

AI-based static analysis for clean code

Sudarshan Bhide – CTO, Acellere GmbH

Agenda

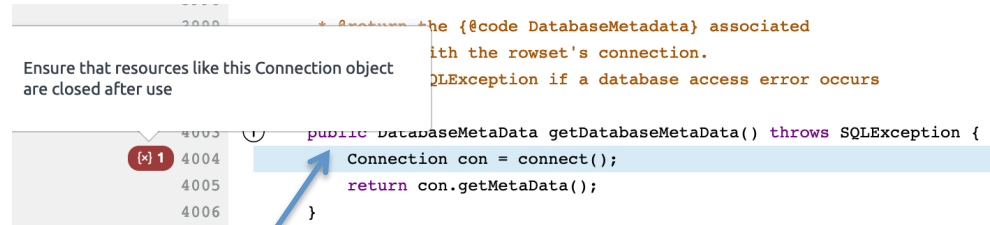
- Introduction
- Gamma's AI Features
- Technical Overview
- Examples
- Benefits

Conventional static analysis – What it can do

Rule-based

- Can find issues which are implemented up-front as rules (e.g. check for a memory leak, or a buffer over-run)
- Proven and powerful technique
- Used for finding robustness issues, compliance checks, etc.

Example: CloseResource rule



Can be detected by
conventional approach

Conventional static analysis – limitation

Memory leak found during testing

Memory leak fixed

Commit Id: 0a7462e Related issue: [KAFKA-6925](#) KAFKA-6925: fix parentSensors memory leak (#5108) **memory**

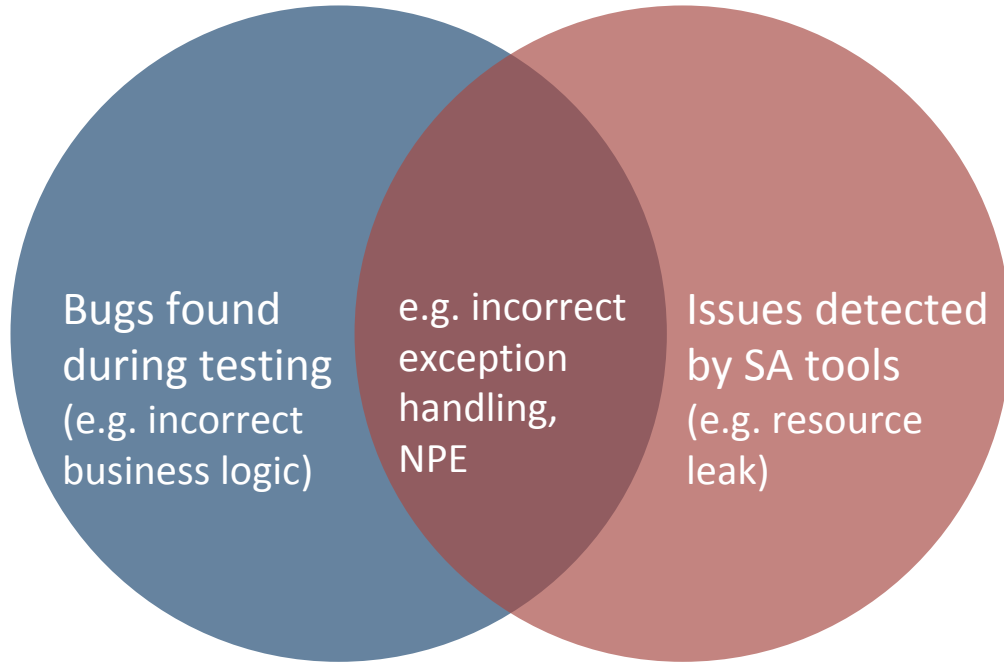
```
if (parent != null) {  
    metrics.removeSensor(parent.name());  
}
```

```
if (parent != null) {  
    metrics.removeSensor(parent.name());  
    parentSensors.remove(sensor);  
}
```

