ANVAYA: An Algorithm and Case-Study on Improving the Goodness of Software Process Models generated by Mining Event-Log Data in Issue Tracking Systems

Prerna Juneja*, Divya Kundra* and Ashish Sureka+

*Indraprastha Institute of Information Technology, Delhi (IIIT-D), India
prerna1399@iiitd.ac.in, divya1395@iiitd.ac.in

+ABB Research, India
ashish.sureka@in.abb.com

COMPSAC 2016
11 June 2016
1. Research Motivation and Aim
2. Research Framework and Solution Approach
3. Experimental Analysis and Results
4. Limitations and Future Work
5. Conclusion
6. References
1. Research Motivation and Aim
2. Research Framework and Solution Approach
3. Experimental Analysis and Results
4. Limitations and Future Work
5. Conclusion
6. References
Software Maintenance

- s/w maintenance is the process of improving a product's quality after its delivery.
- Failures keep on occurring and need of improvement keeps on growing throughout a product's lifecycle.
- It can be of various types:
  - Corrective
  - Adaptive
  - Perfective
  - Preventive

Issue Tracking Systems

• Software applications to update, maintain and resolve an issue.
• Tool to guide the software maintenance process.
• Allows user to raise, describe an issue and track its status and progress until it gets resolved.
• Defect reporting and tracking is made simpler.

Source: [http://www.online-issues.com/](http://www.online-issues.com/)
Issue Tracking Systems

- Software applications to update, maintain and resolve an issue.
- Tool to guide the software maintenance process.
- Allows user to raise, describe an issue and track its status and progress until it gets resolved.
- Defect reporting and tracking is made simpler.

Source: [http://www.online-issues.com/](http://www.online-issues.com/)
Issue Tracking Systems

• Software applications to update, maintain and resolve an issue.

• Tool to guide the software maintenance process.

• Allows user to raise, describe an issue and track its status and progress until it gets resolved.

• Defect reporting and tracking is made simpler.

Source: http://www.online-issues.com/
Issue Tracking Systems

- Software applications to update, maintain and resolve an issue.
- Tool to guide the software maintenance process.
- Allows user to raise, describe an issue and track its status and progress until it gets resolved.
- Defect reporting and tracking is made simpler.

Source: [http://www.online-issues.com/](http://www.online-issues.com/)
Issue Tracking Systems

### Activity

<table>
<thead>
<tr>
<th>Who</th>
<th>When</th>
<th>What</th>
<th>Removed</th>
<th>Added</th>
</tr>
</thead>
<tbody>
<tr>
<td>mconnor</td>
<td>2004-04-03 16:33:00 PST</td>
<td>Status</td>
<td>UNCONFIRMED</td>
<td>NEW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ever confirmed</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>mconnor</td>
<td>2006-08-27 05:57:25 PDT</td>
<td>QA Contact</td>
<td>mconnor</td>
<td>bookmarks</td>
</tr>
<tr>
<td>mak77</td>
<td>2009-03-24 04:59:05</td>
<td>Assignee</td>
<td>p_ch</td>
<td>nobody</td>
</tr>
<tr>
<td>jaws</td>
<td>2013-03-15 10:32:42 PDT</td>
<td>Resolution</td>
<td>Won't fix</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Last resolved</td>
<td>2013-03-15 10:32:42</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Status</td>
<td>NEW</td>
<td>RESOLVED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CC</td>
<td></td>
<td>jAws</td>
</tr>
</tbody>
</table>

### Timestamp

- 2004-04-03 16:33:00 PST
- 2006-08-27 05:57:25 PDT
- 2009-03-24 04:59:05
- 2013-03-15 10:32:42

### Actor

- mconnor
- mak77
- jaws

### Event Log

- Status changed from UNCONFIRMED to NEW
- Assignee changed from mconnor to p_ch
- Resolution changed from WONTFIX

---

Image: Bugzilla@Mozilla screenshot showing activity, timestamp, actor, and event log.
Bug Lifecycle in Bugzilla

Source: https://www.bugzilla.org/docs/2.18/html/lifecycle.html
Process mining is extraction of insights, consumable results and actionable information from event logs recorded by Process Aware Information Systems (PAIS).

- PAIS is a *software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process models*. [1]
Process Mining Techniques

Source: [2]
Problem of Spaghetti Models

- Results in a spaghetti process model consisting of:
  - a large number of activity or task nodes
  - large number of relations (or directed edges) between these nodes.
- Cumbersome to comprehend.
- Difficult to analyse.
- Only by zooming in one can get some level of understandability.
Problem of Spaghetti Models

Source: http://simonandersonfitness.co.uk/i-fd-up/
Problem of Spaghetti Models

Source: http://simonandersonfitness.co.uk/i-fd-up/
1) To study the problem of spaghetti process models by conducting case-study on open-source Firefox browser project.

2) To propose a trace clustering technique based on grouping sequential data and apply it on ITS dataset.

3) To investigate the effectiveness of proposed approach in reducing structural complexity and enhancing process model comprehensibility for a process analyst.
4) To study self-loops, back-and-forth, issue reopen and bottlenecks on discovered process models derived from output of trace clustering.

