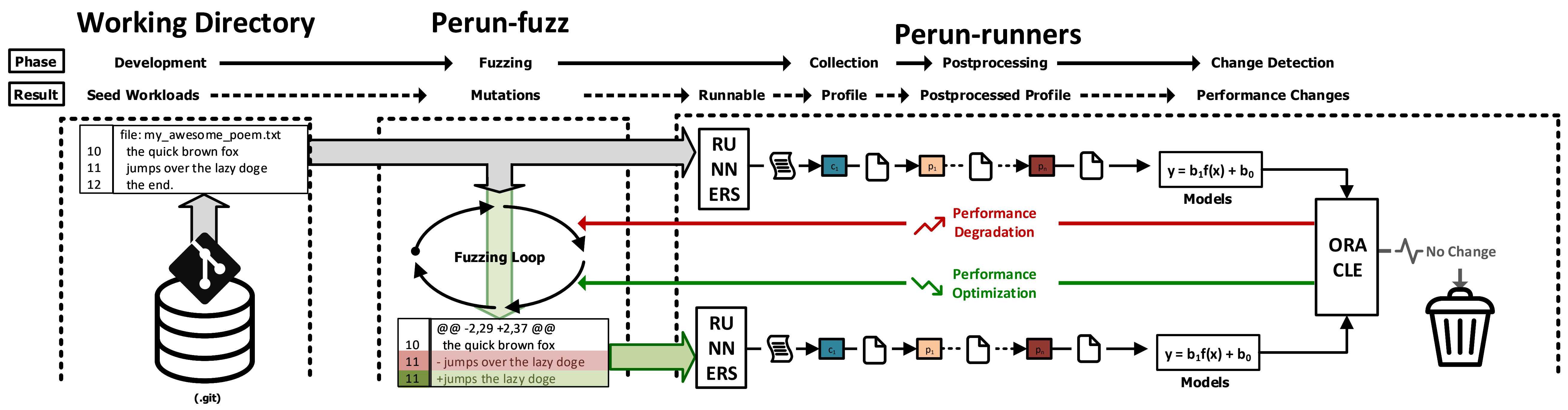
53 Fuzz testing of program performance

Matúš Liščinský
xlisci02@stud.fit.vutbr.cz

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2019

Methodology of Performance Fuzz Testing



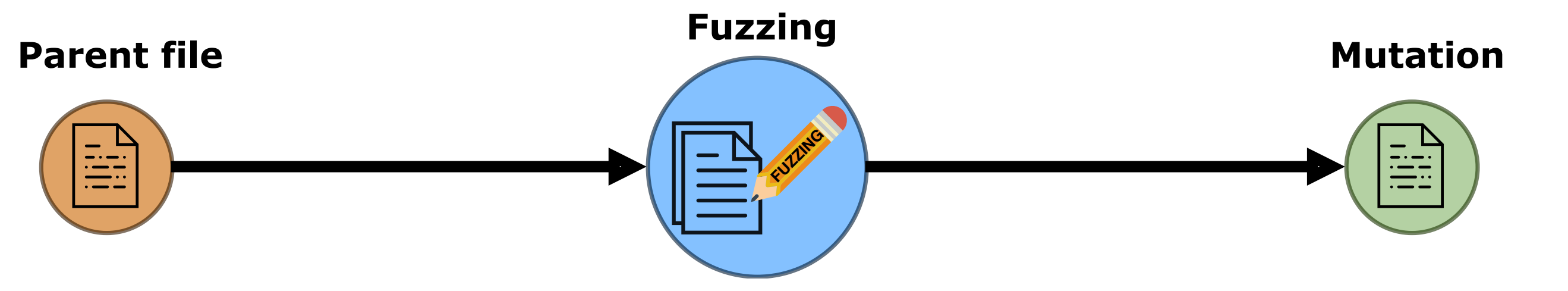
Informed parent selection

Parent files

15%
18%
25%
27%
33%

- Parents are rated by:
  1. Code **coverage**
  2. Caused degradation **ratio**
- Parents are divided into **weighted intervals** followed with **random selection**.

