Preparing for Test
(part 2)

Bojan Cukic

Department of Computer Science and Electrical Engineering
West Virginia University
The big picture

List Associated Systems

- Define Necessary Reliability
- Develop Operational Profiles

Prepare for Test

- Execute Tests
- Apply Failure Data to Guide Decisions

Requirements and architecture
Design and Implementation
Test & Validation
Procedure

Procedure involves:

- Preparing test cases.
- Preparing test procedures.
- Recording time for failure acknowledgement purposes.
Preparing Test Cases

- Recording real inputs in the field.
- Might mirror “old” system usage, old system loads.
- Should not miss rarely occurring but critical runs.
- Steps.
  1. Estimating the number of new test cases needed to test the current release.
  2. Allocating tests among systems to be tested.
  3. Allocating NEW test cases.
  4. Specifying new test cases.
  5. Merging old and new test cases (if multiple releases).
Estimating # new test cases

The state of practice indicates:
- Take a minimum of
  - How much time you have available
    (The time between the stabilization of requirements and the date to start testing).
  - How much is the development of new tests going to cost.

Test case development usually takes 5% - 10% of the overall budget!
- Grows with required reliability level.
Example

- Available time = 600[h], staff = 3.5[person], average time to develop test = 3 hours
  - 2100[staff hours] / 3 = 700 tests

- Available budget = $2m, 10% devoted to test case development, cost of preparing a test = $250.
  - $200,000 / $250 = 800 tests

- Plan to develop 700 tests!

- Run a SANITY CHECK
  - # New tests >> # new operations (50 to 100 more, if frequently occurring operations).

- Experience.
Improvements

- Develop more tests in the given time.
  - Experience.
- Execute many more runs than test cases.
  - Have runs “evenly spaced” across an operation, i.e., equidistant in covering the “inputs” of an operation.
  - This maximizes the likelihood of finding faults.
- Improve (automate) test case generation.
  - Tools + research.
  - Random test case generators.
    - For example, randomly generate indirect variables within a test case.
  - Path coverage generators.
  - *Test trajectory generators.*
Allocating test cases to (sub)systems

- Allocate bulk of tests to the SYSTEM (product).
  - Adjust to subsystems to reflect changes.
- Base the selection on the operational usage.
- Allocate test cases to acquired components in the first release.
  - Exception in cases of substitution, later releases may require additional tests.
- Target risky components and frequently used components.

1. Convert operational profile to tabular representation.
   - Estimate execution probability.

2. Identify rare but critical operations and preassign the number of test cases to each.

3. Determine allocation probabilities for other new operations.

4. Preassign one new test to each new infrequent operation.

5. Assign remaining test cases in accordance with the occurrence probabilities.
500 preassigned test cases in the first release.

For release 2, depending on the amount of change and operational distribution, a different number would be assigned.
Specifying test cases

- Specify test cases in such a way that they “cover” the test space.
  - Long distance calls vs. local calls.
  - Within the same “level” (variable values likely to trigger same faults) assign values to variables randomly.

- Runs are defined by “ranges of values” assigned to variables.
  - Define the set of runs forming an “equivalence class”.
  - Try to cover all combinations of classes.

- Write test scripts.
  - Add new tests to the database and use for future releases.
## Equivalence classes

<table>
<thead>
<tr>
<th>Originator</th>
<th>Forwardee</th>
<th>Billing Type</th>
<th>Dialing Type</th>
<th>Originator Screening</th>
</tr>
</thead>
<tbody>
<tr>
<td>On screenlist</td>
<td>Local calling area</td>
<td>Flat rate</td>
<td>Standard</td>
<td>Yes</td>
</tr>
<tr>
<td>Not on screenlist</td>
<td>Within area code</td>
<td>Per call</td>
<td>Abbreviated</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Outside area code</td>
<td>Discount</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>International</td>
<td>Per minute</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>800 number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Other carrier</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wireless</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Detailing New Test Cases

.authorization

For each new test case, for each new operation:

- Randomly select equivalence class for each direct input variable.
  - Equal probability among possible choices.
  - Reject duplication of equivalence classes.
- Select value of direct input variable within each equivalence class.
  - Strongly failure prone (boundary value), and/or.
  - Random.
Procedure

Procedure involves

- Preparing test cases
- Preparing test procedures
- Recording time for failure acknowledgement purposes
Preparing Test Procedures

One test procedure for each operational mode must be developed.

