Log4j Architecture Analysis

Log4j Version: log4j 2.4.1 and log4j 2.7
Tools: Arcade
Recovery Methods: Relax, ACDC and ARC

Introduction:
Apache log4j has layered architecture. Layered architecture makes the log4j design very flexible and easy to extend in future. Log4j has two types of objects-

(i) core objects: logger, layout and Appender and
(ii) support objects: Log manager, filter, renderer and Level object. In case of architecture drift, we add more functionalities into support objects. Core object are mandatory of log4j's architecture.

1. Comparison between Log4j_2.4.1 and Log4j_2.7 by Relax

(a) Analysis relax cluster and directories:

(i) Apache log4j 2.4.1 cluster and entity information

	Cluster	# Entities
1.	Graphics	37
2.	IO	32
3.	Networking	28
4.	SQL	14
5.	No Match	9

(ii) Apache log4j 2.7 cluster and entity information

	Cluster	# Entities
1.	Graphics	65
2.	IO	44
3.	Networking	35
4.	SQL	20
5.	No Match	7

(iii) As Relax is NLP technique which encapsulates related files into a cluster for a predefined topic, we can say that In Apache log4j 2.4.1(Table 1), we have total 120 entities with 5 topics named as Graphics, IO, Networking, SQL and No Match for unspecified entities. On other hand, in log4j 2.7(Table 2), we have 171 entities with same clusters as 2.4.1. Clearly 2.7 is more descriptive architecture than 2.4.1 and has been evolved a lot from 2.4.1 design. When we look into api_relax_cluster.pdf, we can say that a lot of utilities have been added to the later version of log4j.

(iv) There is a significant architecture drift from log4j 2.4.2 to log4j 2.7. In Log4j 2.4.1, There are two top sub-directories in parallel in test directory from top, *test.org.apache.log4j.util* and *test.org.apache.logging*. However, In 2.7 there is one directory present and *test.org.apache.log4j.util* was submerged into one *test.org.apache.logging.* We can conclude that test directory architecture was re-designed to consolidate into a more compact test environment.

(v) In log4j 2.4.1, Around 80% of entities are connected into the Graphics, IO and Networking clusters. On the other, 2.7 has 84% of total entities distributed among these three clusters with more cohesion and more entities. Moreover, in log4j 2.7 *message* and *ThreadConext* clusters have more connectivity with others. From this statement we can that Apache wanted to provide more utilities for threading and messaging services.

(vi) In log4j 2.7, There are more entities dependent on *logging.log4j.message* as compare to log4j 2.4.1. Moreover, from log4j_directories_x.x, we can say that in 2.4.1 *logging.log4j.message* and *logging.log4j.spi* packages shares major chunk of system dependencies and code **However, in 2.7**, addition to *logging.log4j.message* and *logging.log4j.spi*, *logging.log4j.util* package *also* share major chunk of system logic/code which substantiates the statement (iv)'s conclusion.

(vii) In log4j 2.7, *logging.log4j.ThreadContext* node was replaced by logging.log4j.closeableThreadContext into the logging system to improve threading concept.

2. Comparison between Log4j_2.4.1 and Log4j_2.7 by ACDC

(a) Analysis of api_acdc_cluster.rsf:

(i) From ACDC recovery method we are able to get following insights on the log4j_2.4.1 and log4j_2.7's architecture (clusters and number of entities).

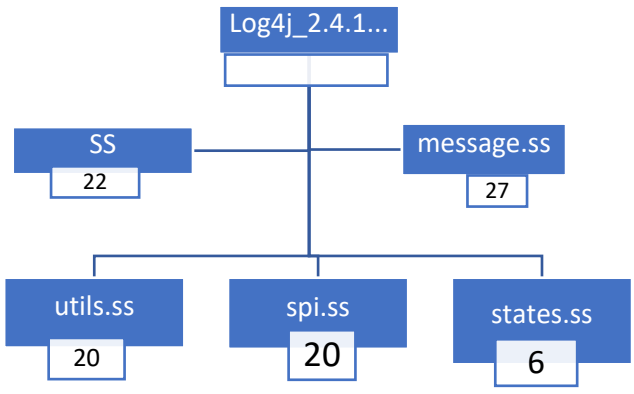

Figure 2.1: **log4j 2.4.1** cluster analysis by ACDC.

