A EVALUATION

A.1 Precision

Table 1: Precision improvements. *n*: no. of tests selected. *f* and $S_f[\%]$: no. of tests selected and the % precision improvement over Ekstazi or STARTS by only leveraging our findings. *fm* and $S_{fm}[\%]$: the respective quantities when leveraging our findings *and* method-level reasoning. $b[\%]$: % of RetestAll tests that are selected.

PID	RetestAll	Ekstazi		FineEkstazi				STARTS		FineSTARTS				HyRTS
	n	n	$b[\%]$	f	$S_f[\%]$	fm	$S_{fm}[\%]$	n	$b[\%]$	f	$S_f[\%]$	fm	$S_{fm}[\%]$	n
P1	5760	1193	20.7	1057	11.4	819	31.3	2103	36.5	1904	9.5	1673	20.4	693
P2	7441	732	9.8	404	44.8	142	80.6	2772	37.3	1599	42.3	1341	51.6	234
P3	8522	486	5.7	371	23.7	167	65.6	738	8.7	556	24.7	362	50.9	329
P4	2300	94	4.1	88	6.4	62	34.0	188	8.2	164	12.8	138	26.6	46
P5	3033	138	4.5	107	22.5	80	42.0	205	6.8	151	26.3	123	40.0	62
P6	8452	1373	16.2	1204	12.3	836	39.1	1981	23.4	1817	8.3	1411	28.8	185
P7	7130	1339	18.8	1218	9.0	954	28.8	2647	37.1	2476	6.5	2089	21.1	636
P8	1199	217	18.1	183	15.7	142	34.6	685	57.1	658	3.9	616	10.1	130
P9	765	248	32.4	239	3.6	152	38.7	287	37.5	276	3.8	204	28.9	84
P10	114848	31187	27.2	29869	4.2	28709	7.9	61267	53.3	59008	3.7	56294	8.1	9746
P11	23382	1851	7.9	1619	12.5	584	68.4	3254	13.9	3069	5.7	2093	35.7	998
P12	2200	167	7.6	118	29.3	93	44.3	193	8.8	144	25.4	119	38.3	63
P13	5130	553	10.8	464	16.1	446	19.3	766	14.9	646	15.7	644	15.9	N/A
P14	2415	168	7.0	106	36.9	79	53.0	746	30.9	652	12.6	643	13.8	124
P15	2131	373	17.5	287	23.1	203	45.6	445	20.9	348	21.8	281	36.9	222
P16	5646	355	6.3	190	46.5	68	80.8	518	9.2	257	50.4	149	71.2	108
P17	1963	930	47.4	755	18.8	609	34.5	1253	63.8	1032	17.6	861	31.3	N/A
P18	1200	1018	84.8	974	4.3	882	13.4	1067	88.9	1019	4.5	927	13.1	N/A
P19	1916	1178	61.5	1142	3.1	884	25.0	1051	54.9	981	6.7	695	33.9	179
P20	1100	130	11.8	106	18.5	97	25.4	208	18.9	176	15.4	145	30.3	78
P21	6000	1746	29.1	1619	7.3	956	45.2	3293	54.9	3080	6.5	2409	26.8	946
P22	5030	328	6.5	291	11.3	159	51.5	391	7.8	315	19.4	168	57.0	212
P23	715	183	25.6	131	28.4	90	50.8	224	31.3	169	24.6	134	40.2	72
Avg	9490.3	1999.4	20.9	1849.7	17.8	1618.0	41.7	3751.4	31.5	3499.9	16.0	3196.5	31.8	-
Sum	218278	45987	21.1	42542	7.5	37213	19.1	86282	39.5	80497	6.7	73519	14.8	-

Table 1 presents the number of tests selected by Ekstazi, $FineEkstazi^F$, FineEkstazi, STARTS, $FineSTARTS^F$, FineSTARTS, and HyRTS on 23 projects.

A.2 Time

Table 2 presents the analysis time, execution time, collection time of Ekstazi, $FineEkstazi^F$, FineEkstazi, STARTS, $FineSTARTS^F$, and FineSTARTS.

Table 3 presents the end-to-end time of Ekstazi, $FineEkstazi^F$, FineEkstazi, STARTS, $FineSTARTS^F$, FineSTARTS and HyRTS.

Table 4 presents the analysis time, execution time, collection time of 23 projects of STARTS, $FineSTARTS^F$, and FineSTARTS.

Table 5 presents the analysis time, execution time, collection time of 23 projects of Ekstazi, $FineEkstazi^F$, and FineEkstazi.

Table 6 presents the offline time of 23 projects of Ekstazi, $FineEkstazi^F$, FineEkstazi, STARTS, $FineSTARTS^F$, and FineSTARTS.

A.3 Tables for comparing with ML models

Table 7 presents the selection rate we used in Fail-Basic^E, Fail-Basic^S, BM25^E, and BM25^S.

Table 8 presents the number of selected test classes of FineEkstazi, FineSTARTS, Fail-Basic^E, Fail-Basic^S, BM25^E, and BM25^S

Table 9 presents the end-to-end time of FineEkstazi, FineSTARTS, Fail-Basic^E, Fail-Basic^S, BM25^E, and BM25^S

A.4 Plots for comparing with ML models

Figure 1 shows the number of selected tests of FineEkstazi, FineSTARTS, Fail-Basic^E, Fail-Basic^S, BM25^E, and BM25^S on 10 projects.

Table 2: AEC time improvements. *t*: RTS tool end-to-end time (s). *f* and S_f[%]: end-to-end time (s) and the % end-to-end time improvements over Ekstazi or STARTS by only leveraging our findings. *fm*, S_{fm}[%], and ${}^{M}S_{fm}$[%]: the respective quantities improvement when leveraging our findings *and* method-level reasoning. ${}^{M}S_{fm}$[%] includes Maven overhead. *b*[%]: % of RetestAll end-to-end time.

PID	RetestAll	Ekstazi		FineEkstazi					STARTS		FineSTARTS				
	t	*t*	*b*[%]	*f*	S_f[%]	*fm*	S_{fm}[%]	${}^{M}S_{fm}$[%]	*t*	*b*[%]	*f*	S_f[%]	*fm*	S_{fm}[%]	${}^{M}S_{fm}$[%]
P1	778	772	99.3	741	4.0	729	5.6	2.4	521	67.0	500	4.1	497	4.7	2.6
P2	831	223	26.8	179	19.6	101	54.6	18.8	280	33.7	187	33.4	164	41.4	22.9
P3	878	206	23.5	170	17.5	123	40.3	10.3	157	17.9	132	15.9	116	25.9	9.1
P4	1004	137	13.7	144	-5.3	111	19.1	4.4	157	15.6	158	-0.7	121	22.9	8.7
P5	1121	296	26.4	250	15.6	205	30.7	15.8	178	15.9	159	11.0	124	30.3	16.1
P6	1260	1036	82.2	991	4.3	799	22.9	14.7	526	41.8	501	4.9	417	20.8	12.7
P7	1447	744	51.5	626	16.0	556	25.4	19.2	774	53.5	697	10.0	572	26.2	21.5
P8	1504	825	54.9	782	5.2	635	23.1	27.2	1176	78.2	1137	3.4	1039	11.7	10.7
P9	2309	2746	118.9	2652	3.4	1741	36.6	33.1	1441	62.4	1406	2.4	1003	30.4	28.4
P10	2391	1298	54.3	1299	-0.1	1277	1.6	1.6	1416	59.2	1387	2.0	1398	1.2	1.4
P11	2516	640	25.4	701	-9.6	418	34.7	23.6	511	20.3	480	6.1	352	31.1	17.8
P12	3114	524	16.8	405	22.7	206	60.8	39.7	276	8.9	217	21.4	117	57.4	37.8
P13	3731	535	14.3	486	9.2	441	17.5	12.9	555	14.9	473	14.8	475	14.4	10.4
P14	4471	618	13.8	476	23.0	374	39.5	35.3	1437	32.1	1269	11.7	1222	14.9	14.0
P15	4643	2891	62.3	1760	39.1	1569	45.7	50.4	1819	39.2	1442	20.7	977	46.3	41.9
P16	5653	737	13.0	528	28.3	192	74.0	54.0	688	12.2	393	42.9	220	68.0	52.8
P17	7861	5036	64.1	4172	17.2	3436	31.8	30.9	5898	75.0	4912	16.7	4120	30.1	29.1
P18	8491	17803	209.7	16969	4.7	15889	10.8	10.3	7784	91.7	7440	4.4	6977	10.4	9.9
P19	11584	22522	194.4	12478	44.6	10842	51.9	14.7	8150	70.4	7676	5.8	5958	26.9	26.2
P20	16692	4302	25.8	3848	10.6	3301	23.3	22.7	5083	30.5	4421	13.0	3810	25.1	24.4
P21	16847	22751	135.0	21296	6.4	14766	35.1	34.4	10879	64.6	10233	5.9	7940	27.0	26.4
P22	59296	5836	9.8	5044	13.6	2480	57.5	55.4	6712	11.3	5280	21.3	2445	63.6	60.6
P23	65432	33763	51.6	25433	24.7	22488	33.4	31.3	32457	49.6	26993	16.8	23067	28.9	28.7
Avg	9733	5489	60.3	4410	13.7	3595	33.7	24.5	3864	42.0	3369	12.5	2745	28.7	22.4
Sum	223857	126242	-	101431	-	82677	-	-	88877	-	77492	-	63132	-	-

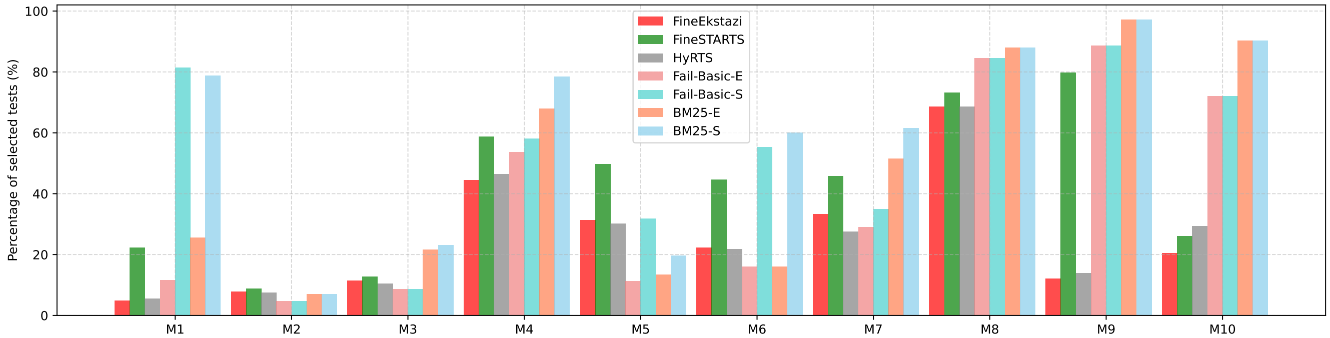

Figure 1: Percentage of number of selected test classes of RTS tools and ML models over RetestAll.

Table 3: Improvements in the end-to-end time of FineEkstazi over Ekstazi and FineSTARTS over STARTS. n: no. of tests selected. f and $S_f[\%]$ show the end-to-end time (s) and the percentage time improvement by only incorporating the findings from our manual analysis. fm and $S_{fm}[\%]$ show the respective quantities when findings from our manual analysis plus method-level reasoning are incorporated. $b[\%]$ means comparing with RetestAll.