Start with:
1. Operational mode’s operational profile for the first release.
2. Test operational profile (or usage profile) from previous releases, if they exist.
3. Account for the rare but critical events.
Example

Unadjusted OP, peak hours operational mode

<table>
<thead>
<tr>
<th>Operation</th>
<th>Occurrence Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice call, no pager, answer</td>
<td>0.22</td>
</tr>
<tr>
<td>Voice call, no pager, no answer</td>
<td>0.18</td>
</tr>
<tr>
<td>Voice call, pager, answer</td>
<td>0.18</td>
</tr>
<tr>
<td>Fax call</td>
<td>0.14</td>
</tr>
<tr>
<td>Voice call, pager, answer on page</td>
<td>0.12</td>
</tr>
<tr>
<td>Voice call, pager, no answer</td>
<td>0.1</td>
</tr>
<tr>
<td>Phone number entry</td>
<td>0.06</td>
</tr>
<tr>
<td>Recover from HW failure</td>
<td>0.000001</td>
</tr>
<tr>
<td>Add subscriber</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1</td>
</tr>
</tbody>
</table>

- These occurrence probabilities are different from the system level (no admin tasks).
- By assigning 2 out of 500 tests to “recover from HW failure”, we “increased” its probability 4000 times! *Its new probability is 0.004.*
- This factor is called *acceleration factor.*
  - Accelerated testing the only way to test rare events.
- How much did we change operational distribution?
  - About 0.4%
  - Changes between 0% and 25% appear to be reasonable.
**Adjusted OP, peak hours operational mode**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Initial Operational Profile</th>
<th>Final operational Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice call, no pager, answer</td>
<td>0.22</td>
<td>0.219</td>
</tr>
<tr>
<td>Voice call, no pager, no answer</td>
<td>0.18</td>
<td>0.179</td>
</tr>
<tr>
<td>Voice call, pager, answer</td>
<td>0.18</td>
<td>0.179</td>
</tr>
<tr>
<td>Fax call</td>
<td>0.14</td>
<td>0.139</td>
</tr>
<tr>
<td>Voice call, pager, answer on page</td>
<td>0.12</td>
<td>0.12</td>
</tr>
<tr>
<td>Voice call, pager, no answer</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Phone number entry</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>Recover from HW failure</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>Add subscriber</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1.04</strong></td>
<td><strong>1</strong></td>
</tr>
</tbody>
</table>

Column “Final Operational Profile indicates the test operational profile For testing of the first release of Fone Follower in the peak hours Operational mode.
Example: second release

<table>
<thead>
<tr>
<th>Operation</th>
<th>Occurrence Probability</th>
<th>Initial operational Profile</th>
<th>Final Operational profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>New operation A</td>
<td>0.1</td>
<td>0.1</td>
<td>0.25</td>
</tr>
<tr>
<td>New operation B</td>
<td>0.1</td>
<td>0.1</td>
<td>0.25</td>
</tr>
<tr>
<td>Voice call, no pager, answer</td>
<td>0.219</td>
<td>0.0438</td>
<td>0.1095</td>
</tr>
<tr>
<td>Voice call, no pager, no answer</td>
<td>0.179</td>
<td>0.0358</td>
<td>0.0895</td>
</tr>
<tr>
<td>Voice call, pager, answer</td>
<td>0.179</td>
<td>0.0358</td>
<td>0.0895</td>
</tr>
<tr>
<td>Fax call</td>
<td>0.139</td>
<td>0.0278</td>
<td>0.0695</td>
</tr>
<tr>
<td>Voice call, pager, answer on</td>
<td>0.12</td>
<td>0.024</td>
<td>0.06</td>
</tr>
<tr>
<td>Voice call, pager, no answer</td>
<td>0.1</td>
<td>0.02</td>
<td>0.05</td>
</tr>
<tr>
<td>Phone number entry</td>
<td>0.06</td>
<td>0.012</td>
<td>0.03</td>
</tr>
<tr>
<td>Recover from HW failure</td>
<td>0.004</td>
<td>0.0008</td>
<td>0.002</td>
</tr>
<tr>
<td>Add subscriber</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1</td>
<td>0.4</td>
<td>1</td>
</tr>
</tbody>
</table>

- Interaction factor 0.2 assumed when creating initial operational profile for the 2nd release.
- Final OP obtained by dividing initial OP with the sum (0.4).
Improvements

- Test automation
  - test management
  - test failure identification
    - automated test oracles, the field still in its infancy
- Improving test efficacy
  - \( n = \frac{1}{f_i} \cdot \frac{df_i}{dt} \), where
    - \( n \) is the relative rate of reduction in failure intensity
    - \( f_i \) is the failure intensity
    - \( t \) is the execution time
  - Define the set of runs forming an “equivalence class”
Automated Generation of Test Trajectories

Brian J. Taylor
Institute for Scientific Research, Inc.

