Machine Learning Software Engineering Patterns: Classification and Practitioners’ Insights

Hironori Washizaki
Professor at Waseda University, Tokyo, Japan

washizaki@waseda.jp
Prof. Dr. Hironori Washizaki

- Professor and the Associate Dean of the Research Promotion Division at Waseda University in Tokyo
- Visiting Professor at the National Institute of Informatics
- Outside Directors of SYSTEM INFORMATION and eXmotion
- Research and education projects
  - Leading a large-scale grant at MEXT enPiT-Pro Smart SE
  - **Leading framework team of JST MIRAI eAI project**
- Professional contributions
  - IEEE Computer Society Vice President for Professional and Educational Activities
  - Editorial Board Member of MDPI Education Sciences
  - Steering Committee Member of the IEEE Conference on Software Engineering Education and Training (CSEE&T)
  - Associate Editor of IEEE Transactions on Emerging Topics in Computing
  - Advisory Committee Member of the IEEE-CS COMPSAC
  - Steering Committee Member of Asia-Pacific Software Engineering Conference (APSEC)
  - Convener of ISO/IEC/JTC1 SC7/WG20
Machine Learning Software Engineering Patterns: Classification and Practitioners’ Insights

Hironori Washizaki
Professor at Waseda University, Tokyo, Japan

washizaki@waseda.jp  http://www.washi.cs.waseda.ac.jp/


Supported by JST MIRAI eAI JPMJMI20B8  https://www.jst.go.jp/mirai/jp/program/super-smart/JPMJMI20B8.html
Agenda

• ML software engineering and patterns
• Literature review of software engineering patterns for ML applications
• Classification of ML patterns
• Practitioners’ insights on ML patterns
Street Cafe

**Problem:** Needs to have a place where people can sit lazily, legitimately, be on view, and watch the world go by…

**Solution:** Encourage local cafes to spring up in each neighborhood. Make them intimate places, with several rooms, open to a busy path …

Towards a pattern language

... OK, so, to attract many people to our city, **Small Public Square**s should be located in the center. At the **Small Public Square**, make **Street Cafes** be **Opening to the Street** ...
**ML-SE: Induction (and abduction)**

Conventional software: Deduction

![Diagram](image)

ML software engineering: Induction (and abduction)

![Diagram](image)

H. Maruyama, “Machine Learning Engineering and Reuse of AI Work Products,” The First International Workshop on Sharing and Reuse of AI Work Products, 2017

ML software engineering needs patterns!

- **Bridge** between abstract paradigms and concrete cases/tools
  - Documenting Know-Why, Know-What and Know-How
  - Reusing solutions and problems
  - Getting consistent architecture

- **Common language** among stakeholders
  - Software engineers, data scientist, domain experts, network engineers, ...
Practices and patterns in ML-SE

• Researchers and practitioners studying best practices strive to design Machine Learning (ML) systems and software.
• Some practices are formalized as patterns.

(Note: NOT handle ML model patterns.)

Data Lake for ML

Different Workloads in Different Computing Environments (e.g., Facebook)

K. M. Hazelwood, et al., Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective, HPCA 2018
Problem and goal

• ML system architecture and design patterns at different abstraction levels are not well classified and studied.
• Thus, we conducted a survey of software developers and an Systematic Literature Review.
Agenda

• ML software engineering and patterns
• Literature review of software engineering patterns for ML applications
• Classification of ML patterns
• Practitioners’ insights on ML patterns
Research questions

• RQ1. Does academic and gray literature address the design of ML systems and software?
  – 19 scholarly and 19 gray documents identified
  – 15 SE patterns for ML applications extracted

• RQ2. Can ML patterns be classified?
  – Categories of scopes: Topology, programming and model

• RQ3. How do practitioners perceive ML patterns?
  – Questionnaire-based survey for 600+ developers
  – Developers were unfamiliar with most ML patterns, although there were several major patterns used by 20%
RQ1. Does academic and gray literature address the design of ML systems and software?

