

Re-Engineering Test Suites

Yvan Labiche
Associate Professor



Software Quality Engineering Laboratory
Department of Systems and Computer Engineering
Carleton University, Ottawa, Canada

March 17th, 2010

Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions



Typical (?) Scenarios

Team Lead

- We have this ~~component from a previous project.~~ open source component
- We want to reuse it for your project.
- Can you look into it?
- Most importantly, can we depend on it?
- Well ... hmm ... not really.
- But we have test cases!
- Well ... hmm ... don't know.

You

- Sure!
- What should I focus on?
- Ok.
- Do we have a documentation?
- How were they created?



Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Issues—Open Source Context

- A study of open source software development projects (2003)
 - Zhao L. and Elbaum S., “Quality assurance under the open source development model,” *Journal of Systems and Software*, vol. 66 (1), pp. 65-75, 2003.
- Observations
 - lack “attention to basic, accepted, and mature testing techniques”
 - resulting in low source code coverage (e.g., 30%)
- Dependability?



Canada's Capital University



Carleton University, Ottawa, Canada
www.sce.carleton.ca/squall

Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Issues—COTS (In-House or not)

- Documentation (functionalities) for the COTS?
- Test documentation?

- Documentation practices
- Personal turn over



Context

Melba

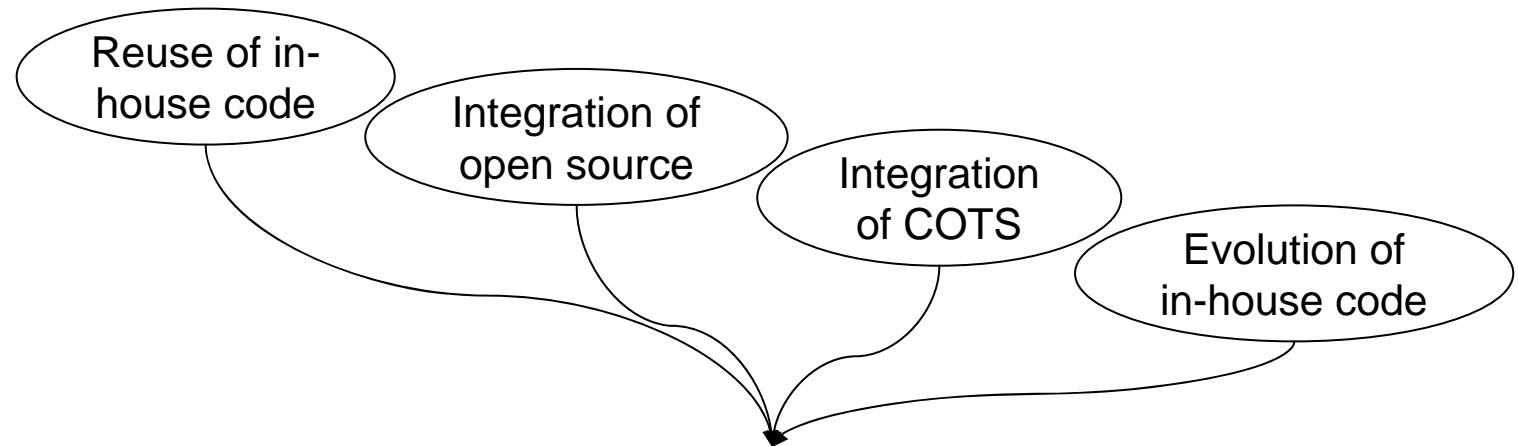
- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Steps to ensure dependability?



1. Read documentation / read test cases / ask colleague
2. Execute existing test case and learn
3. Repeat steps 1 and 2 until sufficient understanding
4. While no sufficient dependability, repeat step 5
5. Add test cases

Notes:

During 1-2-3, one essentially tries to abstracts away from code/tests
(Domain information?)

