Performance Evaluation Engine

M. E. Fayad, Ph.D.

San Jose State University, Computer Engineering, College of Engineering, San José State University, One Washington Square, San José, CA 95192-0180
m.fayad@sjsu.edu

Abstract. This paper specifically deals with patterns topology or pattern language for Performance Evaluation, a cognitive process involved in producing and understanding different concepts of patterns in this discipline. It also explains how this language can be used to obtain the relationship between certain stable patterns, exhibiting a rich set of behaviors capable of handling features that are reusable, adaptable, and maintainable with minimal modification. The main goal of this paper is to overcome the issues with “forgettable” systems. The proposed solution is not intended to replace existing systems; rather, the effort is to define and provide a new methodology for designing such systems, with a rich set of patterns for performance evaluation that can be used efficiently.

1. INTRODUCTION

Developing software systems may involve lot of complexity, but maintaining a piece of software, enhancing it with new, additional requirements, and making it applicable to multiple domains can be of even greater complexity. Designing reusable, object-oriented software that remains stable even after addressing new requirements, is a real complicated issue. There are many applications that are designed, implemented, maintained, and used, with limited purpose and short lifespan. And used by very little number of people. These systems are usually termed “forgettable” or “disposable,” since they are very difficult and tough to maintain, and very expensive to reuse, modify, and extend functionality to in order to handle new requirements. Consequently, such systems are just thrown away to be replaced by entirely new systems. The goal of this paper, “Pattern Language for Performance Evaluation,” is to overcome issues with “forgettable” or “disposable” systems. The solution proposed to deal with these issues never intents to replace existing systems; rather, the effort is to define and provide a new methodology for designing such systems, with a rich set of patterns for performance evaluation for an efficient use.

2. Complexity and Patterns

To overcome the issues related to redesigning an entire system, to handle new functionalities, it becomes very important to develop a core system that usually exhibits a rich set of behaviors capable of handling features, which are reusable, adaptable with few modifications and, ultimately, easy to maintain. These result in recurring patterns of classes that solve specific design problems and make systems more flexible and, finally, reusable.

According to Christopher Alexander, “Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to the problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice” [Alexander, Silverstone, Jacobson, Fiksdahl and Angel,
Set of such related patterns comprise of a patterns language, which is a system used for representing ideas and states related to patterns. In other words, it is certain cognitive processes involved in producing and understanding different concepts of patterns in various disciplines. Pattern language is usually characterized according to the aspects and nature of different stable patterns. The common aspects that all stable patterns share are identified and, as a result, each pattern category with its own aspects is obtained.

There are several methodologies available for developing patterns that systematically name, explain, and evaluate an important and recurring design in a software system. However, developing a good pattern with an appropriate methodology is sometimes very expensive, and may have a number of drawbacks. Moreover, it may not necessarily be applicable to multiple domains. “Stable Patterns,” that handles the methodologies for developing both analysis and design patterns, is used for developing patterns. The concept of Software Stability Modeling is used in these cases [Fayad, 2002]. The patterns thus developed will be very simple enough to learn and still provide enough number of features that can be used quickly to hook multiple applications with few changes. This also ensures that the pattern is flexible enough to be applied to many domains, and remains stable even after addressing new requirements and design changes. Let us next examine why software stability patterns are an essential modeling technique.

3. OVERVIEW OF SOFTWARE STABILITY AND PATTERNS

Software Stability modeling is a modeling technique that uses the concepts of Enduring Business Themes” (EBT’s), “Business Objects” (BO’s), and “Industrial Objects” (IO’s). These are classes that are usually differentiated into three different layers (Figure 1). They are:

Layer 1: Enduring Business Themes (EBT’s): these represent the core knowledge of the system.

Layer 2: Business Objects (BO’s): these form the concrete classes of the system and are generally referred to as the “workhorses” of the system, under consideration.

Layer 3: Industrial Objects (IO’s): these represent those external applications that can be hooked to the pattern, which usually form the leaves and which can be replaced without affecting the system.

Since the pattern that is developed uses a multi-layered approach, it is absolutely important to understand these layers and how these layers are obtained. To obtain EBT’s of the system under consideration, aspects of the system that remain stable over a period of
time are further identified. These aspects form the core of the system, based on the needs of the system and why the system is modeled. This helps us in reducing the reengineering and code modifications necessary, when requirements change. BO’s, on the other hand, are those classes that act as “workhorses” and form concrete classes. They change only internally, but remain stable externally. Because of obtaining these classes, the majority of engineering performed on a modeled system should be to fit the project to those areas that remain stable. This yields a stable core design and thus a stable software product. Changes introduced to the software will then be in the periphery, since the core is based on something that remains stable. Since any changes that must be made to the software in the future will be in the periphery, only these small external modules will be engineered. Thus, the endless circle of reengineering an entire system is avoided for minor changes, and ultimately this reduces the changes that need to be made to a system, when needs and requirements change.

