Pattern Documentation – Detailed Template (preferred):

- **Name**: Presents the name of the presented pattern.

  - Provide short definition of the term (Name)
  - Compare the name of the patterns with other selective name, and conclude with the right selection of the name.
  - Why did you choose that specific name?
  - Justify the name (such as why use “Any…” as a prefix for BO only)

- **Known As**: List all the terms that are similar to the name of the pattern. Two possible sources that one can use to fill this Section are 1- Similar patterns that are proposed in the literature. 2- Other names that you may find relative to the developed pattern. In some cases, several names might make sense, so you can keep a list of few of these names under this Section.

  Discuss the following cases briefly:
  1. Names match the pattern name: Just list similar names and why?
  2. Names match with doubts: List them, describe, and indicate doubts, and why?
  3. Names do not match, but people think they match the pattern name: List them, describe, and show why they do not match.

- **Context**: Gives possible scenarios for the situations, in which the pattern may recur. It is important in this Section that you motivate the problem you solve in an attractive way. For example, if I were writing a pattern about Trust, I would flush the trust in the context of e-commerce. Keep this Section short, yet exciting (This Section somewhat serves as an Introduction in conventional paper).

  - Describe the boundaries
  - List basic scenario – context,
  - Show by good examples, where the pattern can be applied
  - For example “account”… would have ownership and handler context, can be applied to banking Internet Providers, private clubs, etc

- **Problem**: Presents the problem the pattern concentrating on. This is one of the hardest parts in the pattern writing. Do not try to write it quite well in the first iteration, and most probably, you will not be able to! The problem should focus on the core purpose of the pattern, and should be able to answer the question: In what situation, I may benefit from your pattern? Try and keep this Section as short as possible otherwise, reader may get confused.

  **Length**: 1/4 to 1/3 Pgs
  - Has to be about a specific problems and descriptions = actual requirements of the pattern (functional and non-functional requirements of the pattern described in the
• Challenges and Constraints: Illustrates the challenges and the constraints that the pattern needs to resolve. You may create two subsections: (1) Challenges and (2) Constraints. In particular, in this Section, you try to say, this is not a trivial problem, and that trivial solution may not work. Be clear and brief. One major mistake in writing this Section is that you mix the problem statement with the forces themselves. After writing this Section, try to read the problem statement again, and make sure that they are not the same! It always happens!

  • Describe some of the challenges that must be overcome by the pattern
  • Describe the constraints related to the pattern, such as multiplicities, limits and range.
  • Make sure to list the challenges and constraints as bullets.

• Solution:
  1. Pattern Structure and Participants: Gives the class diagram of the pattern (EBT or BO). It also introduces briefly each class and its role. (Associations, aggregations, dependencies, and specializations) should be included in the class diagram. Association classes, constraints, interfaces, tagged values, and notes must be included in the class diagram. Include the hooks (show each of the BOS’ connections to IOs) A full description of the class diagram should be included with the final submission.
  2. CRC- Cards: Summarizes the responsibility and collaboration of each participant (class). Each participant should have only one well-defined responsibility in its CRC-Card. Participants with more than one responsibility should be presented with more than one CRC- Card, when each CRC- Card will handle one of these responsibilities. Refer to Appendix B – CRC Card Layout.
  3. Behavior Model (whenever is possible): If the abstraction of the pattern prevents you from writing an appropriate behavior model, then you can flush the dynamics of the pattern later on within the Example Section.

Description
• Describe the constraints related to the pattern such as multiplicities, limits and range.
• Describe some of the challenge that must be overcome by the pattern
• Note: Not ALL IO & BO may have inheritance

Detail Models
• Describe the model, role story, such as scenarios, how they play together
Participants
  • Each name, and its short description, and how it behaves within the model, such as classes and patterns in the patterns

CRC Cards

• **Consequences:** How does the pattern (EBT or BO) support its objectives or goals? What is the trade-off, and results of using the pattern? It is also important to highlight the things that the pattern does not cover, and reason about why you choose to exclude them. Another point that I found useful in this Section is to highlight other components that may arise from using the proposed patterns. For example, in AnyAccount pattern, we can say that using this pattern for banking systems, will require the integration of entries and logs to keep track of the accounts. However, this does not mean that the pattern is incomplete, but this is the nature of patterns anyway, they need to be used with other components.