5) Illustrate the benefits of trace clustering in the domain of SPI using a real-life case-study.
1. Research Motivation and Aim
2. Related Work and Research Contributions
3. Research Framework and Solution Approach
4. Experimental Analysis and Results
5. Limitations and Future Work
6. Conclusion
7. References
Architecture Diagram
82 unique activities were identified.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Acronym</th>
<th>Count</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assigned to</td>
<td>ASS</td>
<td>4148</td>
<td>Bug is assigned to the proper person for setting its resolution.</td>
</tr>
<tr>
<td>Carbon Copy</td>
<td>CCC</td>
<td>43041</td>
<td>Users who are interested in the progress of the bug are included in the mailing list.</td>
</tr>
<tr>
<td>Resolution Fixed</td>
<td>REF</td>
<td>3869</td>
<td>A fix for the bug is determined and tested.</td>
</tr>
<tr>
<td>Resolution Invalid</td>
<td>REI</td>
<td>2863</td>
<td>The issue raised is not a valid bug and resolution is thus set to 'Invalid'.</td>
</tr>
<tr>
<td>Status Assigned Resolved</td>
<td>SAR</td>
<td>2335</td>
<td>The bug status changes from Assigned, where it was assigned to proper person for fixing its resolution to Resolved where resolution has been performed and is awaiting verification by Quality Assurance.</td>
</tr>
<tr>
<td>Status Reopened New</td>
<td>SRN</td>
<td>41</td>
<td>The bug status changes from Reopened where the bug was reopened as the resolution was later found to be incorrect to New where it is assigned for processing.</td>
</tr>
<tr>
<td>Status Resolved Reopened</td>
<td>SRR</td>
<td>699</td>
<td>The bug status changes from Resolved, where its resolution was set, to Reopened where the bug is reopened as the resolution is found to be incorrect afterwards.</td>
</tr>
<tr>
<td>Status Resolved Verified</td>
<td>SRV</td>
<td>727</td>
<td>The bug status changes from Resolved where resolution has been performed to Verified where Quality Assurance has looked at the bug and its resolution and agrees that the appropriate resolution has been performed.</td>
</tr>
<tr>
<td>Status Unconfirmed Assigned</td>
<td>SUA</td>
<td>73</td>
<td>The bug status changes from Unconfirmed where it is validated whether the bug is true to Assigned where it is assigned to the proper person for processing.</td>
</tr>
</tbody>
</table>
• Each cluster is represented by a medoid.
  • It is the most centrally located data point in a cluster.
  • It’s average similarity to all other data points in that cluster is maximal.

• This algorithm partitions the datasets into k clusters such that distance between the data points assigned in a cluster and center of that cluster is minimized.
**Algorithm 4: k Medoid Clustering**

**Data:** Event log in sequential data format  
**Result:** $k$ clusters

1. input the value of number of clusters to be formed $k$.  
2. read the input event log  
3. randomly select $k$ traces as initial medoids.

4. **foreach** non medoid trace $t_i$ **do**  
    5. **foreach** medoid trace $m_i$ **do**  
       6. calculate similarity score of $t_i$ and $m_i$ using LCS Similarity $lcs_i$ or DTW Similarity score $dtw_i$  
       7. assign $t_i$ to $m_i$ with highest $lcs_i$ or lowest $dtw_i$.  

8. **foreach** medoid trace $m$ **do**  
    9. **foreach** non medoid trace $o$ **do**  
       10. swap $m$ and $o$ compute the total similarity score of the configuration using either $lcs_i$ or $dtw_i$  

11. select the configuration with the highest cost while using LCS Similarity and lowest cost while using DTW Similarity.  
12. Steps 4 to 8 are repeated till there is no change in the medoids.
### K medoid

**Algorithm 4: k Medoid Clustering**

- **Data:** Event log in sequential data format
- **Result:** $k$ clusters

1. input the value of number of clusters to be formed $k$.
2. read the input event log
3. randomly select $k$ traces as initial medoids.

4. foreach non medoid trace $t_i$ do
5.     foreach medoid trace $m_i$ do
6.         calculate similarity score of $t_i$ and $m_i$ using LCS Similarity $lcs_i$ or DTW Similarity score $dtw_i$
7.         assign $t_i$ to $m_i$ with highest $lcs_i$ or lowest $dtw_i$.

8. foreach medoid trace $m$ do
9.     foreach non medoid trace $o$ do
10.        swap $m$ and $o$ compute the total similarity score of the configuration using either $lcs_i$ or $dtw_i$

11. select the configuration with the highest cost while using LCS Similarity and lowest cost while using DTW Similarity.
12. Steps 4 to 8 are repeated till there is no change in the medoids.
K medoid

Algorithm 4: k Medoid Clustering

Data: Event log in sequential data format

Result: k clusters

1. input the value of number of clusters to be formed k.
2. read the input event log
3. randomly select k traces as initial medoids.
4. foreach non medoid trace \( t_i \) do
5.   foreach medoid trace \( m_i \) do
6.     calculate similarity score of \( t_i \) and \( m_i \) using LCS Similarity \( lcs_i \) or DTW
7.     Similarity score \( dtw_i \)
8.     assign \( t_i \) to \( m_i \) with highest \( lcs_i \) or lowest \( dtw_i \)
9. foreach medoid trace \( m \) do
10.   foreach non medoid trace \( o \) do
11.     swap \( m \) and \( o \) compute the total similarity score of the configuration
12.     using either \( lcs_i \) or \( dtw_i \)
13. select the configuration with the highest cost while using LCS Similarity and
14. lowest cost while using DTW Similarity.
15. Steps 4 to 8 are repeated till there is no change in the medoids
Algorithm 4: k Medoid Clustering

