Supplementary Material

In this document, we will supplement some materials about the paper draft.

First, we demo an example to illustrate our method FoSCI.

Then, in the evaluation of FoSCI by compared with the three existing methods, we give the complete measurement results.

Third, we categorize the entities uncovered by execution traces in our experiments, and explain each category with examples.

1 A Running Example of JPetstore

We use an example of *JPetstore*[1] (Monolithic version 6.0.2) to demonstrate our method FoSCI.

1.1 Step 1: Representative Execution Trace Extraction

We instrument the software system by using Kieker. After executing the test cases, monitoring logs were generated and stored in *Log files*. Table 1 illustrates a small piece of the collected logs. An *Execution Trace* can be modeled as $tr : \langle(m_7, m_6), (m_6, m_1), (m_6, m_2), (m_6, m_3), (m_6, m_4), (m_6, m_5)\rangle$.

Processing *Log files*, we obtained an *Execution Trace Set* O_{tr}, including 47 execution traces. By employing Algorithm 1, 15 execution traces form the *Representative Trace Set* R_{tr}.

1.2 Step2: Entity Identification

For simplicity we map the involved *Class* Names into *Class* IDs, as shown in Table 2. We will use these Class IDs through the whole running case.

From R_{tr}, we get a set of *ctr*, illustrated in Table 3. From this table, we obtain T_i for each c_i. For instance of Tc_0 and Tc_2:

$$Tc_0 = \{ctr_0, ctr_1, ctr_2, ctr_9\}$$
$$Tc_1 = \{ctr_0, ctr_1, ctr_2, ctr_5, ctr_9, ctr_{10}\}$$

Then initialize the initial functional atom fa_i according to c_i: $fa_i = \{c_i\}$. Then we get in total 16 functional atoms including $fa_0, fa_1, ..., fa_{15}$ with corresponding $Tc_0, Tc_1, ..., Tc_{15}$. Use the hierarchical clustering algorithm described in section 2.4.1 of our paper by computing $f_{jaccard}(fa_i, fa_j)$ and *newDiff*. For example in the first loop of the clustering:

$$f_{jaccard}(fa_0, fa_1) = \frac{Tc_0 \cap Tc_1}{Tc_0 \cup Tc_1} = \frac{4}{6}$$
$$\textit{newDiff} = min(|Tc_0 \cup Tc_1| - |Tc_0 \cap Tc_1|, |Tc_0 \cup Tc_2| - |Tc_0 \cap Tc_2|, ..., |Tc_{14} \cup Tc_{15}| - |Tc_{14} \cap Tc_{15}|) = 0.$$

1. https://github.com/mybatis/jpetstore-6

Repeat the clustering until *newDiff* is bigger than *diff* ($diff = 3$). We finally get 8 function atoms for `JPetstore`:

$$fa_1 = \{c_3\}$$
$$fa_2 = \{c_4\}$$
$$fa_5 = \{c_9\}$$
$$fa_7 = \{c_{12}\}$$
$$fa_3 = \{c_5, c_6\}$$
$$fa_4 = \{c_7, c_8, c_{15}\}$$
$$fa_0 = \{c_0, c_1, c_2\}$$
$$fa_6 = \{c_{10}, c_{11}, c_{13}, c_{14}\}$$

We proceed search-based functional atom grouping to cluster functional atoms according to the approach and in section 2.4.3 of our paper. The detail parameters of the used NSGA-II algorithm are same with those in section 4.3 of our paper. Assume that $K = 4$ service candidates is expected to be identified in this example. When initializing populations with individuals, we randomly generate K-partitions (k=4) of *FA* as individuals.

