Documentation Reuse: Hot or Not? An Empirical Study

M-A Oumaziz, A. Charpentier, **J-R Falleri**, X. Blanc

Documentation

isEqualCollection

```
public static boolean isEqualCollection(Collection a,
                        Collection b)
```

Returns `true` iff the given `Collection`s contain exactly the same elements with exactly the same cardinalities.

That is, iff the cardinality of *e* in *a* is equal to the cardinality of *e* in *b*, for each element *e* in *a* or *b*.

Parameters:

`a` - the first collection, must not be null

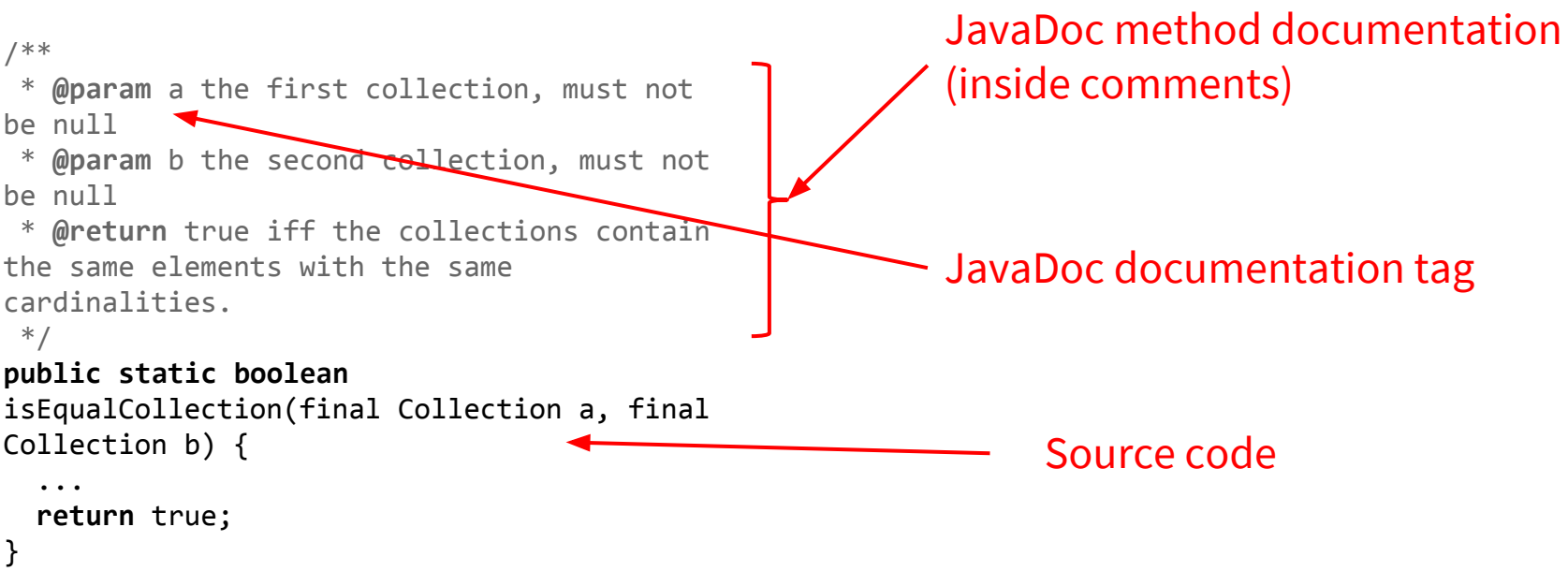
`b` - the second collection, must not be null

Returns:

`true` iff the collections contain the same elements with the same cardinalities.