1-2-3 and 4-5 could be concurrent

May need to iterate all these steps

Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Domain Expert vs. Developer

- Developer
 - Has the expertise to read and execute code/test cases
 - Does not necessarily have the necessary expertise about the Domain
- Domain expert
 - Has the expertise of the Domain
 - Does not necessarily have the technical expertise
 - To read code and execute tests
- Domain expert may be the best at identifying how to ensure dependability (addition of interesting test cases)
- Note: Domain expert may be a senior developer
 - Lost track with current technology



Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Needs

- Need
 - to augment/refine a test suite to ensure sufficient dependability
 - to reduce a test suite to meet tight deadlines
- Who?
 - Ideally the domain expert
- Solution
 - Re-engineering test suite
 - Black box (not tied to development technology)
- Re-engineering?
 - similar to re-engineering source code
 - where code information is extracted,
 - abstracted from a design standpoint, and
 - used to decide about design changes



Context

Melba

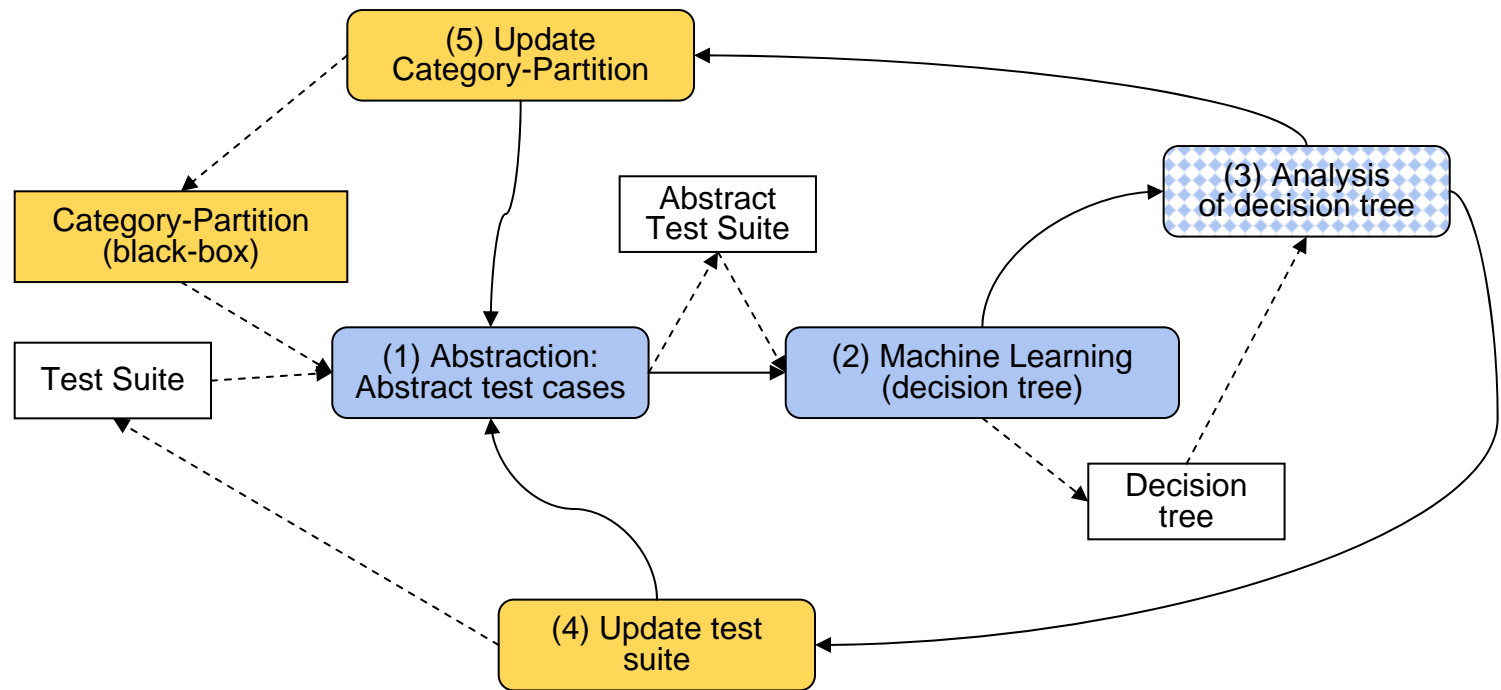
- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Melba (MachinE Learning based refinement of BIAck-box test specification)



- automated
- tool support (semi automated)
- automated or not, depending on context—but this would be done anyway!



