Diffs between candidates of the selected chunks in the RQ5 analysis and the expected resolutions

May 3, 2025

1 RQ5: How effective is parent similarity as an evaluation function for guiding the generation of conflict resolution candidates?

Starting with the chunk with the lowest correlation value (chunk #6) which is not an outlier, Listing 1 shows the non-empty candidate least similar to the parents (similarity of 2.6%). A red strikethrough and the green character represent characters that the developer would need to respectively remove from or add to the candidate to match the resolution. Conversely, black and white (space) characters mean matches between the candidate and the resolution. In this case, the candidate only matches the starting blank spaces. The candidate requires the developer to remove the brackets from the candidate and add many other characters.

Table 1: Difference between a randomly generated candidate and the resolution for chunk #6 with 3.5% of similarity to the resolution and 2.6% to the parents.

```
    }final Collection<ResourcePostProcessor> processors = ProcessorsUtils.filterProcessorsToApply(minimize,
resourceType, allPostProcessors);
  final String output = applyPostProcessors(mergedResource, processors, content);
```

Listing 2 shows a candidate with a similarity of 50% to the chunk's parents. This represents an intermediate case, where an automated approach would still be progressing towards the resolution. In this case, the randomly generated candidate has some correct parts, but the developer would still need to work on it to reach the resolution.

Table 2: Difference between a randomly generated candidate and the resolution for chunk #6 with 55.3% of similarity to the resolution and 50.1% similarity to the parents.

```
    final SCollectriong <Resoutput = applyrcePostProcessors(> processors, content);
  if (!minimize) {
      =pProcessors  =  Utils.filterProcessorsUtiToApplys.getMy(minimizeF,  recPrsourcessType,
allPors(ptProcessors);
  }
finalStringoutput=LOG.debug("applyPostProcessors: {}"(mergedResource, processors, content);
```

Listing 3 shows the case for the candidate with the highest similarity to the chunk's parents (71.2%) for chunk #6. In this case, the random candidate matches most of the resolution. The developer would only need to add a parameter to one of the method calls.

Table 3: Difference between a randomly generated candidate and the resolution for chunk #6 with 96.1% of similarity to the resolution and 71.2% similarity to the parents.

```
    final Collection<ResourcePostProcessor> processors = ProcessorsUtils.filterProcessorsToApply(minimize,
resourceType, allPostProcessors);
  final String output = applyPostProcessors(mergedResource, processors, content);
```

Now we analyze chunk #31, which represents the Q1 of the correlations in the dataset. Listing 4 shows the difference between the non-empty randomly generated candidate with the least parent similarity (7.9%) to the expected resolution. In this case, the generated candidate has only one line with an `else` statement, which is not present in the resolution. Note that, when comparing the candidate to the expected resolution, the diff algorithm matched the '`e`' and the '`lse`' characters to different parts of the resolution characters. To match the resolution, the developer would need to add many new characters.

Table 4: Difference between a randomly generated candidate and the resolution for chunk #31 with 7.9% of similarity to the resolution and to the parents.

```
            node.failureTable.onFinalFailure(key, next, htl, origHTL, timeLeft, FailureTable.REJECT_TIME,
source);
          next.noLongerRoutingTo(origTag, false);
```

Listing 5 shows a candidate with a similarity of 50% to the chunk's parents. In this case, the candidate has many characters that are not used in the expected resolution. To match the resolution, the developer would need to remove these characters and add some others.

Table 5: Difference between a randomly generated candidate and the resolution for chunk #31 with 52.8% of similarity to the resolution and 50% similarity to the parents.

```
            this.norigTagde.failuremovTableR.outnFingTo alFailure(key, next);
          //,FIXMEhtl,weorigHTL,ntimeLed ft o,eFailureTate thble F.REJECT_TIME,entsoury.ce);
          next.noLongerRoutingTo(origTag, false);
```

Finally, Listing 6 shows the candidate with the highest similarity to the chunk's parents (71.6%). The candidate is 82% similar to the resolution, except for the `if/else` statements. Thus, the developer would only need to remove the `if/else` statements to reach the expected resolution.

While Q1 cases (median $\rho = 0.53$) show weaker correlation than the dataset overall, parent similarity still guides the search toward resolutions, albeit with higher variance in candidate quality.

Next, we analyze chunk #7, which represents the median case in the dataset. Listing 7 shows the non-empty candidate least similar to the parents (similarity of 25.1%). In this case, the candidate is 20.7% similar to the resolution. Note that this similarity is not continuous, meaning that the characters that match between the candidate and the resolution might be scattered in the visualization of the differences.