Data: Event log in sequential data format

Result: k clusters

1. input the value of number of clusters to be formed k.
2. read the input event log
3. randomly select k traces as initial medoids.
4. foreach non medoid trace tᵢ do
   5. foreach medoid trace mᵢ do
      6. calculate similarity score of tᵢ and mᵢ using LCS Similarity lcsᵢ or DTW Similarity score dtwᵢ
      7. assign tᵢ to mᵢ with highest lcsᵢ or lowest dtwᵢ.
   8. foreach medoid trace m do
      9. foreach non medoid trace o do
         10. swap m and o compute the total similarity score of the configuration using either lcsᵢ or dtwᵢ

11. select the configuration with the highest cost while using LCS Similarity and lowest cost while using DTW Similarity.
12. Steps 4 to 8 are repeated till there is no change in the medoids.
Each trace is nothing but a sequence of characters. LCS algorithm determines the length of the longest common sequence of characters which need not be consecutive but follow a left to right ordering. The algorithm takes two traces as input and returns their similarity score. Higher the similarity score, more similar are the traces.
Longest Common Subsequence

- Let $S_1$ & $S_2$ be two sequences and $\text{Similarity}[i,j]$ be the length of LCS of sequences $S_{1i}$ and $S_{2i}$.
- The algorithm uses the following recursive formula:

$$\text{Similarity}[i,j] = \begin{cases} 
\text{Similarity}[i-1,j-1] + 1 & \text{if } S_1[i] = S_2[i] \\
\max\{\text{Similarity}[i-1,j], \text{Similarity}[i,j-1]\} & \text{otherwise}
\end{cases}$$
Algorithm 5: LCS Similarity

Data: Trace $s_1$ and $s_2$ from event log in sequential data format.

Result: Similarity score between the two input traces.

1. Find the length $n$ of $s_1$.
2. Find the length $m$ of $s_2$.
3. For $i ← 0$ to $n$ do
   4. $\text{Similarity}[i][0] = 0$.
5. For $j ← 0$ to $m$ do
   6. $\text{Similarity}[0][j] = 0$
7. For $i ← 1$ to $n$ do
   8. For $j ← 1$ to $m$ do
      9. If $s_1[i-1]$ equals $s_2[j-1]$ then
         10. $\text{Similarity}[i][j] = \text{Similarity}[i-1][j-1] + 1$
      11. Else
         12. $\text{Similarity}[i][j] = \max(\text{Similarity}[i-1][j], \text{Similarity}[i][j-1])$
13. Return $\text{Similarity}[n][m]$
Algorithm 5: LCS Similarity

Data: Trace $s_1$ and $s_2$ from event log in sequential data format.

Result: Similarity score between the two input traces.

1. find the length $n$ of $s_1$.
2. find the length $m$ of $s_2$.
3. for $i \leftarrow 0$ to $n$ do
4.     Similarity[$i$][0] = 0.
5. for $j \leftarrow 0$ to $m$ do
6.     Similarity[0][j] = 0
7. for $i \leftarrow 1$ to $n$ do
8.     for $j \leftarrow 1$ to $m$ do
9.         if $s_1[i-1]$ equals $s_2[j-1]$ then
10.            Similarity[$i$][$j$] = Similarity[$i$-1][$j$-1] + 1
11.        else
12.            Similarity[$i$][$j$] = max(Similarity[$i$-1][$j$], Similarity[$i$][$j$-1])
13. return Similarity[$n$][$m$]
### Longest Common Subsequence

**Algorithm 5: LCS Similarity**

**Data:** Trace $s_1$ and $s_2$ from event log in sequential data format.

**Result:** Similarity score between the two input traces.

1. find the length $n$ of $s_1$.
2. find the length $m$ of $s_2$.
3. for $i \leftarrow 0$ to $n$ do
   4.  $\text{Similarity}[i][0] = 0$.
5. for $j \leftarrow 0$ to $m$ do
   6.  $\text{Similarity}[0][j] = 0$
7. for $i \leftarrow 1$ to $n$ do
   8.  for $j \leftarrow 1$ to $m$ do
      9.  if $s_1[i-1]$ equals $s_2[j-1]$ then
         10.  $\text{Similarity}[i][j] = \text{Similarity}[i-1][j-1] + 1$
      11.  else
         12.  $\text{Similarity}[i][j] = \max(\text{Similarity}[i-1][j], \text{Similarity}[i][j-1])$
13. return $\text{Similarity}[n][m]$
Longest Common Subsequence

Algorithm 5: LCS Similarity

Data: Trace s1 and s2 from event log in sequential data format.
Result: Similarity score between the two input traces.

1. find the length n of s1.
2. find the length m of s2.
3. for i ← 0 to n do
   4. Similarity[i][0] ← 0.
5. for j ← 0 to m do
   6. Similarity[0][j] ← 0.

7. for i ← 1 to n do
   8. for j ← 1 to m do
    9. if s1[i-1] equals s2[j-1] then
    10.   Similarity[i][j] ← Similarity[i-1][j-1] + 1
    11. else
    12.   Similarity[i][j] ← max(Similarity[i-1][j], Similarity[i][j-1])

13. return Similarity[n][m]
Dynamic Time Warping

• Finds similarity between temporal sequences having different lengths that are non-linearly warped in time dimension.

