A Cutting-Edge Unified and Stable Rule Design Pattern

Mohamed E. Fayad[1(✉)], Gaurav Kuppa[1], Siddharth Jindal[1], and David Hamu[2]

[1] San Jose State University, San Jose, CA, USA
{m.fayad, gaurav.kuppa}@sjsu.edu,
siddharthajindall@gmail.com
[2] Liberty Consulting, Phoenix, AZ, USA
dave.hamu@gmail.com

Abstract. Often, changing market dynamics require business applications to quickly and efficiently adapt to the needs of the ensuing business environment. Business Rules excel in delivering software solutions that are implicitly adaptable to changing business requirements; thus they can prove to be an effective tool to provide necessary flexibility and control for rapidly deploying changes across a wide array of business operations. When a proper design is employed, business rules provide a robust and capable way of enabling enterprise software that adapts to changing business needs. In other words, business rules find varied applications and ways of use, for example, managing a pending problem, using it as production rules and for facilitating collaboration between various systems, etc. However, despite a plethora of tools and technologies available, most organizations still find it difficult to define or model workable business rules explicitly. Furthermore, from a macroscopic point of view, rules are important and inseparable artifacts in governing a software application to make it comply with the system goals. In this paper, the current ways to manage business rules along with their pitfalls are discussed, and therefore, a new approach for developing rule-based business architectures is proposed. The proposed approach allows for both managing and reusing business rules.

Keywords: Software reuse · Stable design patterns · Software Stability Model · Business Rules · Knowledge map · Adaptability

1 Introduction

Modern business applications require a system that has the ability to efficiently and effectively manage its processes to align itself with the business goals. Pertaining to specific needs and requirements, a number of tools to generate and manage business rules currently exist in the market. However, existing rule generating solutions possess formidable limitations, including:

(1) Very high development and maintenance cost.
(2) Prohibitive demands for scarce technical expertise.

K. Arai et al. (Eds.): FTC 2019, AISC 1069, pp. 644–653, 2020.
https://doi.org/10.1007/978-3-030-32520-6_47

(3) Inadequate expressive semantics for modeling very complex solutions.
(4) Lack of administrative features required to manage complex rule-base deployments and versioning. majority of these issues center around the limits of conventional modeling approaches which contend with tangible rather than changing aspects of a software system. A flexible approach to system modeling is paramount for a successful business rules-based solution.

Some software engineer may counter that rules engines are entirely unnecessary since a programming language is in and of itself a rules engine. This perspective is naïve and is equivalent to asserting that a plumbing system is but an assortment of pipes and fittings.

A rule, in general, is defined either to make an orderly system or to obtain uniformity for reaching the target. It is a verdict that must be followed while carrying out the set of operations for which it is specified. In essence, a rule finds application in almost every domain because it is the basic building block behind managing and controlling a system. Therefore, it is imperative to understand how rules can be generalized and reused to suit any number of applications. According to Software Stability Model (SSM), a rule can be classified as a Business Object (BO) as it has a specific duration and is subject to changes over a period of time [7–9]. AnyRule Stable Design Pattern [2–6] illustrated in this paper, identifies the core knowledge behind a rule, related to its Enduring Business Theme (EBT) which is *Governing*. This clear separation of knowledge from problem-specific artifacts makes it stable and reusable over an unlimited number of applications [11].

In short, we assert that business rules-based software modeling is a wholly unique process unto itself. The sort of generalizations which must be employed to successfully model a dynamic, flexible and adaptable system of rules for a given domain is far more specialized than one will find in a straightforward deterministic algorithm designed for a solution that is not designed to be changed. Furthermore, the modeling approach and methodology is different than what is employed when building software with an object-oriented language. Foremost, the understanding of the solution domain must be far deeper than the analysis that s required for a non-rule-based application. Software Stability Method is a specialized approach to software engineering that targets those essential aspects of a solution that are enduring (and therefore stable over time) and distinguishes those aspects of a model from those aspects of a solution that are dynamic and changing over time.