Stable analysis and design patterns are conceptual models obtained as a result of capturing the core aspects of the problem under consideration. Patterns may vary in the level of abstraction and granularity; capturing the core of the problem is mandatory in order to design the right solution. To reuse the same model to address a similar problem, regardless of the nature in which the problem appears, it is important to extract features that are adaptable to handling multiple domains.

4. ORGANIZATION OF STABLE PATTERN LANGUAGE

The main goal of pattern language is to provide an exhaustive material of related patterns in the field of performance evaluation. Pattern language mainly consists of four different phases. These mainly include the classification, or core, knowledge; properties, which provide a description of the benefits; development; and last but not least, deployment. We will consider the example performance evaluation in this paper, to understand these different phases or steps. Each of these phases is a set of patterns that uniquely identifies that phase. Figure 2 shows how these phases form the pattern language for performance evaluation.

As shown in the figure, performance evaluation forms the basis of the structure, for which relevant analysis and design patterns need to be obtained. The next layer gives all the EBT’s, which form the analysis patterns and support the most common tasks in performance evaluation. These are identified as Evaluation, Measurement, Collection, and Analysis. The next layer, which sits on top of this layer, is the architectural framework. This layer indicates that the layers above and below, which are the analysis and design patterns, collectively form the architectural framework for performance evaluation. The layer above this is the list of all BO’s, which form the design patterns and are primarily identified as Assessment, Metrics, Storage, and Optimization. These uniquely identify the properties for EBT’s obtained earlier.
The details of each layer are provided in the following chapters. Table 1 gives an overview of various analysis and design patterns for performance evaluation.

Table 1. Patterns for Performance Evaluation

<table>
<thead>
<tr>
<th>Patterns</th>
<th>Class Type</th>
<th>Pattern Type</th>
<th>Related Design patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation</td>
<td>EBT</td>
<td>Analysis</td>
<td>AnyPerformance, Assessment</td>
</tr>
<tr>
<td>Measurement</td>
<td>EBT</td>
<td>Analysis</td>
<td>AnyPerformance, Metrics</td>
</tr>
<tr>
<td>Collection</td>
<td>EBT</td>
<td>Analysis</td>
<td>Storage</td>
</tr>
<tr>
<td>Analysis</td>
<td>EBT</td>
<td>Analysis</td>
<td>Optimization</td>
</tr>
<tr>
<td>Assessment</td>
<td>BO</td>
<td>Design</td>
<td>AnyParty, AnyEntity</td>
</tr>
<tr>
<td>Metrics</td>
<td>BO</td>
<td>Design</td>
<td>AnyPerformance</td>
</tr>
<tr>
<td>Storage</td>
<td>BO</td>
<td>Design</td>
<td>AnyMedia, AnyEntity</td>
</tr>
<tr>
<td>Optimization</td>
<td>BO</td>
<td>Design</td>
<td>AnyEntity, AnyParty</td>
</tr>
</tbody>
</table>

As shown in the table, there are four analysis patterns and four different design patterns obtained for performance evaluation. The map (Figure 3) which is provided in the next section, gives a more detailed picture of various analysis and design patterns and phases in pattern language.

4. Stable Pattern Language or Knowledge Map of Performance Evaluation

The map provides the interrelationship between patterns, and their interaction and their properties, etc.
5. Classification and Goals

Classification, which also represents the core knowledge of the system, mainly contains patterns that deal with cooperating EBT’s that make up the pattern language. A number of analysis patterns can be obtained that are related to performance evaluation. A systematic approach to finding the right patterns for this is as follows:

- **Consider what the focus of the pattern language is:** This step helps us to obtain patterns that lead to the core knowledge and objective of the pattern language and, eventually, the system under consideration. For example, in pattern language for performance evaluation, the focus is to provide *evaluation* of performance or a task accomplished by some entity based on the *collection* of data obtained from a particular task. Provide the results of *evaluation* as a *measurement*, based on some *analysis* techniques that are used for following a procedure, optimizing existing processes, etc.

- **Obtain relevant patterns.** Based on the intent and objective of the system, obtain relevant patterns that will solve the problem under consideration.

- **Obtain pattern interrelation.** Once the patterns are obtained, it is important to study how they are related to each other. Studying this relationship will help us in obtaining the right group of patterns. For example, examine, if the patterns obtained for performance evaluation are really related to each other, as they should be.
- **Examine a redesign cause.** Consider and speculate on the reasons that might lead to redesigning of the system and the way the patterns relate to each other. This will help in obtaining a broader perspective of the problem and how the patterns that are already obtained can encompass these issues.

