Unified Software Engineering Reuse (USER) using Stable Analysis, Design and Architectural Patterns

Mohamed E. Fayad, PhD[1] and Charles A. Flood III[2]
Department of Computer Engineering,
Charles W. Davidson College of Engineering,
San Jose State University, San Jose, CA, USA
[1]m.fayad@sjsu.com and [2]charlesFlood3@gmail.com

***Abstract*—The endless pursuit for creating effective systems for software reuse has continued for as long as software has existed. To date, there have been few, if any, such effective systems created for ensuring a high degree of reusability from one project to the next. The inherent tendency for projects to demand substantial alterations, despite being designed for maximum reusability, remains strong evidence of this fact.**

Software reusability makes study of stable analysis, design, and architecture patterns a domain of immense interest. By extrapolating the stable concepts that use software stability model (SSM) and Knowledge Maps (KMs), we can realize software solutions that do not need excessive alterations, changes or additions. Such patterns function as a framework, to which new objects can be added depending only on the uniqueness of the scenario to which it is applied. The effectiveness of these techniques will be demonstrated in this paper.

Keywords—Unified Software Engineering Reuse (USER); Software Reuse; Software Stability Model (SSM); Knowledge Maps (KMs); Stable Analysis Patterns; Stable Design Patterns; Stable Architecture Pattern

I. Introduction to the Software Stability Model

Patterns are models that are reusable in the future. The problem with many existing software engineering patterns is they are generally domain dependent, and using them for entirely different applications could be very hard, because some modifications or changes will have to be made. Using software stability concepts to generate patterns promotes greater reuse as stable models, which use Enduring Business Themes (EBTs) [5, 6, and 7], and Business Objects (BOs) [5, 6, and 7] created for keeping the goal of a system in mind. The goal of a system, which forms the foundation of the pattern rarely changes, and hence the models generated from them are more stable. By stability, we mean that the pattern itself never changes, although it is extendable for use in various applications irrespective of the domain.

Stable Patterns [13 and 14] always depict concepts and theories. Concepts are more likely to be implicit, and they are always associated with semi-tangible objects, through which they will become explicit. The concept could be inherent in the objects – it could represent a property of the object or it could even represent a relationship between the objects. Through the pattern, we may recognize the objects that always appear with the concept. However, the object abstractions that we create should be extendable in a manner that allows them to be application independent. Using the software stability concepts, it becomes very convenient to model patterns that do not change over time, as they encapsulate the core concept of a system. The concepts modeled as Enduring Business Themes (EBT's) and the semi-tangible objects associated with the concepts modeled as Business Objects (BO's) are the foundations on which the stability aspects are developed which are extended to Industrial Objects (IOs) through hooks or extension points, such as the patterns of Gang of Four, Simons Group, etc.

Figure 1 depicts the three layers of the software stability model [4, 5, and 6]. The detailed characteristics of EBTs, BOs, and IOs Layers.

Fig. 1. Layers in SSM

In SSM, no pattern is isolated from other design and analysis patterns. Each pattern is composed of EBT and BO objects which have their own associated patterns. Accordingly, in models figures displaying analysis, design, or architecture patterns, the presence of a secondary pattern will be shown along with the type of object (e.g. <P-BO> means a business object that is also a pattern). These model objects will be appended with the word "Any", in reference to the broad range of options available. For ease of reading, however, all Business Objects will merely be highlighted and dropped rather than prepended.

II. Stable Analysis Patterns

Stable Analysis Patterns (SAPs) are synthesized based on the concept of an Enduring Business Theme, or EBT. These EBTs are intangible, implicit to a given scenario, and extremely stable over time. An EBT is described in the form of an SAP by using some common non-tangible concepts, known as Business Objects. An example of an EBT is the concept of reputation. This concept can be represented as a model, then in software, to apply to many contexts where reputation is a needed factor.