Bojan Cukic
LDCSEE Department
West Virginia University
Introduction

- Safety assessment of safety-critical systems requires a large number of test cases.
  - Experience envelopes, system envelopes
- Testing of closed-loop control systems frequently requires data trajectories
  - Trajectory: sequence (usually time series) of data points (sensor reading snapshots, for example)
  - Each point’s value depends upon the previous data point(s).
Motivation

- Most Automated Test Data Generation algorithms do not handle trajectories of data
- This research proposes a test trajectory generation algorithm.
  - Uses regression analysis to *predict new trajectories from a small set of existing trajectories*.
    - Existing trajectories collected from simulators, observed usage of older versions, handcrafted trajectories.
  - Algorithm composed of several modules allowing for ‘refine and replace’
- Case Study: SFDIA Flight Controller
  - Sensor Failure Detection, Identification, and Accommodation
Test Data Generation (TDG)

- TDGs facilitate generation of large volumes of data
- Markov chain based approaches
  - At each point in the *scenario*, the next point is chosen probabilistically.
  - Applied to telecommunications sw [Arvitzer, Weyuker 95].
  - Not suitable for the generation of long trajectories.
- Pathwise Generators
  - Generates data which tests an execution path through a system.
  - Problem: safety assessment requires extensive coverage of the input domain, rather than only code coverage.
- Random Test Generators
  - Uses an inexpensive random number generator to chose legal program inputs.
  - Problem: Generated data is uncorrelated and of little use for time series.
Algorithm Overview

Collect Test Trajectories

Pre-Process → Cluster Independent Variables → Develop a Regressive Model

Trajectory Generation

Perturb Independent Variables → Apply Model

Acceptability Testing → Valid Trajectory

Model Generation
Test Trajectory Generation

- Existing subset of test trajectories used to build regressive models representing relations between independent (controllable) and dependent (non-controllable) variable sequences.
- New independent variable sequences, given as inputs to the regressive model, used to generate dependent variable sequences.
Collecting & Processing of Existing Trajectories

- Record Sample Test Trajectories
  - Independent variables - Controllable mathematical variables (pilot/autopilot command) inputs.
  - Dependent variables - Variables used in testing.

- What is a Trajectory?
  - Multidimensional sequence (time series) of data.
  - Independent (controllable) variables form trajectories.
  - Dependent variables also form trajectories.

- What is generated?
  - Sequences of dependent variables.

- Processing the existing trajectories
  - Truncation, Interpolation, Noise Removal, Conversion
Cluster trajectories of independent variables into related groups
Cluster upon the independent variables
  - Goal: To develop trajectories of “similar” independent variables

Choosing clustering parameters
  - Inputs should have some variance, low variance may not provide distinguishing characteristics.
  - Limit to a few parameters, too many parameters complicate the clustering process.
  - Scale parameters
    - Percent normalization, Range normalization, Zero-Mean
Clustering: Treatment of Outliers

- An outlier is a trajectory which lies on the outer bounds of the cluster.
  - Outliers may be eliminated to improve uniformity.
  - Outliers may be included into the cluster to increase the parameter variance.
  - Decision is application dependent.
Clustering: Techniques

- Hierarchical vs Non-Hierarchical
  - Non-Hierarchical: trajectories are assigned into $k$ arbitrary clusters until the intra-group variances reach a minimum.
  - Hierarchical: the number of groups, $n$, is decided before the clustering begins and the trajectories are divided into these $n$ groups.

- Interpreting the Results
  - Clusters should have at least 4-5 member trajectories.
  - Select a Representative Trajectory from a cluster
    - The Centroid is the ‘average’ trajectory for a cluster.
    - The representative component is the trajectory of INDEPEDENT data which lies closest to the centroid.
    - The representative trajectory is the test trajectory which corresponds to the representative component.
Regressive Models: Development

- Simple-Linear, 2-Variable, 3-Variable, 4-Variable Multiple Linear, Parametric models, etc.
- Develop a large collection of models, if time permits.
- Use Discriminant Analysis to determine the independent variables that are the best predictors.
- Use cross-correlations between actual values of dependent variables and the predicted values to determine the best model.
- Smooth predicted signals to improve regressive model output
  - Averaging, triangular smoothing, Gaussian.
Regressive Model: Choosing the Best Model