From Apache Common Collections

Writing Documentation



Documentation Reuse : The Problem

```
/**
 * @param a the first collection, must not
be null
 * @param b the second collection, must not
be null
 * @return true iff the collections contain
the same elements with the same
cardinalities.
 */
public static boolean
isEqualCollection(final Collection a, final
Collection b) {
  ...
  return true;
}
```

How to document this method?

```
public static boolean
isEqualCollection(final Collection a, final
Collection b, final Equator equator) {
  ...
  return
isEqualCollection(collect(a, transformer),
collect(b,transformer));
}
```

delegation

Current Solution : Copy-Paste!

```
/**
 * @param a the first collection, must not
be null
 * @param b the second collection, must not
be null
 * @return true iff the collections contain
the same elements with the same
cardinalities.
 */
public static boolean
isEqualCollection(final Collection a, final
Collection b) {
  ...
  return true;
}
```

```
/**
 * @param a the first collection, must not
be null
 * @param b the second collection, must not
be null
 * @param equator the Equator used
for testing equality
 * @return true iff the collections contain
the same elements with the same
cardinalities.
*/
public static boolean
isEqualCollection(final Collection a, final
Collection b, final Equator equator) {
  ...
  return
isEqualCollection(collect(a, transformer),
collect(b,transformer));
}
```

Research Questions

1. **RQ1:** Do developers often copy-paste documentation tags?
2. **RQ2:** What are the causes of documentation tags copy-paste and could they be avoided by a proper usage of documentation tools?

Our Empirical Study: Corpus

Seven very popular Java libraries

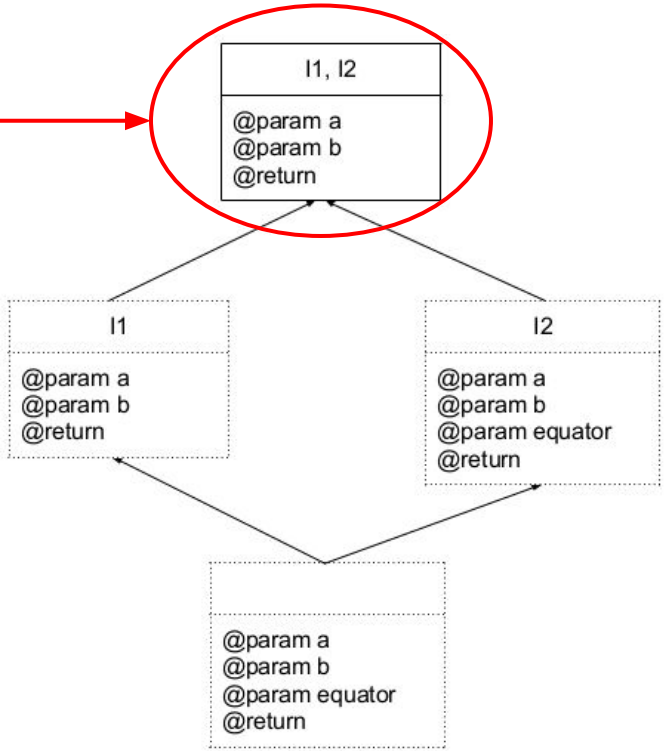
	acc[1]	acio[2]	ggson[3]	Guava	JUnit	Mockito	SLF4J
General							
# of classes	466	119	72	1,203	205	375	33
# of methods	4,078	1,173	569	9,928	1,319	1,716	433
% of documented methods	61.53	97.27	52.55	36.37	43.44	28.15	36.49
Tags							
# of `@description`	1,939	922	265	3,073	448	436	128
# of `@param`	1,199	734	106	749	178	237	51
# of `@throw`	438	209	65	462	12	11	5
# of `@return`	892	322	92	414	90	131	42
inheritDoc							
# of usage	85	18	0	112	2	0	0

[1] Apache Commons Collections
[2] Apache Commons IO
[3] google-gson

Our Empirical Study: Duplication Detector

Documentation duplication

	@param a	@param b	@return	@param equator
I1: isEqualCollection(final Collection a, final Collection b)	X	X	X	
I2: isEqualCollection(final Collection a, final Collection b, ...)	X	X	X	X