978-1-5090-4171-8/16/$31.00 ©2016 IEEE

A. Sample Applications

In an online business scenario, such as eBay or Amazon, reputation is a critical component for carrying out profitable business operations (***Type***). In this scenario, a seller (***Type***) and buyer (***Party***) use eBay or Amazon (***Mechanism***) to sell a product (Entity). The sale is usually contingent reflected on the degree of satisfaction (Factor) of others buyers which they express as a positive rating (Rate) through their opinions of others.

In the domain of personal music player marketing of the popular media player, iPod (***Entity***/***Mechanism***), Apple (***Party***) is the undisputed market leader (***State***) that is known to generate significant annual revenue (***Rate***) due to the very high demand (***Factor***), by customers (***Party***), within the said business segment (***Type***). This demand is based on Apple's reputation for producing high quality players.

B. Sample Solution

Using the two examples above, it is possible to model workable solutions for the given contexts.

In the first context, the model can be assembled as shown in Figure 2 below. The core concepts are extended with Industrial Objects (IOs) in order to better fulfill the requirements of the scenario. A short description of these basic requirements is as follows:

- The ***Party*** (Seller) sells the Entity (Product) through the ***Mechanism*** (eBay or Amazon).
- The ***Mechanism*** (eBay) is visited by the ***Party*** (Buyer), who finds the ***Entity*** (Product), sold by the ***Party*** (Seller) worth buying.
- The ***Party*** (Buyer) buys the ***Entity*** (Product).
- The ***Party*** (Buyer) likes the ***Entity*** (Product), which creates ***Factor*** (Satisfaction) in ***Party*** (Buyer).
- The ***Party*** (Buyer), through the ***Mechanism*** (Feedback), provided in the eBay/Amazon website records his ***State*** (Opinion).
- ***State*** (Opinion) impacts the ***Rate*** (Positive Rating) given to ***Party*** (Seller).
- The ***Rate*** (Positive Rating) causes good Reputation to ***Party*** (Seller)

The second scenario can be similarly modeled like before as seen in Fig 3.

In this scenario, much like the last one, the IOs extend the common core classes, such as Mechanism, Entity, and Actor, to satisfy the requirements. One can also easily observe that the core classes are shared across both contexts. The resulting solution can be described as follows:

- The ***Party*** (Apple Inc) captures ***Factor*** (Market Demand) for ***Entity*** (Music Player).
- The Party (Customer) is looking for a good quality Entity (Music Player).