• Systematic Literature Review (SLR)
  – Scholar papers: Engineering Village
  – Gray documents: Google

• 19 scholarly papers and 19 gray documents identified
• 15 patterns extracted

Engineering Village

(((system) OR (software)) AND (machine learning) AND (implementation pattern) OR (pattern) OR (architecture pattern) OR (design pattern) OR (anti-pattern) OR (recipe) OR (workflow) OR (practice) OR (issue) OR (template))) WN ALL) + ((cpx OR ins OR kna) WN DB) AND (((ca} OR (ja} OR {ip} OR {ch}) WN DT)

Google

(system OR software) "Machine learning" (pattern OR "implementation pattern" OR "architecture pattern" OR "design pattern" OR anti-pattern OR recipe OR workflow OR practice OR issue OR template)

"machine implementation pattern" OR "architecture pattern" OR "design pattern" OR anti-pattern OR recipe OR workflow OR practice OR issue OR template
Numbers of Documents per Year

- ML application systems have recently become popular due to the promotion of artificial intelligence.
- Since 2008, academic and gray documents have discussed good (bad) practices of ML application systems design.

Agenda

• ML software engineering and patterns
• Literature review of software engineering patterns for ML applications
• Classification of ML patterns
• Practitioners’ insights on ML patterns
RQ2. Can ML patterns be classified?

• Model operation patterns that focus on ML models
• Programming patterns that define the design of a particular component
• Topology patterns that define the entire system architecture.

H. Washizaki, et al., Practitioners’ insights on machine-learning software engineering design patterns: a preliminary study, ICSME 2020
<table>
<thead>
<tr>
<th>Pattern</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Different Workloads in Different Computing Environments</strong></td>
<td>It is necessary to separate and quickly change the ML data workload ...</td>
<td>Physically isolate different workloads to separate machines...</td>
</tr>
<tr>
<td><strong>Distinguish Business Logic from ML Models</strong></td>
<td>The overall business logic should be isolated from the ML models ...</td>
<td>Separate the business logic and the inference engine, loosely coupling the business logic and ML-specific dataflows.</td>
</tr>
<tr>
<td><strong>ML Gateway Routing Architecture</strong></td>
<td>Difficult to set up and manage individual endpoints for each service...</td>
<td>Install a gateway before a set of applications ...</td>
</tr>
<tr>
<td><strong>Microservice Architecture for ML</strong></td>
<td>ML applications may be confined to some “known” ML frameworks ...</td>
<td>Provide well-defined services to use for ML frameworks....</td>
</tr>
<tr>
<td><strong>Lambda Architecture for ML</strong></td>
<td>Real-time data processing requires scalability, fault tolerance, predictability ...</td>
<td>The batch layer keeps producing views while the speed layer creates the relevant real-time views ...</td>
</tr>
<tr>
<td><strong>Kappa Architecture for ML</strong></td>
<td>It is necessary to deal with huge amount of data with less code resource ...</td>
<td>Support both real-time data processing and continuous reprocessing with a single stream processing engine ...</td>
</tr>
</tbody>
</table>

Distinguish Business Logic from ML Models

- **Problem:** Business logic should be isolated from ML models so that they can be changed without impacting rest of business logic.

- **Solution:** Separate the business logic and the inference engine, loosely coupling the business logic and ML-specific dataflows.

---

**Legend**

- **Architectural Layers**
- **Deployed as ML System**

**Data Flow**

- **Business Logic Data Flow**
- **ML Runtime Data Flow**
- **ML Development Data Flow**

---

H. Yokoyama, Machine Learning System Architectural Pattern for Improving Operational Stability, ICSA-C, 2019
H. Washizaki, et al., Software Engineering Patterns for Machine Learning Applications (SEP4MLA), AsianPLoP 2020
Usage of **Distinguish Business Logic from ML Models**

**Presentation Layer**
- User Interface (Chatbot UI)
  - Web App Front-end
  - Slack

**Logic Layer**
- Business Logic (Chatbot Logic)
  - Web App Back-end
  - Slack

**Data Layer**
- Database (Previous Q&A Store)
  - DB Server
  - (None)

**Architectural Elements**
- Data Collection (Dataset)
  - Datasets
  - Nagoya Univ. Conversation Corpus

- Data Processing (Text to Vector Transformer)
  - NN Model pre- and post-processing
  - TensorFlow

- Inference Engine (Language Model)
  - NN Model
  - TensorFlow

**Legend**
- Architectural Elements (Example Role as Chatbot)
  - What
  - How

- Business Logic Data Flow
- ML Runtime Data Flow
- ML Development Data Flow
# Programming patterns