Cannot be detected by conventional approach, without writing a complex rule

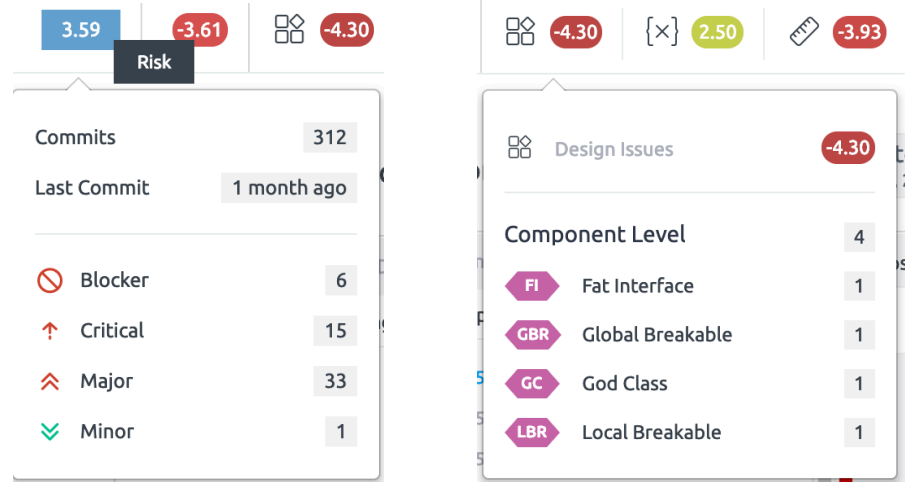
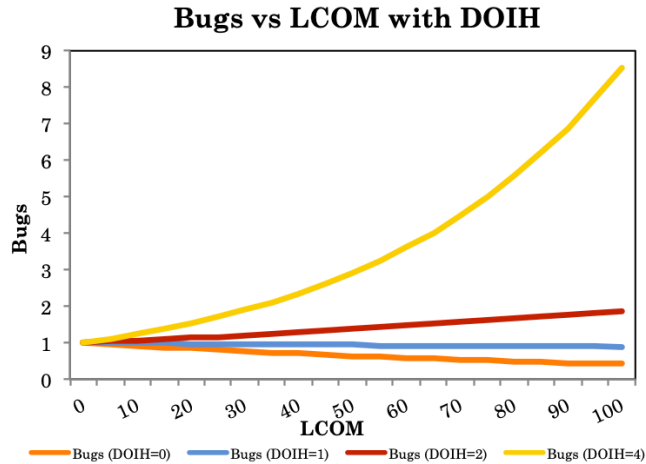
- Although rule-based analysis is a proven and powerful technique, it cannot find issues (functional or non-functional) beyond the checks implemented as rules
- Writing new rules for every such condition is not feasible / possible

Coverage of Issues by static analysis tools



- Several issues found during testing are not detectable by existing tools at development time
- Some issues detected by tools also manifest as functional issues, if not addressed (e.g. bad exception handling can lead to unwanted functional behavior)
- If more such issues are detected at development time, testing effort will reduce

Design has a strong correlation with bugs



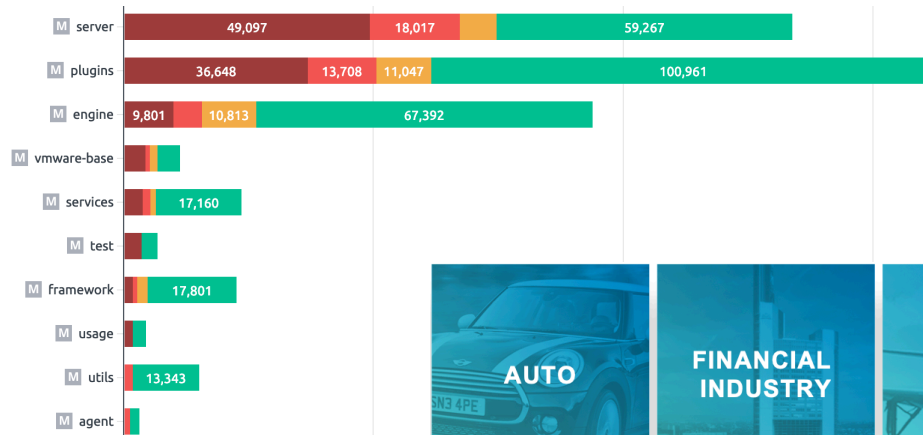
- Amount of bugs increase with lack of cohesion (LCOM) and deep inheritance trees (high DOIH)
- Issues such as lack of separation of concerns, lack of encapsulation contribute to bugs