  • List and briefly describe the Good (the benefits) of this pattern
  • List and briefly describe the Bad (Side-effects) of the pattern with suggested solutions

• Describe briefly and Map 5 different applications using the pattern using the following table format:

<table>
<thead>
<tr>
<th>EBT</th>
<th>BOs</th>
<th>App-1 Name - IOs</th>
<th>App-2 Name - IOs</th>
<th>App-3 Name - IOs</th>
<th>App-4 Name - IOs</th>
<th>App-5 Name - IOs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• **Applicability with Illustrated Examples:** Provides clear and detailed two case studies for applying the pattern in different contexts. The following sub elements represent the required details in one case.
  1. *Case Studies.* Shows the scenario of two cases studies from different contexts
  2. *Class Diagram.* Presents the EBTs, BOs, and IOs
  3. *Use Case Template.* Gives detailed description for a complete Use Case. Include test cases for the EBT and all the BOs – Abstraction of actors, roles, and classes, classes’ type, such as EBT, BOs, IOs, attributes and operations. Refer to Appendix B – Use Case Template.
  4. *Behavior Diagram.* Map the above use case into a sequence diagram.

  • Show 2-3 Distinct Scenarios
  • Description of the problem statement of the particular problem
  • Describe the Model – Class Diagram
  • Use Case Description with test cases (don’t need to do use case diagrams)
  • Sequence Diagram/use case
• **Related Patterns & Measurability:** Shows other patterns that usually interact with the described pattern, and those who are included within the described pattern. Related patterns can be classified as related analysis or/and related design patterns. Related patterns usually share common forces and rationale. In addition, it is possible that you might give some insights of other patterns that can or need be used with the proposed patterns, for example, in the case of AnyAccount pattern; we might point out to the AnyEntry pattern as a complementary pattern. There are rooms for contrasting and comparing the existing patterns with the documented pattern. This section also provides a few metrics for measuring several things related to the pattern structure, such as complexity and size, Cyclomatic Complexity, Lack of Cohesion, Coupling between Object Classes, etc.

This section is divided to two parts:

1. **Related Pattern**
   Two approaches:
   • Search for an existing traditional pattern on the same topic. Compare with traditional existing pattern’s model with reference with ours
   • If existing patterns do not exist, select a single definition of the name of our pattern, develop a traditional model class diagram and describe it briefly.

2. **Measurability:**

   Measurability compares our pattern to other models on the Number of behaviors and Number of classes. Justification of why the numbers of behavior or classes are so high or low.

   You may compare and comments on other quality factors, such as reuse, extensibility, integration, scalability, applicability, etc.

   Two Approaches: Compare the traditional with stability models in two of the following approaches:

   (a) **Quantitative Measurability, such as**

   + Number of behaviors or operations per class
   + Number of attributes per class
   + Number of associations
   + Number of inheritance
   + Number of aggregations
   + Number of interactions per class
   + Number of EBTs vs. Number of Requirements Classes in TM
   + Number of classes
   + Documentation – Number of pages
   + Number of IOs
   + Number of Applications
Estimation Metrics
Measurement Metrics
Etc

(b) Qualitative Measurability

- Scalability
- Maintainability
- Documentation
- Expressiveness
- Adaptability
- Configurability
- Reuse
- Extensibility
- Arrangement & Re-arrangement
- Etc

- **Modeling Issues, Criteria, and Constraints:** There are a number of modeling issues, criteria, and constraints that you need to address, in such a way as to explain them, and make sure that the model satisfies all the modeling criteria and constraints.

**Modeling Issues are:**

- **Abstraction:** describe the abstraction process of this pattern, list, and discuss briefly the abstractions within this pattern.
  + Show the abstractions that are required for the patterns (EBT, BOs, and IOs)
  + Elaborate on the abstraction of why EBTs and BOs are selected?
  + Show examples of unselected EBTs, and Why?
  + Show examples of unselected BOs, and Why?