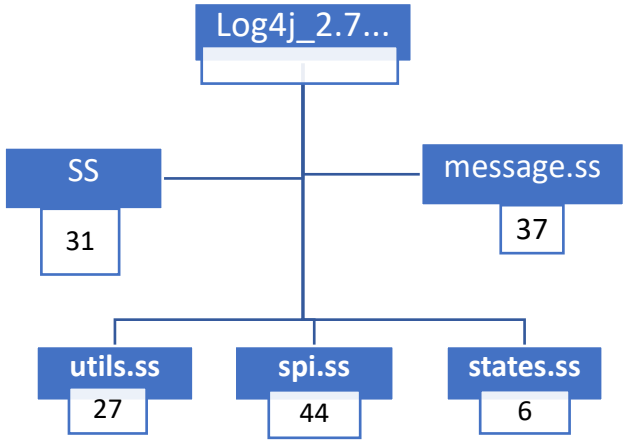

Figure 2.2: **log4j 2.7** cluster analysis by ACDC. ****Number** in the box indicate that particular cluster contains X number of entities.

(ii) Clearly log4j 2.7 is more cohesive as each cluster is formed by relevant libraries and it has more number of entities in each cluster than 2.4.1 except *states.ss* cluster. We also found that both the systems are dependent on other systems for example *log4j.util.ss* is using osgi organization's *org.osgi.framework.*

(iii) Log4j 2.4.1 has 89 entities in total and *message.ss* cluster has around 30% of the entities encapsulated in it. On the other hand, Log4j 2.7 has 145 entities in it. *spi.ss* and *message.ss* clusters have 30% and 25% entities respectively encapsulated in it. Moreover, 2.7's architecture evolved faster as it contains 50% more packages than 2.4.1.

(iv) As ACDC recovers architectures based on connections among entities like libraries, body etc. and then forms clusters. We can observe the introduction of following new entities in log4j 2.7 –

a. *GarbageFreeSortedArrayFactoryContext*
b. *ThreadContextFactory*
c. *MessageFactory2Adaptor*
d. *LoggerRegistry*
e. *DefaultMessageFactory*
f. *ReusableSimpleMessage, ResuableMessageFactory and ReusableObjectMessage.*

3. Comparison between Log4j_2.4.1 and Log4j_2.7 by ARC

(i) When we analyze *api_top_30_words.txt*, we can come up with following details on relevant words –

# of times a word in all clusters/words(2.4.1)	# of times a word in all clusters/words(2.7)
11 (2%) util 11 (2%) messag 11 (2%) log 10 (2%) param 9 (2%) return 9 (2%) format 8 (2%) trace 8 (2%) equal 7 (1%) level	15 (3%) log 13 (3%) equal 9 (2%) messag 9 (2%) overrid 9 (2%) param 8 (2%) test 8 (2%) util 8 (2%) return 8 (2%) format

Table 3.1: Occurrence of each words in a specific log4j version. X (y%) Z: - Z word is part of X clusters with y% of participation

(ii) Table 3.1 gives us lots of information about clusters and their most relevant words. In Log4j 2.4.1 *Util*, *message*, *log* and *param* are connected to 11 clusters However on other hand, *log* and *equal* words are more dominate with 6% of total share in log4j 2.7 architecture.

(iii) In log4j 2.7, there is one cluster in which "box", "unbox" are most relevant words which is not present in log4j 2.4.1. Unlike log4j 2.4.1, log4j 2.7 is dominated by "log" than "util" word.