In the following we will describe how to compute fitness values for an individual, An individual is a partition of *FA*:

$$FA = \{fa_1, fa_2, fa_3, fa_4, fa_5, fa_6, fa_7\}.$$
$$P = \{fa_0, fa_5\}, \{fa_1\}, \{fa_4, f_6\}, \{f_2, f_4, f_7\}\}.$$

Denote the i-th subset of P as $cluster_i$. Then $cluster_0 = \{fa_0, fa_5\}$, $cluster_1 = \{fa_1\}$. There is no edge between fa_0 and fa_5 in terms of structural information from R_{tr}. Thus, for $cluster_0$, its *structural intra-connectivity* is $\frac{0}{2_2} = 0$. Between $cluster_0$ and $cluster_1$, there is method call relation, (c_0,c_3), between $fa_0 \in cluster_0$ and $fa_1 \in cluster_1$, two edges ($fa_0 \rightarrow fa_1, fa_1 \rightarrow fa_0$) exist. Thus *structural inter-connectivity* between $cluster_0$ and $cluster_1$ is $\frac{2}{2 \times 2 \times 1} = \frac{1}{2}$.

The conceptual term sets of fa_0, fa_1 and fa_5 are {*Catalog, Product*},{*Category*} *and* {*Account*} respectively. In terms of the conceptual information, get *conceptual intra-connectivity* is 0. Thus *conceptual inter-connectivity* between $cluster_0$ and $cluster_1$ is 0 because of no edges from $cluster_0$ to $cluster_1$, since the intersection of conceptual term is empty. In this way, the fitness values $F(P)$ for individual P are obtained.

After finishing the NSGA-II algorithm, we choose the knee point from the pareto frontier as the best solution:

$$P_{knee} = \{\{fa_0, fa_1, fa_2, fa_7\}, \{fa_6\}, \{fa_3, fa_5\}, \{fa_4\}\}.$$

Extend P_{knee} with the class set of fa_i. Finally, the class entities for the 4 service candidates are:

$$E_{ser_1} = \{c_0, c_1, c_2, c_3, c_4, c_{12}\}$$
$$E_{ser_2} = \{c_{10}, c_{11}, c_{13}, c_{14}\}$$
$$E_{ser_3} = \{c_5, c_6, c_9\}$$

TABLE 1: A piece of the collected log for `JPetstore`

ID	Method	ThreadID	Eoi	Ess
m_1	org.mybatis.jpetstore.domain.Product.init()	22010695	4	3
m_2	void org.mybatis.jpetstore.domain.Product.setProductId(String)	22010695	5	3
m_3	void org.mybatis.jpetstore.domain.Product.setName(String)	22010695	6	3
m_4	void org.mybatis.jpetstore.domain.Product.setDescription(String)	22010695	7	3
m_5	void org.mybatis.jpetstore.domain.Product.setCategoryId(String)	22010695	8	3
m_6	java.util.List org.mybatis.jpetstore.service.CatalogService.searchProductList(String)	22010695	3	2
m_7	ForwardResolution org.mybatis.jpetstore.web.actions.CatalogActionBean.searchProducts()	22010695	2	1

TABLE 2: ClassID and ClassName for `JPetstore`

Class ID	Class Name
c_0	org.mybatis.jpetstore.web.actions.CatalogActionBean
c_1	org.mybatis.jpetstore.service.CatalogService
c_2	org.mybatis.jpetstore.domain.Product
c_3	org.mybatis.jpetstore.domain.Category
c_4	org.mybatis.jpetstore.domain.Item
c_5	org.mybatis.jpetstore.web.actions.AccountActionBean
c_6	org.mybatis.jpetstore.domain.Account
c_7	org.mybatis.jpetstore.domain.Cart
c_8	org.mybatis.jpetstore.domain.CartItem
c_9	org.mybatis.jpetstore.service.AccountService
c_{10}	org.mybatis.jpetstore.web.actions.OrderActionBean
c_{11}	org.mybatis.jpetstore.service.OrderService
c_{12}	org.mybatis.jpetstore.domain.Sequence
c_{13}	org.mybatis.jpetstore.domain.Order
c_{14}	org.mybatis.jpetstore.domain.LineItem
c_{15}	org.mybatis.jpetstore.web.actions.CartActionBean

$$E_{ser_4} = \{c_7, c_8, c_{15}\}$$

1.3 Step3: Interface Class Identification

In JPetstore, the identified operations O_i and interface classes I_i are public method members and public classes with a prefix of `org.mybatis.jpetstore.web.actions`.

1.4 Summary of the Example

The identified service candidates SC, containing entities E_{ser}, interface classes I and operations O, are shown in Table 4.