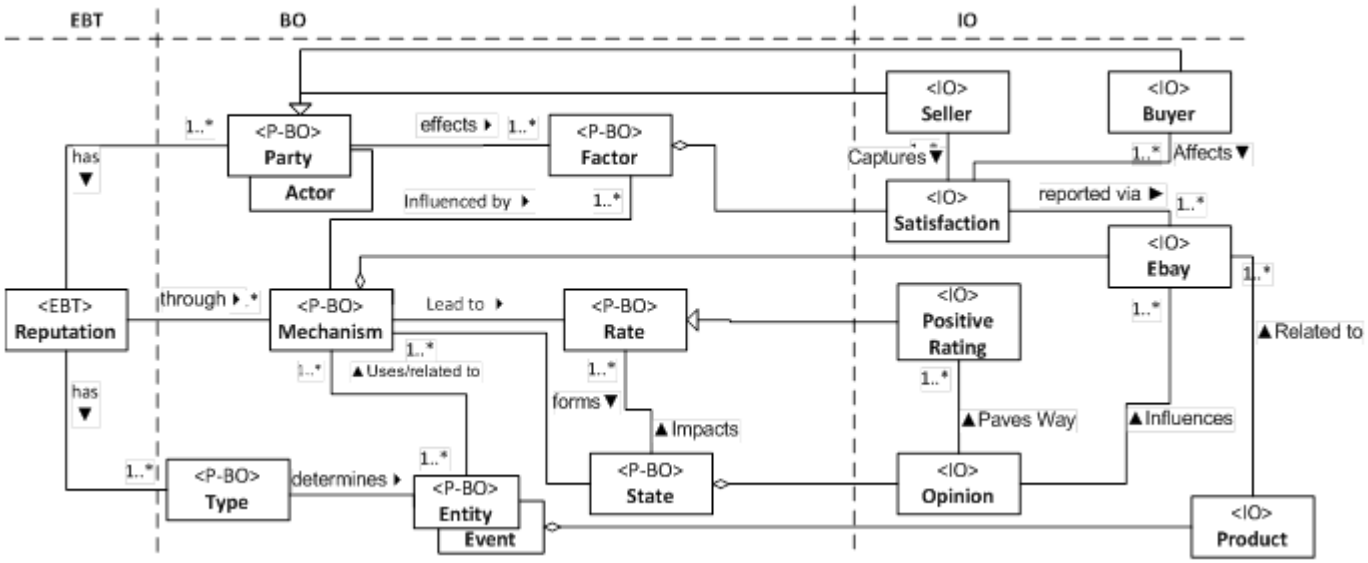

Fig. 2. Reputation Analysis Pattern for online EBay Sales

978-1-5090-4171-8/16/$31.00 ©2016 IEEE

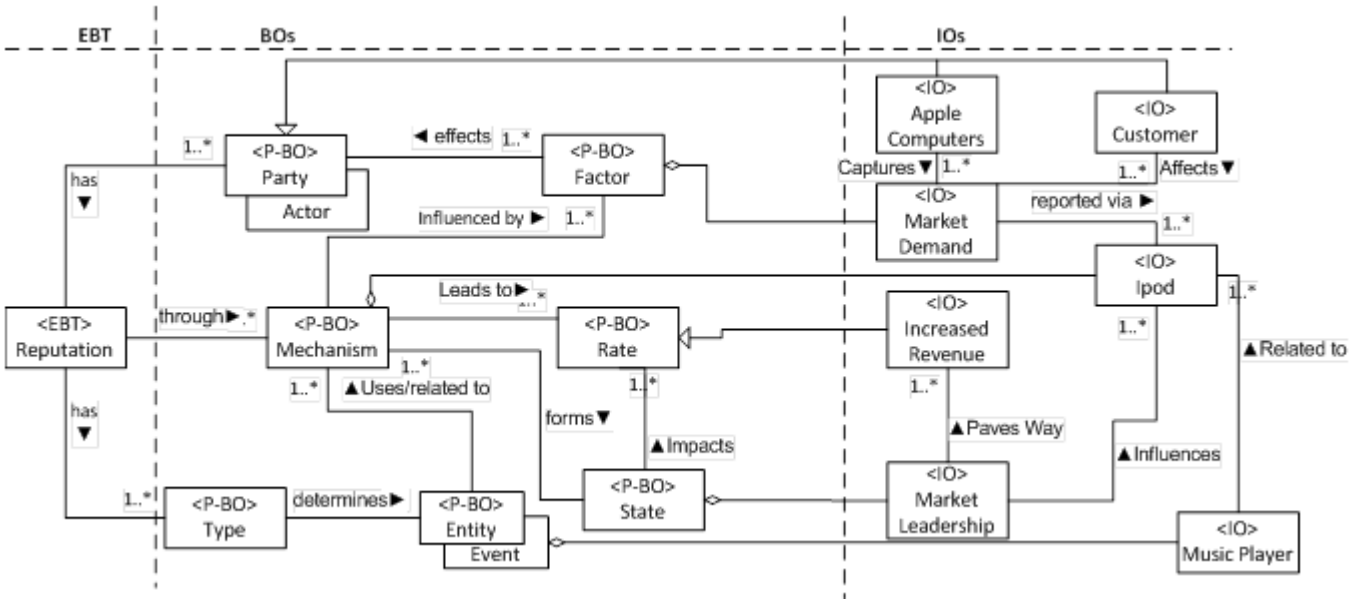

Fig. 3. Reputation analysis pattern for sale of Apple iPods

- The Entity (Music Player) details are obtained by Party (Apple Inc).
- The Party (Apple Inc) develops the Mechanism (IPod).
- The Mechanism (IPod) is bought by the Party (Customer).
- The Party (Customer) is influenced by a Factor (Quality) of the Mechanism (IPod).
- The Factor (Quality) helps the Party (Apple Inc) gain Reputation by increased Rate (Revenue).

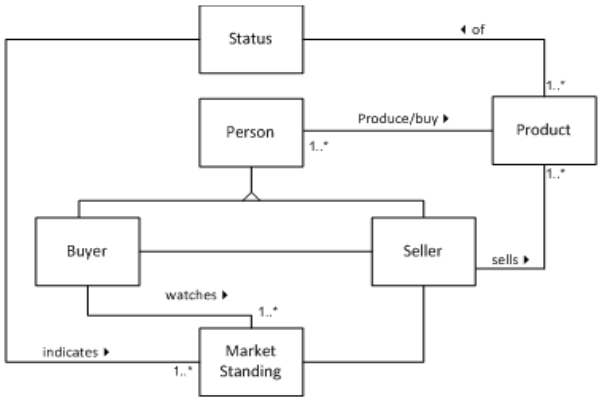

Fig. 4. Traditional Model for Reputation

This Stable Software Model (SSM) can be compared to a traditional model in this scenario as shown in Figure 4. In this model, only the industrial objects are used, which necessitates their replacement in other contexts and scenarios.

III. Stable Design Patterns

The main distinction between a Stable Design Pattern and a Stable Analysis Pattern is that the Design Pattern is focused on or built from a Business Object, rather than an Enduring Business Theme. Business Objects differ from EBTs in their temporal nature, having a beginning and an end. Beyond this, similar rules for representation apply.

It is important to note that Stable Design Patterns are quite different from the "design patterns" that are typically discussed and attributed to the "Gang of Four". While such patterns are used as strategy, observer, and decorator to demonstrate patterns in problem solving techniques, they are conceptually distinct from the Stable Design Pattern, which instead, abstracts an entire set of situations or scenarios, rather than problem solving practices.

To illustrate Stable Design Patterns, let us use as an example the concept of Influence.