• Let two sequences be $S_1$ and $S_2$.

• Warping path consists of index pairs $(i,j)$ if DTW associates $S_1[i]$ with $S_2[j]$.

• Warping distance is the summation of element wise distance between $S_1[i]$ and $S_2[j]$ over all pairs of $(i,j)$ in the warping path.
Dynamic Time Warping

- We assign a cost (distance) 0 if $S_1[i]=S_2[j]$, otherwise 1 is assigned.
- The algorithm returns the similarity score which is the warping distance.
- Lower the warping distance, more similar are the traces.
Dynamic Time Warping

Algorithm 6: DTW Similarity

Data: Sequences $s_1$ and $s_2$ from event log in sequential data format.

Result: Similarity value between the two input sequences

1. find the length $n$ of $s_1$.
2. find the length $m$ of $s_2$.
3. cost=0.
4. for $i \leftarrow 1$ to $n$ do
   5. DTW[i][0]=0
   6. for $j \leftarrow 1$ to $m$ do
      7. DTW[0][j]=0
      8. DTW[0][0]=0
   9. for $i \leftarrow 1$ to $n$ do
      10. for $j \leftarrow 1$ to $m$ do
          11. if $s_1[i-1]$ equals $s_2[j-1]$ then
              12. cost=0
          13. else
              14. cost=1
          15. DTW[i][j]=cost+minimum of (DTW[i-1][j], DTW[i][j-1], DTW[i-1][j-1])
      16. return DTW[n][m]
Algorithm 6: DTW Similarity

Data: Sequences s1 and s2 from event log in sequential data format.

Result: Similarity value between the two input sequences

1. find the length $n$ of $s1$.
2. find the length $m$ of $s2$.
3. cost = 0.
4. for $i \leftarrow 1$ to $n$ do
5.   [ ] $\text{DTW}[i][0] = 0$
6. for $j \leftarrow 1$ to $m$ do
7.   [ ] $\text{DTW}[0][j] = 0$
8. $\text{DTW}[0][0] = 0$
9. for $i \leftarrow 1$ to $n$ do
10.   for $j \leftarrow 1$ to $m$ do
11.     if $s1[i-1]$ equals $s2[j-1]$ then
12.        [ ] cost = 0
13.     else
14.        [ ] cost = 1
15. $\text{DTW}[i][j] = \text{cost} + \text{minimum of}(\text{DTW}[i-1][j], \text{DTW}[i][j-1], \text{DTW}[i-1][j-1])$
16. return $\text{DTW}[n][m]$
Complexity

• Complexity can have unwanted effects on understandability, comprehensibility and correctness of process models.

• McCabe's cyclomatic number determines the number of linearly independent paths in the process model.
• One of the applications of Process Mining is to determine the gaps between the real world as recorded in the event log and the model.

• Fitness metric is used to determine the conformance between an event log and a process model generated from that log.

• The fitness value of a process model can take any values between 0 and 1.
Fitness

Algorithm 8: Fitness

Data: Xml format input of the process model and Event log in sequential format.

Result: Fitness of the process model.

1. read Xml format input file.
2. foreach transition between a source node $n_i$ and target node $n_j$ do
   3. adjacency matrix $a_{n_i,n_j} = 1$
4. read the input event log
5. foreach bug $b_i$ do
   6. add each activity to trace $t_i$
   7. if $t_i$ is unique then
      8. add it to $uniquetrace[]$
      9. Count its frequency $F_i$ in the event log
10. foreach entry $t_i$ in $uniquetraces[]$ do
    11. $Valid_i = 1$
    12. $j = 1$
    13. while till the length of $t_i$ do
        14. if $a_{j,j+1} \neq 1$ then
            15. $Valid_i = 0$
            16. break
        else
            17. $j++$
    18. foreach entry $t_i$ in $uniquetraces[]$ do
    19. $FreqValidProduct = FreqValidProduct + F_i \cdot Valid_i$
    20. $FreqSum = FreqSum + F_i$
    21. $Fitness = FreqValidProduct / FreqSum$
Determine Goodness of Process Models

- Clustering can give many different solutions.

- Objective function:
  - Reduce complexity
  - Increase Fitness

- Find goodness ratio (G_Ratio) by dividing the weighted average of fitness values by the weighted average of complexity values of all clusters present in the given solution set.

- Returns the set having maximum G_Ratio
Determine Goodness of Process Models

**Algorithm 9: Automate Clustering**

**Data:** History data of bugs  
**Result:** Best cluster set

1. *Event Log Creation*(History data of bugs)  
2. *Sequential Data Creation*(Event Log of Bugs)  
3. generate 3 cluster sets $S_1$, $S_2$ and $S_3$ by calling  
   *k Medoid Clustering*(Event log in sequential data format) that uses LCS and DTW similarity for input $k$ value  
4. foreach cluster set $S_i$ consisting of $m$ clusters do  
   5. for $j ← 1$ to $m$ do  
   6. discover process model $P_j$

\[
C_{-}\text{score}_i = \sum_{j=1}^{m} \frac{\text{Complexity(}X\text{ml format input of } P_j \text{)} * t_j}{t_j}
\]

\[
F_{-}\text{score}_i = \sum_{j=1}^{m} \frac{\text{Fitness(}X\text{ml format input of } P_i, \text{ Event log in sequential format of cluster } C_j \text{)} * t_j}{t_j}
\]

\[
G_{-}\text{Ratio}_i = F_{-}\text{score}_i / C_{-}\text{score}_i
\]