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Lake for ML</td>
<td>We cannot foresee the kind of analyses that will be performed on the data ...</td>
<td>Store data, which range from structured to unstructured, as “raw” as possible into a data storage ...</td>
</tr>
<tr>
<td>Separation of Concerns and Modularization of ML Components</td>
<td>ML applications must accommodate regular and frequent changes to their ML components ...</td>
<td>Decouple at different levels of complexity from the simplest to the most complex ...</td>
</tr>
<tr>
<td>Encapsulate ML Models within Rule-based Safeguards</td>
<td>ML models are known to be unstable and vulnerable to adversarial attacks, drifts, ...</td>
<td>Encapsulate functionality in the containing system using deterministic and verifiable rules ...</td>
</tr>
<tr>
<td>Discard PoC Code</td>
<td>The code created for Proof of Concept (PoC) often includes code that sacrifices maintainability ...</td>
<td>Discard the code created for the PoC and rebuild maintainable code ...</td>
</tr>
</tbody>
</table>

Encapsulate ML Models within Rule-based Safeguards

- **Problem**: ML models are known to be unstable and vulnerable to adversarial attacks, noise, and data drift.
- **Solution**: Encapsulate functionality provided by ML models and deal with the inherent uncertainty in the containing system using deterministic and verifiable rules.
- **Know usage**: E.g. Apollos’s object detection [Peng20]

![Diagram showing the flow from Input to Output through Business Logic API, Rule-based Safeguard, Inference (Prediction), Rule, Encapsulated ML model]

# Model operation patterns

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameter-Server Abstraction</strong></td>
<td>For distributed learning, widely accepted abstractions are lacking ...</td>
<td>Distribute both data and workloads over worker nodes, while the server nodes maintain globally shared parameters ...</td>
</tr>
<tr>
<td><strong>Data Flows Up, Model Flows Down</strong></td>
<td>Standard ML approaches require centralizing the training data on one machine ...</td>
<td>Enable mobile devices to collaboratively learn while keeping all the training data on the device as federated learning ...</td>
</tr>
<tr>
<td><strong>Secure Aggregation</strong></td>
<td>The system needs to communicate and aggregate model updates in a secure and scalable way ...</td>
<td>Encrypt data from each device and calculate totals and averages without individual examination ...</td>
</tr>
<tr>
<td><strong>Deployable Canary Model</strong></td>
<td>A surrogate ML that approximates the behavior of best model must be built to provide explainability ...</td>
<td>Run the explainable inference pipeline in parallel to monitor prediction differences ...</td>
</tr>
<tr>
<td><strong>ML Versioning</strong></td>
<td>ML models and their different versions may change the behavior of the overall ML applications ...</td>
<td>Record the ML model, dataset, and code to ensure a reproducible training and inference processes ...</td>
</tr>
</tbody>
</table>

Deployable Canary Model

- **Problem:** A surrogate ML that approximates the behavior of the best ML model must be built to provide explainability.
- **Solution:** Run the explainable inference pipeline in parallel with the primary inference pipeline to monitor prediction differences.
- **Known usage:** Image-based anomaly detection at factory

S. Ghanta et al., Interpretability and reproducibility in production machine learning applications, ICMLA 2018
<table>
<thead>
<tr>
<th>Pattern</th>
<th>Performance</th>
<th>Compatibility</th>
<th>Reliability</th>
<th>Security</th>
<th>Maintainability</th>
<th>Portability</th>
<th>Robustness</th>
<th>Explainability</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Different Workloads in Different Computing Environments</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distinguish Business Logic from ML Models</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ML Gateway Routing Architecture</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microservice Architecture for ML</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lambda Architecture for ML</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kappa Architecture for ML</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Lake for ML</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Separation of Concerns and Modularization of ML Components</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Encapsulate ML Models within Rule-based Safeguards</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discard PoC Code</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Parameter-Server Abstraction</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Flows Up, Model Flows Down</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secure Aggregation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Deployable Canary Model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ML Versioning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Agenda

• ML software engineering and patterns
• Literature review of software engineering patterns for ML applications
• Classification of ML patterns
• Practitioners’ insights on ML patterns
RQ3. Practitioners’ insights on quality