978-1-5090-4171-8/16/$31.00 ©2016 IEEE

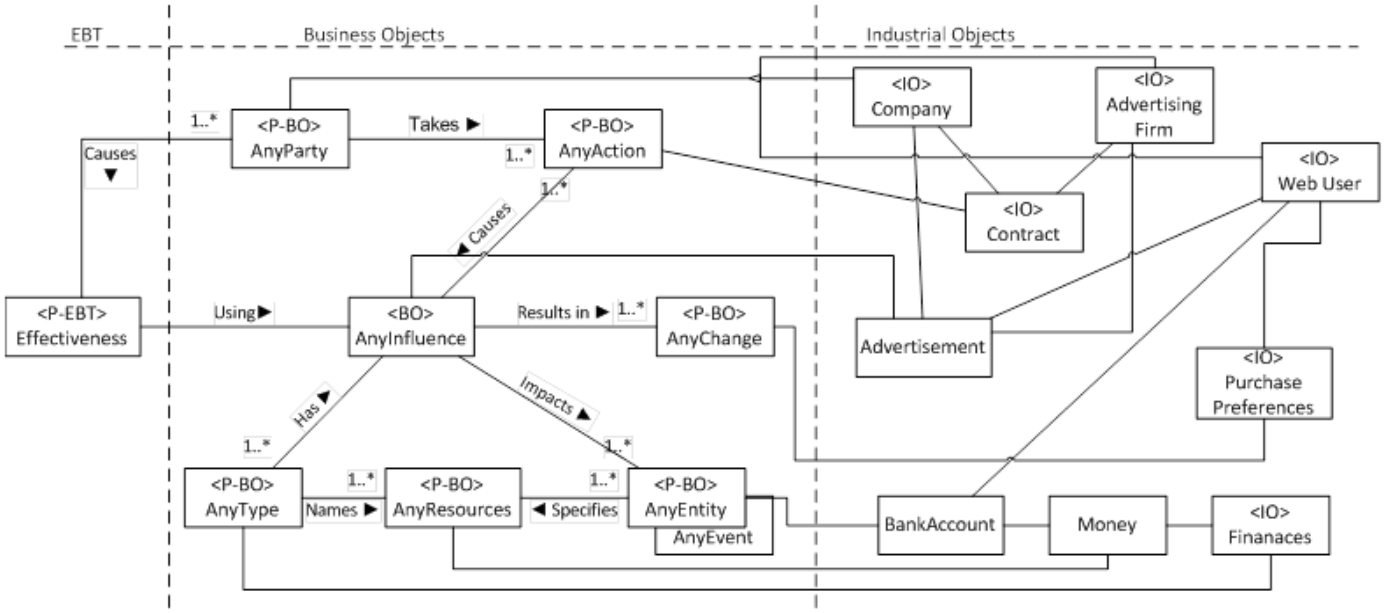

Fig. 5. Influence Design Pattern for Web Advertising

A. Sample Applications

Consider the case of a company choosing advertising on the web to influence the spending habits of the online shoppers. The business will begin by first negotiating a contract with an advertising firm, who will assist in this endeavor, for a negotiated price. The advertising firm will then assist the company in making short videos, banners, and web site side bar. Once the content is ready, the advertising firm places the advertisement in their database for future use by an analytics engine, which will determine web sites where the advertisements are likely to perform better. The company that wants advertisement services will be billed according to the terms of the contract, which will most depend on consumer web response, measured in views, hits, or rollover time. These statistics will be combined with sales data to inform the companies how effective the advertising was and it will eventually help them make decisions concerning any future advertisements.

The role of influence can influence family members at home too, with something as trivial as an argument over which sports team is superior, especially when one's home team is not involved. For example, last year's Super Bowl was played between the Seattle Seahawks and the New England Patriots, but the majority of viewers were not from either Seattle or New England. It is not unthinkable that a family, in itself ready for Thanksgiving dinner would start a minor dispute over which team would emerge victorious. The father may argue his case for the Patriots with their seasonal statistics to back up argument, while the son might argue in favor of the other team. Eventually, one side may win the debate with forced and sufficient arguments even before the game ends or dinner begins.

B. Sample Solution

In the web advertising example, the only objective of advertisement is to inform people (***Party***) that they can chose a product or service (***Resource***) that the advertised company (***Party***) provides. Through the purchase and use (***Action***) of a billboard (***Influence***), the organization exerts financial (***Type***) influence on the potential consumer. After seeing the billboard, the consumer may have a future need (***Event***) for the product or service. Instead (***Change***) of searching or choosing a provider at random, the consumer may now purchase from the organization eventually impacting the bank accounts (***Entity***) of both parties. Figure 5 above models this scenario.

A similar case can be made for the family arguing over which football team in the Super Bowl should win. The influence is bi-directional in this case, as fans of both teams (***Party***) attempt (***Action***) to win over (***Change***) the other side through an informal debate (***Influence***). Either side of the debate will spend their time and energy (***Resources***) toward the goal of winning the argument, until one side wins (***Event***), or until something more important comes up – like an announcement for dinner. This scenario is given in Figure 6.

These scenarios can be compared to a more traditional model of an instance of influence. Such a traditional model is normally composed only of the Industrial Objects, most, if not all, apply only to a single scenario.

