An Economic Model for Software Architecture Decisions

Paul C. Clements
Software Engineering Institute, Carnegie Mellon University
clements@sei.cmu.edu

Abstract

Software architecture is touted as essential for system development, but its benefits are almost never quantified. Further, architects are faced with decisions about architecture (such as deciding when an architecture has outlived its usefulness) that should be answered on an economic basis. This paper presents an simple economic modeling language that has been useful in the realm of software product lines, and argues that a similar language would be equally useful in the realm of architecture decision-making.

1. Introduction: Economics and software architecture

Designing, evaluating, documenting, evolving, and conforming to a software architecture have all been touted as essential activities in large-system development. However, their economic benefit has (to my knowledge) never been quantified or measured empirically. In many cases this lack of economic justification is perfectly acceptable; the alternative to architecture is disaster, and no justification more fine-tuned than that is necessary. However, in many projects, there are several decision points faced over time that pivot around the software architecture. These decision points should be resolved rationally, using economics as the litmus test. For example, when a large change is looming to a system, how do we know if the architecture should be modified or retired in favor of a new one? The SEI's Cost-Based Analysis Method (CBAM) [1] is an excellent approach for deciding how to prioritize changes to an architecture based on perceived difficulty and utility, but does not return dollar-amount results that would quickly enable decision-making.

This paper argues that a straightforward economic model for software architecture decision points would be both useful and easy to accomplish. We use SIMPLE, an economic model for software product line development, as an exemplar of the kind of easy model that would be useful.

2. A SIMPLE approach to economics

Recently, researchers at the Software Engineering Institute, Clemson University, the Fraunhofer Institute for Experimental Software Engineering, and Siemens created the Structured Intuitive Model for Product Line Economics (SIMPLE) [2][3][4]. SIMPLE is a language intended to facilitate decision-making in a product line context by allowing a decision-maker to calculate the costs and benefits of different decision alternatives. An example of a decision that SIMPLE was created to facilitate is: "Should I build my new product on its own, or as a member of an already-existing product line?" SIMPLE employs a small set of basic cost functions and benefit functions to allow a product line decision-maker to decompose the decision into constituent (and easily evaluable) parts.

SIMPLE is based on the observation that establishing and then using a product line engineering capability involves the following four costs:

- C_{org}: The cost to an organization of adopting the product line approach for its products. Such costs can include reorganization, process improvement, training, and whatever other organizational remedies are necessary.
- C_{cab}: The development cost to develop a core asset base suited to support the product line being built. This includes costs such as commonality/variability analysis, a generic software architecture, and the cost of developing the software and its supporting designs, documentation, and test infrastructure. The core asset base may be assembled proactively all at once, or incrementally over time using a reactive strategy as more products are introduced.

- C_{unique}: The development cost to develop unique software that is not based on a product line platform. Usually this will be a small portion of a product but in the extreme it could be a complete product.
- C_{reuse}: The development cost to reuse core assets in a core asset base. This includes the cost of locating and checking out a core asset, tailoring it for use in the intended application (if necessary), and performing the extra integration tests associated with reusing core assets.

To illustrate with an extremely simple example, the cost of establishing a product line consisting of *n* products, can be written as

$$C_{org}() + C_{cab}() + \sum_{i=1}^{n}(C_{unique}(product_i) + C_{reuse}(product_i))$$

where Corg, Ccab, Cunique, and Creuse are cost functions that, given the appropriate parameters, return the corresponding costs. Predictive formulas for many other scenarios are also available.

As one can readily see, SIMPLE does not produce numeric results on its own. Rather, it helps an analysis decompose an economic decision (too complex to quantify by itself) into smaller pieces that can be quantified. SIMPLE produces thought experiments, rendered in a divide-and-conquer fashion to turn large questions into a series of smaller questions that can be quantifiably answered.

In addition to the cost functions enumerated above, SIMPLE uses a set of benefit functions that "return" the quantified benefit of decision alternatives. For example, the true cost of building a product line must be counterbalanced against the benefit of (for instance) the time to market advantage that future products will enjoy. Analysts are free to define their own benefit functions, to capture whatever benefits are germane. They are also free to add new cost functions; a common one is the cost of building a product from scratch without using a product line approach.