Motivation





















- Find new issues based on actual bugs detected (in addition to rule-based analysis)
 - Issues in code are not only limited to what can be found through standard checks
 - Issues can also occur due to usage of incorrect APIs, incorrect decision points, logical flow errors, etc.
 - Since these checks cannot be implemented in advance, an alternate way to uncover such issues is needed
- Suggest fixes
 - When an issue is detected, a possible suggestion to fix the issue is equally important in order to address the issues quickly
- Analyze software against design anti-patterns for long-term maintainability

Introducing Gamma

- Gamma is a code quality improvement platform which analyzes code as well as design quality
- Gamma's rating engine enables identification of the most relevant code components



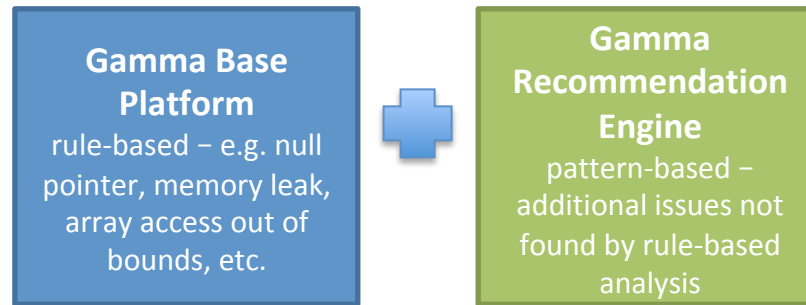
Multi-dimensional Analyzer

Components	Risk	Overall Rating ^	 Design	 Metrics	 Duplication	 Code Issues
 StorageService	5.00	-4.12	-5.00	-4.55	5.00	-2.50
 ModificationStatement	3.32	-3.71	-4.06	-3.52	5.00	-3.42
 SchemaKeyspace	3.16	-3.42	-4.60	-3.70	4.53	-1.56
 StorageProxy	4.10	-3.31	-4.60	-4.39	5.00	-0.50
 ColumnFamilyStore	4.84	-3.29	-4.60	-4.33	5.00	-0.50
 SSTableReader	3.18	-3.08	-2.68	-4.21	3.57	-2.50
 SecondaryIndexManager	3.05	-2.83	-3.96	-3.66	5.00	-0.50
 CQLTester	3.21	-2.82	-2.16	-4.02	4.00	-2.50
 StressProfile	3.05	-2.77	-4.26	-3.04	5.00	-0.50
 QueryProcessor	3.28	-2.60	-1.72	-2.33	5.00	-4.05
 SystemKeyspace	3.22	-2.50	-1.98	-3.82	5.00	-1.87
 Keyspace	3.16	-2.50	-3.88	-3.28	5.00	0.13
 CompactionManager	3.67	-2.45	-2.66	-4.11	5.00	-0.50
 MessagingService	3.67	-2.42	-1.56	-3.87	5.00	-2.12
 FBUtilities	3.44	-2.35	0.20	-3.40	5.00	-4.69
 Gossiper	3.51	-2.31	-2.42	-3.98	5.00	-0.50