IV. STABLE ARCHITECTURE PATTERNS

Stable Architecture Patterns, can be defined as blending of two or more design or analysis patterns. This is done simply

978-1-5090-4171-8/16/$31.00 ©2016 IEEE

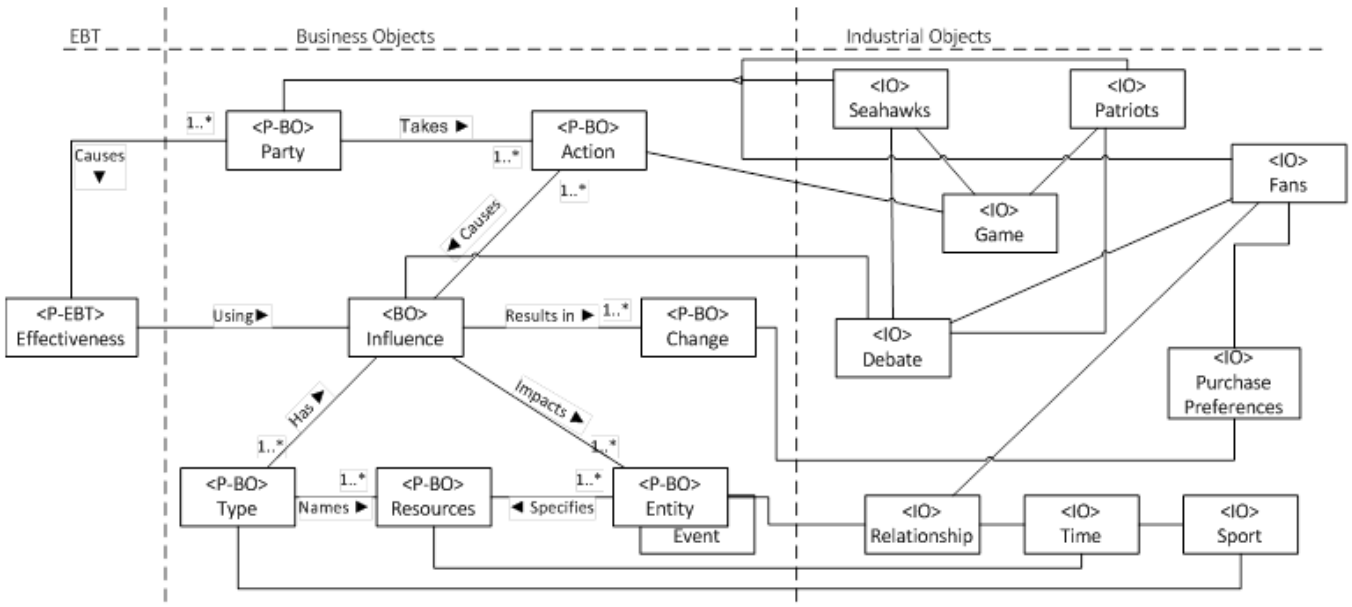

Fig. 6. Influence Design Pattern in Super Bowl Argument

by merging the core concepts shared between patterns, so that classes (such as Actor and Party) are not duplicated. For an example of a Stable Architecture Pattern, consider Conflict Analysis, where analysis is the EBT and conflict, being more temporal, is a BO.

A. Sample Applications

Conflict analysis can have many forms depending on its source of origin. Though there is infinite variety in the types of conflicts that are available for analysis, some of the more accessible ones may arise from highly publicized reality TV shows, where legitimate personal issues are raised to provide entertainment to the viewing public. For example, midway through the filming season of Five of Teen Mom, the MTV producers suddenly changed their opinions of removing Farrah Abraham from the show, because of her other commitments and later invited her back perform again to act in the series. This act made another cast member very upset and the actor decided to terminate her portion of acting because of the confusion caused by differing viewpoints on some of the external issues. To smooth sail through the show and to dispel bitter acrimony, the situation was reviewed again by the cast member, who eventually decided to remove herself from filming process, but agreed with the MTV producer to continue the show without her son.

