International Journal of Core Engineering & Management
Volume-6, Issue-12, 2021 ISSN No: 2348-9510

BUILDING A UNIFIED ERROR HANDLING FRAMEWORK FOR DIVERSE SOFTWARE PRODUCTS

Preeti Tupsakhare
Engineer Sr - Information Technology, Anthem INC.
Buffalo Grove, USA.
pymuley@gmail.com

Saurav Bansal
Application Architect , IT - MasterBrand Cabinets.
saurav.bansal.kbl@gmail.com

Abstract

Error handling is a critical component of software development, particularly in complex systems involving APIs, integrations, ETL processes, EDI, Managed File Transfers (MFT), user interfaces, and payment gateways. This paper presents a comprehensive error handling framework designed to manage errors consistently across these diverse domains. The proposed framework aims to standardize error handling processes, enhance error detection and reporting, improve system reliability, and provide robust monitoring and logging capabilities. By integrating this unified error framework, organizations can achieve consistent and efficient error management across various software products.

Index Terms— Error Handling, Error Framework, API Inte- gration, ETL, EDI, MFT, User Interface, Payment Gateways, Error Detection, Error Reporting

I. INTRODUCTION

In today's interconnected digital landscape, managing errors across various software products is crucial. Errors, if not handled properly, can lead to system failures, data corruption, and a poor user experience. This paper introduces a robust error framework designed to handle errors consistently and effectively across different domains, ensuring system stability and reliability.

II. INTRODUCTION TO ERROR FRAMEWORK

An error framework is a comprehensive, structured approach designed to manage errors within a system effectively. It encompasses procedures for detecting, handling, logging, and reporting errors in a standardized manner. The framework ensures consistency and efficiency in error management across various components of the system, thereby enhancing the overall reliability and maintainability of the application [2].

By implementing an error framework, developers can systematically identify where and why errors occur, ensuring timely and appropriate responses to different types of issues. This includes pinpointing critical errors that need immediate attention and categorizing less severe issues that

can be handled through routine maintenance [1].
Key features of an error framework typically include:
1. Error Detection
2. Error Handling
3. Error Logging
4. Error Reporting
5. Consistency and Standardization
6. Enhanced Maintainability

In summary, an error framework is pivotal in creating robust and resilient applications. It provides developers with the tools necessary to ensure errors are managed consistently and efficiently, improving the overall user experience and reducing downtime caused by unforeseen issues.

III. MOST POPULAR ERROR FRAMEWORK DESIGN PATTERNS

3.1 Exception Handling: Exception handling involves mechanisms provided by programming languages to respond to runtime errors or exceptional conditions that occur during the execution of a program. This design pattern ensures that the application can gracefully handle unexpected scenarios, minimizing disruption and maintaining a controlled execution flow [3].

Implementation:

- Try-Catch Blocks: These blocks are used to encapsulate code that might throw an exception (try) and define how to handle these exceptions (catch).
- Finally Block: Used for cleanup operations that must execute whether an error occurred or not.
- Custom Exceptions: Creating user-defined exceptions specific to application needs for better context and granularity in handling errors.

Benefits: Enhances the robustness and reliability of applications by preventing crashes due to unhandled exceptions and providing meaningful error messages.

3.2 Retry Patterns: The retry pattern is a technique used to automatically retry operations that fail due to transient errors, which are typically temporary and may succeed if attempted again. This pattern is designed to make applications more resilient to temporary failures such as network issues or service unavailability.

Implementation:

- Fixed Retry Interval: Retrying the operation after a specific, constant interval.
- Exponential Backoff: Increasing the wait time between retries exponentially to reduce load and avoid overwhelming the system or service.
- Jitter: Adding randomness to the retry intervals to avoid a thundering herd problem where multiple retries happen at the same time.

Benefits: Helps in maintaining the continuity of operations by providing a simple mechanism to handle temporary disruptions, without requiring manual intervention.

International Journal of Core Engineering & Management
Volume-6, Issue-12, 2021 ISSN No: 2348-9510

3.3 Circuit Breaker: The circuit breaker pattern prevents a system from attempting operations that are likely to fail repeatedly, thereby conserving resources and maintaining system stability. This pattern protects the system from being overwhelmed by repeated failures, offering time for recovery and preventing cascading failures [4].

Implementation:

- Closed State: The circuit breaker allows operations to proceed normally.
- Open State: After a certain number of consecutive failures, the circuit breaker moves to an open state where it immediately rejects further attempts.
- Half-Open State: After a predefined period, the circuit breaker allows a few test requests to determine whether the issue is resolved. If they succeed, it transitions back to the closed state; if they fail, it returns to the open state.

Benefits: Protects critical systems from the ripple effects of persistent failures, thus enhancing overall system resilience.