Section 2 of this paper discusses the problem associated with the existing ways of modeling systems based on traditional methods and elaborates this with the help of a few sample scenarios. It also touches upon the issue of current systems having failure rates and how well-defined rule-based systems can be a solution to this problem. In the latter half of this section, the new Stable Rule Pattern is discussed illustrating its reuse potential. In Sect. 3, we will discuss a related rule-based traditional pattern and measure it against the new stable design pattern: this evaluation, both qualitative and quantitative criteria. Section 4 will then provide a conclusion about this work, followed by references to end the paper.

2 The Study

2.1 Problem

In the Information Technology (IT) sector, various studies have shown high failure rates for the projects due to reasons ranging from confusing or changing requirements to poor design and inappropriate alignment of business processes with the organization's goals and objectives. According to an estimate by the US Government Accounting Office (GAO), it was found that out of a total of 840 federal projects, approximately 49% were either poorly planned or performing poorly. In 2014, a market survey conducted by Project Management Institute showed only 42% of organizations having a high alignment of projects to organizational strategy. Furthermore, according to a worldwide estimate, organizations and governments will spend approximately $1 trillion on IT services and infrastructure in the next few years. Most likely, 5 to 15% of the initiated projects will be found inadequate and subsequently abandoned while many more will be over-budget or over-time and may even need massive reworking. In other words, only a few of these IT projects will truly succeed.

The reasons behind the failure of new business initiatives or improvement of an existing business process are often complex. But it often comes down to the fact that the underlying problems the organization was facing were poorly understood or the process improvement initiatives were poorly attuned to solving the real needs of the organization [1]. This is largely due to the fact that these initiatives were not properly governed or elicited, owing to a lack of clearly defined and stable rules. Most organizations have a hard time identifying and articulating these rules because the rules and logic are scattered across the organization and owned by several stakeholders who may not possess the complete visibility of the interconnected nature of the problems that they are trying to solve. This makes the business applications susceptible to failures as the business itself may lack the agility and domain knowledge needed for adapting to rapidly changing business situations.

2.2 Discussion

Traditionally, a rule is understood in the limited context of a term or a statement which is accepted as true and used as a basis for reasoning or conduct. However, rules are the fundamental unit that governs an application in almost every domain irrespective of the problem concerned. Figure 1 depicts the scenario of a typical Loan Application process.

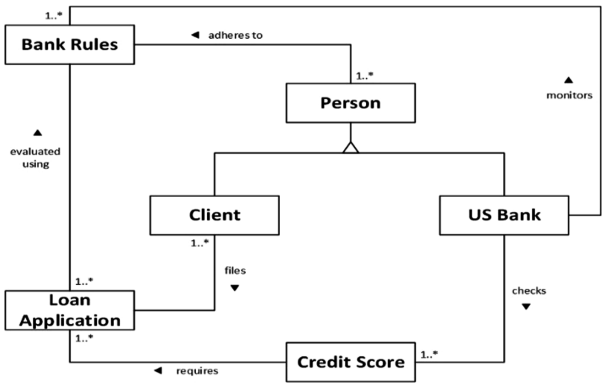

Fig. 1. Traditional Loan Application process

In this example, a Loan Application is submitted to a Bank, which is then evaluated as per the Bank Rules. However, any change in the application requirements or components will lead to the re-designing of the complete application architecture. For example, another application in similar context can be a sports application. In such a scenario, a Coach may record data and select players on the basis of their performance in some Training Match and even finalize their Playing Positions. But if we compare it with the earlier model, we can find striking similarities in the underlying context of the two models. Figure 2 gives an illustration of how another application deals with problem-specific and distinct classes but in a similar context, i.e. rule-based evaluation of players for selection into the final team.