Mutation rules for Performance



- *text file rules:* (e.g. remove whitespace)
  "the quick brown fox" "thequickbrownfox"
- *binary file rules:* (e.g. add zero byte)
  "This is !binary!.\0" "This is !\0binary!.\0"
- *domain-specific file rules:* (e.g. remove attribute)
  <book id="bk1" pgs="457"> <book id="bk1" "457">

Experimental evaluation

- ReDoS inspired regular expressions:

regex	time degradation	coverage increase
\s+$	5.79x	15.59x
^(.*?,){10}P	2 897.23x	2 442.37x
**	940.44x	10 873.94x

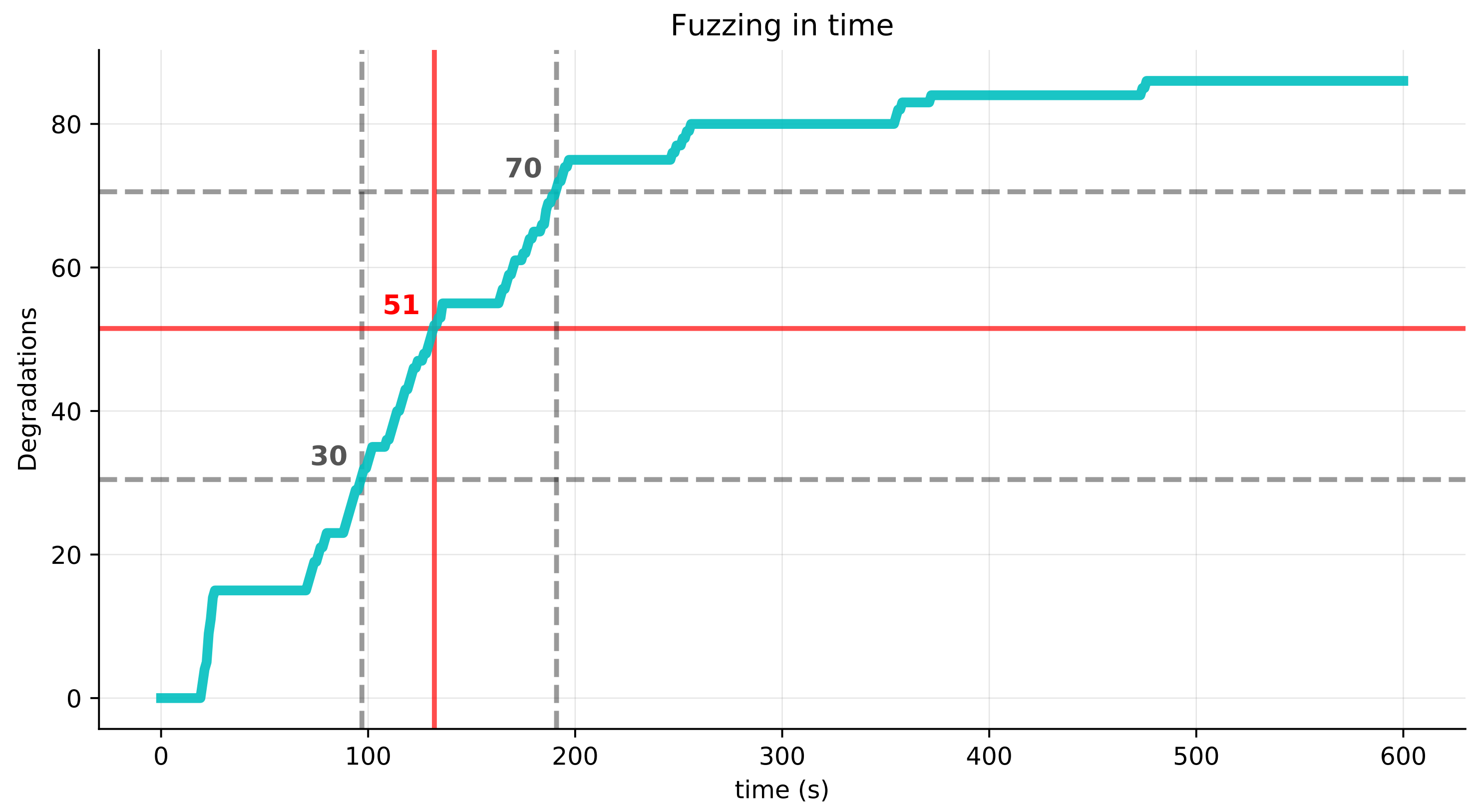
- Selected data structures:

structure	time degradation	coverage increase
unbalanced binary tree	9.28x	41.56x
std::list + std::find	14.01x	15.05x

** <html>.*?<head>.*?</head>.*?<body[^>]*>.*?</body>.*?</html>

Interpretation

Interpretation by (1) **time series**, and (2) **file diff**.



Acknowledgements

The work is supported by various projects, groups and companies including:

- **Red Hat**, Inc.
- **VeriFIT** group (BUT FIT)