**Table 2. Patterns for classification of performance evaluation**

<table>
<thead>
<tr>
<th>Classification</th>
<th>Class Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation</td>
<td>EBT</td>
<td>Process of judging the competence with which a party has performed the task assigned to it, or the process of data collection and analysis, to determine the success or failure of performers for a specific task, as a result of performance.</td>
</tr>
<tr>
<td>Measurement</td>
<td>EBT</td>
<td>Represents the process of obtaining data for measuring any task or process performed by an entity.</td>
</tr>
<tr>
<td>Collection</td>
<td>EBT</td>
<td>Represents the collection of any data that can be obtained during the process of performance evaluation. It includes all relevant information that can be used for any analysis that needs to be undertaken, once a task is performed and evaluated.</td>
</tr>
<tr>
<td>Analysis</td>
<td>EBT</td>
<td>The pattern includes all the analysis techniques used for the performance evaluation process. This can include all the processes that perform analysis of the tasks that are evaluated, or analysis of the performance evaluation process itself. It is also the process of examination and study of a whole, in terms of the parts comprising it or the process of separation of the whole into its component parts. Sometimes, it is an examination of a complex, its elements, and their relations. It can also be a philosophical method of resolving complex expressions into those that are simpler or more basic, or a set of techniques for exploring underlying motives.</td>
</tr>
</tbody>
</table>

**6. Capabilities and Properties**

This phase represents all the properties exhibited by the patterns identified during classification. Below are certain steps for finding a set of capabilities, as per goal or a set of properties, for each pattern obtained during classification.

*Read each pattern to obtain an overview.* Each pattern for which the properties have to be obtained has to be understood thoroughly by paying particular attention to the sections—Consequences, Results, Tradeoffs, and Solution—in the pattern template, as provided in the appendix.

- **Study the CRC cards, Participants, and Pattern structure.** Emphasis is on factors like the classes in the pattern to understand, how they relate to each other with respect to EBT’s and their BO’s. This provides the possible patterns that might represent the properties of the pattern.

- **Consider applicability of the pattern.** Check to see, which domains the pattern can be applied to, in order to obtain the properties appropriate to encompass the features of the domain.

- **Identify the patterns in properties:** Once the properties are identified, examine the recurring classes in those properties to see, if it is a pattern. Choose the appropriate name for the pattern that will also be a property for the pattern under consideration.

Here are several different patterns identified during classification, along with their properties.
Table 3. Patterns for Capabilities/Properties of performance evaluation

<table>
<thead>
<tr>
<th>Classification</th>
<th>Capabilities/Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation</td>
<td>Assessment, Metrics</td>
<td>Assessment: Represents the process or property of observing and evaluating a party’s performance, recording the assessment, providing the status, findings, evidence, and result of any evaluation process. It also indicates how assessment is influenced by the constraints or external factors that are introduced in the system.</td>
</tr>
<tr>
<td>Measurement</td>
<td>Metrics</td>
<td>Indicates performance evaluation as a type of measurement, which provides the metrics or measurements of the task being evaluated.</td>
</tr>
<tr>
<td>Analysis</td>
<td>Optimization</td>
<td>Represents the attributes of all the optimization techniques that are used in any analysis method to enhance its utilization and the process of performance evaluation as a whole.</td>
</tr>
<tr>
<td>Collection</td>
<td>Storage</td>
<td>Represents the property of storage used by data collection for the purpose of performance evaluation.</td>
</tr>
</tbody>
</table>

7. Development

This phase deals with the process of improving the previous phases, to evolve or develop the patterns for classifications, in order to present a more productive and meaningful stage. This provides all the attributes of the patterns in terms of patterns that make them easier to use. In the case of performance evaluation, the development phase represents all the patterns that show the intrinsic behavior of the patterns obtained in the classification and properties phases. Thus, the patterns are applicable, testable, stable, reusable and portable to represent the inherent properties of the patterns obtained during classification. It also addresses the issues related to development of patterns. Providing hooks to relevant patterns based on the application does this. Several patterns for development are provided below.

- **Stable**: Indicates that all the related classes and attributes are stable.
- **Understandable**: Represents the attributes of all those classes that are understandable, in the sense, they are easily grasped.
- **Reusable**: Represents all those properties that make the patterns reusable in any domain.
- **Implementable**: Describes the behavior of any pattern, which indicates that it can be applied in a manner consistent with its purpose and design.
- **Testable**: Represents a pattern that defines the standardized procedure for measuring.

8. Deployment

This phase represents all issues related to deployment. There are certain patterns that are used for representing this phase, which are discussed in the following section.

- **Scalable**: This pattern represents scalability-related properties, with respect to the various patterns and the participants for the same.
Adaptable: This indicates all those properties related to the adaptability of the developed patterns.

9. Analysis and Design Pattern Examples

This section provides an example of an Analysis pattern “Evaluation,” and an example for a design pattern, “AnyPerformance.”