From Table 4, we observe that the produced 4 service candidates are separately dedicated to the business capabilities of "Catalog service", "Order service", "Account service" and "Cart service". Intuitively, even though the classes appear in the same code package of original monolith, the classes are separately grouped into different service candidates in terms of functional logic. It shows that our method based on execution traces help to identify service candidates around business functionality.

2 The Evaluation

The objective of the evaluation is to justify whether FoSCI can produce effective service candidates. In terms of *Independence of Evolvability, Independence of Functionality*, and *Modularity*, we compare FoSCI with the baseline methods (LIMBO, WCA, MEM).

We conducted comparison on six subjects: Springblog, Solo, JForum, Apache Roller, Agilefant, and Xwiki-platform. The baseline methods require configuring the number N of service candidates to be identified. To address the comparison rigorously, we configured five different settings by using $N-2$, $N-1$, N, $N+1$ and $N+2$. Thus we conducted $5 \times 6 = 30$ group experiments in our evaluation.

In terms of *Independence of Evolvability*, Table 6 shows the evaluation results by measuring *ICF, ECF* and *REI* metrics. As indicated by *ICF, ECF* and *REI*, FoSCI can split frequently co-changed entities of a monolith into a service candidate while placing infrequently co-changed entities into different candidates. In contrast, for service candidates generated by LIMBO, WCA and MEM, changes across service boundaries are more common than those within services. Consequently, these services are unable to evolve independently. In conclusion, FoSCI can generate service candidates with significantly greater Independence of Evolvability than the other three baseline methods.

In terms of *Independence of Functionality*, Table 7 shows the evaluation results by measuring *IFN, CHM* and *CHD* metrics. As indicated by *IFN*, FoSCI and LIMBO are more capable of splitting the business responsibilities of a monolith into different services candidates, while MEM and WCA still heavily mix the functions together into fewer services, and even one very large service. Furthermore, in terms of *CHM* and *CHD*, FoSCI performs better than LIMBO excluding WCA and MEM. For WCA and MEM, their *CHM* and *CHD* of WCA (and MEM) may be better than those of FoSCI when they decompose a monolith into one super larger service (i.e., still a "monolith") and lots of tiny services. This phenomenon will be more obvious when decomposing a large-scale software system, such as the Xwiki-platform.

In terms of *Modularity*, Table 8 shows the measurement results by measuring *SMQ* and *CMQ* metrics. Both structural modularity (*SMQ*) and conceptual modularity (*CMQ*) measurements suggest that FoSCI can split a monolith into service candidates with considerably better modularity. Entities inside service candidates by FoSCI tend to function more coherently than those by other three methods.

To sum up, it is evident that our FoSCI can produce efficient service candidates that can consistently exhibit reasonable functionality, modularity and evolution characteristics.

3 The Examples of Uncovered Categories

To figure out why some classes in our experiments were not covered by execution traces, we manually inspected the source code of four[2] investigated projects: Springblog, Solo, JForum, and Apache Roller. We categorize uncovered classes into 8 types, as shown in Table 5. Since no single

2. Due to the manual labor, large project, such as Agilefant and Xwiki-platform, were not included.

ctr	class set	ctr	class set	ctr	class set
ctr_0	$\{c_0, c_1, c_2, c_3\}$	ctr_5	$\{c_1, c_5, c_6, c_9\}$	ctr_{10}	$\{c_1, c_3, c_4, c_5, c_6, c_7, c_8, c_{15}\}$
ctr_1	$\{c_0, c_1, c_2\}$	ctr_6	$\{c_5, c_6, c_7, c_{10}, c_{11}, c_{12}, c_{13}, c_{14}, c_{15}\}$	ctr_{11}	$\{c_4, c_5, c_6, c_7, c_8, c_{10}, c_{13}, c_{14}, c_{15}\}$
ctr_2	$\{c_0, c_1, c_2, c_4\}$	ctr_7	$\{c_4, c_7, c_8, c_{15}\}$	ctr_{12}	$\{c_5, c_6\}$
ctr_3	$\{c_4, c_5, c_6, c_7, c_8\}$	ctr_8	$\{c_4, c_7, c_8, c_{15}\}$	ctr_{13}	$\{c_2, c_4, c_5, c_6, c_{10}, c_{11}, c_{13}, c_{14}\}$
ctr_4	$\{c_5, c_6\}$	ctr_9	$\{c_0, c_1, c_2, c_4\}$	ctr_{14}	$\{c_5, c_6, c_{10}, c_{11}, c_{13}\}$