4. A SIMPLE approach to questions of architecture economics

Obviously, SIMPLE as originally constructed does not directly apply to the kinds of architectural decision points alluded to earlier. It was presented as an example of the economic model that we need: Easily understood, straightforwardly used, and capable of decomposing complex economic questions into a set of smaller and more easily quantified questions.

For example, the benefits of software architecture documentation are often touted but never quantified. It should be straightforward to come up with an expression in the style of SIMPLE that defines the economic benefit of producing documentation. Consider the following [5]:

$$\Sigma_{(i=1,n)}\ \Delta(\text{cost of activity}_i) - \text{cost of documentation}$$

The Δ term refers to the cost of performing the activity without documentation minus the cost of performing the same activity with documentation. Presumably the latter will be lower. Activities whose cost should be affected by the presence of absence of architecture documentation include downstream design, coding, analysis, project management planning, testing, and maintenance. For the activities not affected by the presence of documentation, the delta is zero, and so these do not contribute to the equation.

The equation above can also be used to measure the benefit of high-quality documentation versus low-quality documentation. Presumably, high-quality documentation (the precise meaning of which is context-dependent) will result in lower-cost activities, but at the price of imbuing the documentation with that higher quality.

As before, an equation like this is not meant to be solved in great detail, but is an invitation to do a thought experiment. How much does it cost to provide documentation? This can be roughly estimated by (time required x hourly rate). How much will documentation lower the cost of, say, downstream design? This can be roughly estimated by (number of questions a designer will have x time it takes to answer each one with and without documentation x hourly rate).

Further, the equation need not be settled completely. If we are merely after justification for producing documentation, we can stop as soon as the improvement outweighs the cost. Even if we don't add up all the benefits, the point will have been made.

5. Other questions

What other architectural questions might a SIMPLE-style economic model quickly help us answer? The following seem viable candidates:

- Shall we retire the architecture or continue to evolve it?
- Shall we take the time and expense to subject the architecture to a formal evaluation?

- In our architecture documentation, shall we produce a specialized view for a particular stakeholder, such as a safety or security analyst, or shall we ask that analyst to use views that we have created for other developers?
- How beneficial is capturing architectural rationale?

6. Conclusions and directions

SIMPLE has proved valuable in the realm of software product line decision-making, despite (or more likely, because of) its breathtaking simplicity. Its strength is that it provides a structured way to decompose an economic question into more easily answered components. It also works well because decision-makers aren't in search of answers with great precision, but rather are seeking first-order justifications for taking certain actions. It is the claim of this paper that a similar modeling language could be equally useful in the realm of architecture decision-making.

If this approach seems promising, then we should craft a set of scenarios that our modeling language could answer, create equations to answer them, and then see if a set of common cost functions (similar to SIMPLE's quartet) emerge. Finally, we should try to investigate some industrial situations so that we could apply our approach in practice.

7. References

[1] Asundi, Jayatirtha; Kazman, Rick; & Klein, Mark. Using Economic Considerations to Choose Among Architecture Design Alternatives (CMU/SEI-2001-TR-035).

[2] Clements, Paul C.; McGregor, John D.; & Cohen, Sholom G. *The Structured Intuitive Model for Product Line Economics (SIMPLE)* (CMU/SEI-2005-TR-003).

[3] Böckle, Günter; Clements, Paul; McGregor, John D.; Muthig, Dirk; Schmid, Klaus. "A Cost Model for Software Product Lines," *Fifth International Workshop on Product Family Engineering (PFE-5),* Siena, Italy, November 4-6, 2003.

[4] Böckle, G.; Clements, P.; McGregor, J.D.; Muthig, D.; Schmid, K. "Calculating ROI for Software Product Lines," *IEEE Software*, Volume 21, Issue 3, May-June 2004, pages 23-31.

[5] Results of working group on documentation, 2005 SEI Software Architecture Technology User Network (SATURN) workshop, Pittsburgh. Unpublished.