Object Oriented Design Quality Metrics

References

- [Analyze java package metrics in a graph database](#)
- [Calculate metrics](#)
- [jqassistant](#)
- [notebook walks through examples for integrating various packages with Neo4j](#)
- [OO Design Quality Metrics](#)
- [Neo4j Python Driver](#)

Incoming Dependencies

Incoming dependencies are also denoted as "Fan-in", "Afferent Couplings" or "in-degree". These are the ones that use the listed package.

If these packages get changed, the incoming dependencies might be affected by the change. The more incoming dependencies, the harder it gets to change the code without the need to adapt the dependent code ("rigid code"). Even worse, it might affect the behavior of the dependent code in an unwanted way ("fragile code").

Table 2

- Show the top 20 packages with the most incoming dependencies
- Set the "incomingDependencies" properties on Package nodes.

	packageName	incomingDependencies	incomingDependenciesWeight	incomingDependentTypes	incomingDependentInterfaces	inc
0	org	0	0	0	0	
1	org.axonframework	0	0	0	0	
2	org.axonframework.config	119	1560	35	0	
3	org	0	0	0	0	
4	org.axonframework	0	0	0	0	
5	org.axonframework.modelling	0	0	0	0	
6	org.axonframework.modelling.command	225	917	79	0	
7	org.axonframework.modelling.command.inspection	80	369	43	0	
8	org.axonframework.modelling.command.legacyjpa	2	14	2	0	
9	org.axonframework.modelling.saga	126	622	60	0	
10	org.axonframework.modelling.saga.metamodel	15	60	9	0	
11	org.axonframework.modelling.saga.repository	50	209	28	0	
12	org.axonframework.modelling.saga.repository.in...	5	23	5	0	
13	org.axonframework.modelling.saga.repository.jdbc	16	157	8	0	
14	org.axonframework.modelling.saga.repository.jpa	12	72	5	0	
15	org.axonframework.modelling.saga.repository.le...	3	15	2	0	
16	org	0	0	0	0	
17	org.axonframework	0	0	0	0	
18	org.axonframework.disruptor	0	0	0	0	
19	org.axonframework.disruptor.commandhandling	55	269	21	0	

Outgoing Dependencies

Outcoming dependencies are also denoted as "Fan-out", "Efferent Couplings" or "out-degree". These are the ones that are used by the listed package.

Code from other packages and libraries you're depending on (outgoing) might change over time. The more outgoing changes, the more likely and frequently code changes are needed. This involves time and effort which can be reduced by automation of tests and version updates. Automated tests are crucial to reveal updates, that change the behavior of the code unexpectedly ("fragile code"). As soon as more effort is required, keeping up becomes difficult ("rigid code"). Not being able to use a newer version might not only restrict features, it can get problematic if there are security issues. This might force you to take "fast but ugly" solutions into account which further increases technical dept.

Table 3

- Show the top 20 packages with the most outgoing dependencies
- Set the "outgoingDependencies" properties on Package nodes.

	packageName	outgoingDependencies	outgoingDependenciesWeight	outgoingDependentTypes	outgoingDependentInterfaces	out
0	org	0	0	0	0	
1	org.axonframework	0	0	0	0	
2	org.axonframework.config	467	2924	242	0	
3	org	0	0	0	0	
4	org.axonframework	0	0	0	0	
5	org.axonframework.modelling	0	0	0	0	
6	org.axonframework.modelling.command	231	958	94	0	
7	org.axonframework.modelling.command.inspection	157	723	73	0	
8	org.axonframework.modelling.command.legacyjpa	24	115	20	0	
9	org.axonframework.modelling.saga	141	571	62	0	
10	org.axonframework.modelling.saga.metamodel	23	71	15	0	
11	org.axonframework.modelling.saga.repository	69	397	36	0	
12	org.axonframework.modelling.saga.repository.in...	7	31	5	0	
13	org.axonframework.modelling.saga.repository.jdbc	42	303	25	0	
14	org.axonframework.modelling.saga.repository.jpa	29	133	18	0	
15	org.axonframework.modelling.saga.repository.le...	21	106	17	0	
16	org	0	0	0	0	
17	org.axonframework	0	0	0	0	
18	org.axonframework.disruptor	0	0	0	0	
19	org.axonframework.disruptor.commandhandling	185	869	85	0	

Instability

$$Instability = \frac{Outgoing\ Dependencies}{Outgoing\ Dependencies + Incoming\ Dependencies}$$

Instability is expressed as the ratio of the number of outgoing dependencies of a module (i.e., the number of packages that depend on it) to the total number of dependencies (i.e., the sum of incoming and outgoing dependencies).

Small values near zero indicate low *Instability*. With no outgoing but some incoming dependencies the Instability is zero which is denoted as maximally stable. Such code units are more rigid and difficult to change without impacting other parts of the system. If they are changed less because of that, they are considered stable.

Conversely, high values approaching one indicate high *Instability*. With some outgoing dependencies but no incoming ones the *Instability* is denoted as maximally unstable. Such code units are easier to change without affecting other modules, making them more flexible and less prone to cascading changes throughout the system. If they are changed more often because of that, they are considered unstable.