TABLE 3: ctr obtained from R_{tr} of `JPetstore`

TABLE 4: The service candidates extracted from *JPetstore*

SC	E_{ser}	*I*	*O*
SC_0	org.mybatis.jpetstore.domain.Category org.mybatis.jpetstore.service.CatalogService org.mybatis.jpetstore.web.actions.CatalogActionBean org.mybatis.jpetstore.domain.Product org.mybatis.jpetstore.domain.Item org.mybatis.jpetstore.domain.Sequence	CatalogActionBean	ForwardResolution viewCategory() ForwardResolution searchProducts() ForwardResolution viewProduct() ForwardResolution viewItem()
SC_1	org.mybatis.jpetstore.domain.LineItem org.mybatis.jpetstore.web.actions.OrderActionBean org.mybatis.jpetstore.service.OrderService org.mybatis.jpetstore.domain.Order	OrderActionBean	Resolution newOrder() boolean isConfirmed() org.mybatis.jpetstore.domain.Order getOrder() Resolution newOrderForm() void clear() void setOrderId(int) Resolution viewOrder() Resolution listOrders()
SC_2	org.mybatis.jpetstore.domain.Cart org.mybatis.jpetstore.domain.CartItem org.mybatis.jpetstore.web.actions.CartActionBean	CartActionBean	void clear() Resolution removeItemFromCart() Resolution updateCartQuantities() org.mybatis.jpetstore.domain.Cart getCart() Resolution addItemToCart()
SC_3	org.mybatis.jpetstore.service.AccountService org.mybatis.jpetstore.web.actions.AccountActionBean org.mybatis.jpetstore.domain.Account	AccountActionBean	boolean isAuthenticated() String getUsername() void setPassword(String) void setUsername(String) Resolution newAccount() org.mybatis.jpetstore.domain.Account getAccount() Resolution signoff() void clear()

TABLE 5: Categories of uncovered classes

Type	Description
NoBehavior	A class only having member variables or constructors, but having no member methods.
Exception	A class responsible for exceptions.
3rdPartyService	A class accessing third party services.
Constant	A class only having static variables which are initialized with constants.
Abstraction	An abstract super-class.
ProvidedService	A class responsible for providing APIs or web-services for external systems.
Extension	A class for extending modules.
Other	A class only used by developers, or other reasons.

system presents all types, here we will use three projects—Springblog, Solo and JForum—as examples to explain these types.

For the project Springblog:

1) *NoBehavior* classes. They are in `models.dto.*`, such as `models.dto.PostIdTitleDTO`.

2) *Exception* classes. They are included in `blog.error.*`. These classes are not included because only successful execution paths are taken into consideration.

3) *3rdPartyService* classes. Located in `support.web.*`, these classes are mainly for accessing Markdown (`support.web.MarkdownService`, `support.web.FlexmarkMarkdownService`), YouTube linking (`support.web.extensions.YoutubeLinkTransformer`), and syntax highlighting (`support.web.SyntaxHighlightService`).

4) *Constant* classes. `blog.Constants` defines static members for enumerating configuration or type information.

5) *Abstraction* classes. `models.BaseModel` is the abstract super class of specific models such as `models.User`, `models.Like` and `models.Post`.

For the project Solo:

6) *ProvidedService* classes. They are in package `solo.api.*`, and provide APIs to allow other external systems or programs managing blogs of Solo system.

For the project JForum:

7) *Extension* classes. JForum enables using different database for extensibility. `dao.oracle.*`, `dao.mysql.*` and `dao.sqlserver2000.*` access data in various Database Management Systems.

8) *Other* classes. They are uncovered classes which belong to none of the above seven types.