8. where $t_j$ is total traces in event log of cluster $C_j$ return the cluster set $S_i$ with the maximum $G_{-}\text{Ratio}$.
Outline

1. Research Motivation and Aim
2. Research Framework and Solution Approach
3. Experimental Analysis and Results
4. Limitations and Future Work
5. Conclusion
6. References
# Experimental Dataset Details (Mozilla Firefox Project)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>Firefox</td>
</tr>
<tr>
<td>First Issue Report Date</td>
<td>1 January 2013</td>
</tr>
<tr>
<td>Last Issue Report Date</td>
<td>31 December 2013</td>
</tr>
<tr>
<td>Data Extraction Date</td>
<td>16 October 2014</td>
</tr>
<tr>
<td>Number of Open Issues</td>
<td>3399</td>
</tr>
<tr>
<td>Number of Closed Issues Used</td>
<td>11804</td>
</tr>
<tr>
<td>Number of Activities in Closed Issues</td>
<td>82</td>
</tr>
<tr>
<td>Number of Events Reported for Closed Bugs</td>
<td>178331</td>
</tr>
</tbody>
</table>
• To validate the clustering, k medoid algorithm using LCS and DTW similarity metrics was applied on 1615 process-instances and 6 clusters were obtained.
## Experimental Results

### Cyclomatic Complexity (Percentage decrease in complexity of clusters)

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Complexity</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Model</td>
<td>143 (-)</td>
<td>0.017</td>
</tr>
<tr>
<td>Cluster 2</td>
<td>75 (47.5 %)</td>
<td>0.178</td>
</tr>
<tr>
<td>Cluster 3</td>
<td>82 (42.6 %)</td>
<td>0.085</td>
</tr>
<tr>
<td>Cluster 4</td>
<td>106 (25.8 %)</td>
<td>0.004</td>
</tr>
<tr>
<td>Cluster 5</td>
<td>96 (32.8 %)</td>
<td>0.070</td>
</tr>
<tr>
<td>Cluster 6</td>
<td>83 (41.9 %)</td>
<td>0.015</td>
</tr>
<tr>
<td>Cluster 7</td>
<td>72 (49.6 %)</td>
<td>0.208</td>
</tr>
</tbody>
</table>

Complexity and Fitness Metric of the Spaghetti Model Generated from the entire Event Log as well as the Six Clusters Generated by K-medoid Algorithm using LCS as the Distance Metric.
## Experimental Results

<table>
<thead>
<tr>
<th></th>
<th>Cyclomatic Complexity ( %age decrease in complexity of clusters)</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Model</strong></td>
<td>143 (-)</td>
<td>0.017</td>
</tr>
<tr>
<td><strong>Cluster 2</strong></td>
<td>89 ( 37.7 %)</td>
<td>0.004</td>
</tr>
<tr>
<td><strong>Cluster 3</strong></td>
<td>93 ( 34.9 %)</td>
<td>0.059</td>
</tr>
<tr>
<td><strong>Cluster 4</strong></td>
<td>63 ( 55.9 %)</td>
<td>0.328</td>
</tr>
<tr>
<td><strong>Cluster 5</strong></td>
<td>97 (32.1 %)</td>
<td>0.063</td>
</tr>
<tr>
<td><strong>Cluster 6</strong></td>
<td>78 (45.49 %)</td>
<td>0.052</td>
</tr>
<tr>
<td><strong>Cluster 7</strong></td>
<td>86 (39.8 %)</td>
<td>0.078</td>
</tr>
</tbody>
</table>

Complexity and Fitness Metric of the Spaghetti Model Generated from the entire Event Log as well as the Six Clusters Generated by K-medoid Algorithm using DTW as the Distance Metric.
## Determine Goodness of Process Models

<table>
<thead>
<tr>
<th>Dataset No.</th>
<th>Iteration No.</th>
<th>Weighted Complexity</th>
<th>Weighted Fitness</th>
<th>G_Ratio</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>90.98</td>
<td>0.190</td>
<td>$2.08 \times 10^{-03}$</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>92.38</td>
<td>0.158</td>
<td>$1.71 \times 10^{-03}$</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>90.91</td>
<td>0.227</td>
<td>$2.49 \times 10^{-03}$</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>99.43</td>
<td>0.275</td>
<td>$2.08 \times 10^{-03}$</td>
<td>Selected</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>100.99</td>
<td>0.205</td>
<td>$2.7 \times 10^{-03}$</td>
<td>Selected</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>105.8</td>
<td>0.213</td>
<td>$2.01 \times 10^{-03}$</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>92.05</td>
<td>0.125</td>
<td>$1.35 \times 10^{-03}$</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>91.39</td>
<td>0.106</td>
<td>$1.15 \times 10^{-03}$</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>93.14</td>
<td>0.218</td>
<td>$2.33 \times 10^{-03}$</td>
<td>Selected</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>81.36</td>
<td>0.394</td>
<td>$4.84 \times 10^{-03}$</td>
<td>Selected</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>85.57</td>
<td>0.270</td>
<td>$3.15 \times 10^{-03}$</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>85.40</td>
<td>0.211</td>
<td>$2.47 \times 10^{-03}$</td>
<td>-</td>
</tr>
</tbody>
</table>
### Self Loop Analysis