A conflict arises due to natural complexities involved in the process and most of them are related to inter-personal and emotional issues of the actors involved. Another classical scene would be a complicated labor strike in an auto spare parts supplier unit for the GM; wage reductions could be one of the main reasons for the strike. In this example, an attempt by the spare parts manufacturing American, Axle, to slash labor costs (including pension and health care benefits) by as much as 50%, caused an unprecedented uproar in the United Auto Workers Union. As a result, auto supplies to almost thirty facilities that manufactured GMC vehicles were seriously affected. Eventually, these facilities started operating on just a single shift or they were closed indefinitely.

B. Sample Solutions

In the example of the MTV interpersonal conflict, the story begins with MTV and it's Cast Members (***Party***). When considering the careers (***Aspect***), an analysis of the conditions (***Conflict***) of employment defined by contracts (***Form***), should results in a plan of action (***Consequence***). This plan will need to be developed at meetings (***Event***) with both the producer and editor (***Party***) and consider such things as revenue (***Factor***), and it might ultimately affect subsequent filming sessions (***Media***) of later episodes and future seasons (***Entity***).

In the case of the auto parts manufacture strike, the primary parties involved include workers and other employees, the company board members and the labor union (***Party***). The workers, due to the effect of inflation (***Factor***), submit their demands (***Form***) to management and decide to strike and boycott working (***Conflict***), unless they are given a raise (***Consequence***). The labor union, in accordance with the Labor Beneficiary Act (***Context***), enters into negotiation (***Event***) for a new contract (***Media***).

V. Analysis

In the simple examples given above, it should be noted that the Stable Software Model, though requiring additional initial work than the traditional models, are also more widely applicable to new and different contexts. For more refined evidence, a study of few relevant metrics might be helpful.

A. Quantitative Measurements

Perhaps the easiest measurement is reusable class count. In the SDP example of web advertisement, there are eight classes in the traditional model, of which none are applicable to the alternative influence example of a family debate over sports. The SSM, in contrast, shares all nine core classes (EBTs and

978-1-5090-4171-8/16/$31.00 ©2016 IEEE

BOs) across both scenarios. Likewise in the SAP examples involving product and vendor reputation, the traditional model does not have any adaptable classes that could make it stable and extendable. The SSMs on the other hand, can easily maintain their core classes even with newer scenarios.

There are additional potential metrics, cyclomatic complexity, for example. This metric is used to determine the interconnectivity of the objects in a given design or model. The calculations are included in Table 1, given the equation

$$M = E - N + 2P \quad (1)$$

Where M is the cyclomatic complexity, E is the number of edges (connections between classes), N is the number of classes, and P is the number of connected components, which for software models is almost always one.

TABLE I. CYCLOMATIC COMPLEXITY

Model	***E***	***N***	***P***	***M***
Traditional Influencce Model	10	8	1	4
SSM Influence Design Pattern	9	9	1	2

A greater cyclic complexity has a number of ramifications. First, it shows the complexity of the program itself, usually meaning higher maintenance costs. It also shows strong interconnectivity, which makes the program less modular and more difficult to change and adapt. Finally, it also shows how much testing is required to be reasonably assured of the program's proper execution. For each of these, a smaller cyclomatic complexity is desired. It is most evidently delivered here through the use of SSM.

B. Qualitative Measurements

We can also measure the quality of the various models shown above with respect to some desirable parameters like the reusability factors of various models. In the Stable Architectural Pattern example, a developer can consider a total of 15 core classes along with many peripheral IOs. If one considers only the core classes, then the model is 100% reusable. As shown in table 2 below.

We can see this clearly, given the formula

$$RF = C_R/C_T \quad (2)$$

Where RF is the Reusability Factor, C_R is the number of Reusable Classes, and C_T is Total Class count.