- **Static models:** Illustrate and describe one or two of the static models of this pattern, and list and discuss briefly the complete story of the pattern’s model using actual objects.
  + Determine the sample model that you are planning to use: CRC Cards, Class Diagram, Component Diagram, etc. – Show the model.
  + Tell a complete story of the pattern’s model using objects
  + Repeat a complete story with other objects

- **Dynamic Models:** Illustrate and describe one or two the dynamic models of this pattern, and list and discuss briefly the behavior of the pattern through the selected dynamic models
  + Determine the sample model that you are planning to use: Interaction Diagram or State Transition Diagram. – Show the model.

- **Modeling Essentials:** Examine the pattern using the modeling essentials, and list
and discuss briefly the outcome of this examination
  + List or reference to the model essentials, and use them as criteria to examine
    the pattern
  + Elaborate on how to examine the model of the pattern by using the model
    essential criteria
  + Briefly describe the outcome.

• **Concurrent Development:** Show the role of the concurrent development of
developing this pattern. Describe.
  + Describe and show with illustration of the concurrent development of this
    pattern

• **Modeling heuristics:** Examine the pattern by using the modeling heuristics and
list and discuss briefly the outcome of this examination,
  + List or reference to the modeling heuristic, and use them as criteria to
    examine the pattern
  + Elaborate on how to examine the model of the pattern using the modeling
    heuristics
  + Briefly describe the outcome.
  + Modeling heuristics: such as
    + No dangling
    + No star
    + No tree
    + No sequence
    + General enough to be reused in different applications
    + Etc.

• **Design & Implementation Issues:** For each EBT, discuss and elaborate on the important
issues required for linking the analysis phase to the design phase and for each BO,
discuss the important issues required for linking the design phase to the implementation
phase, for example hooks. Describe the Design Issues (EBT), for example hooking
issues. Alternatively, discuss the Implementation Issues (BO), for example, why using
relationship rather than inheritance, hooking, hot spots problems. Show segments of code
here.

**Here is a list of Analysis Issues:**

- Divide and conquer
- Understanding
- Simplicity
- One unique base that suitable to many applications
- Goals
- Fitting with business modeling
- Requirements Specifications Models
- Packaging
- Components
- Type (TOP) (A)
- Actors/Roles
- Responsibility and Collaborations
- Generic and Reusable models
- Etc,

**Design Issues (EBT)**
- For example hooking issues
- Implementation Issues (BO)
- For example, why using aggregation or delegation rather than inheritance
- For example, hooking, hot spots problems
- Can show code here

**Here is a sample list of design and implementation issues:**
- Framework models (D)
- Classes (TOP) (D)
- Collaborations (D)
- Refinement (D)
- Generic and Reusable Designs (D)
- Precision (I)
- Hooks (I)
- Pluggable Parts (I)
- Navigation (I)
- Object Identity (I)
- Object State (I)
- Associations/Aggregations (I)
- Collections (I)
- Static Invariants (I)
- Boolean Operators (I)
- Collection Operators (I)
- Dictionary (D) (I)
- Behavior models (D) (I)
- Pre-Post-conditions specify actions (I)
- Joint Actions (Use Cases) (D)
- Localized Actions (I)
- Action Parameters (I)
- Actions and Effects (I)
- Concurrent Actions (I)
- Collaborations (I)
- Interaction Diagrams (D)
- Sequence Diagrams with Actions (D) (I)
- Pattern 1: Continuity
- Pattern 2: Performance
- Pattern 3: Reuse
- Pattern 4: Flexibility
• Pattern 5: Orthogonal Abstractions
• Pattern 6: Refinement
• Pattern 7: Deliverables
• Pattern 8: Recursive Refinement
• Package (D) (I)

Here is a list of Java Patterns:

• Fundamental Design Patterns
  1. Delegation (When not to use Inheritance)
  2. Proxy

• Creational Patterns
  1. Abstract Factory
  2. Builder
  3. Factory Method
  4. Object Pool
  5. Prototype
  6. Singleton

• Partitioning Patterns
  1. Composite
  2. Filter
  3. Layered Initialization

• Structural Patterns
  1. Adaptor
  2. Bridge
  3. Cache Management
  4. Decorator
  5. Dynamic Linkage
  6. Façade
  7. Flyweight
  8. Iterator
  9. Virtual Proxy

• Behavioral Patterns
  1. Chain of Responsibility
  2. Command
  3. Little Language / Interpreter
  4. Mediator
  5. Null Object
  6. Observer
  7. Snapshot
  8. State
  9. Strategy
  10. Template Method
  11. Visitor
• **Testability:** Describes the test cases, test scenarios, testing patterns, etc. (this is a very important point, but sometimes it is very hard to write for an isolated pattern, I am not sure what is the best way to write this part!) You can use three ways to document testability: 1. Test procedures and test cases within classes’ members of the patterns, 2. Propose testing patterns that are useful for this pattern and other existing patterns, and 3. Check, if the pattern fit with as many scenarios as possible, without changing the core design.