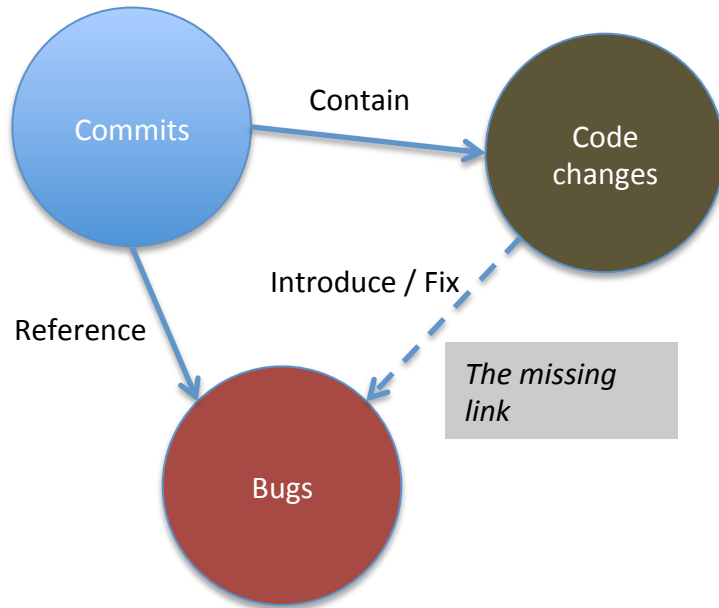
- Gamma scans code to uncover **implementation and design issues**
- It does **intelligent prioritization** to surface the most risky code components
- This enables development teams to focus on the right components when fixing issues

Gamma Recommendation Engine (RE)

- In addition to rule-based analysis, Gamma RE includes a discovery capability, to find new issues which are not explicitly implemented as rules
- Leverages machine learning and the information available in history (code commits and issue databases) to uncover problematic patterns from the past and their possible presence in new code
- Uses this information to suggest fixes to detected issues