Context

Melba

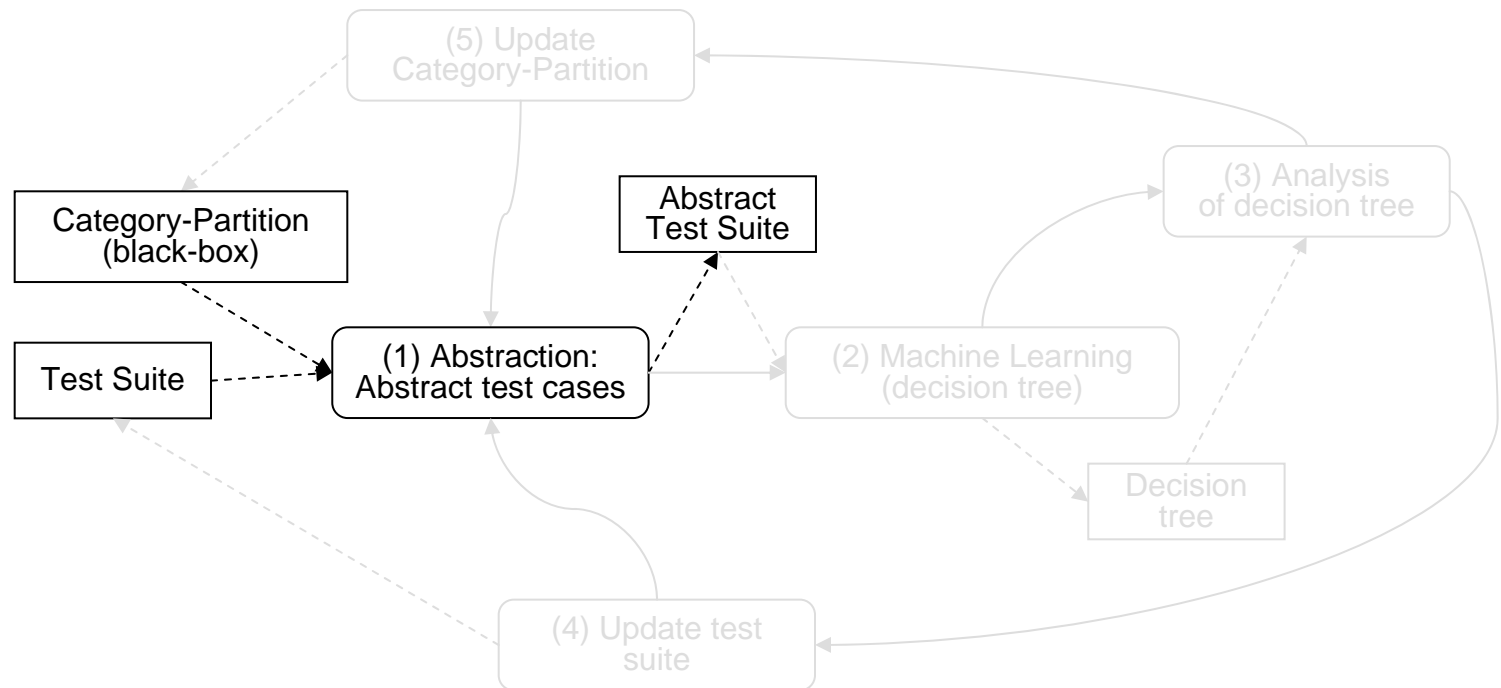
- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Melba (MachinE Learning based refinement of BIAck-box test Specification)



Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Black-Box Abstraction (Category-Partition)

- Category-Partition
 - Black-box testing technique
 - Scales up well: function / component / sub-system / system
 - Functional and non-functional aspects
 - Based on equivalence class partitioning and boundary value analysis
 - Relies on domain experts
- Main steps
 - Identify functions to test
 - Identify parameters and environment variables
 - Identify main characteristics of parameters: categories
 - Class partitioning for characteristics: choices
 - Conditions on choices to prevent their combinations
 - Derive test cases
 - Combinations of choices
 - Satisfy constraints

Environment variable?

- Not a parameter of the function
- Variable of the environment of execution that can impact function's behaviour

Application of boundary value analysis and equivalence class partitioning.



Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions



Category-Partition—example 2

- Consider a search function
`int search(char, string)`
to return the first occurrence of `char` in `string`
- Characteristic of `char`

– position in `string`

choices

first position

last position

intermediate position

– number of occurrences in `string`

choices

0

1

many times

- Choices cannot be combined
- Constraint to prevent it.

- If combining first/last/intermediate with 1 (occurrence), do we need to combine first/last/intermediate with “many times”?
- Constraint to prevent this.

Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Category Partition—another example

- The Triangle program
 - Inputs characterize the length of triangle sides (a, b, c)
 - Output: whether these define an equilateral, isosceles, irregular triangle, or not a triangle
- Parameter a
 - Values for a
 - $a > 0$
 - $a \leq 0$
 - a compared to b
 - $a = b$
 - $a \neq b$
 - a compared to b and c
 - $a \leq b + c$
 - $a > b + c$

...



Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Abstraction with Category-Partition

- Test case
raw values

($a=2, b=3, c=3$)
 - Abstract test case
category/choice combination
with (expected) output equivalence class

($a \leq b+c, b=c, a>0, b>0, c>0, \text{isosceles}$)
 - Characterization of inputs and outputs for a test case
 - Abstraction considering what is deemed important from a testing point of view
-
- Note:
 - Output equivalence classes: isosceles, equilateral, ...
 - Not part of Category-Partition
 - But necessary in Melba



Context

Melba

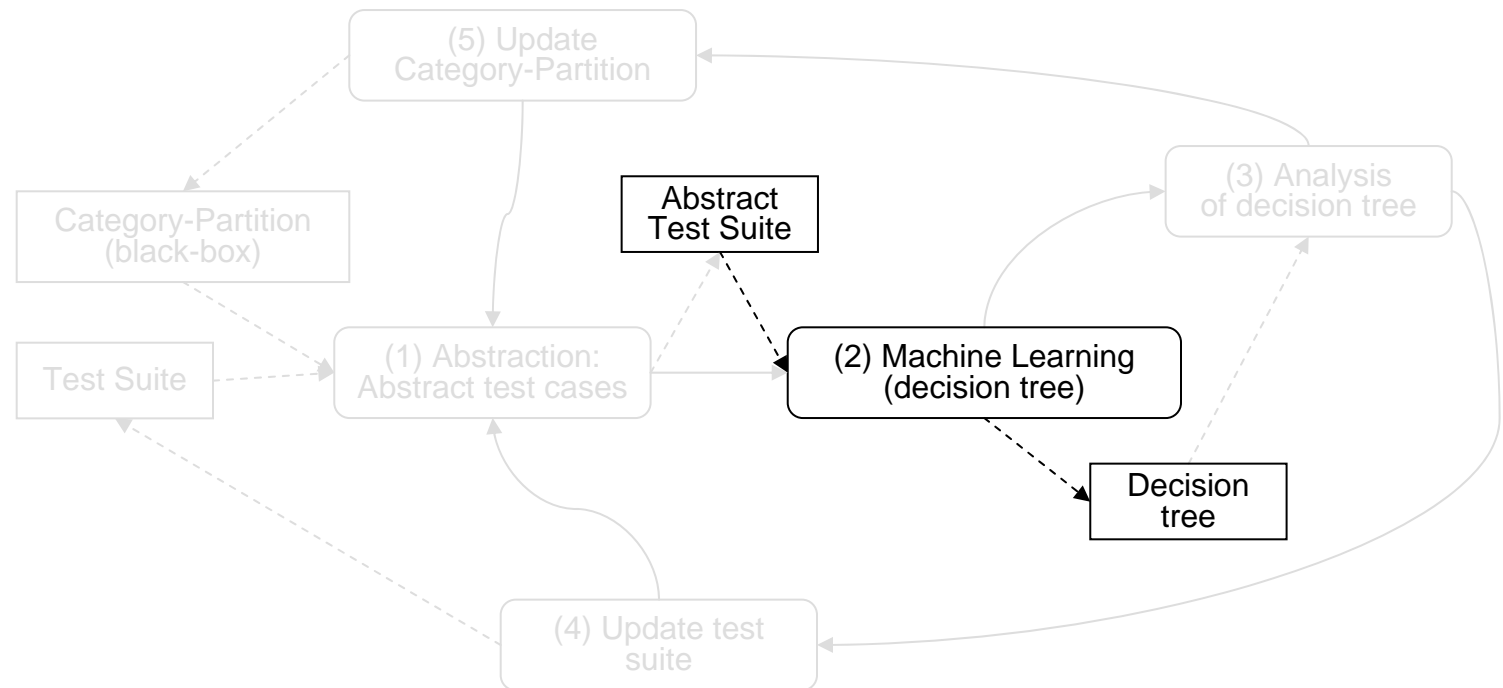
- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Melba (Machine Learning based refinement of Black-box test Specification)



Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Machine Learning

- Learning relationships between input characteristics and output characteristics
 - With a classification algorithm
 - With classification rules
 - Based on heuristics and statistics
- Tool/Algorithm: Weka/C4.5
- Expected output:
 - A set of rules of the form
 - If the value of parameter one is ...
 - And the value of parameter two is ...
 - Then the output is ...



Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

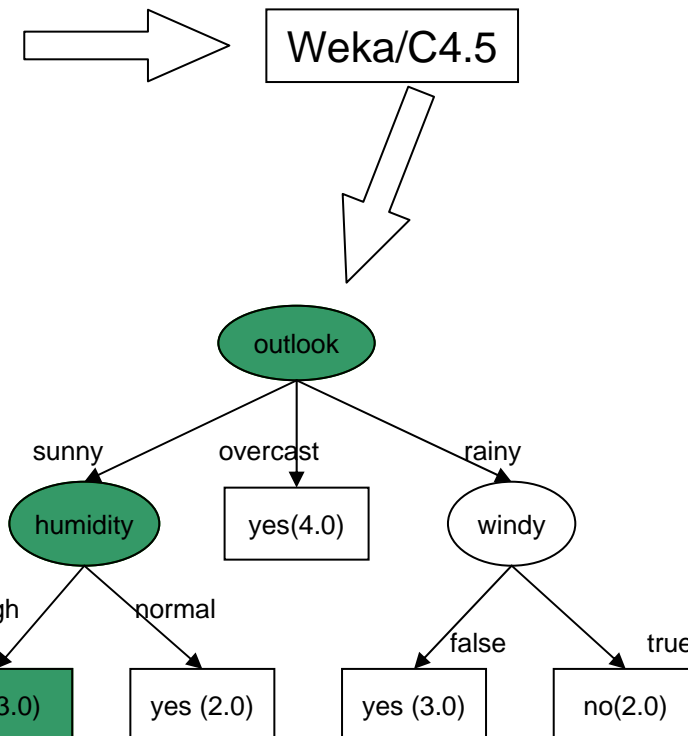
Experiments

Conclusions

Machine Learning—the Weather Example

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Witten I. H. and Frank E., *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufman, 2nd Edition, 2005.



If the outlook is sunny and the humidity is high, then do not play outside

Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Example—Triangle

1. Test cases (with actual values for parameters)
2. Test cases abstracted into Abstract test cases with the help of Category-Partition
3. Abstract test cases given as input to Weka

Result:

```
(a vs. b) = a!=b
| (c vs. a+b) = c<=a+b
| | (a vs. b+c) = a<=b+c
| | | (b vs. a+c) = b<=a+c
| | | | (b vs. c) = b=c
| | | | | (a) = a>0: Isosceles (22.0)
...
```

If $a \neq b$ and $c \leq a+b$ and ... then the output is Isosceles

Identification of categories and choices

Number of (abstract) test cases falling into this rule



Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions



Example—Triangle

- What if we use raw test case data instead of abstractions?

Result (same test cases as previously):

```
(c) <= 1
| (a) <= -1: null (1.0)
| (a) > -1: Equilateral (1.0)
(c) > 1
| (b) <= 1
| | (c) <= 2: Isosceles (1.0)
| | (c) > 2: Not_Triangle (2.0)
| (b) > 1
| | (a) <= 3: Irregular (2.0)
| | (a) > 3: Isosceles (2.0)
```

If $c \leq 1$ and $a > -1$ then the output is Equilateral

So input $a=2, b=3, c=1$ defines an equilateral triangle!