9.1. Evaluation Analysis Pattern

This section details the evaluation analysis pattern, along with its applicability in several domains.

**Pattern Name:** Evaluation Stable Analysis Pattern

The name itself indicates that evaluation can be applied to any domain, as a time-bound activity that attempts to assess, systematically and objectively, the relevance, performance and success of an ongoing or completed activity. Certain functionalities can be evaluated using this pattern. Those functionalities can include periodic checks for accuracy, adaptability, changeability, clarity, compliance, conformity, efficiency, effectiveness, improvability, feasibility, interoperability, maturity, operability, readability, recoverability, security, stability, suitability or testability.

**Known As:**

This pattern describes the process of determining the functionality and/or results of a task in its most fundamental terms.

**Problem:**

This pattern must be flexible and fundamental enough, that it can be applied to any domain. For example, an “account” may be a monetary account, but it can also be an email account, or any other type of account that does not necessarily involve or solely encompass monetary transactions. The pattern must be able to handle any form an account may take and provide a reliable description of its evaluation. The pattern AnyAccount, as often used in banking and financial applications, is too limited and restricted in scope; such a pattern could not be applied to an advertising account and provide any useful information. The evaluation analysis pattern should be able to describe the evaluation process for any type of account regardless of what the account represents.

The evaluation analysis pattern covers many domains that differ in nature and functionality. Therefore, modeling a generic analysis pattern that can be applied to all these domains could be problematic. Moreover, the requirements of each domain vary based on the users of the system and the participants.

The entity that needs to be evaluated can originate from different domains and nature. For example, various generations of wireless networks need to be evaluated based on diverse features; lands need to be evaluated to verify their suitability for agriculture; etc. Hence, obtaining a model or pattern that covers all the functionalities that must be achieved is a complex and tedious task. The greatest challenge or task is in obtaining a model that handles these variations. However, there are certain aspects of evaluation that transcend all application domains, which form the participants of the pattern in the form of “Enduring Business Themes” (EBT’s) and “Business Objects” (BO’s) [1], [2]. For evaluation purposes, the solution section will provide a detailed identification and description of these patterns.
Evaluation is an important concept in any domain requiring a response or feedback for a time-bound exercise that assesses various factors such as relevance, performance, success or failure of an ongoing or completed activity, all in a systematic and objective manner. The entity involved in the activity can be anything from an artist to an employee, an engine, or a network, whose evaluation needs to be done by another party that can be an evaluator like a person or a system. For example, an engine can only be evaluated on an assembly line. The audience in a theatre can only evaluate an artist. Wireless networks or networks in general can be evaluated for comparison, based on certain constraints that influence the assessment provided. These constraints can be any factors that affect overall judgment or assessment, either directly or indirectly.

In general, Evaluation has several distinguishing and notable characteristics related to focus, methodology and function. Thus, Evaluation, as a pattern, assesses the effectiveness of an ongoing activity in achieving its objectives, relies on the standards used by the activity to distinguish any other activity’s effects or external constraints, and aims at improvement of the activity through a modification of current operations. This concept can be applied to multiple fields, as noted previously.

CHALLENGES & CONSTRAINTS

The most difficult part of forming the evaluation pattern is making it general enough to be useful and beneficial across multiple domains. It must also fully describe an evaluation process in yet its most fundamental terms. To model the pattern AnyAccount, would require defining what possible types of accounts AnyAccount actually represents, and determining what possible model could accommodate all of them. This in itself is a major challenge, as it would require extensive knowledge or familiarity with a large variety of applications, where an account may be utilized or defined in some manner. However, the evaluation analysis pattern should be applicable to any entity, not just a particular type, such as an account. Thus, the broader concept of AnyEntity needs to be determined for this pattern; it cannot be limited by definition of application as AnyAccount might be. The pattern needs to describe, in terms generic enough to accommodate any possible scenario, evaluation—when requested by any entity, using any criteria that might be applicable, performing any assessments that might be necessary, and accommodating any party that is required to participate. Ultimately, the pattern must be elemental in form, yet flexible enough, that it can be applicable to any domain, for anything that might require evaluation and analysis. In order to construct such a pattern, one must have a wide ranging knowledge in multiple applications or domains in order to conceptualize the possibilities, where the pattern might apply.

Many contexts and domains that are completely dissimilar in nature are covered by the evaluation analysis pattern. One can accomplish evaluation for any entity or more entities simultaneously, based on multiple constraints. Hence, the pattern must be able to handle multiple constraints and entities. The main challenge lies in dealing with different types of constraints and entities for different domains. Flexibility with respect to handling domain-specific features is another challenge for this model. Assessments can be different based on the context in which it occurs. The features of assessment are domain specific, and addressing them is a limitation faced by this model.
**Pattern Structure and Participants**

The stable pattern that is obtained is a proposed solution for handling the issues mentioned earlier. This concentrates on providing a generic pattern that can be applied to any domain, leaving out the domain-specific features.