Listing 8 shows a candidate with a similarity of 50.3% to the chunk's parents. In this case, the candidate still has characters that are not used in the expected resolution, but some parts of the candidate are converging to the resolution.

Listing 9 shows the candidate with the highest similarity to the chunk's parents (88.5%). In this case, the candidate is 100% similar to the expected resolution. This means that, for this case, which represents the median case in the dataset, the resolution was found by maximizing the similarity of the candidate to the parents.

Table 6: Difference between a randomly generated candidate and the resolution for chunk #31 with 82% of similarity to the resolution and 71.6% similarity to the parents.

```
                node.failureTable.onFinalFailure(key, next, htl, origHTL, timeLeft, FailureTable.REJECT_TIME,
source~~e);~~
            ~~if(!wasFork)~~
                ~~finish(RECENTLY_FAILED, next, fals~~e);
            ~~else~~
——              next.noLongerRoutingTo(origTag, false);
```

Table 7: Difference between a randomly generated candidate and the resolution for chunk #7 with 20.7% of similarity to the resolution and 25.1% similarity to the parents.

```
            if (d) LOG.debug(ts + " - Using PartitionExecutor.dispatchWorkFragments() to execute distributed
queries");
            this.tmp_partitionFragments.clear();
            plan.getWorkFragments(ts.getTransactionId(), this.tmp_partitionFragments);
            if (t) LOG.trace("Got back a set of tasks for " + this.tmp_partitionFragments.size() + " partitions for
" + ts);
```

Next, we analyze chunk #24, which represents the Q3 of the correlations in the dataset. This means that this chunk has a very strong correlation ($\rho = 0.91$), meaning that as the similarity of candidates to the parents increase, the similarity of these candidates to the resolution also increase. Listing 10 shows the non-empty candidate for chunk #24 with the lowest similarity to the parents. We can observe that, in this case, the candidate shared a single line with the resolution.

Listing 11 shows a random candidate for chunk #24 with 50.9% similarity to the parents. It already has many characters which are used in the resolution, but the developer would still need to add other lines. The good news is that this is only an intermediate candidate, since there is still room for improvement in the parents' similarity for this chunk.

Listing 12 shows the random candidate with the highest similarity to the parents (95.1%) for chunk #24 in our dataset. In this case, the candidate has 97% of similarity to the resolution. Analyzing the difference between the candidate and the resolution presented in Listing 12, we can observe that the developer would only need to add a new line to match the expected resolution.

Finally, we analyze chunk #42, which represents the upper whisker of the correlations distribution in our dataset. This chunk represents situations that have a very strong correlation ($\rho = 1.00$) between the similarity of candidates to the parents and candidates to the resolution. Listing 13 shows the non-empty random candidate with the smallest similarity (11.1%) to the parents.

Listing 14 shows a candidate for chunk #42 with a similarity of 50.2% to the parents. Compared to the candidate from Listing 13, we can observe that it became closer to the expected resolution.

Listing 15 shows the random candidate with the highest similarity to the parents (97.1%) for chunk #42 in our dataset. In this case, the candidate has 98.4% of similarity to the resolution. The developer would only need to remove some comments from the candidate to reach the expected resolution.

Our analysis of chunks spanning the correlation spectrum ($\rho = 0.14$ to $\rho = 1.00$) demonstrates that parent similarity is a highly effective evaluation function for automated conflict resolution. Even in low-correlation cases (e.g., chunk #6), maximizing parent similarity yields candidates 96.1% aligned with resolutions. For high-correlation cases (e.g., chunk #42), trivial edits suffice. These findings evidence the effectiveness of using parent similarity in Search-Based Software Engineering (SBSE) approaches, where it can guide search algorithms toward resolutions efficiently.

Table 8: Difference between a randomly generated candidate and the resolution for chunk #7 with 49.3% of similarity to the resolution and 50.3% similarity to the parents.

```
                    if (td) LOG.tracdebug(ts + "G Using PartitionExecut bor.dispatchWork Fra
sgments()tofexecutasksfodistr +          this.tmp_partitionFragments.sizcle() + " partitions for " + ts();
——          if(d)LOG.debug(ts+"-UsingPplartitionExecutor.dispagetchWorkFragments() to exs.getTransacute
ionId(), this.tmp_partibuted querionFragments");
          if(t)LOG.trace("Gotbackasetoftasksfor"+this.tmp_partitionFragments.elsize() + " par(titions for " +
ts);
```