#### Experimental Analysis and Results

### Self Loop

<table>
<thead>
<tr>
<th>Activity</th>
<th>Main Model</th>
<th>Cluster 1</th>
<th>Cluster 2</th>
<th>Cluster 3</th>
<th>Cluster 4</th>
<th>Cluster 5</th>
<th>Cluster 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASS</td>
<td>28, CCC/15</td>
<td>2, ATT/1</td>
<td>5, CCC/1</td>
<td>8, CCC/6</td>
<td>8, QAC/3</td>
<td>5, CCC/7</td>
<td>-,-</td>
</tr>
<tr>
<td>ATT</td>
<td>26, FLA/116</td>
<td>24, FLA/6</td>
<td>20, FLA/6</td>
<td>102, FLA/40</td>
<td>48, FLA/18</td>
<td>70, FLA/43</td>
<td>2, FLA/3</td>
</tr>
<tr>
<td>BLO</td>
<td>152, CCC/39</td>
<td>4, CCC/3</td>
<td>18, CCC/4</td>
<td>72, DEP/18</td>
<td>20, CCC/4</td>
<td>38, CCC/11</td>
<td>-,-</td>
</tr>
<tr>
<td>CCC</td>
<td>6776, FLA/250</td>
<td>524, SNR/60</td>
<td>875, WHI/37</td>
<td>3119, DEP/141</td>
<td>1345, SNR/151</td>
<td>371, FLA/60</td>
<td>42, SUR/53</td>
</tr>
<tr>
<td>CFB</td>
<td>26, CCC/16</td>
<td>1, CCC/1</td>
<td>6, CCC/6</td>
<td>11, CCC/6</td>
<td>-,-</td>
<td>4, CCC/3</td>
<td>4,-</td>
</tr>
<tr>
<td>CFC</td>
<td>1,-</td>
<td>-,-</td>
<td>1,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
</tr>
<tr>
<td>CFS</td>
<td>151, CFT/14</td>
<td>66, CFT/6</td>
<td>4,-</td>
<td>14, SRV/1</td>
<td>-,-</td>
<td>66, CFT/8</td>
<td>1,-</td>
</tr>
<tr>
<td>CFT</td>
<td>38, CCC/15</td>
<td>8, CFS/3</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>29, CCC/10</td>
<td>-,-</td>
</tr>
<tr>
<td>COM</td>
<td>2, QAC/3</td>
<td>-,-</td>
<td>QAC/1</td>
<td>1, QAC/1</td>
<td>1, QAC/1</td>
<td>-,-</td>
<td>-,-</td>
</tr>
<tr>
<td>DEP</td>
<td>704, CCC/110</td>
<td>9, SNR/2</td>
<td>21, CCC/3</td>
<td>576, CCC/83</td>
<td>41, CCC/6</td>
<td>57, CCC/17</td>
<td>CFL/1</td>
</tr>
<tr>
<td>FLA</td>
<td>1612, ATT/464</td>
<td>101, CCC/32</td>
<td>93, ATT/25</td>
<td>614, ATT/168</td>
<td>228, ATT/48</td>
<td>557, ATT/186</td>
<td>19, ATT/12</td>
</tr>
<tr>
<td>ISC</td>
<td>1, SNU/1</td>
<td>-,-</td>
<td>1, SUN/1</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>SUA/1</td>
</tr>
<tr>
<td>KEY</td>
<td>96, CCC/18</td>
<td>20, CCC/4</td>
<td>6, WHI/1</td>
<td>19, FLA/3</td>
<td>3, VER/1</td>
<td>44, CCC/10</td>
<td>4, CCC/2</td>
</tr>
<tr>
<td>OPS</td>
<td>1, PLA/3</td>
<td>-,-</td>
<td>PLA/1</td>
<td>-,-</td>
<td>1,-</td>
<td>-,-</td>
<td>PLA/2</td>
</tr>
<tr>
<td>PRI</td>
<td>3, TAR/2</td>
<td>-,-</td>
<td>TAR/2</td>
<td>-,-</td>
<td>-,-</td>
<td>1,-</td>
<td>2,-</td>
</tr>
<tr>
<td>PRO</td>
<td>3, COM/1</td>
<td>-,-</td>
<td>COM/1</td>
<td>2,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
</tr>
<tr>
<td>QAC</td>
<td>9, ASS/7</td>
<td>3, ASS/1</td>
<td>-,-</td>
<td>1, ASS/2</td>
<td>5, ASS/4</td>
<td>1,-</td>
<td>1,-</td>
</tr>
<tr>
<td>RES</td>
<td>1, SRR/2</td>
<td>1,-</td>
<td>SRR/1</td>
<td>-,-</td>
<td>-,-</td>
<td>SRR/1</td>
<td>-,-</td>
</tr>
<tr>
<td>SEE</td>
<td>3,-</td>
<td>-,-</td>
<td>-,-</td>
<td>1,-</td>
<td>1,-</td>
<td>1,-</td>
<td>-,-</td>
</tr>
<tr>
<td>SRR</td>
<td>-,-</td>
<td>RFF/6</td>
<td>-,-</td>
<td>-,-</td>
<td>RFF/3</td>
<td>RES/3</td>
<td>RFF/2</td>
</tr>
<tr>
<td>SUM</td>
<td>15, CCC/7</td>
<td>1, CCC/2</td>
<td>3, CCC/1</td>
<td>5, CCC/1</td>
<td>2, CCC/3</td>
<td>4, ASS/1</td>
<td>-,-</td>
</tr>
<tr>
<td>TAR</td>
<td>21, CCC/6</td>
<td>-,-</td>
<td>4, CCC/1</td>
<td>12, CCC/2</td>
<td>1, FLA/2</td>
<td>4, CCC/1</td>
<td>1,-</td>
</tr>
<tr>
<td>VER</td>
<td>6, CCC/10</td>
<td>2,-</td>
<td>2,-</td>
<td>1, CCC/2</td>
<td>-,-</td>
<td>PRO/1</td>
<td>1,-</td>