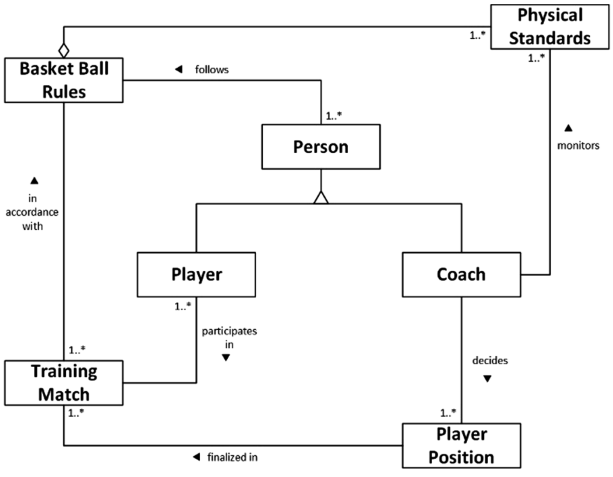

Fig. 2. Traditional model for a sport application

Let us consider one more example of an application dealing with rules in another way. Figure 3 shows an application scenario for a Conference Enrollment system. This model again consists of completely different classes if compared with the earlier two, but as can be seen, the underlying concept is about dealing with a set of rules to enroll for a conference.

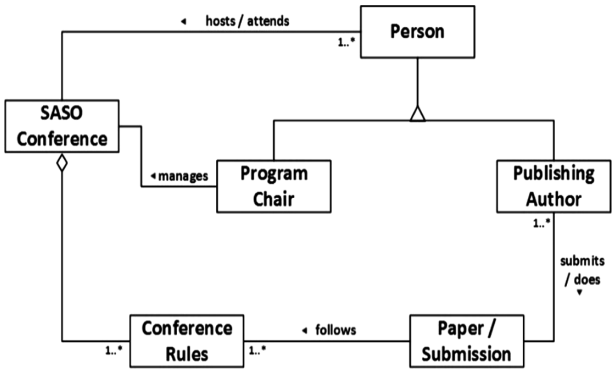

Fig. 3. Traditional conference enrollment model

All the afore-mentioned application scenarios highlight an important and glaring limitation with the traditional modeling approach [10], and that is, *Lack of Stability*. In traditional UML, a rule pattern that is self-adaptive to ever occurring changes and self-organizing to the problem at hand does not exist. Because of this, we need to rethink the problem every time we are required to model an application dealing with rules. This research demonstrates AnyRule as a Stable Design Pattern

2.3 Results

(a) The Stability Approach

As discussed earlier, a rule is the basic building block behind managing and controlling a system. A rule usually consists of a set of conditions and acts as a standard for different activities. It also provides detailed direction on how to convert a given strategy into actions. However, for the rules to have any value, they are required to be collected and stored in a manner that is consistent within the framework of the business requirements. Well-organized and well-structured rules become vital information for defining processes and executing actions. Given below are the stable representations of the application scenarios discussed earlier in this column. These applications are re-modeled using the newly proposed AnyRule Stable Design Pattern [5, 11]:

I. *Rules in Banking:* A bank, for example, Bank of America (AnyParty) defines the minimum criteria like a credit score (AnyConstraint), for or above which only they accept the credit card applications (AnyEntity). For this purpose, they hire professionals like underwriters (AnyParty), who evaluate each credit application

(AnyEntity) based on the applicant's credit score (AnyConstraint). However, to administer these requirements (AnyConstraint), they are governed by a set of Business Rules (AnyType) which are usually statements (AnyRule) consisting of conditions which help them evaluate each application without any bias or prejudice. These rules are generally recorded on rule manuals (AnyMedia) which are a part of the Business Requirement Document (see Fig. 4).