Table 4

- Show the top 20 packages with the lowest *Instability*

	p.fqn	p.name	instability	instabilityTypes	instabilityInterfaces	instabilityPackages	instabilityArtifacts	p.outgoingDe
0	org.axonframework.common	common	0.048421	0.042493	0.0	0.012658	0.142857	
1	org.axonframework.common.transaction	transaction	0.058824	0.050000	0.0	0.034483	0.200000	
2	org.axonframework.eventhandling.scheduling	scheduling	0.090909	0.142857	0.0	0.222222	0.250000	
3	org.axonframework.messaging	messaging	0.119565	0.096154	0.0	0.101695	0.142857	
4	org.axonframework.common.annotation	annotation	0.120000	0.120000	0.0	0.166667	0.250000	
5	org.axonframework.lifecycle	lifecycle	0.172414	0.233333	0.0	0.176471	0.250000	
6	org.axonframework.monitoring	monitoring	0.189189	0.152174	0.0	0.230769	0.200000	
7	org.axonframework.common.stream	stream	0.227273	0.166667	0.0	0.125000	0.250000	
8	org.axonframework.messaging.unitofwork	unitofwork	0.251064	0.196078	0.0	0.121951	0.142857	
9	org.axonframework.serialization	serialization	0.255172	0.250000	0.0	0.214286	0.200000	
10	org.axonframework.common.jpa	jpa	0.272727	0.250000	0.0	0.300000	0.200000	
11	org.axonframework.eventhandling.tokenstore	tokenstore	0.279412	0.325000	0.0	0.333333	0.333333	
12	org.axonframework.tracing	tracing	0.298643	0.225000	0.0	0.333333	0.200000	
13	org.axonframework.common.legacyjpa	legacyjpa	0.300000	0.277778	0.0	0.333333	0.250000	
14	org.axonframework.serialization.upcasting	upcasting	0.312500	0.083333	0.0	0.333333	0.500000	
15	org.axonframework.common.digest	digest	0.333333	0.333333	0.0	0.333333	0.500000	
16	org.axonframework.test	test	0.333333	0.318182	0.0	0.400000	0.666667	
17	org.axonframework.messaging.annotation	annotation	0.349146	0.307339	0.0	0.218750	0.142857	
18	org.axonframework.common.jdbc	jdbc	0.356436	0.431373	0.0	0.357143	0.200000	
19	org.axonframework.commandhandling	commandhandling	0.359551	0.370000	0.0	0.333333	0.142857	

Abstractness

$$Abstractness = \frac{abstract\ classes\ in\ category}{total\ number\ of\ classes\ in\ category}$$

Package *Abstractness* is expressed as the ratio of the number of abstract classes and interfaces to the total number of classes of a package.

Zero *Abstractness* means that there are no abstract types or interfaces in the package. On the other hand, a value of one means that there are only abstract types.

Table 5

- Show the top 30 packages with the lowest *Abstractness*

	fullQualifiedPackageName	packageName	abstractness	numberAbstractTypes	numberTypes
0	org.axonframework.eventsourcing.eventstore.leg...	legacyjpa	0.000000	0	10
1	org.axonframework.commandhandling.distributed....	commandfilter	0.000000	0	7
2	org.axonframework.deadline.dbscheduler	dbscheduler	0.000000	0	7
3	org.axonframework.eventhandling.scheduling.dbs...	dbscheduler	0.000000	0	7
4	org.axonframework.serialization.json	json	0.000000	0	7
5	org.axonframework.serialization.xml	xml	0.000000	0	7
6	org.axonframework.tracing.attributes	attributes	0.000000	0	6
7	org.axonframework.eventhandling.deadletter	deadletter	0.000000	0	5
8	org.axonframework.serialization.converters	converters	0.000000	0	5
9	org.axonframework.test.server	server	0.000000	0	4
10	org.axonframework.commandhandling.callbacks	callbacks	0.000000	0	4
11	org.axonframework.deadline.quartz	quartz	0.000000	0	4
12	org.axonframework.eventhandling.scheduling.java	java	0.000000	0	4
13	org.axonframework.eventhandling.tokenstore.jpa	jpa	0.000000	0	4
14	org.axonframework.modelling.saga.repository.le...	legacyjpa	0.000000	0	3
15	org.axonframework.eventhandling.scheduling.job...	jobrunr	0.000000	0	3
16	org.axonframework.util	util	0.000000	0	3
17	org.axonframework.modelling.command.legacyjpa	legacyjpa	0.000000	0	2
18	org.axonframework.modelling.saga.repository.in...	inmemory	0.000000	0	2
19	org.axonframework.eventsourcing.eventstore.inm...	inmemory	0.000000	0	2
20	org.axonframework.eventhandling.tokenstore.inm...	inmemory	0.000000	0	2
21	org.axonframework.eventhandling.tokenstore.leg...	legacyjpa	0.000000	0	2
22	org.axonframework.messaging.interceptors.legac...	legacyvalidation	0.000000	0	2
23	org.axonframework.common.io	io	0.000000	0	1
24	org.axonframework.common.digest	digest	0.000000	0	1
25	org.axonframework.eventhandling.interceptors	interceptors	0.000000	0	1
26	org.axonframework.disruptor.commandhandling	commandhandling	0.045455	1	22
27	org.axonframework.modelling.saga.repository.jdbc	jdbc	0.100000	1	10
28	org.axonframework.eventhandling.deadletter.jpa	jpa	0.111111	1	9
29	org.axonframework.eventhandling.tokenstore.jdbc	jdbc	0.111111	1	9

Distance from the main sequence

The *main sequence* is a imaginary line that represents a good compromise between *Abstractness* and *Instability*. A high distance to this line may indicate problems. For example is very *stable* (rigid) code with low abstractness hard to change.