**Pattern Structure**

The relationship between enduring business themes (EBT’s), which in this case is the evaluation, and business objects (BO’s) for the evaluation analysis pattern, are shown in the figure 4.

Participants

The participants in the pattern structure are described based on the classes and patterns occurring within the evaluation pattern.

Classes

- **Evaluation**: This is the core of the stable analysis mode and represents a time-bound activity that attempts to assess, systematically and objectively, the relevance, performance, and success of an ongoing or completed activity process.

Patterns

- **AnyParty**: This represents parties that are involved in the task of evaluating any activity performed by any entity or, in some cases, the entity itself, and provide assessments for the same. This party can be an evaluator of some already completed task or one that is being completed. The evaluator can be an employer, an intelligent process that evaluates any task, or a network agent evaluating the performance of network elements. It can also be the audience evaluating an artist performing in a theater, an engineer evaluating land for its suitability for construction, and so on.

![Figure 3: Evaluation Analysis Pattern](image)

- **AnyEntity**: This represents any entity that needs to be critically evaluated. It can be a wireless system, an engine part, or whatever, that needs to be evaluated. Commonly called an “evaluation requestor.”
- **AnyAssessment**: This provides the status, findings, evidence and result of any evaluation process. It also indicates how assessment is influenced by constraints or external factors introduced in the system.

- **AnyConstraints**: This provides details of issues related to the effect of external factors and constraints that affect the efficacy of an ongoing activity in achieving its objectives. It can also be the standards used by the activity, to distinguish any other activity’s effects or external constraints that need to be considered during evaluation.

**CRC Cards**

The CRC cards shown in the following tables provide details on collaboration and responsibilities of various participants in the pattern.

**Table 4: CRC Card for Evaluation**

<table>
<thead>
<tr>
<th>Responsibility</th>
<th>Collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation (Evaluation facilitator)</td>
<td></td>
</tr>
<tr>
<td>Represents a time-bound activity that attempts to assess, systematically and objectively, the relevance, performance, and success of an ongoing or completed activity process</td>
<td>Client</td>
</tr>
<tr>
<td></td>
<td>AnyParty, AnyEntity, AnyAssessment, AnyConstraint</td>
</tr>
</tbody>
</table>

**Table 5: CRC card for AnyEntity**

<table>
<thead>
<tr>
<th>Responsibility</th>
<th>Collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnyEntity (Evaluation Requestor)</td>
<td></td>
</tr>
<tr>
<td>Requests evaluation to be done based on the criteria and standard set</td>
<td>Client</td>
</tr>
<tr>
<td></td>
<td>Evaluation</td>
</tr>
</tbody>
</table>

**Table 6: CRC Card for AnyParty**

<table>
<thead>
<tr>
<th>Responsibility</th>
<th>Collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnyParty (Evaluator)</td>
<td></td>
</tr>
<tr>
<td>Setting the rules, along with identifying the constraints under which the evaluation of any entity is done</td>
<td>Client</td>
</tr>
<tr>
<td></td>
<td>Evaluation, AnyConstraints</td>
</tr>
</tbody>
</table>

**Table 7: CRC Card for AnyConstraints**

<table>
<thead>
<tr>
<th>Responsibility</th>
<th>Collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnyConstraints (Constraint Definer)</td>
<td></td>
</tr>
<tr>
<td>Describes the rules, standards and the external factors that need to be considered before any task can be performed</td>
<td>Client</td>
</tr>
<tr>
<td></td>
<td>Evaluation, AnyParty</td>
</tr>
</tbody>
</table>
Table 8: CRC Card for AnyAssessment

<table>
<thead>
<tr>
<th>Responsibility</th>
<th>Collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provides the status, findings, evidence, and result of any evaluation process</td>
<td>Client Services</td>
</tr>
<tr>
<td>Evaluation, AnyEntity, AnyConstraints</td>
<td>provideAssessment()</td>
</tr>
</tbody>
</table>

**Consequences**

The pattern has certain consequences that aid the pattern in supporting its objectives:

- *It reduces Complexity.* The evaluation pattern modularizes the various aspects involved in the evaluation process, such as assessment or constraints, and makes the pattern simple to understand.

- *It facilitates Adding New Industrial Objects.* The pattern enables adding new “Industrial Objects” (IO’s) based on the application domain. By enabling this capability, it provides the users of the pattern with a core model, which remains stable and encompasses all the important features of evaluation required in any domain.

- *It also Provides Flexibility and Scalability.* It is applicable to various domains, with sufficient flexibility and scalability in terms of constraints, entities, and parties.