TABLE II. REUSABILITY FACTOR OF MODE

Model	C_R	C_T	***RF***
Traditional Influencce Model	2	8	25%
SSM Influence Design Pattern	9	9	100%

Evidently, the SSM model has a higher reusability that the traditional model, which was only given the two classes that were, in a sense, outside the scope of the programs control (money and bank account). All other objects would require significant rewriting to work in a new situation. Meanwhile, the purpose of SSM is to reuse the core classes, extending only as required, which allows us 100% reusability of the core classes.

VI. CONCLUSION

With the sample scenarios and related metrics given above, Stable Software Models are far more adaptable software solutions when compared to traditional modeling techniques. Though we can generate relatively flexible models, such as the one given in Figure 3 that can apply to a range of closely related scenarios, a more effective long-term solution, SSM, still remains more relevant for a variety of scenarios. In the meantime, we recognize the need for some initial startup cost of implementing the core classes involved in the process. However, the core of the program may easily be reused with minimal or no changes which eventually makes this option more attractive.

In addition, one will also find that there is some additional sharing of classes between the analysis and design patterns that are mentioned above. Both of them include ***Actor***, ***Party*** and the ***Entity***. When these classes are implemented, they can be indexed in a pattern library to be used in all future patterns, which further increases reuse thereby reduces software development costs.

REFERENCES

[1] C.A. Flood. Unified Software Engineering Reuse (USER). Master's Thesis, San Jose State University. (In Progress)

[2] W.J. Tracz, Software Reuse Myths. ACM Software Engineering Notes, volume 13, Number l, January, 1988, Pages 17-21

[3] Griss, M.L. "Software reuse: From library to factory", Hewlett-Packard Laboratories, 1501 Page Mill Road, Palo Alto, California 94306, USA

[4] M.E. Fayad, and A. Altman, "Introduction to Software Stability", Communications of the ACM, Vol. 44, No. 9, September 2001.

[5] M.E Fayad. "Accomplishing Software Stability." Communications of the ACM, Vo. 45, No. 1, January 2002, pp 95-98

[6] M.E. Fayad, "How to Deal with Software Stability", Communications of the ACM, Vol. 45, No. 4, April 2002, pp 109-112.

[7] H. Hamza "A Foundation for Building Stable Analysis Patterns." Master thesis. University of Nebraska-Lincoln, 2002

[8] H. Hamza. "Building Stable Analysis Patterns Using Software Stability". 4th European GCSE Young Researchers Workshop 2002 (GCSE/NoDE YRW 2002), October 2002, Erfurt, Germany.

[9] H. Hamza and M.E. Fayad. "A Pattern Language for Building Stable Analysis Patterns", 9th Conference on Pattern Language of Programs (PLoP 02), Illinois, USA, September 2002.

[10] H. Hamza and M.E. Fayad. "Model-based Software Reuse Using Stable Analysis Patterns" ECOOP 2002, Workshop on Model-based Software Reuse, June 2002, Malaga, Spain.

[11] A. Mahdy and M.E. Fayad, "A Software Stability Model Pattern," in Pattern Language of Programs, Monticello (PLoP2002), 2002.

[12] M. E. Fayad, H. A. Sanchez, S. G .K. Hegde, A. Basia, and A. Vakil. "Software Patterns, Knowledge Maps, and Domain Analysis". Boca Raton, FL: Auerbach Publications, December 2014.

[13] Fayad, M. E., "Stable Analysis Patterns for Software and Systems". Boca Raton, FL: Auerbach Publications, August 2015.

[14] Fayad, M. E., "Stable Design Patterns for Software and Systems". Boca Raton, FL: Auerbach Publications, August 2015.

[15] Fayad, M. E., "Stable Software Architecture on-Demand for Software and Systems". Boca Raton, FL: Auerbach Publications, December 2015.

978-1-5090-4171-8/16/$31.00 ©2016 IEEE