TABLE 6: Measurement results of *ICF*, *ECF* and *REI*

Subject	LIMBO			WCA			MEM			FoSCI		
	ICF	ECF	REI	ICF	ECF	REI	ICF	ECF	REI	ICF	ECF	REI
Springblog	0.0856	0.1417	1.6554	0.0270	0.3966	14.6701	0.0940	0.5154	5.4829	0.3795	0.1304	0.3435
	0.0719	0.1295	1.8000	0.0228	0.3689	16.2013	0.1584	0.4575	2.8883	0.4347	0.1488	0.3423
	0.0691	0.2455	3.5554	0.0228	0.3689	16.2013	0.1424	0.5250	3.6878	0.4037	0.1312	0.3251
	0.0692	0.2318	3.3518	0.0228	0.3689	16.2013	0.1300	0.5773	4.4410	0.3645	0.1559	0.4277
	0.0654	0.2141	3.2748	0.0288	0.3253	11.2890	0.1736	0.5289	3.0477	0.3229	0.1415	0.4382
Solo	0.1780	0.1718	0.9650	0.0706	0.1960	2.7745	0.0968	0.7640	7.8958	0.2460	0.1877	0.7627
	0.1632	0.1706	1.0455	0.0288	0.3105	10.7822	0.1479	0.6733	4.5535	0.2789	0.2133	0.7646
	0.1687	0.1712	1.0149	0.0288	0.3105	10.7822	0.1314	0.7106	5.4096	0.3369	0.1669	0.4954
	0.1682	0.1781	1.0584	0.0252	0.5205	20.6533	0.1186	0.7386	6.2265	0.2311	0.1571	0.6797
	0.1687	0.1764	1.0454	0.0252	0.5205	20.6533	0.1082	0.7616	7.0381	0.4001	0.1956	0.4888
Agilefant	0.2214	0.2310	1.0433	0.0509	0.0366	0.7197	0.0264	0.8964	34.0120	2.5298	0.2389	0.0944
	0.2256	0.2319	1.0280	0.0452	0.1422	3.1442	0.0239	0.9060	37.8620	1.8916	0.2528	0.1336
	0.2301	0.2345	1.0190	0.0452	0.1422	3.1442	0.0219	0.9144	41.7175	1.9864	0.2314	0.1165
	0.2264	0.2306	1.0183	0.0407	0.1291	3.1725	0.0203	0.9211	45.4483	2.0392	0.2471	0.1212
	0.2164	0.2216	1.0241	0.0375	0.1180	3.1493	0.0189	0.9268	49.1187	1.5147	0.2254	0.1488
Apache Roller	0.7703	0.7920	1.0282	0.2777	0.6959	2.5064	0.2619	0.9338	3.5661	1.1587	0.9099	0.7853
	0.7714	0.7953	1.0310	0.2182	0.7173	3.2883	0.2444	0.9386	3.8407	0.9236	0.8128	0.8800
	0.7631	0.7910	1.0366	0.2182	0.7173	3.2883	0.2291	0.9423	4.1125	1.0313	0.8567	0.8307
	0.7646	0.7935	1.0378	0.1703	0.7386	4.3377	0.2157	0.9674	4.4855	0.8970	0.7920	0.8829
	0.7816	0.8063	1.0317	0.1596	0.7902	4.9502	0.2035	0.9699	4.7652	0.7591	0.7647	1.0074
Xwiki-platform	0.0083	0.0072	0.8632	0.0402	0.1036	2.5760	0.0621	0.6807	10.9657	0.1461	0.0055	0.0376
	0.0081	0.0071	0.8764	0.0394	0.1016	2.5772	0.0609	0.6869	11.2879	0.1045	0.0060	0.0571
	0.0081	0.0071	0.8797	0.0387	0.1189	3.0747	0.0597	0.6930	11.6101	0.1059	0.0055	0.0516
	0.0083	0.0071	0.8609	0.0379	0.1543	4.0688	0.0714	0.6799	9.5243	0.1038	0.0058	0.0554
	0.0086	0.0071	0.8338	0.0362	0.1517	4.1904	0.0701	0.6858	9.7888	0.1190	0.0052	0.0441
P-value			$>, ***$			$>, ***$			$>, ***$			–

$>$ means the value of former (LIMBO,WCA,MEM) is statistically bigger than the latter (FoSCI), and $<$ vice versa.
$=$ means there is no statistical difference between the two group of results.
*** means the 0.001 significant level (p-value < 0.001).
** means the 0.01 significant level (p-value < 0.01).
* means the 0.05 significant level (p-value < 0.05).
Note: JForum is excluded from this table, since the co-change revision data was missing, as shown in Table **??**.