- *It Presents a Uniform Interface.* The pattern provides a uniform interface for all applications that use evaluation. The underlying interactions between the EBT’s and BO’s, which can be complex, are hidden, and instead a simpler, easier to understand interface, is presented.

- *It Improves Adaptability and Extensibility.* The pattern is adaptable and provides a high level of extensibility, to handle the addition of new and complex features

**Tradeoffs**

The Evaluation pattern has following tradeoffs:

- *Constraints Identification.* The identification of constraints that influence the assessment process in evaluation is actually very difficult to accomplish in certain domains. For example, when evaluating an artist in the theatre, certain assessments may be made based on the artist’s ethnic background or origin, which can be very difficult to capture in the actual code for constraints. The inability to consider such constraints is a drawback of this pattern.

- *Feature extraction.* Different features exist for different types of domains and contexts, e.g., features for industries will be different from features for wireless networks, and hence it is difficult to extract common features and make them applicable to this pattern in an exhaustive manner.

**Results**

The results for evaluation analysis pattern obtained are:
• **Generic Pattern.** It obtained a generic pattern for evaluation that is applicable across various domains.

• **Improves the Evaluation Process.** The pattern obtained also facilitates improving the existing evaluation process. Providing modular layers in form of EBT’s (evaluation) and BO’s that form the core of any system using evaluation do this. Various aspects, such as assessment and constraints related to the evaluation process, are all handled by a single pattern.

• **Improves Reusability.** The pattern promotes a cleaner partitioning of EBT’s similar to evaluation, while remaining stable throughout, as well as BO’s that can change dynamically and internally, thus encouraging reuse. The Industrial Objects for various applications could be added or removed from/to the existing model, in a transparent manner without affecting the standard interface. Thus, any combination of industrial objects related to assessment, party, entity, and constraints can be used with the pattern with few modifications.

**APPLICABILITY WITH ILLUSTRATED EXAMPLES**

**Problem Description**

The pattern must be able to be used for evaluating the task of transmitting data in networks.

**Problem Class Diagram**

Depicted below is a class diagram for the scenario of a manufacturing industry. The stable parts of the system, e.g., the EBT’s and BO’s, are shown in different columns of the table.

![Evaluation class diagram with example](image)

**RELATED PATTERNS & MEASURABILITY**
There are several patterns that usually interact with the above described design pattern. They are usually classified as follows.

Related Analysis Patterns

- **Measurement**: Represents the process of obtaining data for measuring any task or process performed by an entity.

- **Collection**: Represents the collection of any data that can be obtained during the process of evaluation. It includes all relevant information that can be used for any analysis, which needs to be accomplished, once a task has been performed and evaluated.

- **Analysis**: The pattern includes all the analysis techniques used in the evaluation process. This can include all the processes that perform analysis of the tasks, which are being evaluated. It is also the process of examination and study of a whole in terms of the parts it is composed of, or the process of separating a whole into its constituent parts. At times, it is an examination of a complex, its elements, and their relationships. It can also be a philosophical method for resolving complex expressions into those that are simpler or more basic, or a set of techniques for exploring underlying motives.

- **Convenience**: Provides convenience to users of any task.

Related Design Patterns

- **AnyMedia**: Defines the media that is used for evaluation. For instance, one can use video, as the media to evaluate any task.

- **AnyWorkflow**: Represents a workflow that can be applied to any system for defining the proper flow of tasks

- **AnyDomain**: Represents the domain that uses the evaluation

**DESIGN/IMPLEMENTATION ISSUES**

Description of Hooks

This section provides a description of various hooks that can be attached to this pattern.

- **AnyParty**: The party that identifies an evaluator can be extended by attaching an IO source. For example, the party or an evaluator for a wireless network can be an operator that can change over a period of time and be replaced by an administrator. For a land evaluation process, the evaluator can be a different party in charge of carrying out the evaluation task. Thus, the party can be in any domain without requiring the entire code to be rewritten for a particular domain.

- **AnyAssessment**: This BO can have a hook attached to it, to encompass varying assessments that can be used for any domain. For example, in a network scenario, the assessments can change or be modified, while the pattern remains stable, but if the “Industrial Objects” (IO’s) change, minimal modifications are required to accommodate these changes.

- **AnyConstraints**: This can have hooks attached to handle the details of issues related to the effect of external factors and constraints. These can be considered
IO’s that can be replaced or modified without affecting the structure of the overall system. Since it can also represent standards that can be modified or updated based on new requirements of a domain, hooks can be provided to deal with such changes.

9.2 PERFORMANCE STABLE DESIGN PATTERN

This section provides a detailed description of the performance design pattern that is used in performance evaluation pattern language.

**PATTERN NAME: AnyPerformance**

The name indicates that the pattern **AnyPerformance** is the process of accomplishing a task in accordance with a set standard of accuracy and completeness. Since, it begins with “Any”, it indicates that any domain involving performance can utilize this pattern.

