Appendix for
"Large-scale, AST-based API-usage analysis of open-source Java projects"

Ralf Lämmel[1,2] and Ekaterina Pek[2] and Jürgen Starek[1]

[1] Software Languages Team, Universität Koblenz-Landau, Germany

[2] ADAPT Lab, Universität Koblenz-Landau, Germany

See next page.

1. References

[1] T. L. Veldhuizen. Software libraries and their reuse: Entropy, Kolmogorov complexity, and Zipf's law. *CoRR*, abs/cs/0508023, 2005.

	API	Domain	Core	# Projects	# Calls	# Distinct methods called
1	**Java Collections**	Collections	yes	1374	392639	406
2	**AWT**	GUI	yes	754	360903	1607
3	**Swing**	GUI	yes	716	581363	3369
4	**Reflection**	Other	yes	560	15611	154
5	**Core XML**	XML	yes	413	90415	537
6	**DOM**	XML	yes	324	52593	180
7	**SAX**	XML	no	310	13725	156
8	**log4j**	Logging	no	254	43533	187
9	**JUnit**	Testing	no	233	71481	1011
10	**Comm.Logging**	Logging	no	151	21996	88
11	**JNDI**	Networking	yes	101	2900	130
12	**Comm.Lang**	Other	no	93	4620	405
13	**JDOM**	XML	no	86	16770	423
14	**RMI**	Networking	yes	64	1183	46
15	**Hibernate**	Database	no	63	15192	2123
16	**Comm.Beanutils**	Other	no	51	407	67
17	**Xerces**	XML	no	42	3337	213
18	**Comm.Collections**	Collections	no	37	4085	1271
19	**dom4j**	XML	no	37	21874	157
20	**Lucene**	Search	no	36	12302	1684
21	**Comm.IO**	IO	no	34	450	72
22	**Comm.CLI**	Other	no	32	2463	134
23	**Comm.FileUpload**	Networking	no	31	626	49
24	**Axis**	Webservices	no	30	15746	210
25	**SWT**	GUI	no	30	56846	4361
26	**JMF**	Media	no	28	9030	488
27	**Comm.Codec**	Other	no	27	1064	108
28	**Struts**	Web Apps	no	26	11938	227
29	**Comm.Digester**	XML	no	20	1127	68
30	**Jena**	Semantic Web	no	20	7304	787
31	**BC Crypto**	Security	no	16	5147	569
32	**Comm.DBCP**	Database	no	15	119	48
33	**jMock2**	Testing	no	15	3888	46
34	**TestNG**	Testing	no	14	3974	24
35	**Comm.Pool**	Other	no	14	171	43
36	**GWT**	Web Apps	no	13	12987	639
37	**Java 3D**	GUI	no	12	490	80

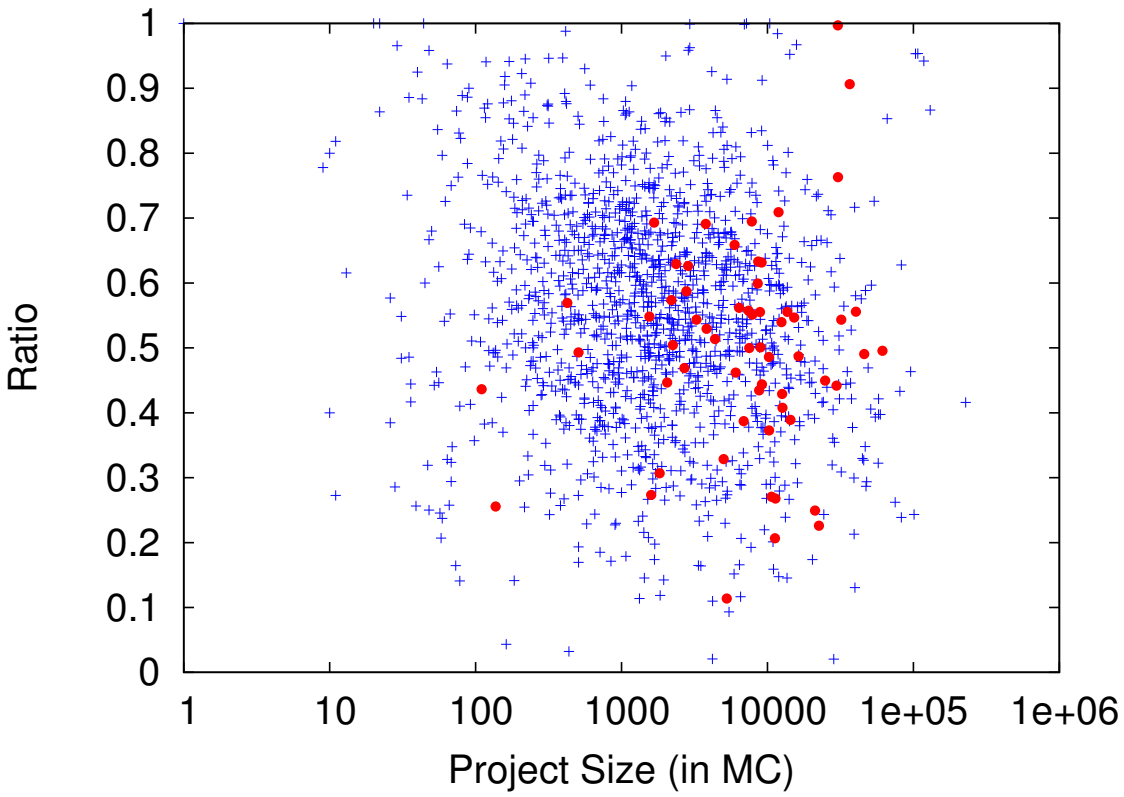
Table 1. List of the known APIs—Part I/II.