Research Question 1: Raw Results

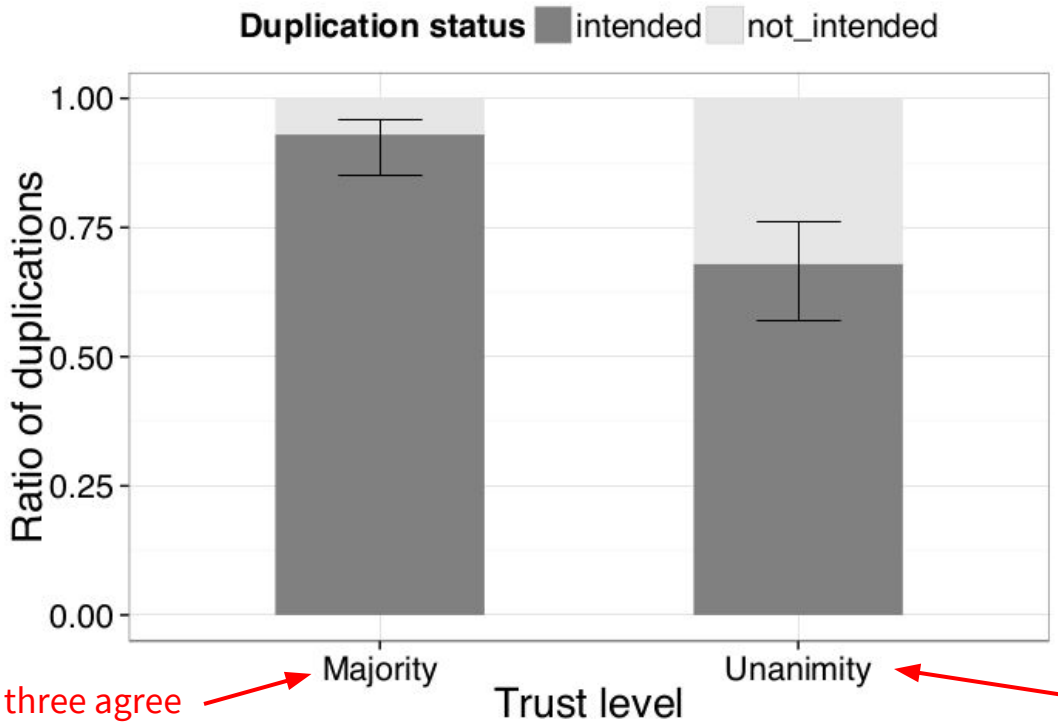
	acc[1]	acio[2]	ggson[3]	Guava	JUnit	Mockito	SLF4J
Documentation							
# of dupl.	1,137	684	59	630	134	86	36
% of complete dupl.	1.51	4.03	0.67	0.80	0.00	0.00	0.00
% of partial dupl.	75.53	77.83	48.49	38.72	42.76	58.59	73.42
Documentation tags							
% of dupl. `@description`	11.24	14.53	8.30	6.57	14.96	4.13	3.91
% of dupl. `@param`	41.78	42.78	22.64	30.71	37.08	17.72	39.22
% of dupl. `@throw`	49.09	55.98	21.54	33.33	8.33	27.27	40.00
% of dupl. `@return`	27.35	31.37	15.22	25.85	14.44	19.85	26.19

[1] Apache Commons Collections
[2] Apache Commons IO
[3] google-gson

Accidental Duplications vs. Copy-Paste

- There are many duplications
- However duplications might be only accidental
- We study the status of duplications
  - Three experts look at a random subset of 100 duplications
  - Analyze source code and documentation
  - Label the duplication as accidental or intended
  - Intended duplication are copy-paste
- We use bootstrapping to estimate the ratio of copy-paste in the corpus

More Than Half Duplications Are Copy-paste!