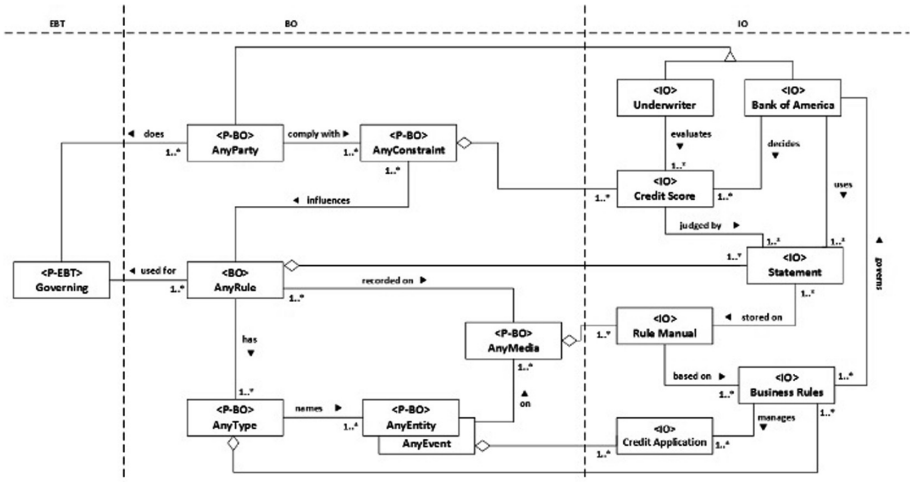

Fig. 4. Stable pattern: Loan Application process

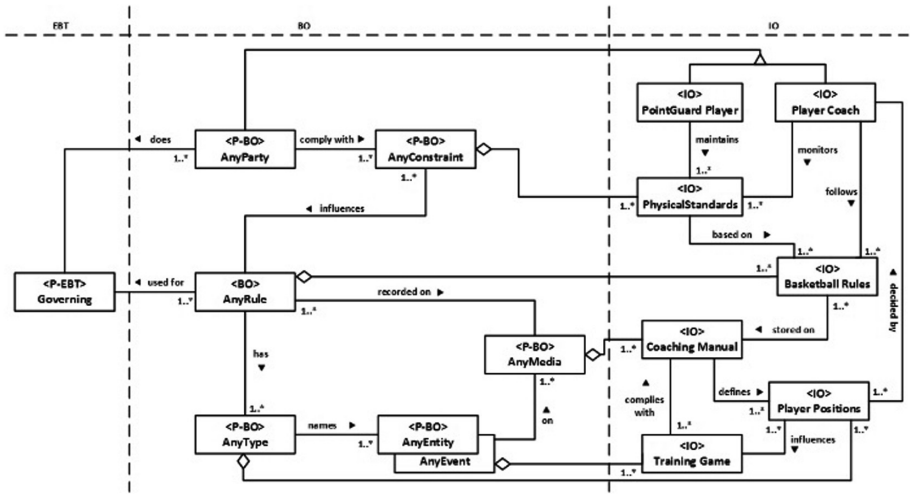

Fig. 5. Stable model for a sport application

II. *Rules in Sports:* During the trial and selection of basketball players (AnyParty), a coach (AnyParty) defines the minimum physical standards (AnyConstraint) for selection, based on which the players appear for trials. For selecting the best players, the coach (AnyParty) uses a defined set of rules (AnyRule) as recorded on the coaching manual (AnyMedia). The selection of the players has to be carried in accordance with the standard international basketball rules (AnyType). The players (AnyParty) performing best in practice matches (AnyEvent) and meeting the minimum physical requirements (AnyConstraint) are shortlisted for selection (see Fig. 5).

III. *Rule in Conference Enrollment:* Usually in technical conferences like the Self-Adaptive and Self-Organizing (SASO) Systems (AnyEvent) which is being held at Massachusetts Institute of Technology (MIT) in 2015, there are certain specific Conference Rules (AnyRule) that are needed to be followed by the Authors (AnyParty) to enroll and submit their work. These rules are often specified and approved by the Program Chair (AnyParty) and are also responsible for managing the conference. The authors can complete their Enrollment (AnyType) by completing an Electronic Form (AnyMedia), which also highlights the Submission Guidelines (AnyConstraint) to be followed (see Fig. 6).