PID	RetestAll	Ekstazi		FineEkstazi				STARTS		FineSTARTS				HyRTS
	t	t	$b[\%]$	f	$S_f[\%]$	fm	$S_{fm}[\%]$	t	$b[\%]$	f	$S_f[\%]$	fm	$S_{fm}[\%]$	t
P1	778.0	618.1	79.4	606.1	1.9	603.0	2.4	783.9	100.8	765.3	2.4	763.5	2.6	618.3
P2	831.5	312.0	37.5	291.8	6.5	253.4	18.8	501.5	60.3	409.6	18.3	386.7	22.9	281.0
P3	878.2	337.1	38.4	321.5	4.6	302.4	10.3	431.5	49.1	409.5	5.1	392.2	9.1	340.5
P4	1004.3	238.5	23.7	243.0	-1.9	228.0	4.4	406.9	40.5	408.8	-0.5	371.5	8.7	256.7
P5	1121.3	293.4	26.2	268.9	8.3	247.0	15.8	321.1	28.6	305.1	5.0	269.3	16.1	513.4
P6	1260.2	816.1	64.8	795.3	2.6	696.3	14.7	862.0	68.4	836.2	3.0	752.2	12.7	2042.2
P7	1446.8	566.0	39.1	491.5	13.2	457.1	19.2	927.3	64.1	853.6	7.9	727.8	21.5	2885.0
P8	1503.8	725.7	48.3	673.8	7.1	528.5	27.2	1271.7	84.6	1232.3	3.1	1135.3	10.7	448.8
P9	2309.3	1517.3	65.7	1471.3	3.0	1014.4	33.1	1543.7	66.8	1507.7	2.3	1104.9	28.4	1401.3
P10	2391.3	1377.2	57.6	1378.4	-0.1	1355.0	1.6	1523.7	63.7	1489.4	2.3	1502.6	1.4	837.7
P11	2516.3	712.9	28.3	699.5	1.9	545.0	23.6	871.0	34.6	843.7	3.1	715.8	17.8	914.6
P12	3113.7	397.8	12.8	338.7	14.9	239.7	39.7	416.6	13.4	358.8	13.9	259.1	37.8	614.0
P13	3731.4	714.1	19.1	665.1	6.9	622.1	12.9	786.4	21.1	703.1	10.6	704.7	10.4	N/A
P14	4471.4	689.0	15.4	546.7	20.7	445.6	35.3	1527.4	34.2	1359.6	11.0	1313.7	14.0	512.7
P15	4642.5	2174.9	46.8	1198.5	44.9	1078.7	50.4	2004.7	43.2	1631.7	18.6	1164.5	41.9	1571.5
P16	5652.7	630.1	11.1	490.1	22.2	289.8	54.0	884.5	15.6	591.2	33.2	417.2	52.8	333.3
P17	7861.5	5179.5	65.9	4314.7	16.7	3578.1	30.9	6097.6	77.6	5111.0	16.2	4320.9	29.1	N/A
P18	8491.3	9205.0	108.4	8789.1	4.5	8256.6	10.3	8136.7	95.8	7796.0	4.2	7332.2	9.9	N/A
P19	11584.1	12902.8	111.4	12647.1	2.0	11011.7	14.7	8382.6	72.4	7906.5	5.7	6185.0	26.2	1822.4
P20	16691.6	4406.4	26.4	3952.5	10.3	3405.2	22.7	5211.5	31.2	4548.9	12.7	3938.6	24.4	2741.7
P21	16847.2	11582.4	68.7	10860.4	6.2	7595.6	34.4	11109.8	65.9	10465.6	5.8	8171.5	26.4	17788.9
P22	59296.0	6060.6	10.2	5264.0	13.1	2700.3	55.4	7037.5	11.9	5610.3	20.3	2771.8	60.6	14697.9
P23	65432.5	32962.0	50.4	24604.8	25.4	22634.1	31.3	32677.2	49.9	27215.9	16.7	23287.0	28.7	490.1
Avg	9732.9	4105.2	45.9	3517.9	10.2	2960.3	24.5	4074.6	51.9	3580.9	9.6	2956.0	22.4	-
Sum	223856.8	94418.9	42.2	80912.8	14.3	68087.5	27.9	93716.8	41.9	82359.8	12.1	67988.0	27.5	-

Table 4: AEC Time of STARTS Family. S stands for STARTS, F stands for $FineSTARTS^F$, F+M stands for FineSTARTS. s(%) means the percentage saving compared with STARTS. All the numbers are in seconds.

PID	analysis time					execution time					collection time					total time					maven time		
	S	F	s(%)	F+M	s(%)	S	F	s(%)	F+M	s(%)	S	F	s(%)	F+M	s(%)	S	F	s(%)	F+M	s(%)	S	F	F+M
P1	4.0	6.7	-67.5	13.4	-236.9	503.1	480.7	4.5	470.0	6.6	14.4	12.8	11.4	13.3	7.5	521.5	500.1	4.1	496.7	4.7	783.9	765.3	763.5
P2	5.8	8.0	-36.5	13.4	-128.7	261.4	167.1	36.1	139.0	46.8	13.0	11.6	10.8	11.8	9.3	280.2	186.7	33.4	164.2	41.4	501.5	409.6	386.7
P3	6.1	8.2	-33.8	14.4	-135.8	130.3	104.2	20.0	82.2	36.9	20.6	19.7	4.4	19.7	4.4	157.0	132.1	15.9	116.3	25.9	431.5	409.5	392.2
P4	4.4	10.8	-148.5	17.8	-308.6	134.9	131.0	2.9	87.7	35.0	17.4	15.9	8.7	15.3	12.2	156.6	157.7	-0.7	120.8	22.9	406.9	408.8	371.5
P5	3.5	4.9	-41.5	6.1	-76.8	166.7	146.5	12.1	110.7	33.6	8.0	7.3	9.4	7.4	7.9	178.2	158.7	11.0	124.2	30.3	321.1	305.1	269.3
P6	5.5	7.2	-32.1	17.0	-211.2	496.8	469.1	5.6	374.2	24.7	24.1	24.3	-0.7	25.7	-6.4	526.4	500.6	4.9	416.9	20.8	862.0	836.2	752.2
P7	5.6	8.6	-52.6	17.6	-212.6	748.8	669.3	10.6	536.4	28.4	20.0	19.2	4.2	17.7	11.5	774.5	697.1	10.0	571.7	26.2	927.3	853.6	727.8
P8	5.0	7.4	-47.8	10.0	-98.7	1159.2	1117.5	3.6	1016.5	12.3	12.0	11.7	2.7	12.1	-0.8	1176.3	1136.6	3.4	1038.6	11.7	1271.7	1232.3	1135.3
P9	3.7	7.1	-90.1	9.6	-156.5	1405.4	1368.1	2.7	966.4	31.2	32.1	31.2	2.7	27.4	14.5	1441.2	1406.4	2.4	1003.4	30.4	1543.7	1507.7	1104.9
P10	24.1	42.4	-75.8	120.0	-398.0	1247.2	1205.3	3.4	1144.9	8.2	144.4	139.1	3.7	133.3	7.7	1415.7	1386.8	2.0	1398.3	1.2	1523.7	1489.4	1502.6
P11	10.6	15.3	-43.8	32.8	-208.8	466.6	432.4	7.3	288.9	38.1	34.1	32.5	4.7	30.6	10.3	511.3	480.2	6.1	352.4	31.1	871.0	843.7	715.8
P12	2.4	6.2	-155.5	7.6	-213.3	266.4	204.4	23.3	103.8	61.0	6.9	6.1	11.0	6.0	13.5	275.8	216.8	21.4	117.4	57.4	416.6	358.8	259.1
P13	3.2	4.5	-38.1	7.1	-118.4	543.6	461.4	15.1	460.7	15.2	8.2	6.9	15.0	7.1	13.2	555.0	472.8	14.8	474.9	14.4	786.4	703.1	704.7
P14	3.3	5.6	-73.1	10.2	-212.1	1418.6	1249.3	11.9	1198.2	15.5	15.0	13.9	7.6	13.8	8.1	1436.8	1268.8	11.7	1222.2	14.9	1527.4	1359.6	1313.7
P15	2.5	4.0	-56.6	6.0	-135.6	1808.4	1431.7	20.8	965.1	46.6	8.1	6.8	15.9	6.2	22.9	1818.9	1442.4	20.7	977.3	46.3	2004.7	1631.7	1164.5
P16	4.1	5.0	-22.1	6.3	-54.9	671.2	376.2	44.0	201.9	69.9	12.9	12.0	7.2	11.9	8.0	688.2	393.1	42.9	220.1	68.0	884.5	591.2	417.2
P17	2.7	4.3	-57.3	7.9	-192.3	5882.2	4894.9	16.8	4100.3	30.3	13.5	12.4	8.3	11.9	12.2	5898.4	4911.6	16.7	4120.1	30.1	6097.6	5111.0	4320.9
P18	6.1	11.0	-79.8	20.9	-242.8	7750.6	7403.5	4.5	6929.1	10.6	27.2	25.9	4.7	27.3	-0.3	7783.9	7440.4	4.4	6977.2	10.4	8136.7	7796.0	7332.2
P19	2.3	4.7	-109.3	7.6	-237.4	8114.6	7638.2	5.9	5918.8	27.1	33.1	32.6	1.4	31.2	5.7	8150.0	7675.6	5.8	5957.7	26.9	8382.6	7906.5	6185.0
P20	2.0	3.5	-72.1	4.7	-131.2	5074.9	4412.4	13.1	3800.1	25.1	6.2	5.5	11.7	4.9	21.6	5083.2	4421.5	13.0	3809.7	25.1	5211.5	4548.9	3938.6
P21	7.3	10.2	-39.7	23.2	-219.5	10849.1	10201.6	6.0	7893.9	27.2	22.7	21.7	4.4	22.6	0.5	10879.1	10233.4	5.9	7939.7	27.0	11109.8	10465.6	8171.5
P22	5.8	8.9	-53.4	12.8	-119.3	6697.2	5262.8	21.4	2424.5	63.8	9.2	7.9	13.9	7.6	17.3	6712.2	5279.7	21.3	2444.9	63.6	7037.5	5610.3	2771.8
P23	1.7	3.2	-81.9	4.3	-146.5	32445.6	26981.3	16.8	23053.1	28.9	10.0	8.6	14.4	9.6	4.8	32457.4	26993.1	16.8	23067.0	28.9	32677.2	27215.9	23287.0
Avg	5.3	8.6	-65.6	17.0	-182.4	3836.6	3339.5	13.4	2707.2	31.4	22.3	21.1	7.7	20.6	8.9	3864.2	3369.2	12.5	2744.8	28.7	4074.6	3580.9	2956.0
Sum	121.8	197.6	-62.2	390.8	-220.8	88242.5	76808.7	13.0	62266.5	29.4	513.2	485.5	5.4	474.2	7.6	88877.5	77491.9	12.8	63131.5	29.0	93716.8	82359.8	67988.0

Table 5: AEC Time of Ekstazi Family. E stands for Ekstazi, F stands for *FineEkstazi*^{F}, F+M stands for FineEkstazi. s(%) means the percentage saving compared with Ekstazi. All the numbers are in seconds.