Two experts out of three agree

All experts agree

Research Question 2: Causes of Copy-Paste

- We take as input all copy-paste from our random subset
- One expert analyzed the documentation and code
- And build a taxonomy of underlying reasons behind the copy-paste

Cause 1 : Delegation (60%)

```
/**
 * @param a the first collection, must not
be null
 * @param b the second collection, must not
be null
 * @return true iff the collections contain
the same elements with the same
cardinalities.
 */
public static boolean
isEqualCollection(final Collection a, final
Collection b) {
  ...
  return true;
}
```

```
/**
 * @param a the first collection, must not
be null
 * @param b the second collection, must not
be null
 * @param equator the Equator used
for testing equality
 * @return true iff the collections contain
the same elements with the same
cardinalities.
*/
public static boolean
isEqualCollection(final Collection a, final
Collection b, final Equator equator) {
  ...
  return
isEqualCollection(collect(a, transformer),
collect(b,transformer));
}
```

Cause 2: Code Cloning (28%)

```
/**
 * Writes a String to the {@link
StringBuilder}.
 *
 * @param value The value to write
 */
@Override
public void write(final String value) {
  if (value != null) {
    builder.append(value);
  }
}
```

```
/**
 * Writes a portion of a character
array to the { @link
StringBuilder }.
 *
 * @param value The value to write
 * @param offset The index of the
first character
 * @param length The number of
characters to write
 */
@Override
public void write(final char [] value,
final int offset, final int length) {
  if (value != null) {
    builder.append(value, offset, length);
  }
}
```

Cause 3: Subtyping (8%)

```
/**
 * Gets the value from the pair.
 *
 * @return the value
 */
V getValue () ;
```

```
/**
* Gets the value from the pair.
*
* @return the value
*/
public V getValue () {
  return value;
}
```

Cause 4: Same Intent Code (3%)

```
/**
 * Delegates to { @link
     EndianUtils#readSwappedInteger(InputStream)}
 * @return the read long
 * @throws IOException if an I/O error occurs
 * @throws EOFException if an end of
     file is reached unexpectedly
 */
public int readInt() throws IOException,
EOFException {
  return EndianUtils.readSwappedInteger(in);
}
```

```
/**
 * Delegates to {@link
   EndianUtils#readSwappedFloat(InputStream)}
 * @return the read long
 * @throws IOException if an I/O error
occurs
 * @throws EOFException if an end of
file is reached unexpectedly
 */
public float readFloat() throws IOException,
EOFException {
  return EndianUtils.readSwappedFloat(in);
}
```

Existing Reuse Support

Tool	Language	#Questions
JavaDoc	Java	2,022
Doxygen	C, C++, Java, C#, VBScript, IDL Fortran, PHP, TCL	1,894
phpDocumentor	PHP	636
JSDoc	JavaScript	574
Doc++	C, C++, IDL, Java	570

No reuse support

	Reuse granularity	
Source location	**Whole**	**Choice**
Override	JavaDoc, JSDoc	JavaDoc
Anywhere	Doxygen, JSDoc	

No support at all for delegation, cloning and same-intent code

Revisiting Documentation Reuse

```
/**
 * @param a the first collection, must not
be null
 * @param b the second collection, must not
be null
 * @return true iff the collections contain
the same elements with the same
cardinalities.
 */
public static boolean
isEqualCollection(final Collection a, final
Collection b) {
  ...
  return true;
}
```

```
/**
 * @param a {@reuse
    isEqualCollection(Collection, Collection)}
 * @param {@reuse isEqualCollection(Collection,
    Collection)}
 * @param equator the Equator used
for testing equality
 * @return {@reuse isEqualCollection(Collection,
    Collection)}
 */
public static boolean
isEqualCollection(final Collection a, final
Collection b, final Equator equator) {
  ...
  return
isEqualCollection(collect(a, transformer),
collect(b,transformer));
}
```

Wrapping Up

- **Copy-Paste** of documentation is **very frequent** in real-world code
- They are caused by **four causes** in the underlying source code
- Existing popular tools **do not provide any support** for most of the cases
- A simple mechanism can dramatically improve the situation, but there are still open questions:
  - What is the situation in other languages than Java?
  - Can we provide better mechanisms?
  - How to migrate existing code?