduction: CBAM evaluation of technology segmented portfolios or portfolios segmented by business domain?
- Can we vary the chosen architectural strategies to reflect a more balanced approach to risk reduction on
any given portfolio? Should the AS reflect the needs of a particular business domain, or should there be an enterprise AS with a subset of domain-specific AS attributes?

References:

[1] Kazman, R; Asundi, J; Klein, M; Quantifying the costs and benefits of architectural decisions, 2001 IEEE. 0-7695-1050-7/01


ACM 2000 1-58113-253-0/00/6


<table>
<thead>
<tr>
<th>Quality Attribute (QA)</th>
<th>Stakeholder 1</th>
<th>Stakeholder 2</th>
<th>Stakeholder 3</th>
<th>Stakeholder 4</th>
<th>Stakeholder 5</th>
<th>Avg Contribution</th>
<th>QA Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modifiability</td>
<td>0.9</td>
<td>-0.8</td>
<td>1</td>
<td>-0.6</td>
<td>0.4</td>
<td>0.18</td>
<td>10</td>
</tr>
<tr>
<td>Security</td>
<td>0.4</td>
<td>0.6</td>
<td>0.7</td>
<td>0.5</td>
<td>1</td>
<td>0.64</td>
<td>10</td>
</tr>
<tr>
<td>Usability</td>
<td>0.9</td>
<td>0.8</td>
<td>1</td>
<td>1</td>
<td>0.5</td>
<td>0.84</td>
<td>10</td>
</tr>
<tr>
<td>Performance</td>
<td>0.3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
<td>1</td>
<td>0.28</td>
<td>10</td>
</tr>
<tr>
<td>Availability</td>
<td>0.6</td>
<td>0.5</td>
<td>0.4</td>
<td>0.6</td>
<td>0.4</td>
<td>0.5</td>
<td>5</td>
</tr>
<tr>
<td>Interoperability</td>
<td>0.1</td>
<td>0.2</td>
<td>0.8</td>
<td>0.6</td>
<td>0.2</td>
<td>0.38</td>
<td>10</td>
</tr>
<tr>
<td>Reliability</td>
<td>1</td>
<td>1</td>
<td>0.7</td>
<td>0.7</td>
<td>0.9</td>
<td>0.86</td>
<td>10</td>
</tr>
<tr>
<td>Supportability</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.8</td>
<td>10</td>
</tr>
<tr>
<td>Functionality</td>
<td>1</td>
<td>1</td>
<td>0.9</td>
<td>0.9</td>
<td>0.8</td>
<td>0.92</td>
<td>15</td>
</tr>
<tr>
<td>Maintainability</td>
<td>0.7</td>
<td>0.8</td>
<td>0.8</td>
<td>0.9</td>
<td>0.6</td>
<td>0.76</td>
<td>10</td>
</tr>
</tbody>
</table>

Figure 5: CBAM Calculations for one software in selected portfolio