PID	analysis time					execution time+ collection time					total time					maven time		
	E	F	s(%)	F+M	s(%)	E	F	s(%)	F+M	s(%)	E	F	s(%)	F+M	s(%)	E	F	F+M
P1	17.2	22.1	-28.5	27.7	-60.7	396.3	378.2	4.6	368.9	6.9	772.3	741.3	4.0	729.2	5.6	618.1	606.1	603.0
P2	21.2	24.8	-16.8	26.9	-26.4	120.0	95.8	20.2	55.3	53.9	222.9	179.2	19.6	101.2	54.6	312.0	291.8	253.4
P3	26.6	28.6	-7.5	40.1	-50.7	102.5	82.7	19.3	52.2	49.0	206.3	170.2	17.5	123.1	40.3	337.1	321.5	302.4
P4	19.4	24.0	-23.7	31.9	-64.7	71.4	71.7	-0.4	48.7	31.8	137.1	144.4	-5.3	110.9	19.1	238.5	243.0	228.0
P5	17.1	18.8	-10.1	18.8	-10.1	153.8	128.7	16.3	106.4	30.8	296.3	250.1	15.6	205.3	30.7	293.4	268.9	247.0
P6	57.9	65.1	-12.3	53.5	7.6	513.6	487.0	5.2	396.4	22.8	1036.5	991.4	4.3	799.4	22.9	816.1	795.3	696.3
P7	31.1	34.4	-10.6	36.6	-17.8	410.5	332.4	19.0	295.4	28.0	744.4	625.6	16.0	555.6	25.4	566.0	491.5	457.1
P8	39.6	48.2	-21.7	40.2	-1.4	613.0	552.3	9.9	414.4	32.4	825.3	782.1	5.2	634.9	23.1	725.7	673.8	528.5
P9	28.8	34.3	-19.3	33.0	-14.7	1410.6	1358.6	3.7	902.6	36.0	2746.4	2651.8	3.4	1741.2	36.6	1517.3	1471.3	1014.4
P10	220.7	235.0	-6.5	279.3	-26.6	1077.3	1064.2	1.2	997.6	7.4	1298.0	1299.2	-0.1	1276.9	1.6	1377.2	1378.4	1355.0
P11	30.8	37.2	-20.8	57.9	-87.8	431.5	409.6	5.1	237.8	44.9	639.7	700.9	-9.6	417.7	34.7	712.9	699.5	545.0
P12	22.9	26.5	-15.6	27.7	-20.8	260.9	197.9	24.1	97.1	62.8	524.2	405.2	22.7	205.6	60.8	397.8	338.7	239.7
P13	17.5	19.6	-12.1	22.4	-27.9	517.7	466.2	9.9	419.0	19.1	535.1	485.8	9.2	441.3	17.5	714.1	665.1	622.1
P14	28.3	32.0	-13.0	31.5	-11.5	589.4	443.7	24.7	342.3	41.9	617.7	475.7	23.0	373.8	39.5	689.0	546.7	445.6
P15	28.4	30.7	-7.8	29.3	-2.8	1999.5	1020.5	49.0	901.9	54.9	2890.7	1760.3	39.1	1568.8	45.7	2174.9	1198.5	1078.7
P16	28.6	34.8	-21.6	32.1	-12.3	444.4	297.9	33.0	100.8	77.3	736.9	528.5	28.3	191.8	74.0	630.1	490.1	289.8
P17	63.3	68.7	-8.5	52.1	17.7	4972.9	4103.0	17.5	3383.6	32.0	5036.2	4171.7	17.2	3435.7	31.8	5179.5	4314.7	3578.1
P18	58.3	66.0	-13.2	80.0	-37.2	8898.4	8476.3	4.7	7928.8	10.9	17802.7	16968.9	4.7	15888.8	10.8	9205.0	8789.1	8256.6
P19	244.0	252.9	-3.6	373.2	-52.9	12484.9	12224.6	2.1	10468.4	16.2	22521.8	12477.6	44.6	10841.6	51.9	12902.8	12647.1	11011.7
P20	17.4	19.1	-9.7	19.4	-11.0	4284.1	3828.5	10.6	3281.2	23.4	4301.5	3847.6	10.6	3300.6	23.3	4406.4	3952.5	3405.2
P21	28.9	36.5	-26.3	46.0	-59.5	11389.0	10655.9	6.4	7383.0	35.2	22751.1	21296.1	6.4	14766.3	35.1	11582.4	10860.4	7595.6
P22	23.9	26.3	-10.1	29.2	-22.1	5812.1	5018.0	13.7	2450.4	57.8	5836.0	5044.3	13.6	2479.6	57.5	6060.6	5264.0	2700.3
P23	19.9	20.7	-4.2	20.3	-2.0	32762.1	24403.8	25.5	22434.7	31.5	33763.2	25433.1	24.7	22487.8	33.4	32962.0	24604.8	22634.1
Avg	47.5	52.4	-14.1	61.3	-25.9	3900.7	3308.6	14.1	2742.0	35.1	5488.8	4410.1	13.7	3594.7	33.7	4105.2	3517.9	2960.3
Sum	1091.9	1206.3	-10.5	1409.0	-29.0	89715.9	76097.7	15.2	63066.9	29.7	126242.1	101431.3	19.7	82677.0	34.5	94418.9	80912.8	68087.5

Table 6: Improvements in the offline time (analysis + execution) of FineEkstazi over Ekstazi and FineSTARTS over STARTS. *t* shows the offline time (s) spent by each baseline tool. *f* and S_f[%] show the offline time (s) and the percentage time improvement by only incorporating the findings from our manual analysis. *fm* and S_{fm}[%] show the respective quantities when findings from our manual analysis plus method-level reasoning are incorporated.

PID	RetestAll	Ekstazi		FineEkstazi								STARTS		FineSTARTS							
	t	t	b[%]	f	b[%]	pp_E[%]	S_f[%]	fm	b[%]	pp_E[%]	S_{fm}[%]	t	b[%]	f	b[%]	pp_S[%]	S_f[%]	fm	b[%]	pp_S[%]	S_{fm}[%]
P1	778.0	376.3	48.4	363.4	46.7	1.7	3.4	360.6	46.4	2.0	4.2	507.1	65.2	487.4	62.6	2.5	3.9	483.4	62.1	3.0	4.7
P2	831.5	103.0	12.4	83.5	10.0	2.3	18.9	45.9	5.5	6.9	55.4	267.2	32.1	175.1	21.1	11.1	34.5	152.4	18.3	13.8	43.0
P3	878.2	103.9	11.8	87.6	10.0	1.9	15.7	70.9	8.1	3.8	31.8	136.4	15.5	112.3	12.8	2.7	17.6	96.6	11.0	4.5	29.2
P4	1004.3	65.7	6.5	72.7	7.2	-0.7	-10.7	62.2	6.2	0.4	5.4	139.2	13.9	141.8	14.1	-0.3	-1.8	105.5	10.5	3.4	24.2
P5	1121.3	142.7	12.7	121.5	10.8	1.9	14.8	99.0	8.8	3.9	30.6	170.1	15.2	151.4	13.5	1.7	11.0	116.8	10.4	4.8	31.4
P6	1260.2	523.4	41.5	504.8	40.1	1.5	3.5	403.4	32.0	9.5	22.9	502.3	39.9	476.4	37.8	2.1	5.2	391.2	31.0	8.8	22.1
P7	1446.8	334.2	23.1	293.5	20.3	2.8	12.2	260.5	18.0	5.1	22.1	754.5	52.1	677.9	46.9	5.3	10.1	554.0	38.3	13.9	26.6
P8	1503.8	212.4	14.1	230.1	15.3	-1.2	-8.3	220.6	14.7	-0.5	-3.9	1164.2	77.4	1124.9	74.8	2.6	3.4	1026.4	68.3	9.2	11.8
P9	2309.3	1337.1	57.9	1294.4	56.1	1.8	3.2	839.4	36.3	21.6	37.2	1409.1	61.0	1375.2	59.5	1.5	2.4	976.0	42.3	18.8	30.7
P10	2391.3	791.6	33.1	274.6	11.5	21.6	65.3	323.1	13.5	19.6	59.2	1271.3	53.2	1247.7	52.2	1.0	1.9	1264.9	52.9	0.3	0.5
P11	2516.3	208.3	8.3	291.6	11.6	-3.3	-39.9	180.0	7.2	1.1	13.6	477.2	19.0	447.7	17.8	1.2	6.2	321.8	12.8	6.2	32.6
P12	3113.7	263.5	8.5	207.5	6.7	1.8	21.3	108.5	3.5	5.0	58.8	268.9	8.6	210.7	6.8	1.9	21.7	111.5	3.6	5.1	58.5
P13	3731.4	106.7	2.9	175.7	4.7	-1.8	-64.7	95.0	2.5	0.3	11.0	546.8	14.7	465.8	12.5	2.2	14.8	467.8	12.5	2.1	14.5
P14	4471.4	573.7	12.8	435.1	9.7	3.1	24.2	333.5	7.5	5.4	41.9	1421.8	31.8	1254.9	28.1	3.7	11.7	1208.4	27.0	4.8	15.0
P15	4642.5	892.0	19.2	740.6	16.0	3.3	17.0	667.5	14.4	4.8	25.2	1810.9	39.0	1435.6	30.9	8.1	20.7	971.1	20.9	18.1	46.4
P16	5652.7	292.7	5.2	230.8	4.1	1.1	21.2	91.1	1.6	3.6	68.9	675.2	11.9	381.1	6.7	5.2	43.6	208.2	3.7	8.3	69.2
P17	7861.5	65.9	0.8	70.6	0.9	-0.1	-7.2	53.4	0.7	0.2	19.0	5884.9	74.9	4899.2	62.3	12.5	16.8	4108.3	52.3	22.6	30.2
P18	8491.3	8913.1	105.0	8501.0	100.1	4.9	4.6	7967.9	93.8	11.1	10.6	7756.7	91.3	7414.5	87.3	4.0	4.4	6950.0	81.8	9.5	10.4
P19	11584.1	10046.6	86.7	252.9	2.2	84.5	97.5	373.2	3.2	83.5	96.3	8116.9	70.1	7642.9	66.0	4.1	5.8	5926.4	51.2	18.9	27.0
P20	16691.6	4266.1	25.6	3810.9	22.8	2.7	10.7	3264.1	19.6	6.0	23.5	5076.9	30.4	4415.9	26.5	4.0	13.0	3804.8	22.8	7.6	25.1
P21	16847.2	11373.4	67.5	10650.8	63.2	4.3	6.4	7390.6	43.9	23.6	35.0	10856.4	64.4	10211.7	60.6	3.8	5.9	7917.2	47.0	17.4	27.1
P22	59296.0	5655.2	9.5	4881.9	8.2	1.3	13.7	2387.2	4.0	5.5	57.8	6703.0	11.3	5271.8	8.9	2.4	21.4	2437.3	4.1	7.2	63.6
P23	65432.5	1002.1	1.5	1030.3	1.6	-0.0	-2.8	53.1	0.1	1.5	94.7	32447.3	49.6	26984.5	41.2	8.3	16.8	23057.4	35.2	14.4	28.9
Avg	9732.9	2071.7	26.7	1504.6	20.9	5.9	9.6	1115.2	17.0	9.7	35.7	3841.9	41.0	3348.1	37.0	4.0	12.6	2724.2	31.3	9.7	29.2
Sum	223856.8	47649.8	21.3	34605.9	15.5	5.8	27.4	25650.7	11.5	9.8	46.2	88364.3	39.5	77006.3	34.4	5.1	12.9	62657.3	28.0	11.5	29.1

Table 7: selection rates of ML models

PID	Name	Fail-Basic^E	Fail-Basic^S	BM25^E	BM25^S
M1	Asterisk	0.13	0.33	0.15	0.20
M2	Bukkit	0.17	0.56	0.17	0.61
M3	Config	0.29	0.35	0.52	0.62
M4	Csv	0.58	0.60	0.73	0.80
M5	Lang	0.12	0.82	0.26	0.79
M6	Net	0.05	0.05	0.09	0.09
M7	Validator	0.09	0.09	0.23	0.24
M8	Gedcom4j	0.80	0.80	1.00	1.00
M9	Vectorz	0.91	0.91	0.99	0.99
M10	Zt-exec	0.90	0.90	0.92	0.92

Table 8: Comparsion of number of selected tests of our RTS tools with ML models

Project	#SHA	FineEkstazi	FineSTARTS	HyRTS	Fail-Basic^E	Fail-Basic^S	BM25^E	BM25^S
Asterisk	4	56	89	54	20	57	24	35
Bukkit	5	42	84	41	30	104	30	113
Config	8	450	618	371	392	472	696	832
Csv	18	111	147	116	134	145	170	196
Lang	36	217	996	248	518	3654	1142	3531
Net	17	57	64	55	34	34	51	51
Validator	15	119	132	109	90	90	225	240
Gedcom4j	21	495	633	711	1751	1751	2192	2192
Vectorz	20	169	1123	195	1248	1248	1368	1368
Zt-exec	14	120	128	120	148	148	154	154
Sum	158	1836	4014	2020	4365	7703	6052	8712

Table 9: Comparsion of execution time (s) of our RTS tools with ML models

Project	#SHA	FineEkstazi	FineSTARTS	HyRTS	Fail-Basic^E	Fail-Basic^S	BM25^E	BM25^S
Asterisk	4	18.09	30.17	18.09	0.50	12.63	21.90	31.29
Bukkit	5	0.41	0.93	0.41	0.29	0.93	0.34	1.29
Config	8	93.58	124.80	95.68	129.74	131.91	109.97	132.15
Csv	18	29.94	32.33	29.95	38.99	39.02	39.15	39.19
Lang	36	21.08	126.72	21.72	79.07	365.42	80.25	280.14
Net	17	113.84	117.87	113.81	4.22	4.22	186.64	186.64
Validator	15	4.06	4.80	3.26	5.13	5.13	4.17	4.33
Gedcom4j	21	36.85	41.80	39.73	96.33	96.33	97.10	97.10
Vectorz	20	4.38	9.02	5.72	12.51	12.51	12.35	12.35
Zt-exec	14	114.89	114.89	114.89	146.82	146.82	137.23	137.23
Sum	158	437.13	603.33	443.24	513.60	814.92	689.10	921.71

B MANUAL ANALYSIS

B.1 Change Type Examples

In this section, we present examples for some change types.