- Selection of model is based upon:
  - Average time to generate a trajectory using the regressive model
    - *Faster - better*, facilitates faster testing
  - Average correlation for the model between actual trajectory and predicted trajectory
Generating New Trajectories: Perturbation

- Perturbation can be thought of as sliding a window of size $n$ along the trajectories of independent variables
  - Multiplication by a constant
  - Increasing time intervals
  - Adding in noise
  - Combinations

- Model coefficients are used for developing the new trajectory.
- Smoothing can be applied to generated trajectories, if needed.
Acceptance Rule(s) and Reasonableness Checks

- Discriminate between automatically generated valid and invalid test trajectories.
- Indicate the success rate of the model.
  - Indicate when to stop generation of new trajectories:
  - Falling below a certain acceptance limit.
  - Generated enough new trajectories.

Acceptance Rules Upon the Independent Variables

- Do the new perturbations stay within the clustered regions?
- Do the new perturbations violate allowed values of independent variables?

Acceptance Rules Upon the Dependent Variables

- System Specific domain predicates
  - Does the trajectory satisfy the input domain predicates?
The scheme provides Sensor Failure Detection, Identification, and Accommodation.

Intelligent fault tolerant flight control scheme can detect failures among the roll, pitch, and yaw rate gyros.

Faults masked through the application of analytical redundancy.
Case Study: Assessment of SFDIA

- How to automatically generate a large number of flight paths for the assessment of the SFDIA
  - Use recorded data from actual commercial jetliners.
  - Use of a model aircraft - record flight data.
  - *Use of an advanced flight simulator.*

- Three possible simulator approaches:
  - Human Pilot (User) flies simulator thousands of times
  - Program a flight simulator to generate thousands of flight paths
  - *Use of statistical methods to generate thousands of flight paths from a few dozen simulated/actual trajectories*

- Tools
  - AVDS : Aviator Visual Design Simulator
  - Matlab 5.0 : signal processing toolbox
Processing the Data

- Recorded over 20 variables
  - P, Q, R
  - aileron, elevator, rudder deflections
  - positional data
  - mach
- Truncated to minimum data set size
  - Typical time: 25 seconds
- Recorded 17 different flight maneuvers
Clustering the Data

- Clustered trajectories by three pilot inputs
  - Aileron, elevator, rudder deflections
  - Used Euclidean distance metric
# Regressive Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Description (independent variable(s))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>Simple Linear (roll)</td>
</tr>
<tr>
<td>Model 2</td>
<td>Simple Linear (pitch)</td>
</tr>
<tr>
<td>Model 3</td>
<td>Simple Linear (yaw)</td>
</tr>
<tr>
<td>Model 4</td>
<td>Simple Linear (mach)</td>
</tr>
<tr>
<td>Model 5</td>
<td>2-variable Multiple Linear (roll, pitch)</td>
</tr>
<tr>
<td>Model 6</td>
<td>2-variable Multiple Linear (roll, yaw)</td>
</tr>
<tr>
<td>Model 7</td>
<td>2-variable Multiple Linear (roll, mach)</td>
</tr>
<tr>
<td>Model 8</td>
<td>2-variable Multiple Linear (pitch, yaw)</td>
</tr>
<tr>
<td>Model 9</td>
<td>2-variable Multiple Linear (pitch, mach)</td>
</tr>
<tr>
<td>Model 10</td>
<td>2-variable Multiple Linear (yaw, mach)</td>
</tr>
<tr>
<td>Model 11</td>
<td>3-variable Multiple Linear (roll, pitch, yaw)</td>
</tr>
<tr>
<td>Model 12</td>
<td>3-variable Multiple Linear (roll, pitch, mach)</td>
</tr>
<tr>
<td>Model 13</td>
<td>3-variable Multiple Linear (roll, yaw, mach)</td>
</tr>
<tr>
<td>Model 14</td>
<td>3-variable Multiple Linear (pitch, yaw, mach)</td>
</tr>
<tr>
<td>Model 15</td>
<td>4-variable Multiple Linear (roll, pitch, yaw, mach)</td>
</tr>
</tbody>
</table>
Linear Correlation Results