Table 9: Difference between a randomly generated candidate and the resolution for chunk #7 with 100% of similarity to the resolution and 88.5% similarity to the parents.

```
          if (d) LOG.debug(ts + " - Using PartitionExecutor.dispatchWorkFragments() to execute distributed
queries");
          this.tmp_partitionFragments.clear();
          plan.getWorkFragments(ts.getTransactionId(), this.tmp_partitionFragments);
          if (t) LOG.trace("Got back a set of tasks for " + this.tmp_partitionFragments.size() + " partitions for
" + ts);
```

Table 10: Difference between a randomly generated candidate and the resolution for chunk #24 with 9.8% of similarity to the resolution and 10.7% similarity to the parents.

```
    public static final Option<Boolean> SMART_INDENT       = bool("smartindent",  true,  "si");
    public static final Option<Boolean> AUTO_INDENT        = bool("autoindent",   false, "ai");
    public static final Option<Boolean> ATOMIC_INSERT      = bool("atomicinsert", true,  "ati");
    public static final Option<Boolean> IGNORE_CASE        = bool("ignorecase",   false, "ic");
    public static final Option<Boolean> SMART_CASE         = bool("smartcase",    false, "scs");
    public static final Option<Boolean> SANE_CW            = bool("sanecw",       false);
    public static final Option<Boolean> SANE_Y             = bool("saney",        false);
    public static final Option<Boolean> SEARCH_HIGHLIGHT   = bool("hlsearch",     false, "hls");
    public static final Option<Boolean> SEARCH_REGEX       = bool("regexsearch",  false, "rxs");
    public static final Option<Boolean> INCREMENTAL_SEARCH = bool("incsearch",    false, "is");
    public static final Option<Boolean> LINE_NUMBERS       = bool("number",       false, "nu");
    public static final Option<Boolean> SHOW_WHITESPACE    = bool("list",         false, "l");
    public static final Option<Boolean> IM_DISABLE         = bool("imdisable",    false, "imd");
    public static final Option<Boolean> VISUAL_MOUSE       = bool("visualmouse",  true,  "vm");
    public static final Option<Boolean> AUTO_CHDIR         = bool("autochdir",    false, "acd");
    public static final Option<Boolean> HIGHLIGHT_CURSOR_LINE = bool("cursorline",   false, "cul");
    // TODO: This is an Eclipse setting under Window->Preferences->Editors->Text Editors->"Insert spaces
for tabs"
    //       Changing this value should change the Eclipse configuration too. -- BRD
    public static final Option<Boolean> EXPAND_TAB         = bool("expandtab",    true,  "et");
```

Table 11: Difference between a randomly generated candidate and the resolution for chunk #24 with 47.2% of similarity to the resolution and 50.9% similarity to the parents.

```
    public static final Option<Boolean> SMART_CASINDE——NT      = bool("smartindent",  true,  "si");
    publicastatic final Option<Boolean> AUTO_INDENT        = bool("autoindent",-   false, "scsai");
    public static final Option<Boolean> ATOMIC_INSERT      = bool("atomicinsert", true, "ati");
    public static finalOption<Boolean> IGNORE_CASE        = bool("ignorecase",   false, "ic");
    public static final Option<Boolean> SMART_CASE         = bool("smartcase",    false,"scs");
    public static final Option<Boolean> SANE_CW           =bool("sanecw",       false);
    public static final Option<Boolean> SANE_Y            = bool("saney",       false);
    public static final Option<Boolean> SEARCH_HIGHLIGHT—   = bool("hlsearch",     false,"hls");
    public static final Option<Boolean> SEARCH_REGEX       =bool("regexsearch", false,"rxs");
    public static final Option<Boolean> INCREMENTAL_SEARCH =bool("incsearch",    false, "is");
    public static final Option<Boolean> LINE_NUMBERS—       = bool("number",       false, "nu");
    public static final Option<Boolean> SHOW_WHITESPACE    = bool("list",         false, "l");
    public static final Option<Boolean> IM_DISABLE—         = bool("imdisable",    false,"imd");
    public static final Option<Boolean> VISUAL_MOUSE       =bool("visualmouse",  true,  "vm");
    public static final Option<Boolean> AUTO_CHDIR         = bool("autochdir",    false, "acd");
    public static final Option<Boolean> HIGHLIGHT_CURSOR_LINE = bool("cursorline",   false,"cul");
    // TODO: This is an Eclipse setting under Window->Preferences->Editors->Text Editors->"Insert spaces
fortabs"
    //      Changing this value should change the Eclipse configuration too. -- BRD
    public static final Option<Boolean> EXPAND_TAB         = bool("expandtab",    true,  "et");
```