F7 a: Modify field holding version

Method name does not change but the imported package changes.

e.g. commons-beanutils2, SHA: 2ad4ad3c, file: src/main/java/org/apache/commons/beanutils2/BeanPredicate.java

```
+    import java.util.function.Predicate;
-    import org.apache.commons.collections4.Predicate;
-    public Predicate getPredicate() {
+    public Predicate<Object> getPredicate() {
```

F7 c: Modify utilized API interface

The field initialization does not change but the imported package changes.

e.g. commons-beanutils2, SHA: 2ad4ad3c, file: src/main/java/org/apache/commons/beanutils2/BeanPredicate.java

```
+import java.util.function.Predicate;
-import org.apache.commons.collections4.Predicate;
-private Predicate predicate;
+private Predicate<Object> predicate;
```

F13 a: Replace parameter with lambda expression

A lambda expression is passed as a parameter to a changed API, as a result, an anonymous class is created.

e.g. commons-beanutils2, SHA: 2ad4ad3c, file: src/test/java/org/apache/commons/beanutils2/BeanPredicateTestCase.java

```
-final BeanPredicate predicate =
-new BeanPredicate("stringProperty", NullPredicate.INSTANCE);
+Predicate<String> p = (s) -> s == null;
+final BeanPredicate predicate = new BeanPredicate("stringProperty", p);
```

F13 b: Modify by dependency

change caused by compiler such as constant pool propagation or synthetic methods.

e.g. commons-pool, SHA:7d033060, file: src/main/java/org/apache/commons/pool2/impl/GenericKeyedObjectPool.java

```
  lock.unlock();
  lock = keyLock.writeLock();
  lock.lock();
- if (objectDeque.getCreateCount().get() == 0 && objectDeque.getNumInterested().get() == 0) {
+ if (objectDeque.getCreateCount().get() == 0 && objectDeque.getNumInterested().get() == 0
+   && objectDeque.allObjects.isEmpty()) {
```

What changed is within outer class GenericKeyedObjectPool However, the constant pool of inner class (GenericKeyedObjectPool$ObjectDeque.class) changed.

B.2 Finding to Implementation

Table 10 presents a mapping from finding to example/implementation. The precondition is that the project after change compiles successfully. We use the ASM library. We implement a CleanCodeUtil by extending Printer to collect the content of fld, con, and mtd.

B.3 Applicability of Findings

Table 11 presents the number of occurrence of each change for each project. Note that three change types in F8 are counted together as “Change signature”. F7 is counted together with F10; F3, F6, and F9 are counted together as “No change”.

Table 10: Mapping findings to implementation

ID	Kind of Change	Implementation
F1	Add class	Check if the .class file is newly added
F2	Add instance method Remove instance method Remove static method Add constructor Add static method	When adding a new method (the method is not a test method) or deleting a new method, if current class does not have any subclass or superclass, this change can be ignored.
F3	Sort members	check if metadata (fld, con, mtd) of old version and new version equal to each other, if metadata of two versions equal to each other, this change can be ignored. Specifically, the bodies of constructor and static initializer are sorted so that the change of order of fields are ignored.
F4	Add field Remove field Add static initialized block	When adding or deleting a field/static block, if the calss has no subclass or superclass and there is no usage of this field, this change can be ignored.
F6	Rename class Rename instance method Rename static method	It is viewed as deleting a class, then adding a new class. It is viewed as deleting a method, then adding a new method.
F7	Modify field holding version Modify field initialization Modify utilized API interface	If the field is non-static, select tests affected by constructors(`<init>`); if the field is static, select tests affected by static initializer(`<clinit>`).
F8	Add exception to method Modify `throws` clause Modify method parameter	Ignore the exception and parameter type when comparing if the method changes in the new version, i.e., if the method body of old version is same as that of new version, this method is viewed as no change.
F9	Change class modifier Add final to field Adjust modifier of field	Ignore the modifier of class, field and method.
F10	Change constructor	Check if the method body of constructors(`<init>`) changes.
F11	Specialize parameter type Add/Change base class to hierarchy	No need to re-run tests if these are the only changes because bytecode of affected(dependent) class has changed.

Table 11: Applicabilty of our findings. *#F*: no. of source files with each kind of code change; *#S*: no. of revisions with each kind of change.

ID	Kind of Change	P1		P2		P3		P4		P5		P6		P7		P8		P9		P10		P11		P12		P13		P14		P15		P16		P17		P18		P19		P20		P21		P22		P23	
		#F	#S	#F	#S	#F	#S	#F	#S	#F	#S	#F	#S	#F	#S	#F	#S	#F	#S	#F	#S	#F	#S	#F	#S	#F	#S	#F	#S	#F	#S	#F	#S	#F	#S	#F	#S	#F	#S	#F	#S	#F	#S	#F	#S	#F	#S
F1	Add class	22	11	2	1	13	3	22	9	16	4	4	2	52	8	33	15	9	7	65	30	52	8	2	2	43	8	37	12	0	0	33	8	0	0	293	24	3132	20	0	0	32	9	63	3	127	6
F2	Add instance method	24	10	11	11	56	15	55	28	14	8	19	7	71	23	41	26	17	13	8	7	65	14	11	8	10	6	62	27	0	0	12	5	0	0	76	35	40	16	1	1	41	13	43	4	11	8
	Remove instance method	19	5	0	0	13	2	26	10	6	2	6	2	26	8	0	0	11	8	1	1	62	8	2	2	8	5	41	11	0	0	9	2	0	0	22	11	21	10	1	1	11	5	34	1	3	2
	Remove static method	1	1	0	0	0	0	1	1	2	1	0	0	11	5	2	2	1	1	1	1	17	7	1	1	1	1	2	2	0	0	0	0	0	0	0	0	4	3	1	1	3	3	6	1	1	1
	Add constructor	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Add static method	16	4	8	8	41	7	7	7	2	2	7	3	19	7	4	4	11	11	2	2	36	7	6	2	5	3	17	11	0	0	6	3	0	0	4	4	16	11	1	1	2	1	7	2	1	1
F4	Add field	28	5	4	4	12	3	29	19	1	1	5	2	22	8	33	21	13	12	1	1	5	5	4	2	6	3	22	13	0	0	0	0	0	0	55	38	11	8	0	0	7	4	10	2	5	5
	Remove field	2	1	0	0	2	2	5	2	4	2	3	2	8	3	2	2	8	7	0	0	6	3	5	3	0	0	14	5	0	0	0	0	0	0	0	0	10	6	0	0	1	1	3	1	1	1
	Add static initialized block	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F8	Change signature	16	3	1	1	15	3	11	10	1	1	9	3	7	4	1	1	1	1	0	0	12	3	2	2	3	3	2	2	0	0	3	2	0	0	8	5	6	3	0	0	16	6	8	1	3	3
	Change method exception	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/
	Add exception to method	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/
	Change method parameter	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/
F10	Change constructor body	78	13	4	4	31	4	27	18	14	4	24	9	90	18	22	18	33	23	12	6	110	8	50	10	14	10	61	16	0	0	18	4	0	0	35	29	51	15	0	0	15	10	52	7	8	7
F7	Change field holding version	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/
	Change field initialization	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/
	Change utilized API interface	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/
F11	Specialize parameter type	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Change base class	0	0	0	0	4	1	1	1	0	0	3	2	5	4	1	1	0	0	0	0	0	0	0	0	2	2	2	2	0	0	11	1	0	0	6	5	23	4	0	0	0	0	1	1	0	0
	No change	681	4	727	4	7	2	3	3	11	3	178	4	566	4	157	3	79	3	4	2	4	3	301	3	35	5	6	5	0	0	336	2	0	0	8	6	40	5	0	0	10	5	41	2	10	7
F3	Sort members	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/
F6	Rename instance method	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/
	Rename static method	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/
F9	Change class modifier	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/
	Make field final	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/
	Change field modifier	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/
	Methods	233	30	69	17	135	19	131	36	88	16	178	31	470	38	102	35	164	40	64	32	506	28	99	20	122	17	176	33	0	0	52	15	0	0	263	36	374	32	8	4	190	33	284	7	93	21
	Summary	1126	36	826	27	332	28	324	49	164	25	443	33	1369	41	403	43	357	48	160	50	954	35	496	26	250	25	465	43	0	0	481	21	0	0	787	50	3729	50	12	4	331	43	571	11	265	33

C PLOTS OF PRECISION AND TIME FOR 23 PROJECTS

C.1 Plots for number of selected tests and Maven time

In this section, we present the plots of 23 projects. The bar plot shows number of selected test classes for each tool, and the line plot shows maven end-to-end time for each tool. Tools include Ekstazi, FineEkstazi, STARTS, and FineSTARTS.

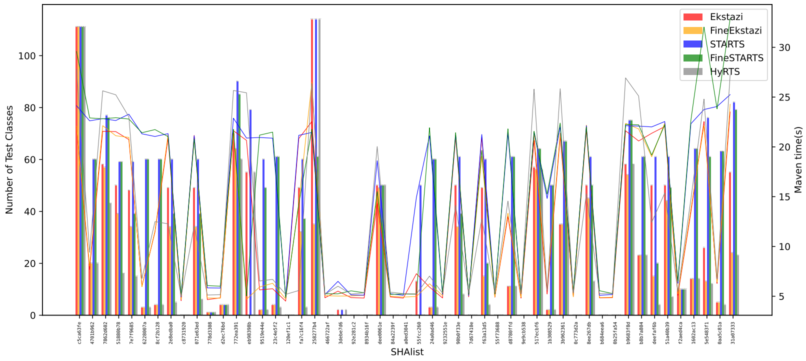

Figure 2: imaging Comparison of maven time and number of test classes among RTS tools.

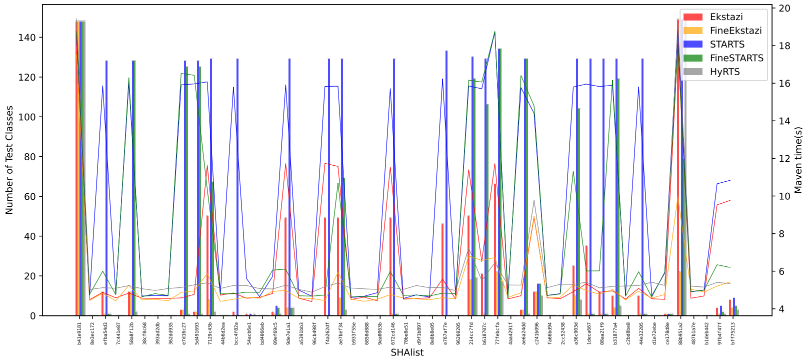

Figure 3: lang Comparison of maven time and number of test classes among RTS tools.

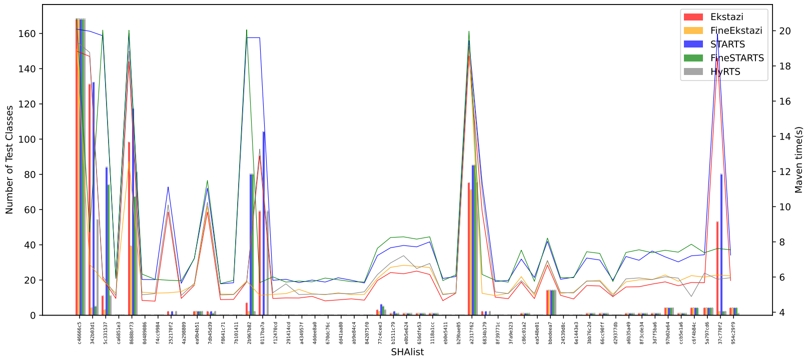

Figure 4: collections Comparison of maven time and number of test classes among RTS tools.