- Surveyed 300+ developers, 46 answered in ML development

**What product quality attributes considered?**
- Maintainability, reliability, security, and usability

**What model and prediction quality attributes?**
- Robustness, accuracy, and explainability

- Maintainability, reliability, robustness and accuracy are well handled by ML patterns. **There are demands for having ML patterns addressing security, usability, and explainability, which are not handled well now.**

![Bar chart showing practitioner insights on quality attributes](chart.png)

H. Washizaki, et al., Practitioners’ insights on machine-learning software engineering design patterns: a preliminary study, ICSME 2020
Practitioners’ insights on ML design patterns

- Surveyed 600+ developers, 118 answered

- Have you ever referred to ML patterns?
  - Major:
    - ML Versioning
    - Microservice Architecture for ML
  - None:
    - Secure Aggregation
    - Data Flows Up (aka. Federated Learning)

---

**Survey Results**

- **ML Versioning**: 80% Used it, 20% Never used it
- **Microservice Architecture for ML**: 70% Used it, 30% Never used it
- **Discard PoC code**: 90% Consider using it, 10% Not consider
- **Data Lake for ML**: 60% Used it, 40% Never used it
- **Distinguish Business Logic from ML Models**: 80% Used it, 20% Never used it
- **Separation of Concerns and Modularization of ML**: 75% Used it, 25% Never used it
- **Lambda Architecture for ML**: 90% Used it, 10% Never used it
- **ML Gateway Routing Architecture**: 80% Used it, 20% Never used it
- **Encapsulate ML models within rule-base...**: 70% Used it, 30% Never used it
- **Different Workloads in Different Computing...**: 80% Used it, 20% Never used it
- **Parameter-Server Abstraction**: 70% Used it, 30% Never used it
- **Kappa Architecture for ML**: 85% Used it, 15% Never used it
- **Deployable Canary Model**: 90% Used it, 10% Never used it
- **Secure Aggregation**: 80% Used it, 20% Never used it
- **Data Flows Up, Model Flows Down**: 70% Used it, 30% Never used it
Practitioners’ insights on ML design patterns

- Have you ever referred to ML patterns?
  - Developers were unfamiliar with most ML patterns, although there were several major patterns used by 20+% of the respondents.
  - For all patterns, most respondents indicated that they would consider using them in future designs.
  - Promoting existing ML patterns will increase their utilization.

- How do you solve and share design challenges of ML application systems?
  - 37 (i.e., 31%) organized design patterns and past design results.
  - As respondents become more organized in their approach to design problems by reuse, the pattern usage ratio increased.
  - Development teams and organizations will reuse more ML patterns as they become more consistent in their reuse approach.

<table>
<thead>
<tr>
<th>Design solution and reuse practice</th>
<th>#Respondents</th>
<th>#Patterns used</th>
<th>Pattern usage ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lv3. Organizing, reusing patterns (and past results)</td>
<td>37</td>
<td>64</td>
<td>11.5%</td>
</tr>
<tr>
<td>Lv2. Reusing externally documented patterns</td>
<td>31</td>
<td>50</td>
<td>10.8%</td>
</tr>
<tr>
<td>Lv1. Resolving problems in an ad-hoc way</td>
<td>37</td>
<td>35</td>
<td>6.3%</td>
</tr>
<tr>
<td>Others</td>
<td>13</td>
<td>3</td>
<td>1.5%</td>
</tr>
</tbody>
</table>
Conclusion

- ML software engineering needs patterns!
  - As bride and common language

- Literature review of academic and gray literature
  - 19 scholarly and 19 gray documents identified.
  - 15 SE patterns for ML applications extracted.
  - Patterns at [https://eai-transfer.github.io/ml-design-pattern/en/](https://eai-transfer.github.io/ml-design-pattern/en/)
  - ML patterns can be classified by scopes and quality attributes

- Survey of practitioners’ insights
  - Developers were unfamiliar with most ML patterns, although there were several major patterns used by 20% (such as ML Versioning and Microservice Architecture for ML)
  - 31% organized design patterns and past design results.
  - As respondents become more organized in their approach to design problems by reuse, the pattern usage ratio increased.
Future research direction

• Identify ML patterns addressing specific quality attributes that are not handled well now
  – Security, usability, and explainability

• Investigate the impact of patterns on quality attributes of systems

• Analyze relationships among patterns including related ones towards a pattern language

• Integration into framework to handle from requirements to implementations and testing/debugging