Evaluating testing strategies for imaging software by means of Mutation Analysis

René Just & Franz Schweiggert

Ulm University, Germany

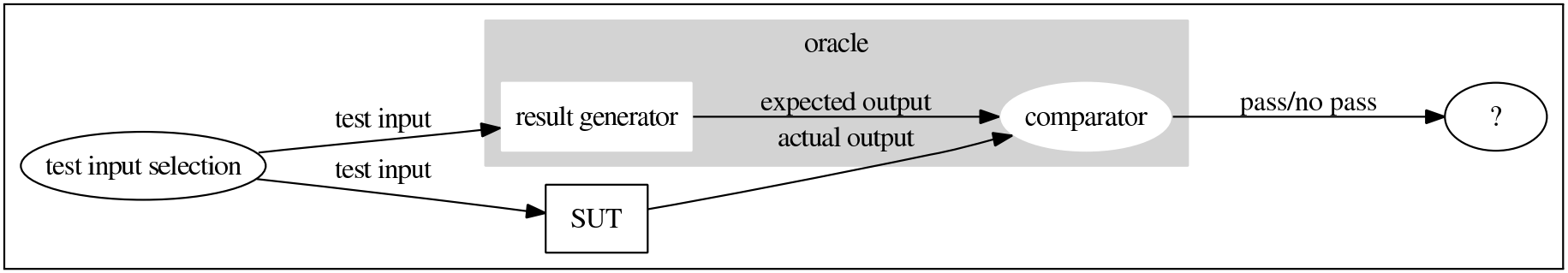
April 4, 2009

ulm university universität

uulm

Outline

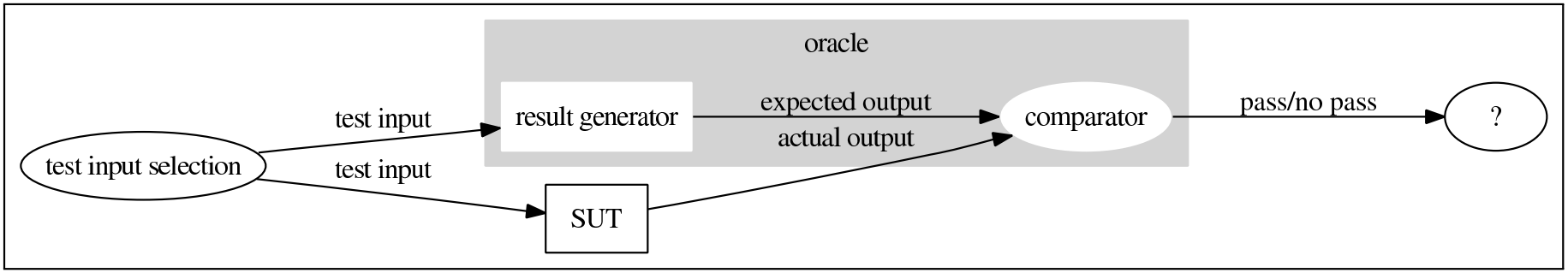
Testing Process



Testing Strategy consists of two parts:

1. method to generate test inputs.
2. (partial) oracle.

Testing Process



Testing Strategy consists of two parts:

1. method to generate test inputs.
2. (partial) oracle.

Partial oracles

Properties:

- exploit necessary conditions of SUT.
- aleviate the oracle problem.
- sufficient conditions cannot be verified.
- may produce false negative results.

Partial oracles

Properties:

- exploit necessary conditions of SUT.
- aleviate the oracle problem.
- sufficient conditions cannot be verified.
- may produce false negative results.

Evaluation is necessary!

Mutation Analysis

- based on an arbitrary (defect-free) program P: $\mathcal{I} \rightarrow \mathcal{O}$.
- systematic error seeding $\Rightarrow$ mutants M_i.
- testing strategy generates $x \in \tilde{\mathcal{I}} \subseteq \mathcal{I}$.
- M_i is equivalent to $P \Leftrightarrow M_i(x) = P(x) \; \forall \; x \in \mathcal{I}$.
- M_i is killed $\Leftrightarrow \exists x \in \tilde{\mathcal{I}} : \; M_i(x) \neq P(x)$.