Context

Melba

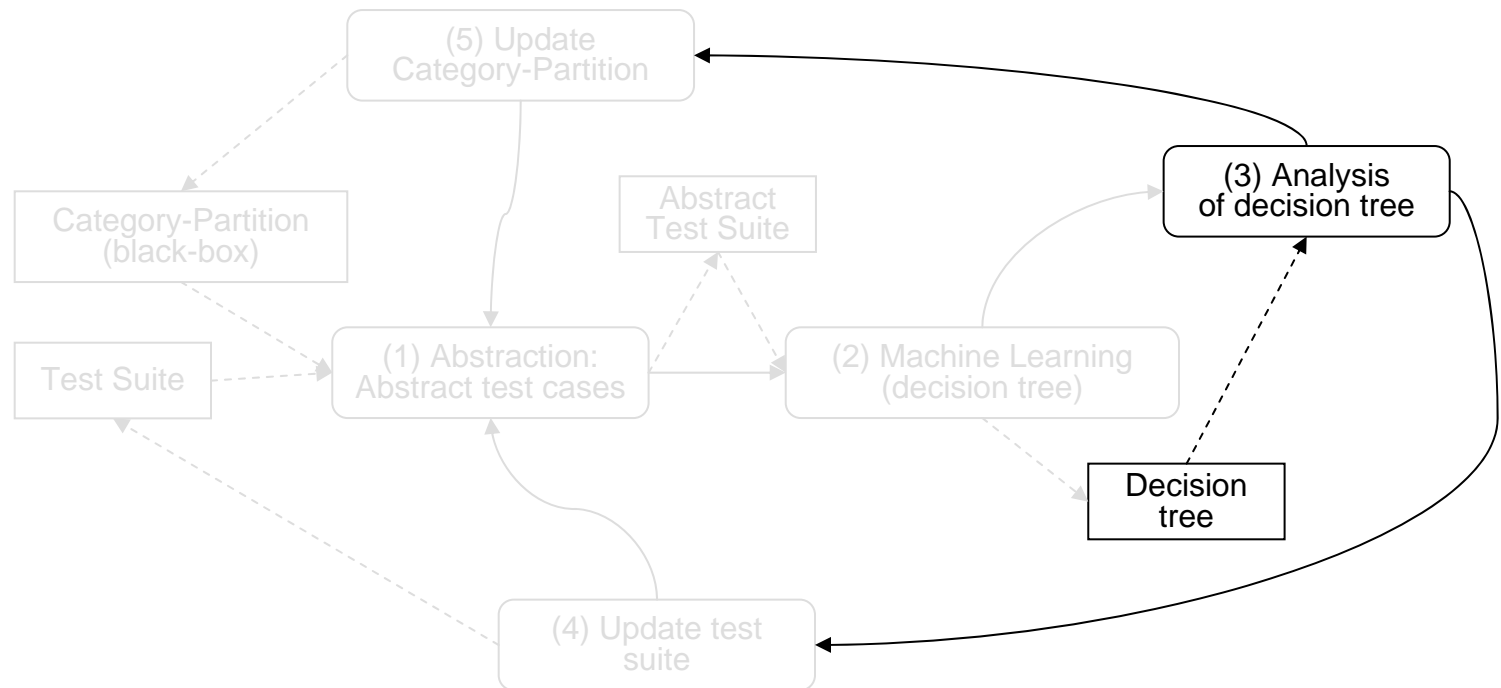
- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Melba (MachinE Learning based refinement of BIAck-box test Specification)



Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions



What can we Learn from a Decision Tree?

Recall:

Abstraction = what is deemed important from a testing point of view

What if ... ?

- A choice of a category, or a category is missing!
 - Missing test case?
- An output equivalence class is missing!
 - Missing test case?
- Combination of choices is missing!
 - Missing test case?
- Zero (abstract) test case for rule!
 - Missing test case?
- Many, e.g., 100, (abstract) test cases for rule!
 - Too many test case?

Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

What can we Learn from a Decision Tree?