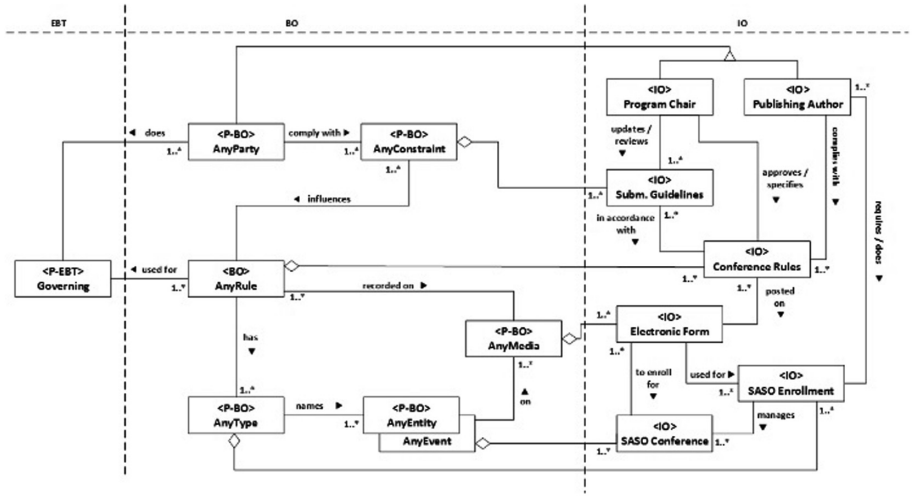

Fig. 6. Stable conference enrollment model

From the different applications of the AnyRule pattern discussed above, it is quite evident that the proposed pattern can be utilized to generate problem-specific solutions for different systems in a similar context. This capability gives the AnyRule Stable Design Pattern the required flexibility of unlimited reuse and adaptability to varying requirements [12–14].

3 Related Pattern and Measurability

3.1 Related Pattern

The pattern given below gives an abstract view of how the rules have been modeled traditionally. This pattern consists of a number of dependencies in the form of aggregation and inheritance. The model also includes tangible classes like the *client, GUI, Property List,* etc. (see Fig. 7).

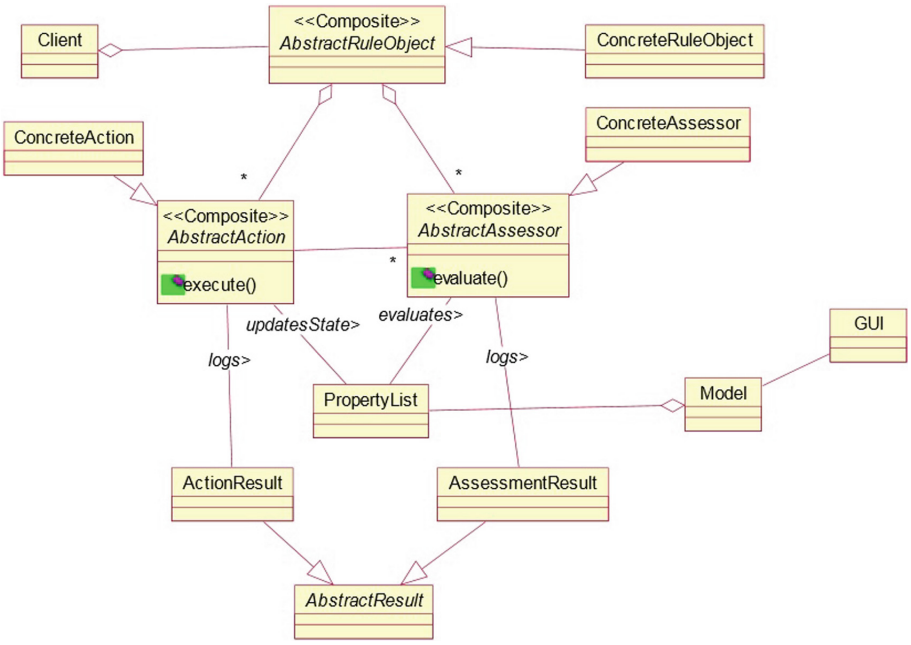

Fig. 7. Traditional compound rule object model [15].

3.2 Measurability

Table 1. Measurability study

Feature	TM	SSM
Number of tangible classes	4	0
Number of inheritances	5	0
Number of attributes per class	5–7	3
Number of operations per class	1–5	2
Number of applications	1	Unlimited

1. **Quantitative Measurability:** The total number of methods in any system can be calculated using the formula:

   T = C * M; where,
   T = total number of operations
   C = total number of classes
   M = number of methods per class

(a) **Traditional Rule Model**

C = 13
M = 3
T = 13 * 3 = 39

(b) **Stable Rule Model**

C = 8
M = 2
T = 8 * 2 = 16

Stable Rule model is more applicable and accurate compared to traditional model since the level of complexity is way less as evident by the above calculations. The stable model based on three level architecture acquires a more detailed understanding of the problem requirements with lesser complexity. The Traditional Rule Model includes tangible classes, which make it vulnerable in the event of any change.