**Problem**

Since the pattern **AnyPerformance** spans and covers many contexts that are completely different in nature, modeling a generic concept that can be applied to all domains, is the immediate and urgent problem. This is because the requirements differ based on the domain or context.

The performance of a system or party can differ based on its requirements, objectives and measures as well. For example, objectives for performing a task for wireless networks will be different from that of a performer in a theatre or an employee at work. So, obtaining a generic model or pattern, which in this case that encompasses all the features of different domains can be a difficult task to accomplish. How a single model addresses these variations is the unique challenge faced by this pattern. This leads to an area that requires a good solution to the problem of how a model that handles performance for different applications be obtained. Nevertheless, there are certain aspects of performance that transcend all application domains, which form the participants of the pattern in the form of EBT’s and BO’s. The solution section of this paper will identify and describe the above-mentioned factors.

**Context**

Performance is an important and critical concept in any domain that needs its party to perform some task according to the desired standards. A reason why the performance task needs to be carried out may also be specified to the performer along with the criterion for the desired performance. In general, this term may be either a criterion objective or an enabling objective. For example, the objective for an actor to perform on stage may be to entertain people; for a network, the objective for performing a particular task may be specified by the domain, based on the type of services desired. There might be some actions or data that can be objectively observed, collected, and measured to determine, if a task performer has executed the task to the prescribed standard, or that can be used for further analysis and evaluation.

In areas such as telecommunication networks, performance is an important concept for carrying out the task of delivering data and voice in a wireless mode according to a set standard. There can be many other aspects, like specifying channel allocation strategy, protocols, architecture, or security features, that form the requirements for a particular performance. Performing a task will also be required in industries that are involved in
assembling or manufacturing. Therefore, evaluating performance across domains becomes easier using a stable pattern. This pattern can be reused with many applications, by using simple hooks that require minimal changes without the entire system being rewritten.

**CHALLENGES & CONSTRAINTS**

[MF: Constraints and Challenges – (I would like to use Constraints and Challenges instead of Forces): Illustrates the challenges and the constraints that the pattern needs to resolve.]

In particular, in this Section you attempt to say, this is not a trivial problem, and thus trivial solution may not work. Be clear and very brief. One major blunder in writing this Section is that you mix the problem statement with the forces themselves. After writing this Section, try to read again the problem statement and make sure that they are not the same! It always happens!

- **The length is 1/2 Page**
- Describe the constraints related to the pattern such as multiplicities, limits and range.
- Describe some of the challenge must be over come by the pattern.

The design pattern obtained for performance spans many contexts that are completely different in nature. Performance of any task can be done by one or more entities simultaneously that can be requested by multiple parties. Thus, the pattern needs to handle multiple parties and entities; how they are handled or managed is a unique challenge faced by this model. The pattern is also not flexible enough to address the requirements of domain-specific features. Criteria can be different based on the context in which it occurs. The features for criteria, defined by a party, are domain specific, and addressing these features is a perceived limitation faced by this model.

**PATTERN STRUCTURE AND PARTICIPANTS**

The solution that is proposed here, concentrates on obtaining a generic pattern for performance that can be used with any domain, leaving out the domain-specific features. This allows any application to be hooked to the pattern with few changes.

Figure 5 shows the diagram of the *AnyPerformance* pattern.
The pattern **AnyPerformance** consists of the following participants:

**Classes:**
- **AnyPerformance**: This is the core of the stable design mode and it represents the process of accomplishing a task, in accordance with a set standard of accuracy and completeness.

**Patterns:**
- **Measurement**: This represents the EBT that the business object performance is derived from. It also utilizes the data and observations obtained because of some task being performed, and checks if the task is performed to the set standard provided by the BO criteria.
- **AnyParty**: This represents parties that are involved in the task of performance, either directly or indirectly. They can be involved in the process of requesting a task be performed, or in the process of observation and providing measures. For example, the party can be an employer, who requests the performance of some task to be performed in order to make relevant observations and data collection, or it can also be an audience in a performing theater. In technical aspects, it can be an operator requesting a network to transmit data.
- **AnyEntity**: This represents any entity that needs to perform a task. It can be a wireless system, an engine part, etc., that needs to finish or perform an assigned task. Engines may perform the task of running some machinery in an industry, or a network, wireless or wired, may perform the task of transmitting data.
- **AnyCriteria**: This provides the details of all those issues that affect performance, directly or indirectly. It can be a standard that needs to be achieved, some conditions, and a reason for the performance that needs to be carried out as a task. Its also specifies a clear-cut criterion for the desired performance by the performer. In general, this term may be either a criterion or an enabling objective. It also represents all separate acts or things that are required to complete any party’s performance on the job. It also includes the act (behavior), the conditions under which the behavior is performed, and the standard of performance required by the incumbent.
- **AnyMetrics**: It describes the actions and data objectively observed, collected or measured to check, if a task performed confirms to the prescribed standard. In general, this represents all the assessment data that can be collected, or the observations that can be made after the completion of a performance. These metrics can be further used for analysis and evaluation.