- Machine Learning algorithm provides additional data
- A rule can look like:

```
(a vs. b) = a=b
| (b vs. c) = b!=c
| | (a vs. b+c) = a<=b+c
| | | (c vs. a+b) = c<=a+b: Isosceles (24.0/2.0)
```

If $a=b$ and $b \neq c$ and
 $a \leq b+c$ and $c \leq a+b$ then
the output is Isosceles

- 2 miss-classified (abstract) test cases among 26 (abstract) test cases.
- 2 (abstract) test cases satisfy the conditions but have a different output class!
- $a=2, b=2, c=0 \Rightarrow$ not a triangle

Question the definition of categories/choices/output equivalence classes.



Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

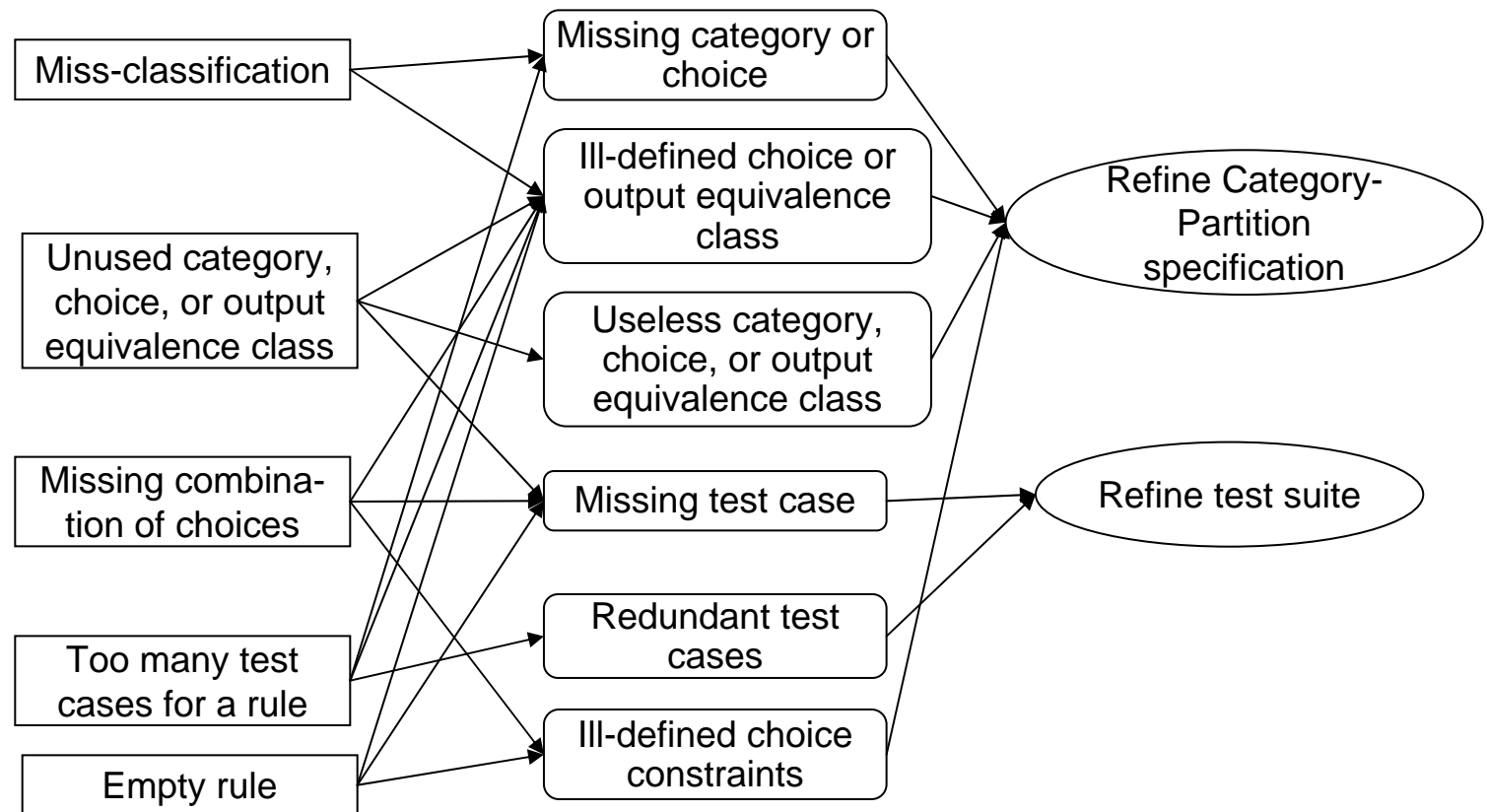
Conclusions

Melba Improvement Process

Observations (problems)
from decision tree (rules)