Read more details on that in OO Design Quality Metrics and Calculate metrics.

Table 6

- Show the top 30 packages with the highest distance from the "main sequence"

	artifactName	fullQualifiedPackageName	packageName	distance	abstractness	instability	typesInPackage
0	axon-messaging-4.9.0	org.axonframework.common.io	io	1.000000	0.000000	0.000000	1
1	axon-messaging-4.9.0	org.axonframework.common.digest	digest	0.666667	0.000000	0.333333	1
2	axon-eventsourcing-4.9.0	org.axonframework.eventsourcing.eventstore.jdb...	statements	0.482759	1.000000	0.482759	15
3	axon-messaging-4.9.0	org.axonframework.monitoring	monitoring	0.477477	0.333333	0.189189	6
4	axon-messaging-4.9.0	org.axonframework.common.jpa	jpa	0.477273	0.250000	0.272727	4
5	axon-test-4.9.0	org.axonframework.test	test	0.466667	0.200000	0.333333	5
6	axon-messaging-4.9.0	org.axonframework.messaging.unitofwork	unitofwork	0.463222	0.285714	0.251064	14
7	axon-messaging-4.9.0	org.axonframework.common.lock	lock	0.454545	0.181818	0.363636	11
8	axon-messaging-4.9.0	org.axonframework.serialization	serialization	0.450710	0.294118	0.255172	34
9	axon-messaging-4.9.0	org.axonframework.common.legacyjpa	legacyjpa	0.450000	0.250000	0.300000	4
10	axon-messaging-4.9.0	org.axonframework.common.transaction	transaction	0.441176	0.500000	0.058824	4
11	axon-messaging-4.9.0	org.axonframework.tracing	tracing	0.438200	0.263158	0.298643	19
12	axon-messaging-4.9.0	org.axonframework.eventhandling.tokenstore	tokenstore	0.434874	0.285714	0.279412	7
13	axon-test-4.9.0	org.axonframework.test.matchers	matchers	0.423387	0.125000	0.451613	24
14	axon-modelling-4.9.0	org.axonframework.modelling.saga.repository.in...	inmemory	0.416667	0.000000	0.583333	2
15	axon-messaging-4.9.0	org.axonframework.serialization.xml	xml	0.414634	0.000000	0.585366	7
16	axon-messaging-4.9.0	org.axonframework.util	util	0.400000	0.000000	0.600000	3
17	axon-messaging-4.9.0	org.axonframework.commandhandling.callbacks	callbacks	0.380952	0.000000	0.619048	4
18	axon-messaging-4.9.0	org.axonframework.commandhandling.distributed....	commandfilter	0.375000	0.000000	0.625000	7
19	axon-messaging-4.9.0	org.axonframework.messaging.annotation	annotation	0.373076	0.277778	0.349146	54
20	axon-test-4.9.0	org.axonframework.test.server	server	0.333333	0.000000	0.666667	4
21	axon-messaging-4.9.0	org.axonframework.eventhandling.async	async	0.316667	0.133333	0.550000	15
22	axon-messaging-4.9.0	org.axonframework.common	common	0.308722	0.642857	0.048421	28
23	axon-eventsourcing-4.9.0	org.axonframework.eventsourcing.snapshotting	snapshotting	0.303030	0.333333	0.363636	3
24	axon-messaging-4.9.0	org.axonframework.messaging.correlation	correlation	0.300000	0.250000	0.450000	4
25	axon-messaging-4.9.0	org.axonframework.messaging	messaging	0.280435	0.600000	0.119565	35
26	axon-messaging-4.9.0	org.axonframework.common.property	property	0.271930	0.333333	0.394737	9
27	axon-messaging-4.9.0	org.axonframework.messaging.deadletter	deadletter	0.270632	0.368421	0.360947	19
28	axon-messaging-4.9.0	org.axonframework.commandhandling	commandhandling	0.265449	0.375000	0.359551	32
29	axon-messaging-4.9.0	org.axonframework.eventhandling.gateway	gateway	0.256250	0.600000	0.656250	5

Abstractness vs. *Instability* Plot with "Main Sequence" line as reference

Figure 1

- Plot *Abstractness* vs. *Instability* of all packages
- Draw the "main sequence" as dashed green line
- Scale the packages by the number of types they contain
- Color the packages by their distance to the "main sequence" (blue=near, red=far)

```
'io'
```

Abstractness vs. Instability ("Main Sequence")