**CRC Cards**

<table>
<thead>
<tr>
<th>Table 9: CRC Card for AnyPerformance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Responsibility</strong></td>
</tr>
<tr>
<td>Defines the process of accomplishing</td>
</tr>
</tbody>
</table>
a task in accordance with a set standard of accuracy and completeness | AnyParty, AnyEntity, AnyCriteria, Measurement, AnyMetrics | definePerformance(), analyzeEntityPerformance(), performanceResult(), calculatePerformance(), criteria(), metrics()  

**Table 10: CRC Card for AnyEntity**

<table>
<thead>
<tr>
<th>Responsibility</th>
<th>Collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performs a task to a standard set</td>
<td></td>
</tr>
<tr>
<td>AnyPerformance</td>
<td>performTask(), providePerformanceData()</td>
</tr>
</tbody>
</table>

**Table 11: CRC Card for AnyParty**

<table>
<thead>
<tr>
<th>Responsibility</th>
<th>Collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting the rules under which the performance of any entity is measured</td>
<td></td>
</tr>
<tr>
<td>AnyPerformance, AnyCriteria</td>
<td>evaluateEntityPerformance( ), defineCriteria( )</td>
</tr>
</tbody>
</table>

**Table 12: CRC Card for AnyCriteria**

<table>
<thead>
<tr>
<th>Responsibility</th>
<th>Collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describes the rules, standards and external factors that need to be considered, before any task can be performed</td>
<td></td>
</tr>
<tr>
<td>AnyPerformance, Measurement, AnyParty</td>
<td>defineCriteria(), modifyCriteria()</td>
</tr>
</tbody>
</table>

**Table 13: CRC Card for Measurement**

<table>
<thead>
<tr>
<th>Responsibility</th>
<th>Collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Utilizes data and observations obtained as a result of some task being performed to verify the task is performed to the set standard</td>
<td></td>
</tr>
<tr>
<td>AnyPerformance, AnyMetrics, AnyCriteria</td>
<td>specifyMeasurement(), compareMetricsResults()</td>
</tr>
</tbody>
</table>

**Table 14: CRC Card for AnyMetrics**

<table>
<thead>
<tr>
<th>Responsibility</th>
<th>Collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describes the actions and data that can be objectively observed, collected and measured to determine if a task that a performer has performed is to the prescribed standard</td>
<td></td>
</tr>
<tr>
<td>AnyPerformance, AnyEntity, Measurement</td>
<td>provideMetrics( )</td>
</tr>
</tbody>
</table>
CONSEQUENCES

The pattern supports its objectives in the following ways:

- The pattern achieves its objectives or goals by providing a more generalized process for performance evaluation, in various domains with sufficient flexibility.
- It is also applicable to various domains that use performance.
- Sufficient flexibility and scalability are provided by this pattern for performance.

The performance pattern has the following benefits:

- Adaptable for different types of Entities. The performance pattern has a high level of adaptability, making it possible to adapt this pattern for different kinds of entities. For example, from base standards like the IEEE 802.3 and IEEE 802.11, different protocols are developed. This performance pattern is used to evaluate all the protocols developed from both these standards.
- Easy to use: The performance pattern developed here is a general pattern to be adapted for different parties and for different types of entities. This provides a high level of extensibility to handle adding new and complex features to this model.

Tradeoffs

- This pattern also introduces a trade-off between generalization and flexibility. Achieving generalization across domains at the cost of less flexibility does this.
- Different features exist for different types of domains and contexts, e.g., the features for industries will be different from the features for wireless networks, and hence it is difficult to extract common features and make them applicable in this pattern, in an in-depth manner.

Results

- Obtained a generic pattern for performance evaluation applicable across various domains.
- Obtained a stable, reusable pattern for performance to which various applications can be hooked.
- Obtained a scalable pattern, in terms of entities, metrics, participants, and media.

APPLICABILITY WITH ILLUSTRATED EXAMPLES

Problem Description

The pattern can be used for performance of the task of transmission of data in networks.

Problem Class Diagram
The diagram below depicts the class diagram for the scenario of data transmission in networks. The stable parts of the system, i.e., the EBT’s and BO’s, are shown in different columns of the table.

![Diagram of class diagram for Performance](image)

**Figure 6: Stability model class diagram for Performance**

10. CONCLUSION

Pattern Language for Performance Evaluation provides relationships, properties, development, and deployment of all possible patterns applicable to multiple domains that use performance evaluation. Hence, it provides a perfect pool of stable patterns that can be uniquely chosen for a particular application to suit its requirements.

References