Mutation Score

$$S = \frac{M_k(\mathit{Number\ of\ killed\ mutants})}{M_t(\mathit{Number\ of\ non-equivalent\ mutants})}$$

Mutation Analysis

- based on an arbitrary (defect-free) program P: $\mathcal{I} \rightarrow \mathcal{O}$.
- systematic error seeding $\Rightarrow$ mutants M_i.
- testing strategy generates $x \in \tilde{\mathcal{I}} \subseteq \mathcal{I}$.
- M_i is equivalent to $P \Leftrightarrow M_i(x) = P(x) \; \forall \; x \in \mathcal{I}$.
- M_i is killed $\Leftrightarrow \exists x \in \tilde{\mathcal{I}} : \; M_i(x) \neq P(x)$.

Mutation Score

$$S = \frac{M_k(Number\ of\ killed\ mutants)}{M_t(Number\ of\ non-equivalent\ mutants)}$$

Overview

Objectives:

- assess quality of partial oracles.
- compare class based and traditional mutants.
- compare results with real faults.

Preconditions:

- complex and object-oriented implementation.
- dissociate non-terminating mutants.

Overview

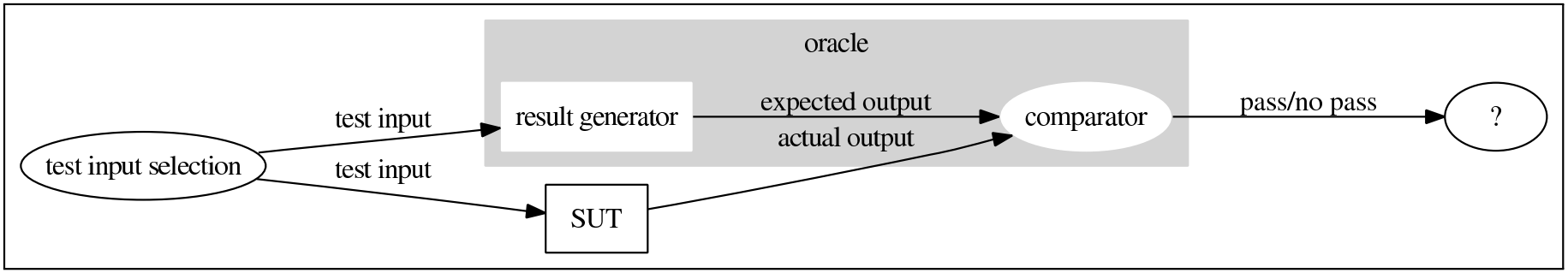
Objectives:

- assess quality of partial oracles.
- compare class based and traditional mutants.
- compare results with real faults.

Preconditions:

- complex and object-oriented implementation.
- dissociate non-terminating mutants.

Methodology



Process is devided into two parts:

1. ascertain suitable input values.
2. assess the quality of applied partial oracles.

Methodology

Ascertain suitable inputs:

- generate images with different properties. (size, color depth, color offset, etc.)
- use original implementation as perfect oracle.
- determine mutation score for each image.
- classify categories of effectiveness.
- combine classes if necessary.

Methodology

Assess quality of partial oracle:

- use images derived from previous step.
- execute SUT and dissociate non-terminating mutants.
- verify only necessary conditions.
- determine mutation score for partial oracle.

Overview

Environment:

- SUT: jpeg2000 decomposition (part of JJ2000 library).
  - 413 traditional mutants.
  - 101 class based mutants.
- Random input generation.
- Metamorphic Relations as partial oracles.

Testing process

1. ascertain suitable input values.
2. assess the quality of applied partial oracles.

Overview

Environment:

- SUT: jpeg2000 decomposition (part of JJ2000 library).
  - 413 traditional mutants.
  - 101 class based mutants.
- Random input generation.
- Metamorphic Relations as partial oracles.