</tr>
<tr>
<td>WHI</td>
<td>71, CCC/40</td>
<td>5, CCC/1</td>
<td>4, CCC/5</td>
<td>21, CCC/16</td>
<td>4, WHI/5</td>
<td>32, CCC/12</td>
<td>5, CCC/10</td>
</tr>
<tr>
<td>Activity</td>
<td>Main Model</td>
<td>Cluster 1</td>
<td>Cluster 2</td>
<td>Cluster 3</td>
<td>Cluster 4</td>
<td>Cluster 5</td>
<td>Cluster 6</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-----------</td>
<td>-----------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>ASS</td>
<td>28, CCC/15</td>
<td>2, ATT/1</td>
<td>5, CCC/1</td>
<td>8, CCC/6</td>
<td>8, QAC/3</td>
<td>5, CCC/7</td>
<td>-</td>
</tr>
<tr>
<td>ATT</td>
<td>266, FLA/116</td>
<td>24, FLA/6</td>
<td>20, FLA/6</td>
<td>102, FLA/40</td>
<td>48, FLA/18</td>
<td>70, FLA/43</td>
<td>2, FLA/3</td>
</tr>
<tr>
<td>BLO</td>
<td>152, CCC/39</td>
<td>4, CCC/3</td>
<td>18, CCC/4</td>
<td>72, DEP/18</td>
<td>20, CCC/4</td>
<td>38, CCC/11</td>
<td>-</td>
</tr>
<tr>
<td>CCC</td>
<td>6776, FLA/250</td>
<td>524, SNR/60</td>
<td>875, WHI/37</td>
<td>3119, DEP/141</td>
<td>1345, SNR/151</td>
<td>871, FLA/60</td>
<td>42, SUR/53</td>
</tr>
<tr>
<td>CFB</td>
<td>26, CCC/16</td>
<td>1, CCC/1</td>
<td>6, CCC/6</td>
<td>11, CCC/6</td>
<td>-</td>
<td>4, CCC/3</td>
<td>4,-</td>
</tr>
<tr>
<td>CFC</td>
<td>1,-</td>
<td>1,-</td>
<td>1,-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CFS</td>
<td>151, CFT/14</td>
<td>66, CFT/6</td>
<td>4,-</td>
<td>14, SRV/1</td>
<td>-</td>
<td>66, CFT/8</td>
<td>1,-</td>
</tr>
<tr>
<td>CFT</td>
<td>38, CCC/15</td>
<td>8, CFS/3</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>29, CCC/10</td>
<td>-,-</td>
</tr>
<tr>
<td>COM</td>
<td>2, QAC/3</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
</tr>
<tr>
<td>DEP</td>
<td>704, CCC/110</td>
<td>9, SNR/2</td>
<td>21, CCC/3</td>
<td>576 , CCC/83</td>
<td>41, CCC/6</td>
<td>57, CCC/17</td>
<td>CFL/1</td>
</tr>
<tr>
<td>FLA</td>
<td>1612, ATT/464</td>
<td>101, CCC/32</td>
<td>93, ATT/25</td>
<td>614, ATT/168</td>
<td>228, ATT/48</td>
<td>557, ATT/186</td>
<td>19, ATT/12</td>
</tr>
<tr>
<td>ISC</td>
<td>1, SNU/1</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
</tr>
<tr>
<td>KEY</td>
<td>96, CCC/18</td>
<td>20, CCC/4</td>
<td>6, WHI/1</td>
<td>19, FLA/3</td>
<td>3, VER/1</td>
<td>44, CCC/10</td>
<td>4, CCC/2</td>
</tr>
<tr>
<td>OPS</td>
<td>1, PLA/3</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
</tr>
<tr>
<td>PRI</td>
<td>3, TAR/2</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
</tr>
<tr>
<td>PRO</td>
<td>3, COM/1</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
</tr>
<tr>
<td>QAC</td>
<td>9, ASS/7</td>
<td>3, ASS/1</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
</tr>
<tr>
<td>RES</td>
<td>1, SRU/2</td>
<td>1,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
</tr>
<tr>
<td>SRR</td>
<td>3,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
<td>-,-</td>
</tr>
<tr>
<td>SUM</td>
<td>15, CCC/7</td>
<td>1, CCC/2</td>
<td>3, CCC/1</td>
<td>5, CCC/1</td>
<td>2, CCC/3</td>
<td>4, ASS/1</td>
<td>-,-</td>
</tr>
<tr>
<td>TAR</td>
<td>21, CCC/6</td>
<td>-,-</td>
<td>4, CCC/1</td>
<td>12, CCC/2</td>
<td>1, FLA/2</td>
<td>4, CCC/1</td>
<td>-,-</td>
</tr>
<tr>
<td>VER</td>
<td>6, CCC/10</td>
<td>2,-</td>
<td>-,-</td>
<td>1, CCC/2</td>
<td>-,-,-</td>
<td>2, CCC/5</td>
<td>-,-</td>
</tr>
<tr>
<td>WHI</td>
<td>71, CCC/49</td>
<td>5, CCC/1</td>
<td>4, CCC/5</td>
<td>21, CCC/16</td>
<td>4, WHI/5</td>
<td>32, CCC/12</td>
<td>5, CCC/10</td>
</tr>
</tbody>
</table>
Reopen Analysis