- Mention to people to try to find scenarios within the context that cannot work with this pattern.
- Show how you can test the requirements and the design artifacts within use cases
- Can also use exhaustive testing of behaviors (may require more pages) by Using testing patterns

• **Formalization Using Z++, Object Z, or Object-Constraints Language (OCL) (Optional):** Describes the pattern structure by using the formal language (Z++ or Object Z), BNF, EBNF, and/or XML.

• **Business Issues:** Cover one or more of the following issues.
  - **Business Rules:** Describe and document the business rules, and how you can extend them in the context & scenarios that are listed.
    - Define the business rules, business policies, business facts, in relation to the pattern
    - Illustrate the business rules that derived from the pattern.
  
  Check The following links:
  - [http://www.businessrulesgroup.org/bra.shtml](http://www.businessrulesgroup.org/bra.shtml)
  - [http://www.businessrulesgroup.org/first_paper/br01c0.htm -- pdf format file.](http://www.businessrulesgroup.org/first_paper/br01c0.htm)
  - [http://www.businessrulesgroup.org/brmanifesto.htm](http://www.businessrulesgroup.org/brmanifesto.htm)

  Define the business rules in relation to the pattern
  Illustrate the business rules that derived from the pattern.

  - **Business Models:** Issues:
    - Business Model Design and Innovation
      - Subscription business model
      - Razor and blades business model (bait and hook)
      - Pyramid scheme business model
      - Multi-level marketing business model
      - Network effects business model
      - Monopolistic business model
      - Cutting out the middleman model
Describe the same for the following business issues:

- **Business Standards**
  - Vertical Standards vs. Horizontal Standards
- **Business Integration**
  - Data Integration
  - People Integration
  - Tools Integration
- **Business Processes or Workflow**: Here are some of the business processes issues:
  - Business process management (BPM) is a systematic approach to improving those processes.
  - Business Process Modeling and Design,
  - Business Process Improvement,
  - Continuous Business Process Improvement,
  - Business Process ROI,
  - Business Process Rules,
  - Business Process Mapping,
- **e-Business**
  - e-Commerce
    - E-shops
    - E-commerce
    - E-procurement
    - E-malls
    - E-auctions
    - Virtual Communities
- Collaboration Platforms
- Third-party Marketplaces
- Value-chain Integrators
- Value-chain Service Providers
- Information Brokerage
- telecommunication
    - business-to-business (B2B)
    - business-to-consumer (B2C)
    - business-to-employee (B2E)
    - business-to-government (B2G)
    - government-to-business (G2B)
    - government-to-government (G2G)
    - government-to-citizen (G2C)
    - consumer-to-consumer (C2C)
    - consumer-to-business (C2B)
  - Web Applications
  - Business Patterns
    - Business Modeling with UML
    - Business Knowledge Map
  - Business Strategies
    - Business Strategy Modeling
    - Business Strategy Frameworks
    - Strategic Management
    - Strategic Analysis
    - Strategy Implementation
    - Strategy Global Business
  - Business Performance Management (BPM)
    - Methodologies
    - BPM Framework
    - BPM Knowledge Map
    - Assessment and Indication
  - Business Transformation
  - Enduring Business Themes
  - Security & Privacy

- Known usage: Give examples of the use of the pattern within existing systems or examples of known applications that may benefit from the proposed pattern. Mention some projects that used it.

- Tips and Heuristics: List and briefly describe all the lessons learned, tips, and heuristics from the utilization of this pattern, if any.

  - What did you discover?
  - Why did you included or excluded different classes?
| Are there any tips on usage such as scaling, adaptability, flexibility? |

| References |