3.4 Compensation Transactions: Compensation transactions are designed to undo or compensate for the effects of previously successful operations when a subsequent operation fails. This pattern ensures data integrity and consistency in distributed systems where a series of operations need to be rolled back if a failure occurs in the middle of a transaction sequence.

Implementation:

- Compensating Actions: Define undo operations for each step that might fail to ensure the system can revert to a previous state.
- Saga Pattern: A sequence of local transactions where each step has a corresponding compensating transaction to undo its effects if necessary.

Benefits: Provides a way to maintain system integrity and consistency, especially in complex transactional workflows involving multiple services [2].

3.5 Fallback: The fallback pattern provides an alternative action or result when the primary operation fails. This pattern ensures that the system can continue to function, albeit in a degraded mode, instead of failing outright.

Implementation:

- Default Values: Returning default values when the primary operation cannot retrieve the expected data.
- Alternative Service: Redirecting the request to an alternative service or a backup implementation that provides similar functionality.
- Cache Fallback: Serving data from a cache if the primary data source is unavailable.

Benefits: Enhances the resilience and user experience of the application by ensuring that failures lead to an acceptable alternative rather than a total failure.

Each of these patterns offers unique strategies to improve error handling and ensure system reliability. They are often used in combination, depending on the specific requirements and architecture of the system.

International Journal of Core Engineering & Management
Volume-6, Issue-12, 2021 ISSN No: 2348-9510

IV. KEY OBJECTIVES OF IMPLEMENTING A COMMON ERROR FRAMEWORK

- Standardization: Ensuring all API errors follow a consistent format with standardized HTTP status codes and error messages.
- Detection: Implementing real-time monitoring to detect file transfer failures immediately.
- Reporting: Generating detailed error reports for failed EDI transactions, including timestamps and error details.
- Logging: Capturing detailed logs of user interface errors to facilitate debugging and analysis.
- Recovery: Implementing retry mechanisms for transient API errors to improve reliability.
- User Notification: Displaying user-friendly error messages in the application interface to inform users of issues without exposing technical details.
- Compliance: Ensuring error logs comply with regulations such as GDPR by not storing sensitive information.

Basic Design Elements of Common Error Framework

- Consistent Error Format: A standardized format for errors ensures consistency across different components and systems. Example: Using a JSON format for API error responses with fields for error code, message, and details [3].
- Real-time Monitoring: Implementing monitoring tools to detect errors as they occur and alert relevant stakeholders. Example: Using application performance monitoring (APM) tools to monitor API endpoints and detect errors in real- time [3].
- Detailed Logging: Capturing comprehensive logs of errors to aid in troubleshooting and root cause analysis. Example: Logging detailed information about file transfer errors, including file names, timestamps, and error messages.
- User-friendly Error Messages: Providing clear and concise error messages to end-users to improve the user experience. Example: Displaying a message like "Unable to load data. Please try again later." instead of a technical error code.
- Retry Mechanisms: Implementing automatic retry mechanisms for transient errors to improve system reliability. Example: Automatically retrying failed EDI transactions, a specified number of times before escalating the error.
- Compliance: Ensuring the error framework complies with relevant industry standards and regulations. Example: Ensuring that error logs do not contain sensitive information in compliance with GDPR

IV. FRAMEWORK COMPONENTS

- Error Detection Module: Monitors system components to detect errors in real-time. for example., An API gateway that monitors incoming and outgoing API requests and detects errors based on HTTP status codes.
- Logging Module: Captures detailed logs of errors, including contextual information for troubleshooting. Example: A logging service that stores error logs in a centralized location for easy access and analysis.
- Notification Module: Sends alerts and notifications to relevant stakeholders when errors occur. Example: An email notification system that alerts the IT team when a critical error is

International Journal of Core Engineering & Management
Volume-6, Issue-12, 2021 ISSN No: 2348-9510

detected in the EDI system.

- Reporting Module: Generates detailed reports on error occurrences, trends, and resolutions. Example: A reporting tool that generates weekly error reports, highlighting the most common errors and their impact.
- Recovery Module: Implements mechanisms for automatic error recovery, such as retries and fallbacks. Example: A retry mechanism that attempts to resend failed API requests up to three times before logging the error.
- User Interface Module: Displays user-friendly error messages and provides options for error resolution. Example: A user interface component that displays a message and a "Retry" button when a file transfer fails [4].

Table I below details out the Framework Components

TABLE I. FRAMEWORK COMPONENT

Component	Function
Error Detection Module	Real-time error detection
Logging Module	Detailed error logging
Notification Module	Alerting stakeholders
Reporting Module	Generating error reports
Recovery Module	Automatic error recovery
User Interface Module	Displaying user-friendly error messages

V. COMMUNICATION PROTOCOLS USED FOR ERROR FRAMEWORK

- SMTP: For sending email notifications.
- HTTP/S: For transmitting error reports to monitoring systems [5].
- Syslog: For centralized logging of system errors [6].
- SNMP: For network error monitoring and alerts.