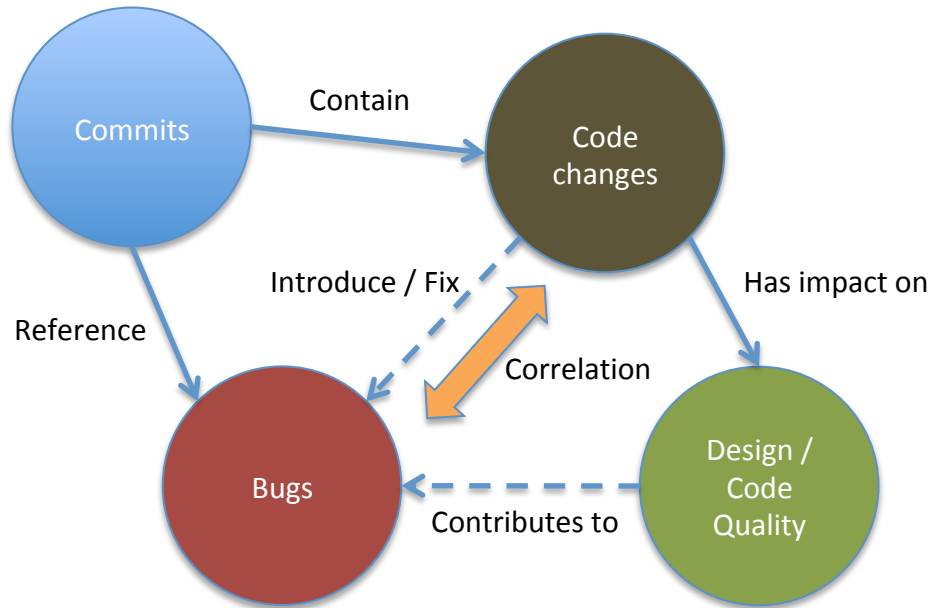


Relationships between software assets



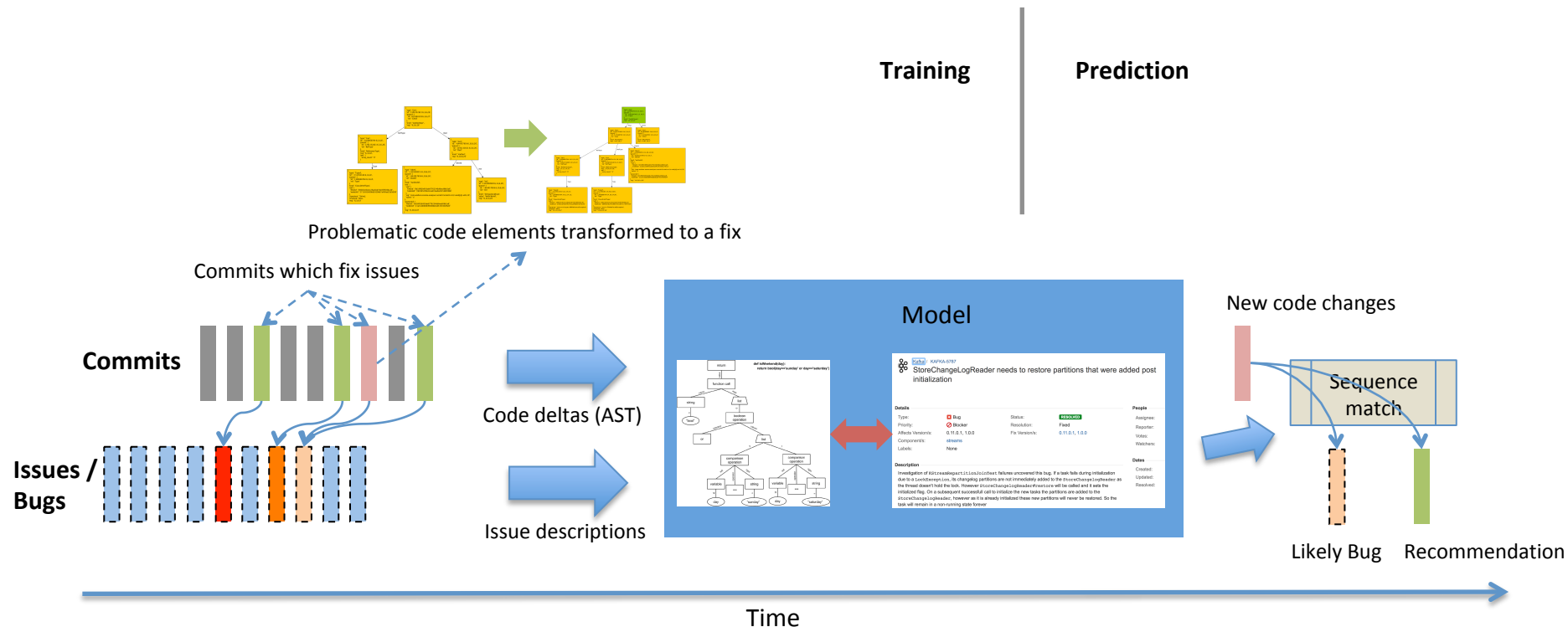
- Commits contain code changes made by developers
- Code changes are made to add features or **fix bugs (relevant here)**
- Code changes may also introduce new bugs
- Bugs are managed in issue management system
- It is not easy / possible to predict if a code change might introduce a new bug before it is committed
- The change may propagate upto QA or even field before it is detected

Gamma RE



- Gamma RE learns from historical data and matches new code with patterns previously learned
- In addition to Gamma's Design and Code Quality analysis, it builds a correlation between code changes and Bugs
- Gamma RE achieves this by using deep learning techniques
- A neural network is trained using an encoder/decoder architecture to associate code change patterns with bug descriptions
- This is used to predict if a new code change is likely to introduce a bug

Gamma RE processing - Schematic



Prediction - Detection and Reporting of Issues

Example: Missing conditional check

KAFKA-7192 Follow-up: update ch...
c8c3a7d
Guozhang Wang committed 1 month ago
3 changed +75 / -54 LOC
7 Likely Bugs Locations

90 - 92
144 - 145
151 - 152
159 - 160
193 - 196
214 - 215
219 - 226

Missing null check

```
146 // as timeout exception gets thrown we just give up this time and rely on the next run loop
147 log.debug("Could not fetch end offset for {}; will fall back to partition by partition fetching", initializable);
148 return;
149 }
150
151 final Iterator<TopicPartition> iter = initializable.iterator();
152 while (iter.hasNext()) {
153     final TopicPartition topicPartition = iter.next();
154     final Long endOffset = endOffsets.get(topicPartition);
155
156     // offset should not be null; but since the consumer API does not guarantee it
157     // we add this check just in case
158     if (endOffset != null) {
159         final StateRestorer restorer = stateRestorers.get(topicPartition);
160         if (restorer.checkpoint() >= endOffset) {
161             restorer.setRestoredOffset(restorer.checkpoint());
162             iter.remove();
163             completedRestorers.add(topicPartition);
164         } else if (restorer.offsetLimit() == 0 || endOffset == 0) {
165             restorer.setRestoredOffset(0);
166             iter.remove();
167             completedRestorers.add(topicPartition);
168         } else {
169             restorer.setEndingOffset(endOffset);
170         }
171     }
172 }
```