	API	Domain	Core	# Projects	# Calls	# Distinct methods called
38	**JFace**	GUI	no	10	3588	346
39	**Batik**	GUI	no	10	336	70
40	**Comm.Net**	Networking	no	10	2016	646
41	**LWJGL**	GUI	no	10	2988	325
42	**Berkeley DB**	Database	no	9	769	128
43	**Comm.Configuration**	Other	no	9	485	55
44	**JAI**	GUI	no	9	111	50
45	**XMLBeans**	XML	no	9	9891	154
46	**jogl**	GUI	no	9	261	45
47	**MySQL Connector/J**	Database	no	8	17738	1288
48	**JavaHelp**	GUI	no	8	29	10
49	**XMLPull**	XML	no	7	444	20
50	**Comm.Math**	Other	no	6	1608	487
51	**Jaxen**	XML	no	6	93	19
52	**Comm.Email**	Networking	no	5	64	28
53	**XOM**	XML	no	5	5801	98
54	**Comm.DbUtils**	Database	no	5	196	13
55	**Axis2**	Webservices	no	5	18794	164
56	**GNU Trove**	Collections	no	4	105	44
57	**AXIOM**	XML	no	4	3010	65
58	**j2ssh**	Networking	no	4	4238	1075
59	**Comm.Betwixt**	Other	no	3	209	28
60	**OFBiz**	e-Business	no	3	170713	6781
61	**Xalan**	XML	no	3	6	2
62	**Java Expression Language**	Other	no	2	9	6
63	**StAX**	XML	no	2	16	9
64	**Struts2**	Web Apps	no	2	25	8
65	**Express4J**	GUI	no	2	31331	4417
66	**Guice**	Other	no	2	50	9
67	**Comm.Discovery**	Other	no	2	11	3
68	**WSMO4J**	Webservices	no	2	774	155
69	**QuickFIX**	e-Business	no	1	62	30
70	**Comm.Transaction**	Other	no	1	6	4
71	**Comm.Chain**	Other	no	1	12	6
72	**Comm.EL**	Other	no	0	0	0
73	**Comm.Daemon**	Other	no	0	0	0
74	**Comm.Exec**	Other	no	0	0	0
75	**Comm.Proxy**	Other	no	0	0	0
76	**Comm.Primitives**	Other	no	0	0	0
77	**Comm.Attributes**	Other	no	0	0	0

Table 2. List of the known APIs—Part II/II.

API	# Projects			# Methods		# Distinct methods		# Derived types		# API types	
	impl.	**ext.**	**any**	**impl.**	**over.**	**impl.**	**over.**	**interf.**	**classes**	**interf.**	**classes**
Swing	173	381	391	2512	11150	305	645	443	1859	39	92
AWT	194	75	225	4201	756	593	176	651	120	31	24
Java Collections	120	0	120	986	0	16	0	208	0	3	0
SAX	28	21	42	428	90	85	21	37	29	12	3
JUnit	3	38	40	4	344	4	19	3	46	2	2
Core XML	11	5	14	89	13	17	4	14	5	9	3
SWT	5	8	10	37	86	4	13	25	11	3	3
log4j	1	8	8	25	87	7	9	2	9	2	3
Reflection	7	0	7	10	0	1	0	7	0	1	0
JMF	4	2	6	8	6	6	3	4	3	3	3
DOM	6	0	6	572	0	41	0	15	0	10	0
GWT	5	6	6	442	143	59	36	18	26	22	11
Hibernate	5	1	5	136	1	78	1	17	1	12	1
Lucene	1	5	5	1	22	1	11	1	13	1	7
Xerces	4	3	4	915	143	46	6	9	1	8	1
Axis	2	1	3	8	4	2	1	5	1	1	1
JNDI	3	0	3	52	0	18	0	3	0	3	0
Struts	0	2	2	0	14	0	2	0	2	0	2
Commons Beanutils	0	1	1	0	1	0	1	0	1	0	1
JFace	0	1	1	0	4	0	1	0	1	0	1
RMI	0	1	1	0	1	0	1	0	1	0	1
Commons Collections	1	0	1	2	0	1	0	2	0	1	0
jMock2	0	1	1	0	7	0	1	0	1	0	1
GNU Trove	0	1	1	0	5	0	2	0	1	0	1
Commons Digester	0	1	1	0	2	0	1	0	1	0	1
Commons Logging	1	0	1	18	0	1	0	1	0	1	0
Bouncy Castle Crypto	0	1	1	0	2	0	1	0	1	0	1
Jena	1	0	1	3	0	1	0	1	0	1	0
Commons Pool	0	1	1	0	3	0	2	0	1	0	1
Commons Chain	1	0	1	1	0	1	0	1	0	1	0
Commons DbUtils	0	1	1	0	1	0	1	0	1	0	1
Commons Lang	0	1	1	0	4	0	3	0	2	0	1
Berkeley DB	0	1	1	0	1	0	1	0	1	0	1
Commons Net	0	1	1	0	3	0	2	0	1	0	1
LWJGL	0	1	1	0	1	0	1	0	1	0	1

Table 3. Summary of implementations and overridings of APIs.