Testing process

1. ascertain suitable input values.
2. assess the quality of applied partial oracles.

Input generation

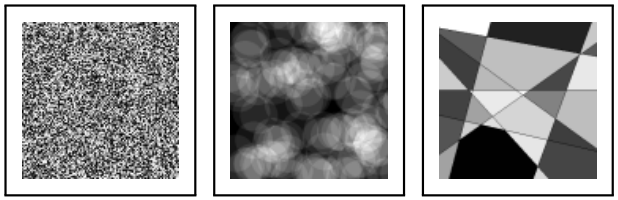
Random image generation:

- use RGB color model.
- extend model for gray images.

Input generation

Random image generation:

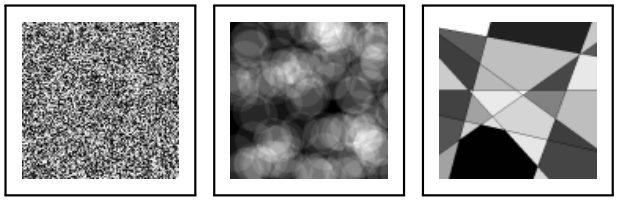
- use RGB color model.
- extend model for gray images.

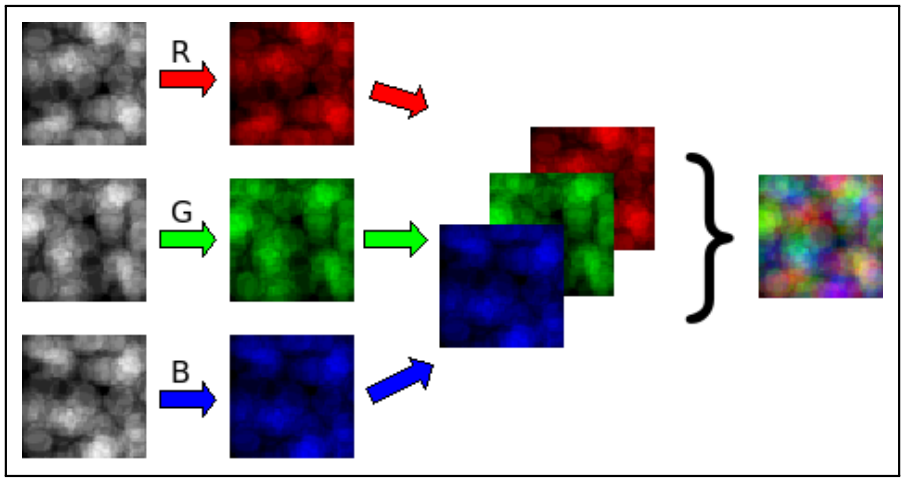


Input generation

Random image generation:

- use RGB color model.
- extend model for gray images.





Ascertain suitable input values

Categories of effectiveness (514 non-equivalent mutants)