Likely issue (prediction)

Sequence match (model)

Proposed fix (recommendation)

New, uncommitted code change

Suggestions

StoreChangelogReader needs to restore part...
This change is likely to introduce an issue similar to KAFKA-5787 concurrency

```
final StateRestorer restorer = stateRestorers.get(topicPartition);
final StateRestorer restorer = needsInitializing.get(topicPartition);
// might be null as has been initialized in a previous invocation.
if (restorer != null) {
```

- Above picture shows how new code changes are matched against previous signatures for known issues and any likely issues are reported along with suggested fixes

Example 1 – HTTP Error Handling

- Project: Apache Cloudstack, commit: 6e09529, file: HttpsDirectTemplateDownloader.java

Call to HTTPS Client

```
86  @Override
87  public boolean downloadTemplate() {
88      CloseableHttpResponse response;
89      try {
90          response = httpsClient.execute(req);
91      } catch (IOException e) {
92          throw new CloudRuntimeException("Error on HTTPS request: " + e.getMessage());
93      }
94      return consumeResponse(response);
95  }
```

Similar code match (commit: 09e27fd)

```
res = getHttpClient().execute(req);
s_logger.info("ssp api call:" + req + " status=" + res.getStatusLine());
} catch (IOException e) {
s_logger.error("ssp api call failed: " + req, e);
}
```

Fix pattern (add error handling with proper HTTP Response)

```
try {
content = getHttpClient().execute(req, new BasicResponseHandler());
s_logger.info("ssp api call: " + req);
} catch (HttpResponseException e) {
s_logger.info("ssp api call failed: " + req, e);
if (e.getStatusCode() == HttpStatus.SC_UNAUTHORIZED && login()) {
req.reset();
content = getHttpClient().execute(req, new BasicResponseHandler());
s_logger.info("ssp api retry call: " + req);
}
```

Example 2 – State Machine Handling

- Project: Apache Kafka, commit: 8e4799b, file: KafkaStreams.java

synchronized setStateListener

```
317     public void setStateListener(final KafkaStreams.StateListener listener) {  
318         synchronized (stateLock) {  
319             if (state == State.CREATED) {  
320                 stateListener = listener;  
321             } else {  
322                 throw new IllegalStateException("Can only set StateListener in CREATED state. Current state is: " + state);  
323             }  
324         }  
325     }
```

Similar code match (commit: 343a8ef)

Commit Id: 343a8ef

HOTFIX: Cherry picking state deadlock fix from 0.11

concurrency

```
synchronized (stateLock) {  
    if (state == State.CREATED) {  
        state = State.RUNNING;  
    } else {  
        throw new IllegalStateException("Cannot start again.");  
    }  
}
```

Fix pattern: use setState instead of directly changing state

```
try {  
    if (setState(RUNNING)) {  
        return;  
    }  
} catch (StreamsException e) {  
    // do nothing, will throw  
}
```



Example 3 – Exception Handling

- Project: Apache Kafka, commit: 460e46c, file: KafkaConsumer.java

IllegalArgumentException thrown

```
1081         for (TopicPartition tp : partitions) {
1082             String topic = (tp != null) ? tp.topic() : null;
1083             if (topic == null || topic.trim().isEmpty())
1084                 throw new IllegalArgumentException("Topic partitions to assign to cannot have null or empty topic");
1085         }
```

Similar code match (commit: 842da7b)

Commit Id: 842da7b

Related issue: [KAFKA-350](#)

KAFKA-350 Enable message replication in the presence of failures; patched by Neha Narkhede; reviewed by Jun Rao and Jay Kreps

```
if (taskId == null) {
    throw new IllegalArgumentException(
        "Configutaion does not contain the property
        mapred.task.id");
}
```