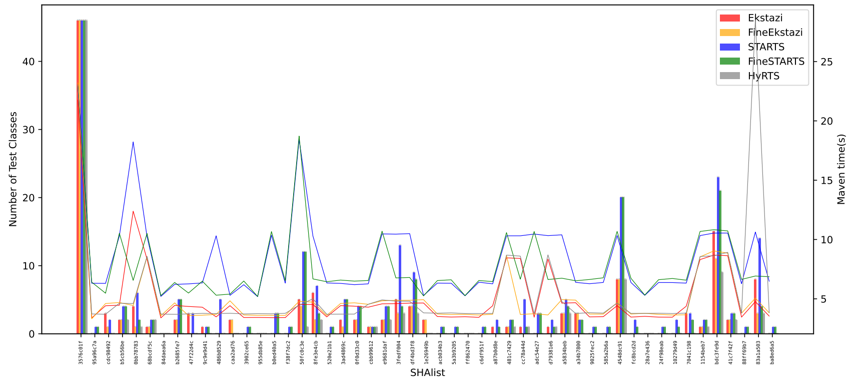

Figure 5: asterisk-Java Comparison of maven time and number of test classes among RTS tools.

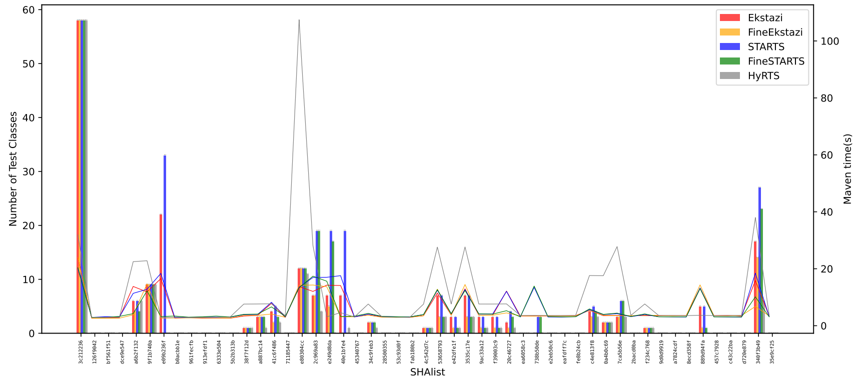

Figure 6: codec Comparison of maven time and number of test classes among RTS tools.

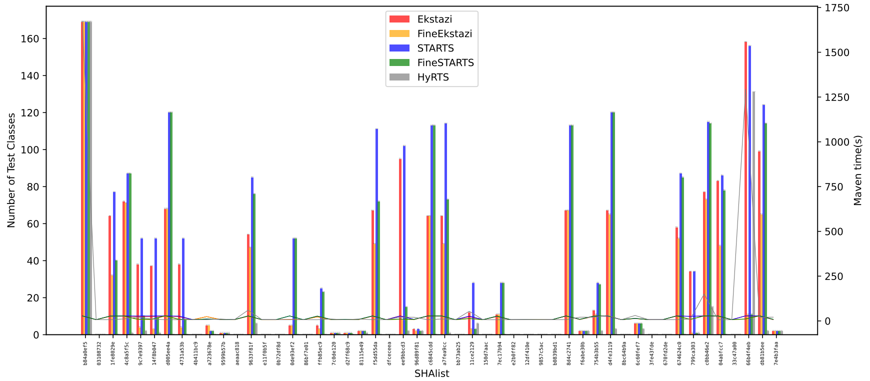

Figure 7: configuration Comparison of maven time and number of test classes among RTS tools.

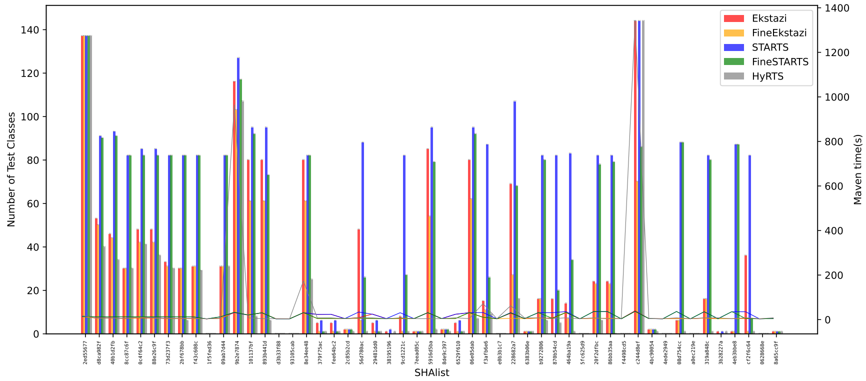

Figure 8: compress Comparison of maven time and number of test classes among RTS tools.

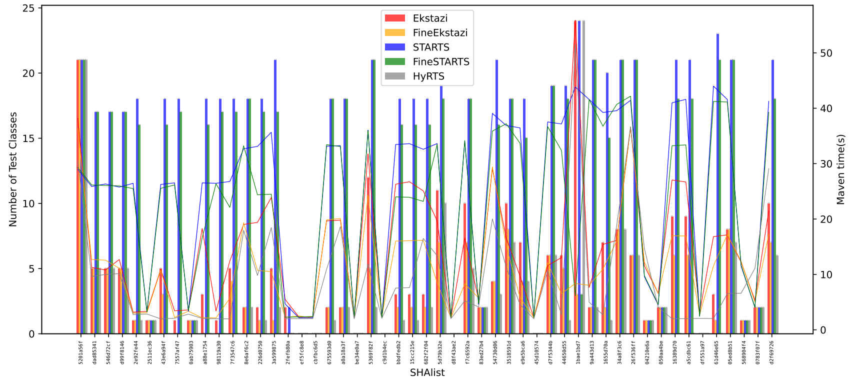

Figure 9: gerrit-events Comparison of maven time and number of test classes among RTS tools.

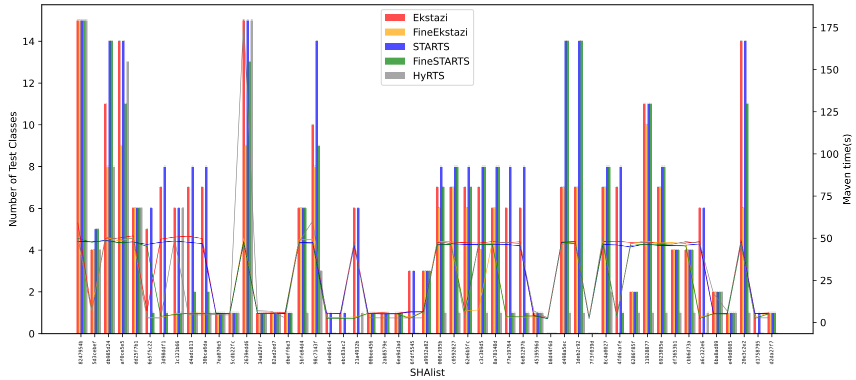

Figure 10: tabula-java Comparison of maven time and number of test classes among RTS tools.

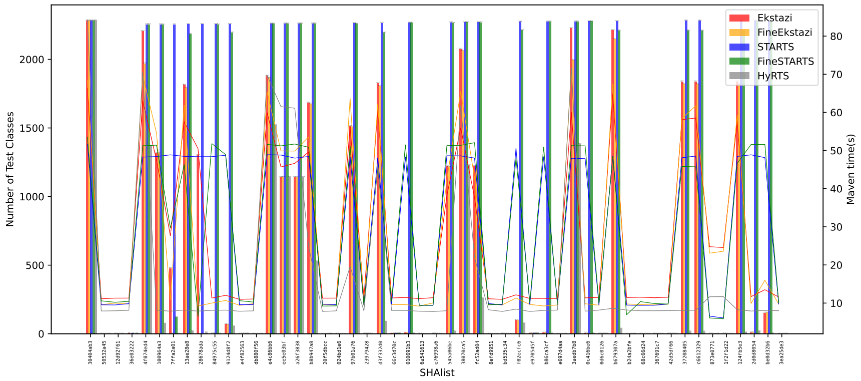

Figure 11: fastjson Comparison of maven time and number of test classes among RTS tools.

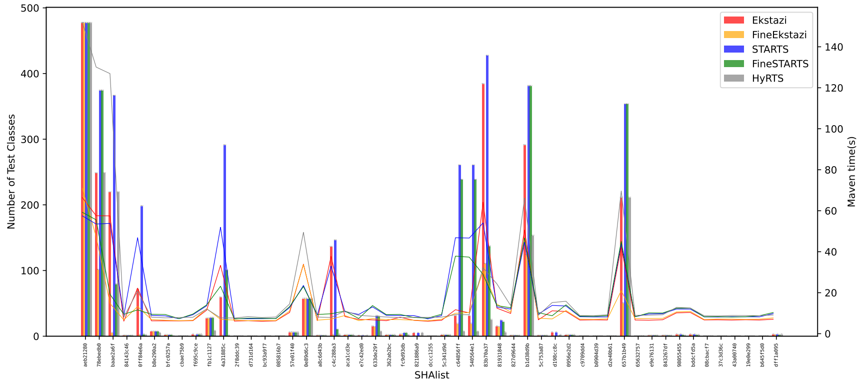

Figure 12: math Comparison of maven time and number of test classes among RTS tools.

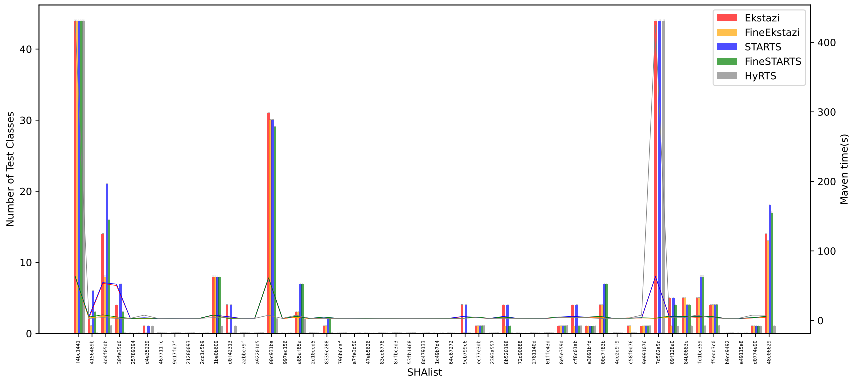

Figure 13: net Comparison of maven time and number of test classes among RTS tools.

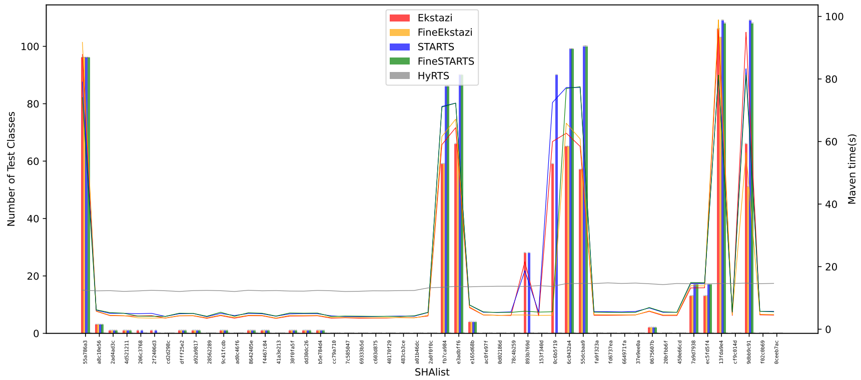

Figure 14: beanutils Comparison of maven time and number of test classes among RTS tools.