Image dimensions			
	x<y	**x=y**	**x>y**
x<y	487	489	514
x=y		465	489
x>y			487

Metamorphic Relations

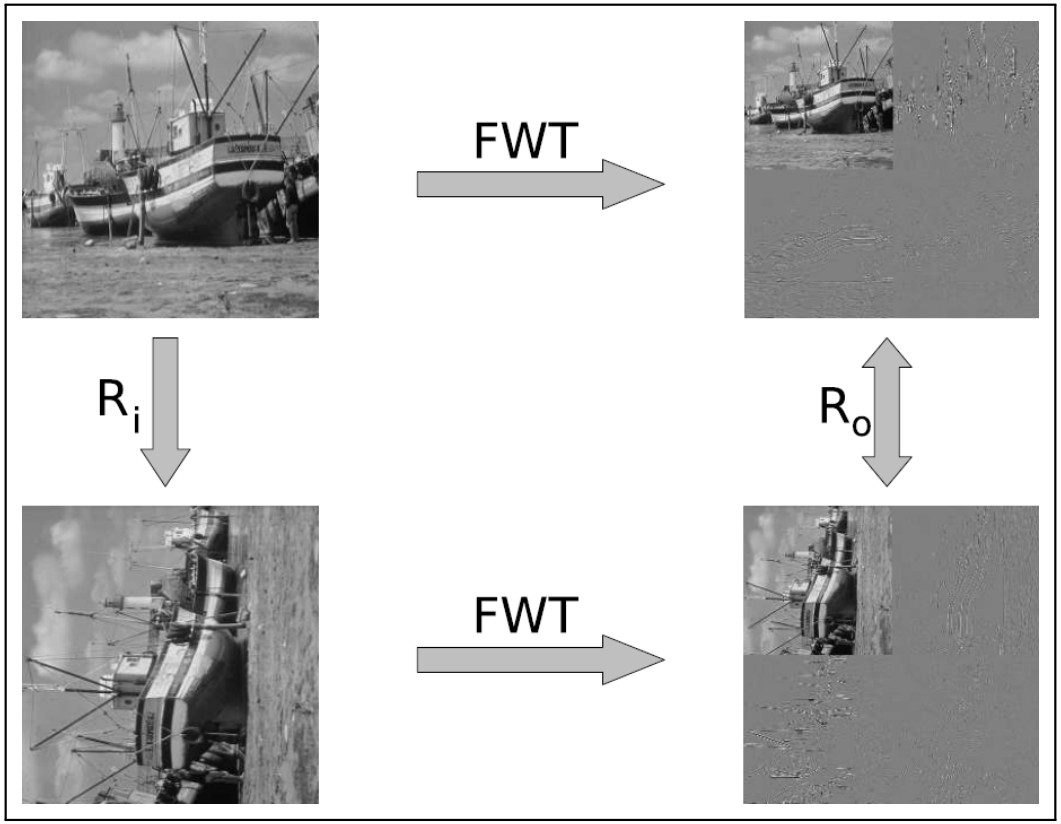
- input domain I.
- output range O.
- mapping $f : I \rightarrow O$.
- two relations $R_I \subseteq I^n$ and $R_O \subseteq I^n \times O^n$.

Metamorphic Relation (Chen et al.)

The pair (R_I, R_O) is called Metamorphic Relation if and only if the following implication is fulfilled.

$$(i_1, \ldots, i_n) \in R_I \Rightarrow (i_1, \ldots, i_n, f(i_1), \ldots, f(i_n)) \in R_O$$

Metamorphic Relations



Applied relations

R1: Add an offset to the color values of each component.

R2: Multiply the color values of each component by a coefficient.

R3: Transpose the pixel array of each color component.

R4: Enlarge the image height (with zeros).

Assess Quality of partial oracles

Quality of applied relations (514 non-equivalent mutants)

All mutants			
Relation	**Total**	**Exceptions**	**Timeouts**
R1	484	312	8
R2	461	311	8
R3	480	312	8
R4	448	312	8

Metamorphic Relations				
	R1	**R2**	**R3**	**R4**
R1	484	493	509	492
R2		461	495	476
R3			480	487
R4				448

ulm university universität ulm

Assess Quality of partial oracles

Quality of applied relations (413/101 non-equivalent mutants)

Traditional mutants			
Relation	**Total**	**Exceptions**	**Timeouts**
R1	402	312	8
R2	381	311	8
R3	413	312	8
R4	398	312	8

Class based mutants			
Relation	**Total**	**Exceptions**	**Timeouts**
R1	82	30	0
R2	80	30	0
R3	67	30	0
R4	50	30	0

Conclusion

- use class based and traditional mutation operators.
- partial oracles should be combined.
- effectiveness of smoke-tests may be language dependent.

Future work:

- confirm results with non-artificial faults.
- generalize approach.
- apply approach to other type of software.

Conclusion

- use class based and traditional mutation operators.
- partial oracles should be combined.
- effectiveness of smoke-tests may be language dependent.

Future work:

- confirm results with non-artificial faults.
- generalize approach.
- apply approach to other type of software.

Evaluating testing strategies for imaging software by means of Mutation Analysis

Thank you for your attention!

ulm university universität ulm