Mutant testing data warehouse testdata

Peter Lichtenveldt

Testdag 2019

Utrecht

Who I am

- I studied Theoretical Chemistry
- I started in IT in 1989
- I worked at Nederlandse Spoorwegen
- I’ve worked for 20 years at ING
- TMap Next Certified Test Engineer

What am I going to tell you?

- Data warehouse in SAS
- ETL gets data from sources and integrates them
  - Filters, mappings, aggregations …
- Tool: Data Integration Studio
- Data are needed to test the functionality
- Anonymized production data, no test design
- How good are the testdata?

Mutant testing

- Mutant testing creates clones of the system ….
- … in which bugs are introduced
- The clones are called mutants
- The mutants are tested the same way the system has been tested
- How many mutants are killed?

Mutant testing our ETL

Create mutants of our ETL

Are the mutants killed by the testdata?

Mutant testing our test data

- We had two challenges
  - ETL takes a long time
    - … so testing many mutants would be very time-consuming
  - Parallel running was not an option in our environment
- Alternative approach: test the ***data***
  - Write queries that simulate part of the functionality
  - Create mutants that simulate common mistakes in DI Studio

Case 1: Columns with indicator values

- We have mappings of indicator columns
- Only possible values are ‘Y’ and ‘N’

Source		Target
• A	->	A_IND
• B	->	B_IND
• C	->	C_IND
• D	->	D_IND

Columns with indicator values : mutant 1

- Two mappings swapped

Source		Target
• A	->	B_IND
• B	->	A_IND
• C	->	C_IND
• D	->	D_IND

- If the number of columns is small, create all
- If the number of columns is large, take the adjacent ones

Indicator columns: mutant 2

- One mapping is a constant

Source		Target
• A	->	A_IND
•		B_IND ='N'
• C	->	C_IND
• D	->	D_IND

- Chance is not so big
- Take one or two

Indicator columns: bad testset

A	B	C	D
N	N	N	N
N	N	N	N
N	N	N	N
N	N	N	N

A_IND	B_IND	C_IND	D_IND
N	N	N	N
N	N	N	N
N	N	N	N
N	N	N	N

A_IND	B_IND	C_IND	D_IND
N	N	N	N
N	N	N	N
N	N	N	N
N	N	N	N

Indicator columns: good testset

A	B	C	D
Y	N	N	N
N	Y	N	N
N	N	Y	N
N	N	N	Y

A_IND	B_IND	C_IND	D_IND
N	Y	N	N
Y	N	N	N
N	N	Y	N
N	N	N	Y

A_IND	B_IND	C_IND	D_IND
Y	N	N	N
N	N	N	N
N	N	Y	N
N	N	N	Y

Results on indicator columns

- Result was that the testset was very strong
- The queries were very straightforward …
- ... so the result was trustworthy

Case 2: Conditions

- Records from sources are selected if they fulfill certain conditions
- But some joining was needed to simulate this
- This led to more complex queries
- Possible errors/mutants:
  - OR instead of AND
  - forget parentheses in p AND (q OR r)
  - forget condition

Conditions: bad testset

p	q	r
True	True	True
True	True	True
True	True	True
True	True	True

(p or q) and r	(p or q) or r	p or q and r	p or q
Y	Y	Y	Y
Y	Y	Y	Y
Y	Y	Y	Y
Y	Y	Y	Y

Conditions: good testset

p	q	r
False	True	True
True	False	True
True	True	False
True	True	True

(p or q) and r	(p or q) or r	p or q and r	p or q
Y	Y	Y	Y
Y	Y	Y	Y
N	Y	Y	Y
Y	Y	Y	Y

Results on conditions

- Result was that the testset was very poor
- But was the query correct?
- Risk: *regressio ad infinitum …*
- … better known as *Droste effect*

Conclusion

- Mutant testing can be applied to assess testdata that are not designed
- Create queries that simulate common mistakes
- Beware of the Droste effect

Thank you very much!

Any questions?