Usage Behaviour Modelling
accuracy iobserve-analysis

1 General

Code: `https://github.com/Nicolas-Boltz/iobserve-analysis-JSS-2020/tree/scalability-usability-1`
Experiment data and results: `https://github.com/research-iobserve/iobserve-architectural-runtime-models-JSS-2020/tree/master/scalability_experiment`

2 Experiments:

The experiments on accuracy of usage behaviour modelling with iobserve-analysis is split up in two sub-experiments. **Detecting and Representing User Behaviours** evaluates if simple or nested patterns in the user behaviour are detected correctly and whether the created Usage Model represents the patterns correctly. Additionally the accuracy of detecting and representing workload specifications is evaluated. **Detecting User Groups** evaluates the clustering of similar usage behaviour into user groups.

3 Detecting and Representing User Behaviour

3.1 Detecting and Representing Behaviour Patterns

To evaluate the accuracy of finding patterns in the user behaviour, we define test cases for each of the possible patterns. For each execution of a test case the usage behaviour to be analysed, as well as a corresponding correct reference usage model, is created according to the constraints of the test case:

- **Simple Sequence**: A single call sequence with 1 to 5 calls, without loops or branches.
- **Simple Branch**: A branch with 2 to 5 transitions, each containing a single call. After the branch, there is a single collective call.
- **Simple Loop**: A loop with 2 to 5 iterations, containig a call sequence with 1 to 5 calls.
- **Overlapping Iteration**: A loop with 2 OR 3 iterations. Call sequences that share 4 to 6 overlapping calls. One iterated sequence consists of more calls than the other.

- **Branch Within Branch**: A branch with 2 to 3 branch transitions. Each transition contains a call sequence with a single call, followed by another branch with 2 to 3 branch transitions, containing a call sequence with a single call.
- **Loop Within Branch**: A branch with 2 to 3 branch transitions, each containing a call and a loop with 2 to 3 iterations. The loop is made up of a call sequence with a single call.
- **Loop Within Loop**: A loop with 2 to 4 iterations, containing a loop with 2 to 4 iterations followed by a call sequence with 1 to 2 calls. The internal loop contains a call sequence with a single call.
- **Branch Within Loop**: A loop with 2 to 3 iterations, containing a branch with two branch transitions. Each transition contains a call sequence with a single call.

The amount of user sessions in the created usage behaviour also depends on the test case:
Simple Sequence, Simple Loop, Overlapping Iteration and Nested Loops create a usage behaviour with between 1 - 200 user sessions.
Simple Branch, Nested Branches and Loop Within Branch the amount of user sessions has to be a minimum of $number\ of\ branch\ transitions * 10$ so there is at least one session covering each branch transition. This results in 20 - 200 user sessions.
Branch Within Loop the amount of user sessions has to be $2^{number\ of\ Loops} * user\ Factor$ so each branch transition in the loop has at least one session covering it. This results in 4 - 200 user sessions.
To achieve a good coverage of possible combinations of constraints, each test case is executed 5000 times.
After a run of iobserve-analysis, the resulting Usage Model is compared to the reference model, created by the test case. As metrics to measure similarity we use the Jaccard Coefficient (JC) and the Spearman?s Rank Correlation Coefficient (SRCC). Because the approach claims to create exact representations of the user behaviour we only consider a JC and SCC of 1 as an exact representation.

3.2 Results

RQ	JC			SCC		
	Min.	Max.	Mean	Min.	Max.	Mean
Simple Sequence	1.0	1.0	1.0	1.0	1.0	1.0
Simple Branch	1.0	1.0	1.0	1.0	1.0	1.0
Simple Loop	1.0	1.0	1.0	1.0	1.0	1.0
Overlapping Iteration	1.0	1.0	1.0	1.0	1.0	1.0
Nested Branches	1.0	1.0	1.0	1.0	1.0	1.0
Loop Within Branch	1.0	1.0	1.0	1.0	1.0	1.0
Nested Loops	1.0	1.0	1.0	1.0	1.0	1.0
Branch Within Loop	1.0	1.0	1.0	1.0	1.0	1.0

3.3 Representing Workload Specifications

To evaluate the correct representation of workload specifications, we use the same setup as test case 'Simple Sequence'. The test case is executed with either an open workload specification or an closed workload specification.
Because we only use an approximation for the population count of closed workload specifications, but an exact measurement of the mean inter arrival time, we expect no deviation for open workload specifications. The think time of a closed workload is passed by the user and therefore not considered within the equation.
We measure the relative measurement error (RME) between the approach?s generated workload specification and a reference. The reference is obtained during the creation of the usage behaviour for the test case. A RME value close to 0 indicates a low measurement error and therefore good representation of the workload specification.

3.3.1 Results

Evaluated Workload Specification	**RME**
Open Workload Specification	0.0%
Closed Workload Specification	13.1%

4 Detecting User Groups

To evaluate the accuracy of detecting user groups within usage behaviour, we want to find out how accurately iobserve-analysis clustering assigns user sessions of the same user group (UG) to the same cluster. We define three exemplary behaviours of three user groups (Customer (CR), Store Manager (SEM), Stock Manager (SKM)).
Within the experiment we use three different user group mixes, where one user group is represented by double as many user sessions as the two other groups. This enables to evaluate whether the user group mix affects the detection accuracy. The call sequence of each user session is randomly generated according to the behaviour of its user group.
The 'variance in user groups' value defines in how far the clustering algorithm is allowed to vary the amount of clusters from a predefined amount. A variance value of 0 allows for no variation, which in our case is 3 fixed clusters, because of the 3 defined user groups. A variance value of 10 allows for a variation of ± 10 from the predefined value, which in our case is between 2 and 13 clusters (2 because the minimum amount of clusters is 2). The predefined amount of clusters is normally gathered by first reading an existing UsageModel, but for this experiment is fixed to 3.
All inputs for this experiment are generated automatically, no external inputs, like models or monitoring data are needed.
As metrics we measure the percentage of misclassified user sessions (MC) and sum of the squared error (SSE).

4.1 Results

MC and SSE presented here are the average results from 100 runs of the setup. Values in columns C1, C2, C3 (and C4) only show an single exemplary result of one clustering run.

4.1.1 variance in user groups = 0

UG	UGM	C1	C2	C3	MC	SSE
CR	50%	0	4000	0		
SKM	25%	0	0	2000	0.0%	177.039830
SEM	25%	2000	0	0		
CR	25%	0	0	2000		
SKM	50%	4000	0	0	0.0%	190.368080
SEM	25%	0	2000	0		
CR	25%	2000	0	0		
SKM	25%	0	2000	0	0.0%	202.309114
SEM	50%	0	0	4000		

4.1.2 variance in user groups = 10

UG	UGM	C1	C2	C3	C4	MC	SSE
CR	50%	0	4000	0	0		
SKM	25%	2000	0	0	0	6.008625%	141.317894
SEM	25%	0	0	641	1359		
CR	25%	0	0	0	2000		
SKM	50%	886	3114	0	0	10.737249%	134.331437
SEM	25%	0	0	2000	0		
CR	25%	0	0	0	2000		
SKM	25%	0	0	2000	0	13.772875%	159.150583
SEM	50%	1215	2785	0	0		