Fix pattern: Use base KafkaException for better handling

```
if (taskId == null) {
    throw new KafkaException("Configuration does not contain
    the property mapred.task.id");
}
```



Example 3 – Exception Handling

- Project: Apache Kafka, co



Kafka / KAFKA-350

Enable message replication in the presence of controlled failures

IllegalArgumentException thrown

```
1081         for (TopicPartitic
1082             String topic =
1083             if (topic == r
1084             throw new
1085         }
```

Details

Type: Bug
Priority: Major
Affects Version/s: 0.8.0
Component/s: None
Labels: None

Status: **RESOLVED**
Resolution: Fixed
Fix Version/s: None

Similar code match (commit: 842da7b)

Commit Id: 842da7b

Related issue: [KAFKA-350](#)

```
if (taskId == null) {
    throw new IllegalArgumentException(
        "Configutaion does not contain the prop
        mapred.task.id");
}
```

- getMaxErrorCodeValue - seems to be recomputed for each TopicMetadata. Let's get rid of this, I don't think we need it. We already have an unknown entry in the mapping, we should use that and get rid of the Utils.getShortInRange
- If we do want to keep it, fix the name
- **We should really give a base class KafkaException to all exceptions so the client can handle them more easily**
- Instead of having the client get an IllegalArgumentException we should just throw new KafkaException("Unknown server error (error code " + code + ")")
- The file NoLeaderForPartitionException seems to be empty now, I think you meant to delete it

Example 4 – Concurrency

- Project: Apache ActiveMQ, commit: 2a815c2, file: DefaultJDBCAdapter.java

Missing “unlock”

```
808 public void doDeleteSubscription(TransactionContext c, ActiveMQDestination destination, String clientId,
809     String subscriptionName) throws SQLException, IOException {
810     PreparedStatement s = null;
811     try {
812         s = c.getConnection().prepareStatement(this.statements.getDeleteSubscriptionStatement());
813         s.setString(1, destination.getQualifiedName());
814         s.setString(2, clientId);
815         s.setString(3, subscriptionName);
816         s.executeUpdate();
817     } finally {
818         close(s);
819     }
820 }
```

Similar code match (commit: 06cbebc)

Commit Id: 06cbebc

Related issue: [AMQ-2980](#)

```
s.close();
s =
c.getConnection().prepareStatement(this.statements.getRemove
s.setString(1, destinationName.getQualifiedName());
s.executeUpdate();
} finally {
close(s);
}
```

Fix pattern: Missing call to release lock

further resolution to <https://issues.apache.org/activemq/browse/AMQ-2980>, concurrent producers was still problematic as the store needed to be traversed multiple times, requiring ack locations per priority. Resolution to <https://issues.apache.org/activemq/browse/AMQ-2551> fell out of decreasing the cleanup interval in some of the tests, cleanup needs an exclusive lock so it won't cause contention

concurrency

```
s.close();
s =
c.getConnection().prepareStatement(this.statements.getRemove
s.setString(1, destinationName.getQualifiedName());
s.executeUpdate();
} finally {
cleanupExclusiveLock.readLock().unlock();
close(s);
}
```

1/1

Outcome

- This approach is able to identify important issues which are otherwise not detected through traditional tools without writing new rules:
 - Missing conditional checks
 - Alternative API recommendations
 - Missing function calls (e.g. DB connection is opened but not closed)
- Since it is based on scanning commits and defects, it gets better with more commits as software evolves over time
- It also gets better with more software systems passed through it as it sees more similarities and variations between code elements which introduce/fix defects

Benefits

- Enhanced issue detection
 - Finds more issues than purely rule-based checks, and which have a direct correlation with defects
- Early feedback
 - Helps identify potential bugs before the code is committed to the repository
- Suggestions
 - Suggests fixes / alternatives to detected issues based on historical information
 - Feedback delivered through IDE plugins (e.g. Eclipse)

Thank You

Sudarshan Bhide
Co-Founder and CTO, Acellere
sudarshan.bhide@acellere.com