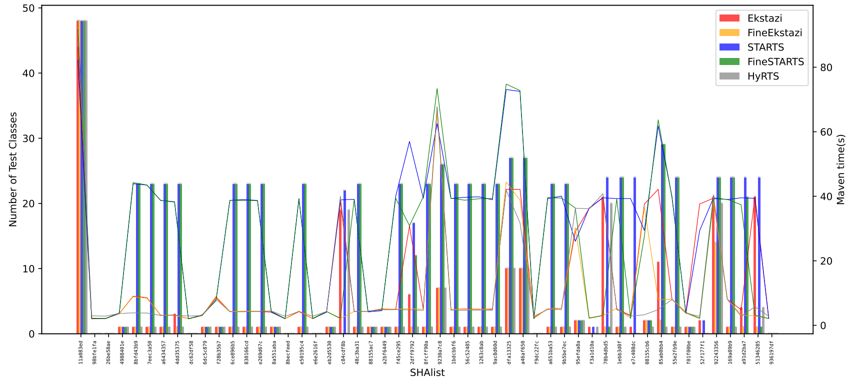

Figure 15: rxjava-extras Comparison of maven time and number of test classes among RTS tools.

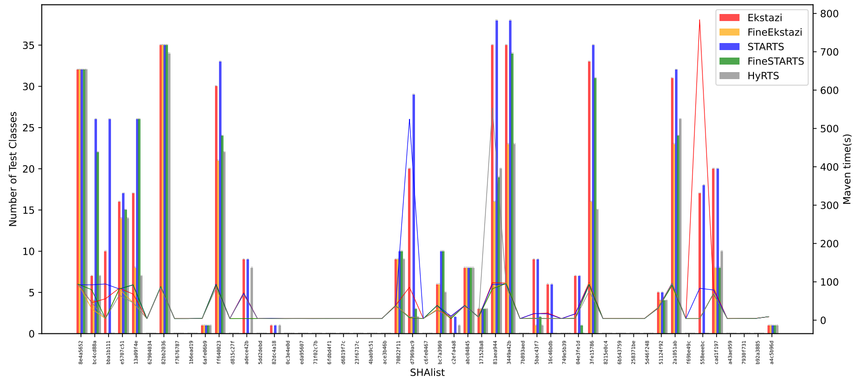

Figure 16: dbcp Comparison of maven time and number of test classes among RTS tools.

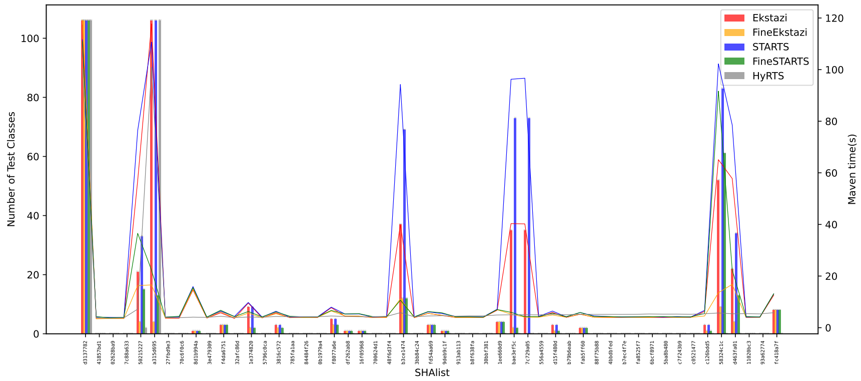

Figure 17: io Comparison of maven time and number of test classes among RTS tools.

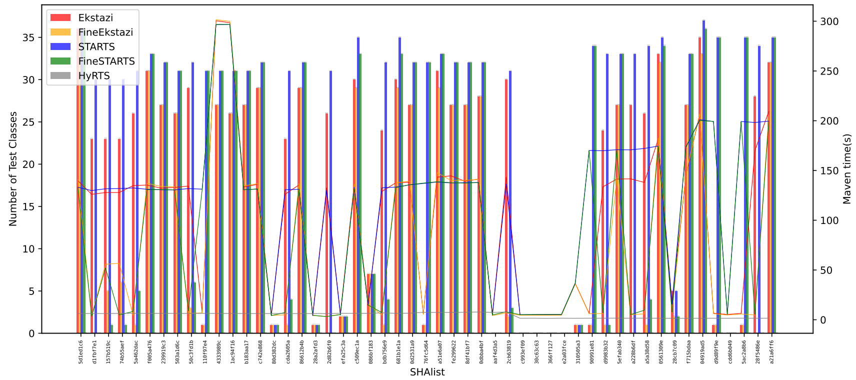

Figure 18: b.HikariCP Comparison of maven time and number of test classes among RTS tools.

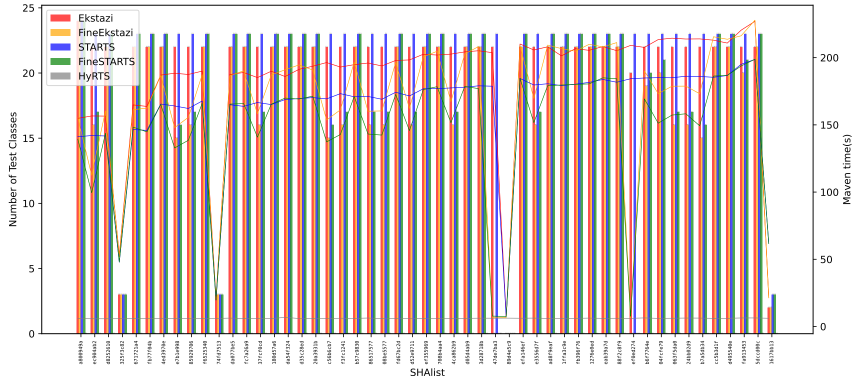

Figure 19: sdk-rest Comparison of maven time and number of test classes among RTS tools.

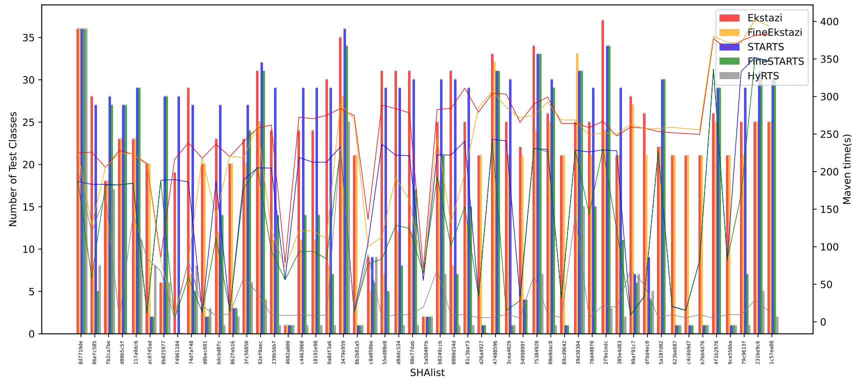

Figure 20: email-ext-plugin Comparison of maven time and number of test classes among RTS tools.

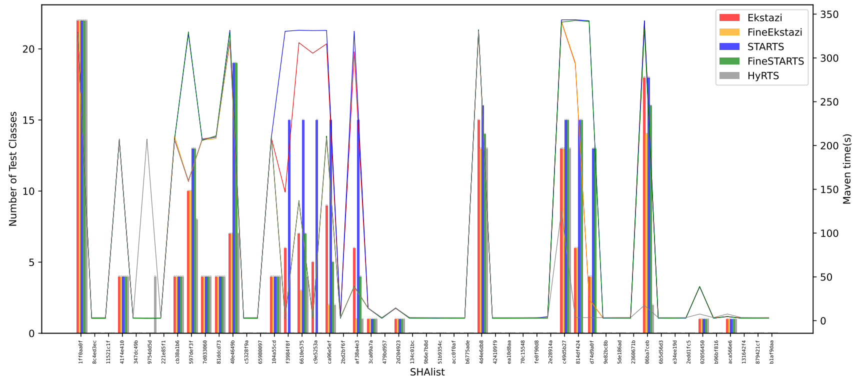

Figure 21: pool Comparison of maven time and number of test classes among RTS tools.

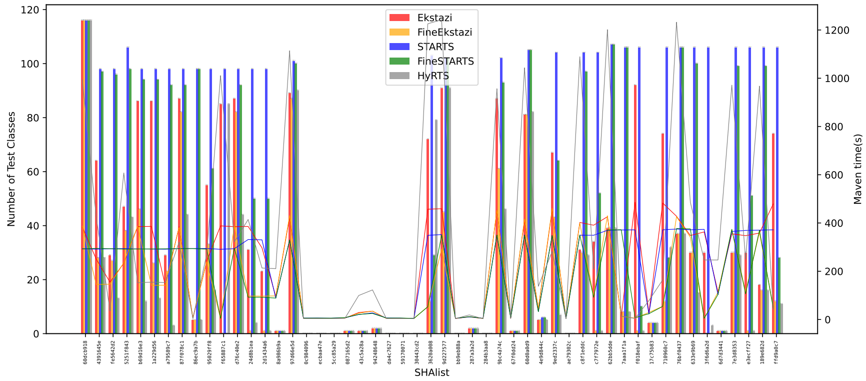

Figure 22: LogicNG Comparison of maven time and number of test classes among RTS tools.

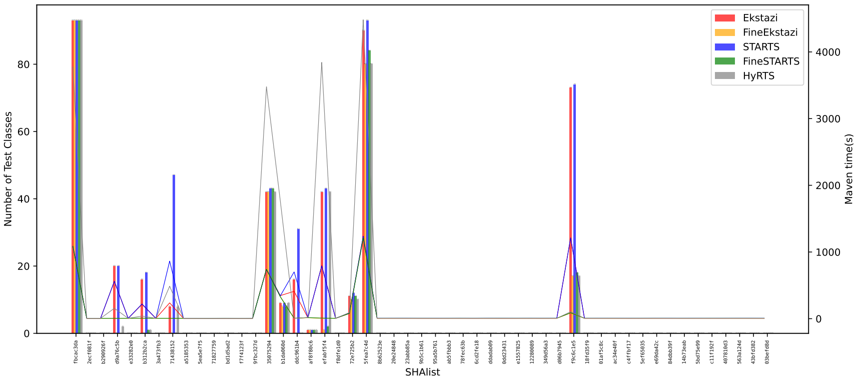

Figure 23: finmath-lib Comparison of maven time and number of test classes among RTS tools.

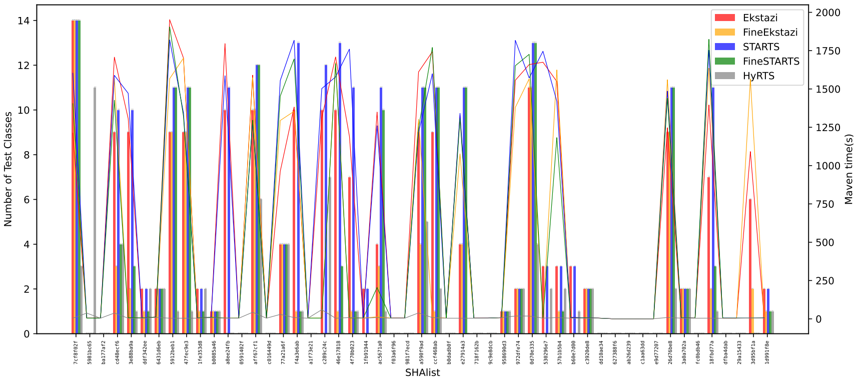

Figure 24: lmdbjava Comparison of maven time and number of test classes among RTS tools.

C.2 Plots for number of selected tests and tool AEC time

In this section, we present the plots of 23 projects. The bar plot shows number of selected test classes for each tool, and the line plot shows tool end-to-end time(analysis time + execution time + collection time) for each tool. Tools include Ekstazi, FineEkstazi, STARTS, and FineSTARTS.