2. **Qualitative Measurability**

Whereas a traditional rule model can only be used for a particular application, the stable rule model is usable in multiple applications. When requirements change, the traditional rule model becomes obsolete or requires massive changes. On the other hand, stable rule model can be used again and again by simply adjusting it to the application scenario. This is further emphasized by the following quality measure.

R = Tc – Tn; where,
R = Reusability Factor
Tc = Total Number of Classes
Tn = Number of classes not reused

(a) **Traditional Rule Model**

Total Number of Classes = 13
Number of classes not reused = 13
Reusability of classes = 13 – 13 = 0

(b) **Stable Rule Model**

Total Number of Classes = 8
Number of classes not reused = 0
Reusability of classes = 8 − 0 = 8

From the above calculations, it is clear that stability model has much more reusable classes than the traditional model, and hence it has wide applicability (Table 1).

4 Conclusion

This paper represents the core knowledge about rules and proposes the AnyRule Stable Design Pattern. The usefulness and varied applicability of the pattern can be illustrated by the fact that it is based on the core knowledge underlying any type of rules which makes it reusable in unlimited applications and as much as required. Another goal of our research has been to contribute in the development of a stable and comprehensive Stable Business Rule Engine (SBRE) where both stable classes and application classes are designed separately from each other and whose composition is formally supported to ensure correctness and exactness. This separation of concerns allows for reusability and enables the building of Stable Business Rules that are adaptable and extendable to an unlimited number of applications. This engine will be accessible online by any business, thereby removing huge costs of ownership and maintenance by only paying a minimal license fee. Using the system, any user can set up the rules as per his or her requirements through direct interaction with the portal simply by choosing required features or answering a few questions.

References

1. Griss, M.L.: Software reuse: from the library to the factory. IBM Syst. J. **32**(4), 1–23 (1993)
2. Griss, M.L., Wentzel, K.D.: Hybrid domain-specific kits for a flexible software factory. In: Proceedings: SAC 1994, Phoenix, Arizona, March 1994
3. Gaffney, J.E., Cruickshank, R.D.: A general economics model of software reuse. In: Proceedings: 14th ICSE, Melbourne Australia, May 1992
4. Mahdy, A., Fayad, M.E., Hamza, H., Tugnawat, P.: Stable and reusable model-based architectures. In: ECOOP 2002, Workshop on Model-Based Software Reuse, Malaga, Spain, June 2002
5. Hamza, H., Fayad, M.E.: Model-based software reuse using stable analysis patterns. In: ECOOP 2002, Workshop on Model-based Software Reuse, Malaga, Spain, June 2002
6. Fayad, M.E.: Stable Design Patterns for Software and Systems. Auerbach Publications, Boca Raton, FL (2015)
7. Fayad, M.E., Altman, A.: Introduction to software stability. Commun. ACM **44**(9) (2001)
8. Fayad, M.E.: Accomplishing software stability. Commun. ACM **45**(1) (2002a)
9. Fayad, M.E.: How to deal with software stability. Commun. ACM **45**(4) (2002b)
10. Ross, R.G.: Business Rule Concepts: Getting to the Point of Knowledge, 4th edn. Business Rule Solutions, LLC, April 2013
11. Jindal, S.: Stable Business Rule Standards. Masters Thesis, San Jose State University (2015)
12. Fayad, M.E., Sanchez, H.A., Hegde, S.G.K., Basia, A., Vakil, A.: Software Patterns, Knowledge Maps, and Domain Analysis. Auerbach Publications, Boca Raton (2014)
13. Fayad, M.E., Cline, M.: Aspects of software adaptability. Commun. ACM **39**(10), 58–59 (1996)
14. Muehlen, Z., Michael, M.I., Kittel, K.: Towards integrated modeling of business processes and business rules. In: ACIS 2008 Proceedings, vol. 108 (2008). [Arch]
15. Arsanjani, A.: Rule object 2001: a pattern language for adaptive and scalable business rule construction. In: PLoP 2001 Conference on Business Rule Construction. National EAD Center of Competency, p. 12. IBM, Raleigh (2001)