TABLE 7: Measurement results of *IFN*, *CHM* and *CHD*

Subject	IFN				CHM				CHD			
	LIMBO	WCA	MEM	FoSCI	LIMBO	WCA	MEM	FoSCI	LIMBO	WCA	MEM	FoSCI
Springblog	2.600	13.000	4.000	2.600	0.494	0.297	0.656	0.681	0.472	0.308	0.556	0.632
	2.167	13.000	3.000	2.167	0.512	0.297	0.747	0.573	0.468	0.308	0.670	0.558
	2.167	13.000	3.000	2.167	0.512	0.297	0.747	0.589	0.468	0.308	0.669	0.607
	2.167	13.000	3.000	2.167	0.512	0.297	0.747	0.552	0.468	0.308	0.670	0.595
	2.167	13.000	2.400	1.625	0.512	0.297	0.705	0.678	0.468	0.308	0.698	0.666
Solo	3.286	23.000	23.000	5.750	0.823	0.847	0.847	0.864	0.452	0.449	0.449	0.508
	2.875	23.000	23.000	4.600	0.849	0.847	0.847	0.880	0.526	0.449	0.449	0.595
	2.556	23.000	23.000	4.600	0.868	0.847	0.847	0.894	0.579	0.449	0.449	0.648
	2.300	23.000	23.000	3.286	0.882	0.847	0.847	0.907	0.580	0.449	0.449	0.615
	2.091	23.000	23.000	3.286	0.926	0.847	0.847	0.943	0.688	0.449	0.449	0.761
JForum	3.857	13.500	27.000	4.500	0.531	0.758	0.508	0.701	0.227	0.578	0.152	0.463
	3.375	13.500	27.000	3.857	0.526	0.758	0.508	0.610	0.233	0.578	0.152	0.440
	3.000	13.500	27.000	3.857	0.575	0.758	0.508	0.559	0.225	0.578	0.152	0.457
	2.700	13.500	27.000	3.000	0.613	0.758	0.508	0.682	0.305	0.578	0.152	0.492
	2.455	13.500	27.000	3.375	0.625	0.758	0.508	0.527	0.315	0.578	0.152	0.322
Agilefant	3.889	17.500	35.000	4.375	0.727	0.619	0.735	0.721	0.282	0.433	0.199	0.388
	3.889	17.500	35.000	4.375	0.727	0.619	0.735	0.861	0.282	0.433	0.199	0.486
	3.500	17.500	35.000	4.375	0.740	0.619	0.735	0.809	0.283	0.433	0.199	0.509
	3.182	17.500	35.000	5.000	0.720	0.619	0.735	0.732	0.291	0.433	0.199	0.408
	2.917	17.500	35.000	3.182	0.691	0.619	0.735	0.672	0.289	0.433	0.199	0.370
Apache Roller	1.667	15.000	15.000	2.143	0.744	0.779	0.779	0.748	0.558	0.385	0.385	0.579
	1.667	15.000	15.000	2.143	0.744	0.779	0.779	0.724	0.558	0.385	0.385	0.525
	1.500	15.000	15.000	1.667	0.753	0.779	0.779	0.791	0.602	0.385	0.385	0.585
	1.364	15.000	15.000	2.500	0.761	0.779	0.779	0.814	0.660	0.385	0.385	0.628
	1.364	15.000	15.000	1.875	0.761	0.779	0.779	0.796	0.660	0.385	0.385	0.523
Xwiki-platform	16.580	21.256	27.667	16.939	0.176	0.367	0.701	0.256	0.091	0.372	0.751	0.253
	16.255	21.256	27.667	16.600	0.177	0.367	0.701	0.231	0.091	0.372	0.751	0.216
	15.942	20.725	26.774	16.275	0.178	0.361	0.711	0.233	0.092	0.378	0.759	0.250
	15.642	20.220	25.938	15.660	0.176	0.401	0.698	0.198	0.091	0.417	0.745	0.216
	15.352	20.220	25.152	15.660	0.176	0.401	0.707	0.225	0.091	0.417	0.753	0.213
P-value	$<, ***$	$>, ***$	$>, ***$	$-$	$<, ***$	$=$	$=$	$-$	$<, ***$	$=$	$=$	$-$