<table>
<thead>
<tr>
<th>Man.</th>
<th>Roll Model Selected</th>
<th>Pitch Model Selected</th>
<th>Yaw Model Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>9</td>
<td>14</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Avg.</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
## Autoregressive Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Driving input</th>
<th>Model Orders</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 16</td>
<td>roll</td>
<td>p = 5, q = 5</td>
</tr>
<tr>
<td>Model 17</td>
<td>pitch</td>
<td>p = 5, q = 5</td>
</tr>
<tr>
<td>Model 18</td>
<td>yaw</td>
<td>p = 5, q = 5</td>
</tr>
<tr>
<td>Model 19</td>
<td>mach</td>
<td>p = 5, q = 5</td>
</tr>
<tr>
<td>Model 20</td>
<td>roll</td>
<td>p = 10, q = 5</td>
</tr>
<tr>
<td>Model 21</td>
<td>pitch</td>
<td>p = 10, q = 5</td>
</tr>
<tr>
<td>Model 22</td>
<td>yaw</td>
<td>p = 10, q = 5</td>
</tr>
<tr>
<td>Model 23</td>
<td>mach</td>
<td>p = 10, q = 5</td>
</tr>
<tr>
<td>Model 24</td>
<td>roll</td>
<td>p = 5, q = 10</td>
</tr>
<tr>
<td>Model 25</td>
<td>pitch</td>
<td>p = 5, q = 10</td>
</tr>
<tr>
<td>Model 26</td>
<td>yaw</td>
<td>p = 5, q = 10</td>
</tr>
<tr>
<td>Model 27</td>
<td>mach</td>
<td>p = 5, q = 10</td>
</tr>
<tr>
<td>Model 28</td>
<td>roll</td>
<td>p = 10, q = 10</td>
</tr>
<tr>
<td>Model 29</td>
<td>pitch</td>
<td>p = 10, q = 10</td>
</tr>
<tr>
<td>Model 30</td>
<td>yaw</td>
<td>p = 10, q = 10</td>
</tr>
<tr>
<td>Model 31</td>
<td>mach</td>
<td>p = 10, q = 10</td>
</tr>
</tbody>
</table>
Autoregressive Results

![Graph showing Yaw Rate vs. ARMA Model]

<table>
<thead>
<tr>
<th>Man.</th>
<th>ARMA Roll Model Selected</th>
<th>ARMA Pitch Model Selected</th>
<th>ARMA Yaw Model Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23</td>
<td>19</td>
<td>26</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td>19</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>29</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>24</td>
<td>27</td>
<td>24</td>
</tr>
<tr>
<td>7</td>
<td>18</td>
<td>27</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>9</td>
<td>20</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>27</td>
<td>16</td>
</tr>
<tr>
<td>11</td>
<td>16</td>
<td>27</td>
<td>16</td>
</tr>
<tr>
<td>12</td>
<td>27</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>13</td>
<td>24</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>14</td>
<td>18</td>
<td>19</td>
<td>30</td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>16</td>
<td>19</td>
<td>24</td>
</tr>
<tr>
<td>Avg.</td>
<td>16</td>
<td>19</td>
<td>16</td>
</tr>
</tbody>
</table>
Test Generation Speed

- Time to simulate 1 test case by a flight simulator
  - 25 seconds
- Time to generate 1 Linear Model test trajectory
  - 0.6 seconds
- Execution of a test case on SFDIA system takes around 2 seconds
  - 13 new flight maneuvers in 25 seconds
  - Acceptance tests executed fast, delay negligible.
  - Even at a 25% acceptance rate, 3 new flight maneuvers can be generated and tested at the time of 1 generated by the flight simulator
Model Advantages

- When applying linear models to the angular rates, the simple-linear approach was selected more often.
- The non-linear yaw angular rate was best predicted using the parametric models.
- Parametric models are faster. On average they performed twice as fast as linear models.
Acceptability Rates: Examples

- Do the perturbed independent variables exceed min/max thresholds?
  - Eliminated no generated trajectory
- Does perturbed independent variables remain in the cluster?
  - Eliminated 2% of the generated trajectories
- Do the generated trajectories correlate to the dependent variables in the cluster?
  - Eliminated 6% of the generated trajectories
    - 8% roll generations eliminated
    - 4% pitch generations eliminated
    - 7% yaw generations eliminated
Summary

- The Trajectory Generation algorithm is able to produce new test trajectories faster than traditional test trajectory collection.
- Experimental tools support the approach.
- Applicability may vary between different domains.
- Further Work
  - More detailed Acceptability Rules for the SFDIA scheme
    - Flying Quality Requirements
      - Dutch-Roll
      - Rolling Time Constant
      - Phugoid Damping
  - Investigation of Autoregressive-moving average, Non-Linear Regression
  - Analysis and improvement of the cost function
Discussion

Is a tool like this suitable for reliability testing?