Projects	Min	1st Q	Median	Mean	3rd Q	Max
All	0.0205	0.4461	0.5551	0.5567	0.6724	1
Reference	0.1138	0.4332	0.5026	0.5015	0.5704	0.9969

Fig. 1 shows the usage of known API methods relative to all methods in a project—both in terms of calls. The smaller the ratio (the closer to zero), the lower the contribution of API calls. The quartiles show that in most projects, about each second method call is an API call. As far as instance-method calls are concerned, the figure distinguishes API vs. project-based method calls solely on the grounds of the static receiver type of methods.

Figure 1. Ratio of API method calls to all method calls.

Fig. 2 shows the relative frequency of API-interface implementations for all the known APIs (including Core APIs). The picture is dominated by AWT handler types, the interface for iterators, and a few XML-related types.

Figure 2. Tag cloud of implemented API interfaces.

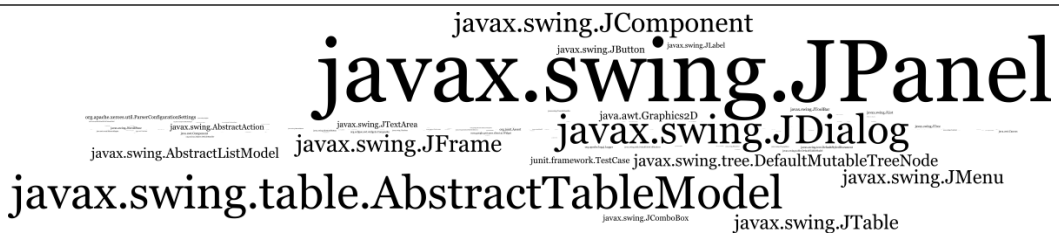
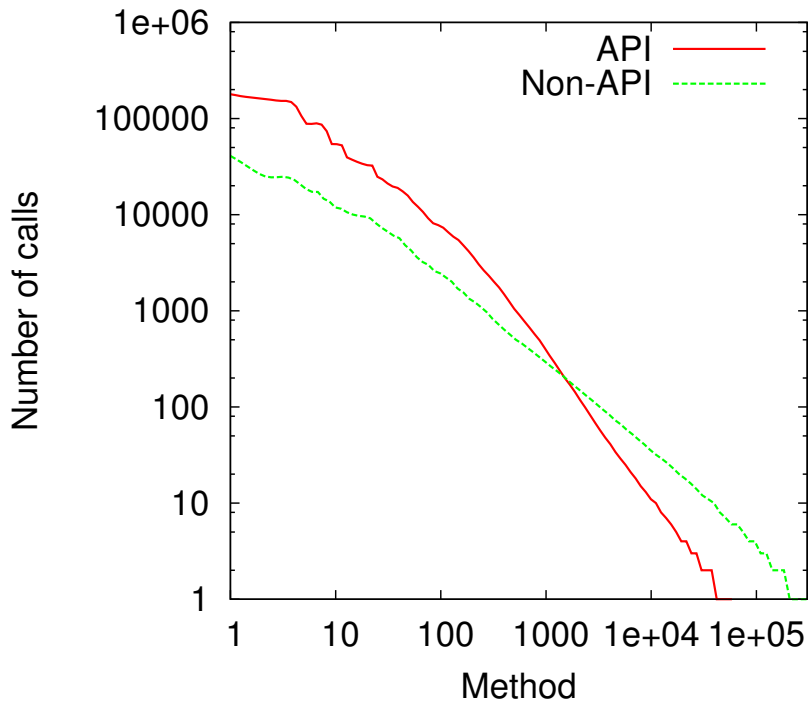


Fig. 3 shows the relative frequency of API-class extensions. The picture is entirely dominated by Swing's types for GUIs. Many other API types are implemented or extended, but only with a marginal frequency.

Figure 3. Tag cloud of overridden API classes.



In [1], library reuse is studied at a level of shared objects in the operating systems Sun OS and Mac OS X. One of the observations is that reuse seems to be low in the sense of Zipf's law. Thus the most frequent function will be referenced approximately twice as often as the second most frequent function, which occurs twice as often as the fourth most frequent function, etc. Fig. 4 shows the distribution of frequency of method calls in the corpus of this paper. Only 0.03% of methods are called more than 10 000 times, while 98.2% of methods are called less than 100 times. The plot suggests a Zipf-style distribution.

Figure 4. Frequency of calling API vs. non-API methods.