Experimental Analysis and Results

Anvaya
Activity Frequency Analysis

![Graph showing activity frequency analysis with different clusters and activities.]

- **Cluster 1**
- **Cluster 2**
- **Cluster 3**
- **Cluster 4**
- **Cluster 5**
- **Cluster 6**
Event Analysis

The figure shows a histogram of events per case across different clusters. The x-axis represents the number of events per case, and the y-axis represents the percentage of cases. The bars are color-coded to indicate different clusters:

- **main model**: navy blue
- **cluster 1**: blue
- **cluster 2**: cyan
- **cluster 3**: yellow
- **cluster 4**: green
- **cluster 5**: red
- **cluster 6**: brown

The distribution varies across the clusters, with some clusters showing a higher concentration of events in specific ranges.
Bottlenecks

Experimental Analysis and Results

Bottlenecks
The bottlenecks found in clusters (not observed in main model) taking mean time greater than 1000 days are:

- ASS $\rightarrow$ SNR, ATT $\rightarrow$ SNR, CFB $\rightarrow$ SNR, TAR $\rightarrow$ SNR
- QAC $\rightarrow$ SNR
- CCC $\rightarrow$ REF, CCC $\rightarrow$ REW, CCC $\rightarrow$ REX
Outline

1. Research Motivation and Aim
2. Research Framework and Solution Approach
3. Experimental Analysis and Results
4. Limitations and Future Work
5. Conclusion
6. References
Limitations and Future Work

- Small dataset.
- Apply the proposed technique on datasets of different domains.
- K medoid algorithm is sensitive to initial cluster center assignment.
- Running time of clustering algorithm is high.
- The process analyst has to provide number of clusters (k) as input.
Outline

1. Research Motivation and Aim
2. Research Framework and Solution Approach
3. Experimental Analysis and Results
4. Limitations and Future Work

5. Conclusion
6. References
Conclusion

• Novel approach for clustering structurally similar traces to simplify complex models.

• Adaptation of k medoid with two different distance metrics: LCS and DTW similarity.

• Algorithm to automatically determine the goodness of process models.

• Experimental results show that process models generated from clusters are less complex and more fitter than the main model.

• Clustering enables better analysis making it easier to identify bottlenecks, study reopening of bugs and loops.
1. Research Motivation and Aim
2. Research Framework and Solution Approach
3. Experimental Analysis and Results
4. Limitations and Future Work
5. Conclusion
6. References
W.M.P. van der Aalst  
Process-Aware Information Systems: Lessons to Be Learned from Process Mining  

Wil M. P. van der Aalst and Christian W. Günther  
Finding Structure in Unstructured Processes: The Case for Process Mining  
Seventh International Conference on Application of Concurrency to System Design, 2007

R. P. Jagadeesh Chandra Bose and Wil M. P. van der Aalst  
Context Aware Trace Clustering: Towards Improving Process Mining Results  
Proceedings of the {SIAM} International Conference on Data Mining, {SDM} 2009
References II

Ferreira, Diogo and Zacarias, Marielba and Malheiros, Miguel and Ferreira, Pedro
Approaching Process Mining with Sequence Clustering: Experiments and Findings

Gabriel M. Veiga and Diogo R. Ferreira
Understanding Spaghetti Models with Sequence Clustering for ProM

Minseok Song and C.W. Günther
Trace Clustering in Process Mining
Business Process Management Workshops, {BPM} 2008 International Workshops.
Rozinat, A. and Aalst.
Conformance Testing: Measuring the Fit and Appropriateness of Event Logs and Process Models
First International Workshop on Business Process Intelligence (BPI'05)

Halverson, Christine A. and Ellis, Jason B. and Danis, Catalina and Kellogg, Wendy A.
Designing Task Visualizations to Support the Coordination of Work in Software Development
Emad Shihab, Akinori Ihara, Yasutaka Kamei, Walid M. Ibrahim, Masao Ohira, Bram Adams, Ahmed E. Hassan and Ken-ichi Matsumoto

Studying Re-opened Bugs in Open Source Software
Springer, 2013

Thomas Zimmermann, Nachiappan Nagappan, Philip J. Guo and Brendan Murphy

Characterizing and Predicting Which Bugs Get Reopened
Proceedings of the 34th International Conference on Software Engineering (ICSE 2012 SEIP Track)