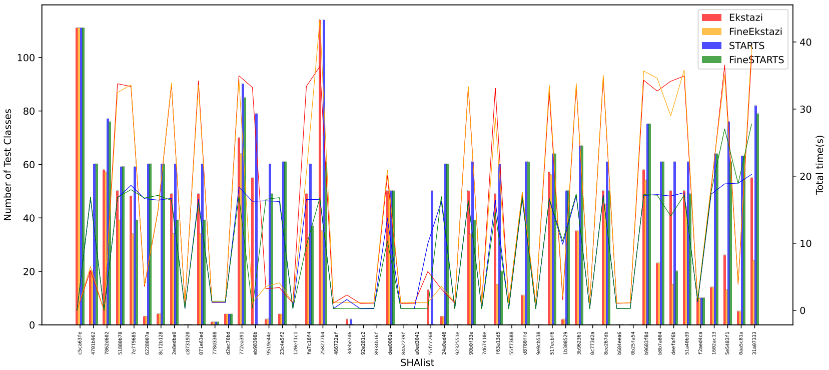

Figure 25: imaging Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

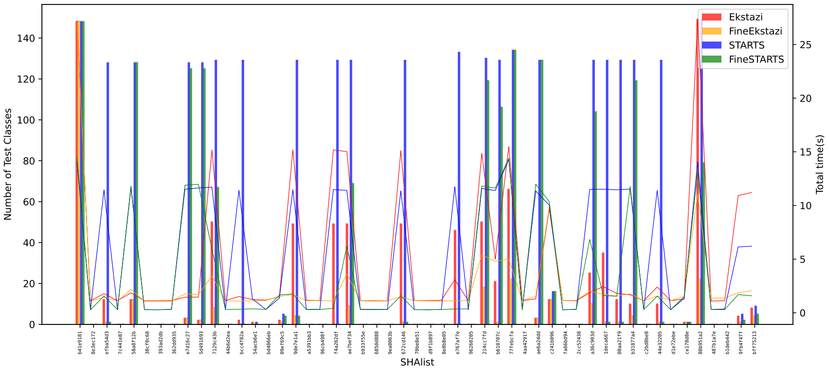

Figure 26: lang Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

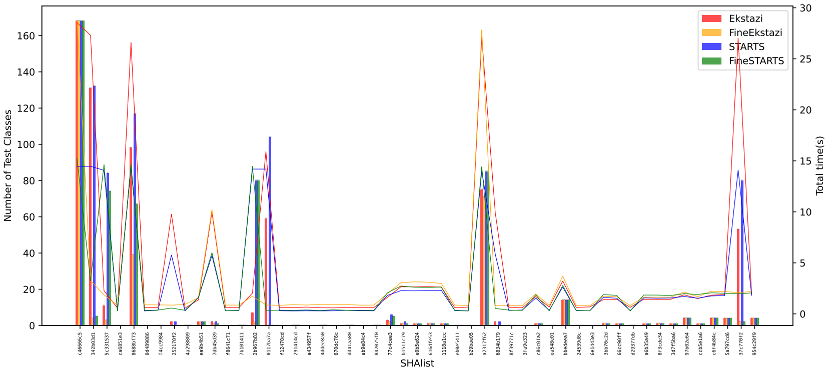

Figure 27: collections Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

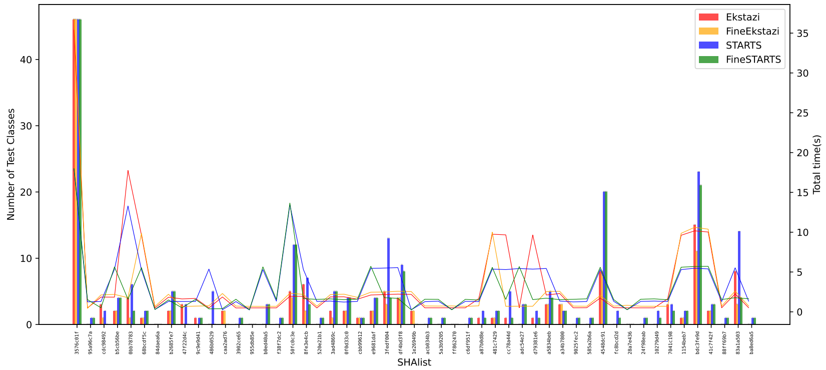

Figure 28: asterisk-Java Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

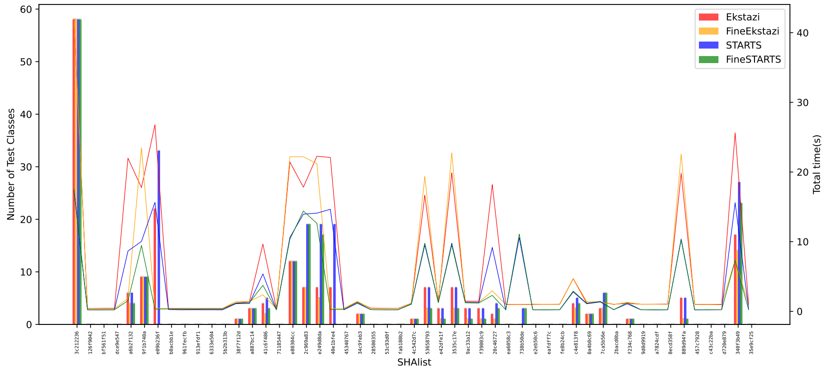

Figure 29: codec Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

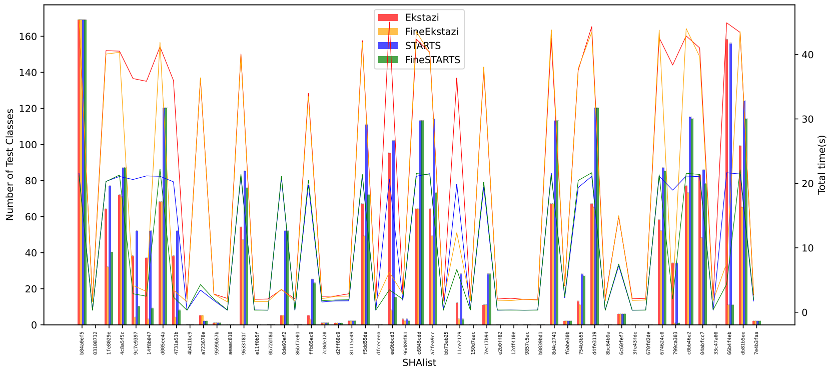

Figure 30: configuration Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

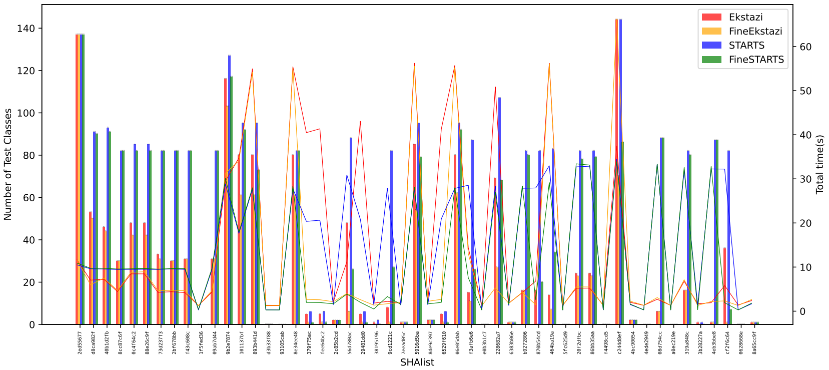

Figure 31: compress Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

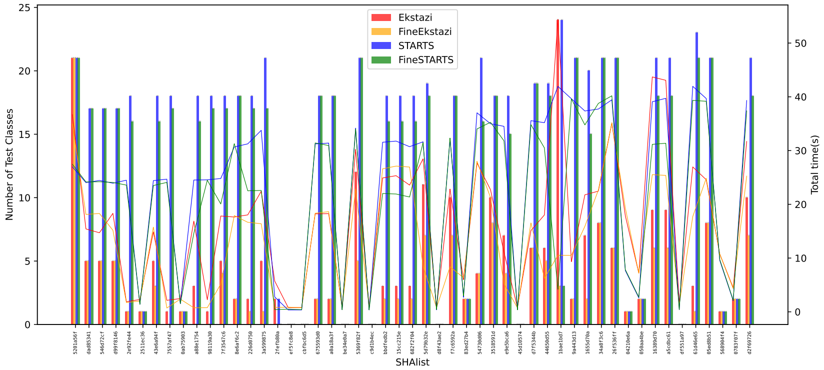

Figure 32: gerrit-events Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

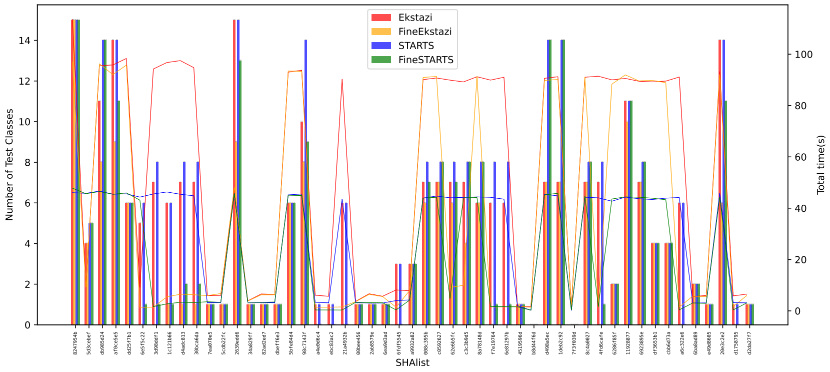

Figure 33: tabula-java Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

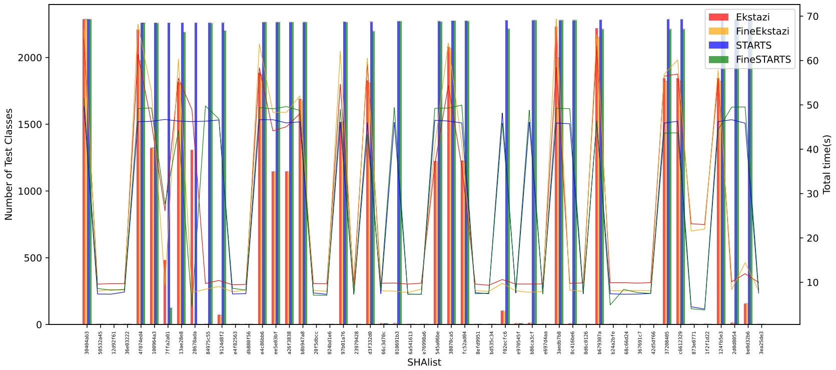

Figure 34: fastjson Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

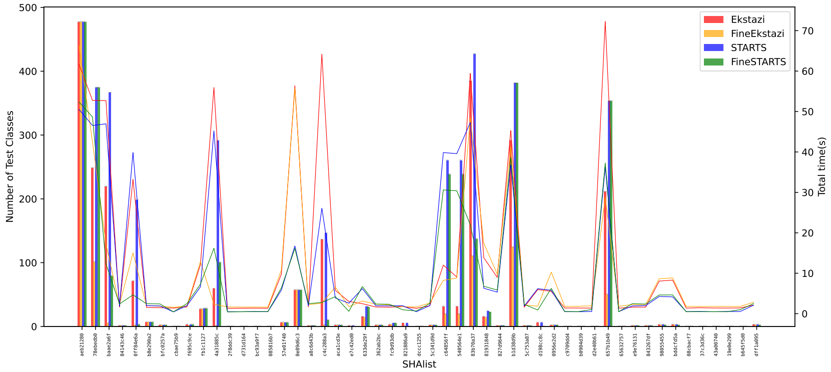

Figure 35: math Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

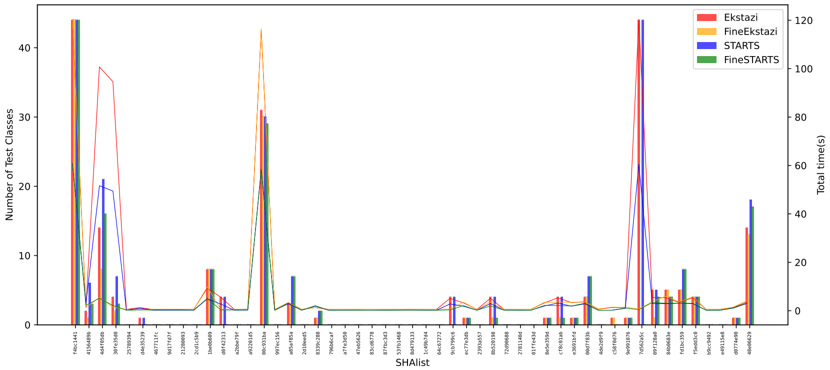