(iv) When we analyze api_deps_cluster.pdf, We can clearly say that log4j 2.7 clusters are more connected to each other than 2.4.1. There has been an architecture drift from log4j 2.4.1 to log4j 2.7 in which lots of utilities related to threads and memory management have been added. For example, log4j 2.7 has following new

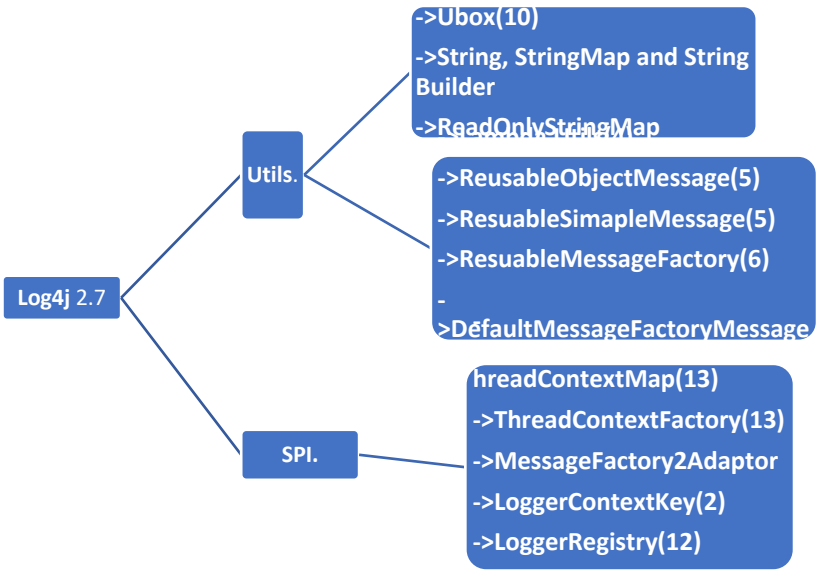

Fig 1.3: New feature in Log4j 2.7 as comparison to 2.4.1.
LoggerRegistry(X): where x is number of functionalities in each module.

new features like Lambda Util, String buffer, String Map, Reusable message objects and factories to increase the reusability of the existing components. Moreover in 2.7 *log4j.message.MultiFormatMessage* become more connected with other entities.

(v) With ARC recovery method we can clearly get more insight about inter-dependencies of clusters. Moreover, we can also advice developers to be cautious about a cluster which has more dependencies with other clusters and entities. In log4j 2.4.1, there is only **one pair** of clusters which is not interlinked or doesn't have any dependencies on each other are *org.apache.logging.log4j.message.ObjectMessage* and *org.apache.logging.log4j.util.Strings*. However, on the other hand in log4j 2.7 **there two pairs** of clusters which are not connected to each other are first, *org.apache.logging.log4j* and *org.apache.logging.log4j.spi* and second *org.apache.logging.log4j.util.SortedArrayStringMap* and *org.apache.logging.log4j.message.ReusableMessageFactory*

(vi) In log4j 2.4.1, There are 4 clusters which have single dependency or one direct link on other nodes However, in log4j 2.7 dependencies among clusters has been increased significantly and there is only one such cluster with single dependencies. In log4j 2.4.1 (from api_deps_cluster_2.4.1.pdf) clusters form a sparse connectivity graph However, 2.7 clusters form a dense graph which can be substantiated by their connectivity with other clusters from api_deps_clsuter.pdf.

4. Conclusion:

All three recovery methods give a very good sense of a system However, in my views, Relax is better recovery method to recover any complex architecture because it gives more inside view of a complex architecture like clusters and their relationship with other clusters.

Relax's graphical representation gives overall system architecture, nodes/clusters, entities, relationship/link among them. Moreover, Relax's recovery methods give information about architecture drift, re-designed components, directories, SQL, UI, IO connectivity of the clusters which we have already discussed. In 2.7 two parallel sub-directories from log4j 2.4.1 architecture were merged into one

Its cluster circle graph gives details about dependencies on one cluster. Here in 2.4.1 and 2.7, *"message*" and *"util"* clusters are more connected with others than any other node hence most of other clusters are depended on "*message"* and *"util"* cluster. This tells us that anything wrong in any of the most congested node or cluster could affect connected clusters in the system.