Table 12: Difference between a randomly generated candidate and the resolution for chunk #24 with 97% of similarity to the resolution and 95.1% similarity to the parents.

```
    public static final Option<Boolean> SMART_INDENT       = bool("smartindent",  true,  "si");
    public static final Option<Boolean> AUTO_INDENT        = bool("autoindent",   false, "ai");
    public static final Option<Boolean> ATOMIC_INSERT      = bool("atomicinsert", true,  "ati");
    public static final Option<Boolean> IGNORE_CASE        = bool("ignorecase",   false, "ic");
    public static final Option<Boolean> SMART_CASE         = bool("smartcase",    false, "scs");
    public static final Option<Boolean> SANE_CW            = bool("sanecw",       false);
    public static final Option<Boolean> SANE_Y             = bool("saney",        false);
    public static final Option<Boolean> SEARCH_HIGHLIGHT   = bool("hlsearch",     false, "hls");
    public static final Option<Boolean> SEARCH_REGEX       = bool("regexsearch",  false, "rxs");
    public static final Option<Boolean> INCREMENTAL_SEARCH = bool("incsearch",    false, "is");
    public static final Option<Boolean> LINE_NUMBERS       = bool("number",       false, "nu");
    public static final Option<Boolean> SHOW_WHITESPACE    = bool("list",         false, "l");
    public static final Option<Boolean> IM_DISABLE         = bool("imdisable",    false, "imd");
    public static final Option<Boolean> VISUAL_MOUSE       = bool("visualmouse",  true,  "vm");
    public static final Option<Boolean> AUTO_CHDIR         = bool("autochdir",    false, "acd");
    public static final Option<Boolean> HIGHLIGHT_CURSOR_LINE = bool("cursorline",   false, "cul");
    // TODO: This is an Eclipse setting under Window->Preferences->Editors->Text Editors->"Insert spaces
for tabs"
    //      Changing this value should change the Eclipse configuration too. -- BRD
    public static final Option<Boolean> EXPAND_TAB         = bool("expandtab",    true,  "et");
```

Table 13: Difference between a randomly generated candidate and the resolution for chunk #42 with 10.8% of similarity to the resolution and 11.1% similarity to the parents.

```
        final String arrayKey = rs.getString("array_key");//NOI18N
        final Integer key = new Integer(rs.getInt("child"));//NOI18N
        final String father_pk = rs.getString("father_pk");//NOI18N
        final int father =rs.getInt("father");//NOI18N
        final String attribute = rs.getString("attribute");//NOI18N
        final String child_table= rs.getString("child_table");//NOI18N
        final String father_table = rs.getString("father_table");//NOI18N
        final String child_pk = rs.getString("child_pk");//NOI18N

          final String value = "Select " + father + " as class_id ," + father_pk + " as object_id" + " from
"//NOI18N
                + father_table
                + " where " + attribute + " in "
                + " (select " + arrayKey + " from " + child_table + " where  " + child_pk + " = ";// ? )
```

Table 14: Difference between a randomly generated candidate and the resolution for chunk #42 with 50.7% of similarity to the resolution and 50.2% similarity to the parents.

```
        final String arrayKey = rs.getString("array_key");//NOI18N
        final Integer key = new Integer(rs.getInt("child"));//NOI18N
        final String father_pk = rs.getString("father_pk");//NOI18N
        final Strngtfather= rs.getInt("fabther");//NOI18N
        finalString attribute = rs.getString("attribute");//NOI18N
        final String child_table = rs.getString("child_table");//NOI18N
        final Strintgfather_table= rs.getInString("father_table");//NOI18N
        final String child_pk = rs.getString("child_pk");//NOI18N

          final String value = "Select " + father + " as class_id ," + father_pk + " as object_id" + " from
"//NOI18N
                + father_table
                + " where " + attribute + " in "
                + " (select "+ arrayKey+ " from" + child_table + " where  " + child_pk + " ="; // ? )
```

Table 15: Difference between a randomly generated candidate and the resolution for chunk #42 with 98.4% of similarity to the resolution and 97.1% similarity to the parents.

```
        final String arrayKey = rs.getString("array_key");//NOI18N
        final Integer key = new Integer(rs.getInt("child"));//NOI18N
        final String father_pk = rs.getString("father_pk");//NOI18N
        final int father = rs.getInt("father");//NOI18N
        final String attribute = rs.getString("attribute");//NOI18N
        final String child_table = rs.getString("child_table");//NOI18N
        final String father_table = rs.getString("father_table");//NOI18N
        final String child_pk = rs.getString("child_pk");//NOI18N

          final String value = "Select " + father + " as class_id ," + father_pk + " as object_id" + " from
"//NOI18N
—               + father_table
—               + " where " + attribute + " in "//NOI18N
—                + " (select " + arrayKey + " from " + child_table + " where  " + child_pk + " = "; // ?
)//NOI18N
```