TABLE 8: Measurement results of *SMQ* and *CMQ*

Subject	SMQ				CMQ			
	LIMBO	WCA	MEM	FoSCI	LIMBO	WCA	MEM	FoSCI
Springblog	0.0116	−0.0180	0.0928	0.3403	−0.0106	−0.037	0.2071	0.4713
	−0.0044	−0.0130	0.1479	0.3490	−0.0111	−0.0060	0.2508	0.4914
	−0.0028	−0.0130	0.1361	0.3725	−0.0165	−0.0060	0.2207	0.3762
	0.02	−0.0130	0.1262	0.2697	−0.0169	−0.0060	0.1700	0.2605
	0.0186	−0.0141	0.1572	0.3713	−0.0045	0.0347	0.2071	0.3368
Solo	−0.0166	−0.0004	−0.0009	0.3007	−0.0110	0.0060	0.0742	0.2494
	−0.0288	−0.0001	−0.0004	0.2948	−0.0211	0.0052	0.1229	0.1207
	−0.0188	−0.0001	−0.0007	0.3033	−0.0145	0.0052	0.1103	0.1863
	−0.0211	0.0001	0.0000	0.1745	−0.0156	0.0049	0.0999	0.4299
	−0.0170	0.0001	0.0002	0.2500	−0.0117	0.0049	0.0914	0.2215
JForum	−0.0048	0.0053	0.0359	0.1110	0.0024	−0.0060	0.1713	0.1929
	−0.0025	0.0035	0.0291	0.3044	0.0067	−0.0086	0.1489	0.3727
	−0.0024	0.0036	0.0172	0.0756	0.0042	−0.0077	0.1921	0.1334
	−0.0030	0.0034	0.0219	0.3652	0.0046	−0.0067	0.1967	0.3117
	0.0005	0.0032	0.0069	0.3872	0.0048	−0.0059	0.1800	0.3199
Agilefant	−0.0035	0.0003	−0.0549	0.1337	−0.0072	−0.0234	0.0018	0.3264
	−0.0018	0.0004	−0.0437	0.0755	−0.0050	−0.0358	−0.0009	0.2694
	−0.0021	0.0004	−0.0358	0.1099	−0.0045	−0.0358	−0.0011	0.3348
	−0.0028	0.0004	−0.0297	0.1167	−0.0049	−0.0366	−0.0006	0.3477
	−0.0021	0.0004	−0.0250	0.0849	−0.0070	−0.0341	−0.0146	0.3466
Apache Roller	−0.0013	0.0070	0.0557	0.1609	0.0013	0.0355	0.0878	0.0500
	0.0012	0.0069	0.0527	0.1240	0.0008	0.0084	0.0831	0.1021
	0.0003	0.0069	0.0471	0.1750	0.0003	0.0084	0.0703	0.1045
	0.0006	0.0066	0.0450	0.1446	0.0000	0.0055	0.0637	0.3504
	−0.0002	0.0064	0.0421	0.1202	−0.0008	0.0011	0.0545	0.3965
Xwiki-platform	0.0001	0.0004	0.0486	0.0489	0.0001	0.0002	0.1124	0.1276
	0.0001	0.0004	0.0476	0.0329	0.0000	0.0006	0.1061	0.1445
	0.0004	0.0005	0.0468	0.1121	−0.0003	0.0010	0.1024	0.1556
	0.0004	0.0004	0.0483	0.0773	−0.0003	0.0083	0.1167	0.1420
	0.0005	0.0004	0.0471	0.0816	0.0000	0.0083	0.1151	0.1152
P-value	<, ***	<, ***	<, ***	−	<, ***	<, ***	<, ***	−