Figure 36: net Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

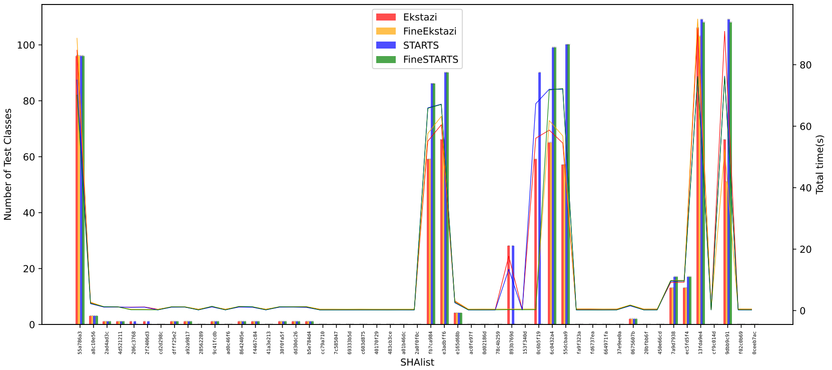

Figure 37: beanutils Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

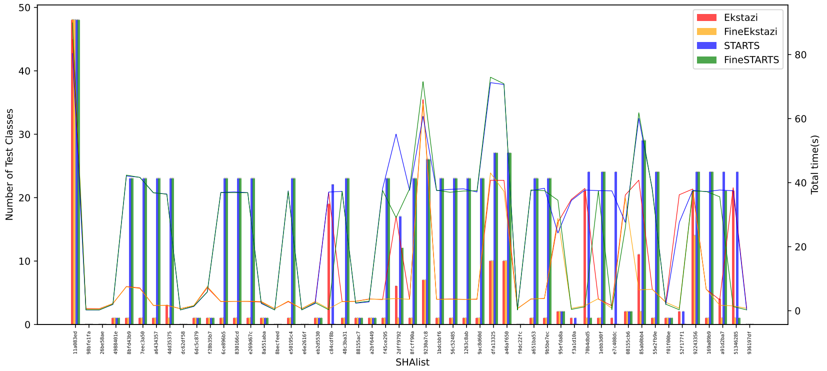

Figure 38: rxjava-extras Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

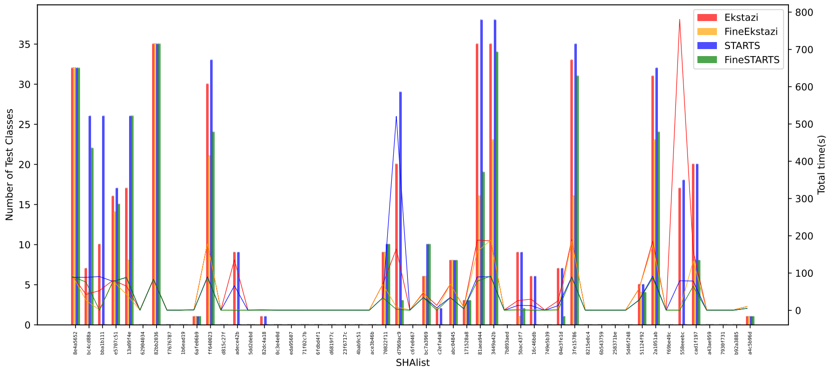

Figure 39: dbcp Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

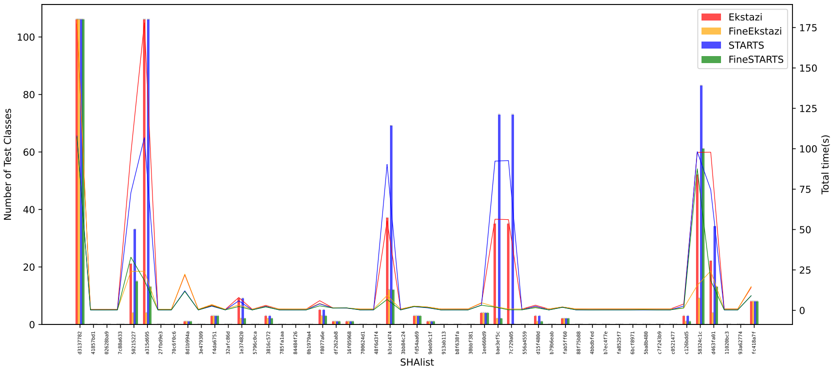

Figure 40: io Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

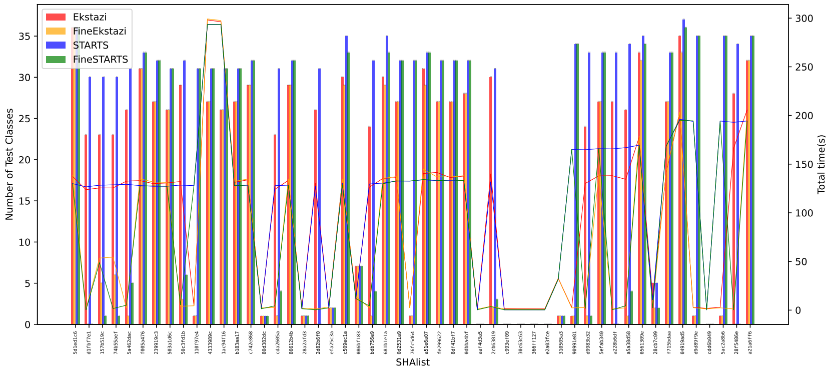

Figure 41: b.HikariCP Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

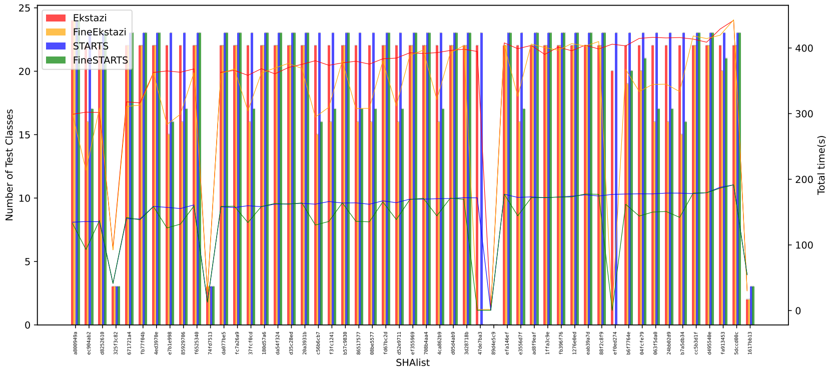

Figure 42: sdk-rest Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

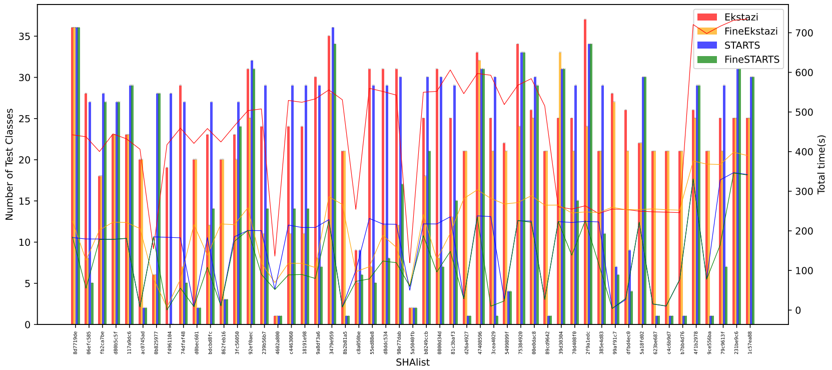

Figure 43: email-ext-plugin Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

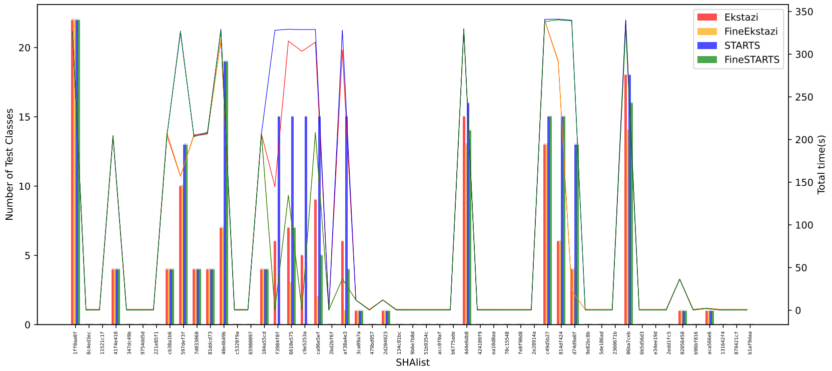

Figure 44: pool Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

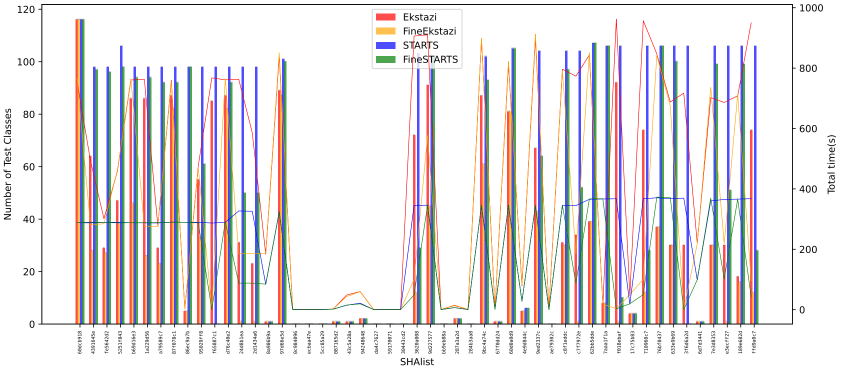

Figure 45: LogicNG Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

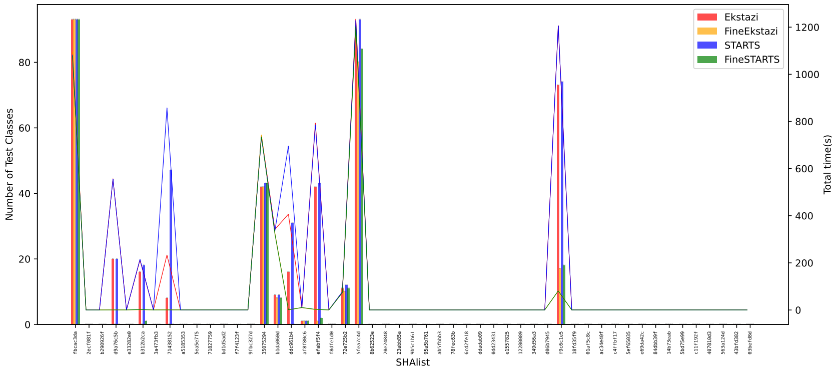

Figure 46: finmath-lib Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.

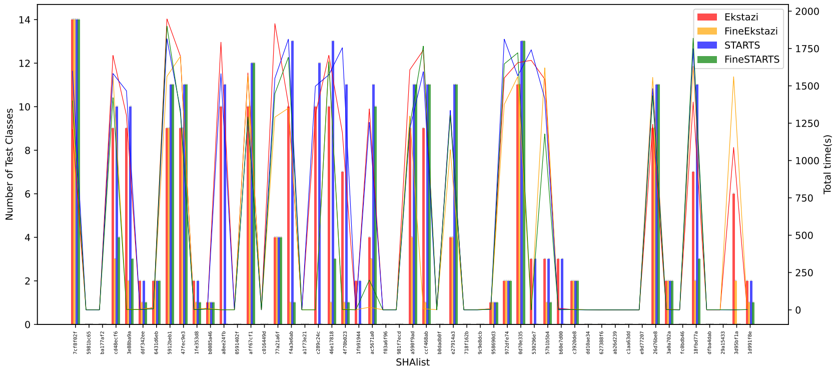

Figure 47: lmdbjava Comparison of tool analysis+execution+collection time and number of test classes among RTS tools.