Table II details out the Pros and Cons of Communication Protocols

TABLE II. PROS AND CONS OF COMMUNICATION PROTOCOLS

Protocol	Pros	Cons
SMTP	Widely supported, simple to implement	Slow, not suitable for real-time alerts
HTTP/S	Secure, real-time capabilities, widely supported	Requires internet connectivity
Syslog	Centralized logging, enterprise support	Complex setup, requires infrastructure [6]
SNMP	Real-time network monitoring, widely supported	Primarily for network devices

VII. KPIS FOR MEASURING SUCCESS

- Error Detection Rate: The percentage of errors detected out of the total errors occurring.
- Mean Time to Detect (MTTD): Average time taken to detect an error after it occurs.
- Mean Time to Resolve (MTTR): Average time taken to resolve an error after detection.
- Error Recovery Rate: The percentage of errors success- fully recovered without manual intervention.

- User Impact: The number of users affected by errors and the severity of the impact.

VIII. IMPLEMENTATION STRATEGIES INTEGRATION

Integrating the error framework into an application involves configuring the detection, logging, notification, and recovery modules based on the application's needs. APIs and SDKs for various programming languages facilitate seam- less integration. Example: A Java-based web application can use the error framework's Java SDK to integrate error handling capabilities, allowing it to log errors to a centralized logging service.

Configuration: Developers configure the framework using a configuration file or environment variables. Key configuration parameters include error thresholds, notification settings, and recovery strategies. Example: A configuration file can specify error thresholds for API request failures, enable email notifications for critical errors, and set retry strategies for transient errors [2].

Invocation: The framework can be invoked at different points in the application, such as within error-handling routines, scheduled tasks, or user-initiated actions. When an error is detected, the framework handles logging, notification, and recovery processes. Example: In a data processing pipeline, the error framework can be invoked to handle errors during data validation, logging them, sending notifications, and retrying the process if possible [3]

IX. CASE STUDY

E-commerce Platform Error Handling

An e-commerce platform needs to handle errors in API integrations, file transfers, payment gateways, and the user interface effectively. By integrating the error framework, the platform can ensure consistent error management across all components.

Implementation

The platform configures the error framework to monitor API endpoints, log errors in file transfers, handle payment gateway errors, and display user-friendly error messages in the interface. The notification module sends alerts to the IT team for critical errors, while the recovery module implements retry mechanisms for transient errors.

X. CONCLUSION

The proposed unified error handling framework offers a comprehensive solution for managing errors across APIs, integrations, ETL, EDI, MFT, user interfaces, and payment gateways. By integrating this framework, organizations can achieve consistent, reliable, and efficient error handling, enhancing overall system stability and user satisfaction. Future work includes extending the framework's capabilities to support emerging technologies and platforms, ensuring its relevance in the ever-evolving digital landscape.

REFERENCES

1. de Sousa, D.B., Maia, P., Rocha, L.S. et al. Studying the evolution of exception handling anti-patterns in a long-lived large-scale project. J Braz Comput Soc 26, 1 (2020). https://doi.org/10.1186/s13173-019-0095-5
2. Google, 2021. [Online]. Available: https://cloud.google.com/logging/docs/
3. 2020. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/standard/exceptions/
4. Aquino, G., Queiroz, R., Merrett, G., Al-Hashimi, B. (2019). The Circuit Breaker Pattern Targeted to Future IoT Applications. In: Yangui, S., Bouassida Rodriguez, I., Drira, K., Tari, Z. (eds) Service-Oriented Computing. ICSOC 2019. Lecture Notes in Computer Science(), vol 11895. Springer, Cham. https://doi.org/10.1007/978-3-030-33702-5_30
5. H. Derhamy, J. Eliasson, J. Delsing, P. P. Pereira and P. Varga, "Translation error handling for multi-protocol SOA systems," 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), Luxembourg, Luxembourg, 2015, pp. 1-8, doi: 10.1109/ETFA.2015.7301473.
6. keywords: {Protocols;Quality of service;Delays;Computer architecture;Servers;Monitoring;Service-oriented architecture;Error handling;Protocol translation;Arrowhead;Internet of Things;Cyber-physical systems;SOA;Translation},
7. Hui, H., Bai, L. (2014). Research and Development of Centrlized Log Management System Based on Syslog Protocol. In: Zhong, S. (eds) Proceedings of the 2012 International Conference on Cybernetics and Informatics. Lecture Notes in Electrical Engineering, vol 163. Springer, New York, NY. https://doi.org/10.1007/978-1-4614-3872-4_249