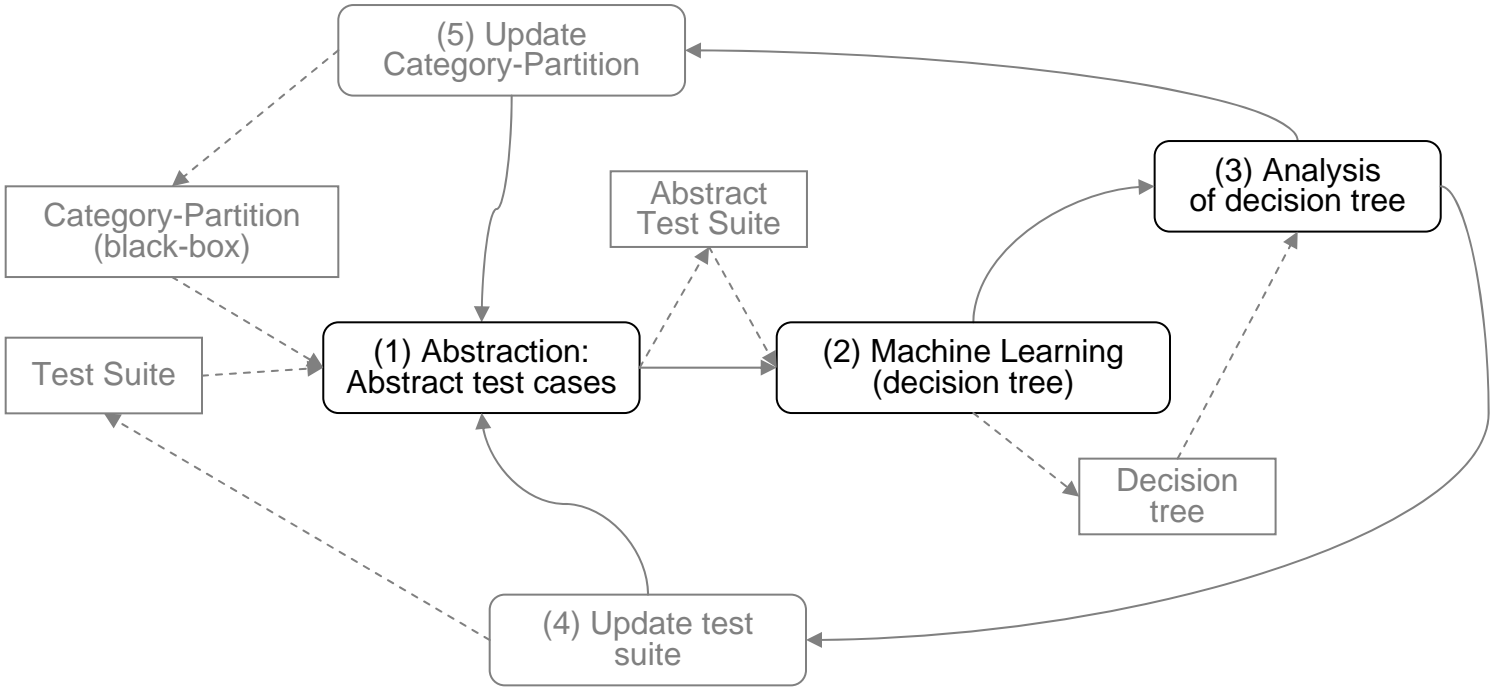
Causes (potential)

Remedies



- Context
- Melba
 - Abstraction
 - Learning
 - Improvements
- Tool overview
- Experiments
- Conclusions

Melba (MachinE Learning based refinement of BIAck-box test Specification)



Tool support



Melba—Architecture

Context

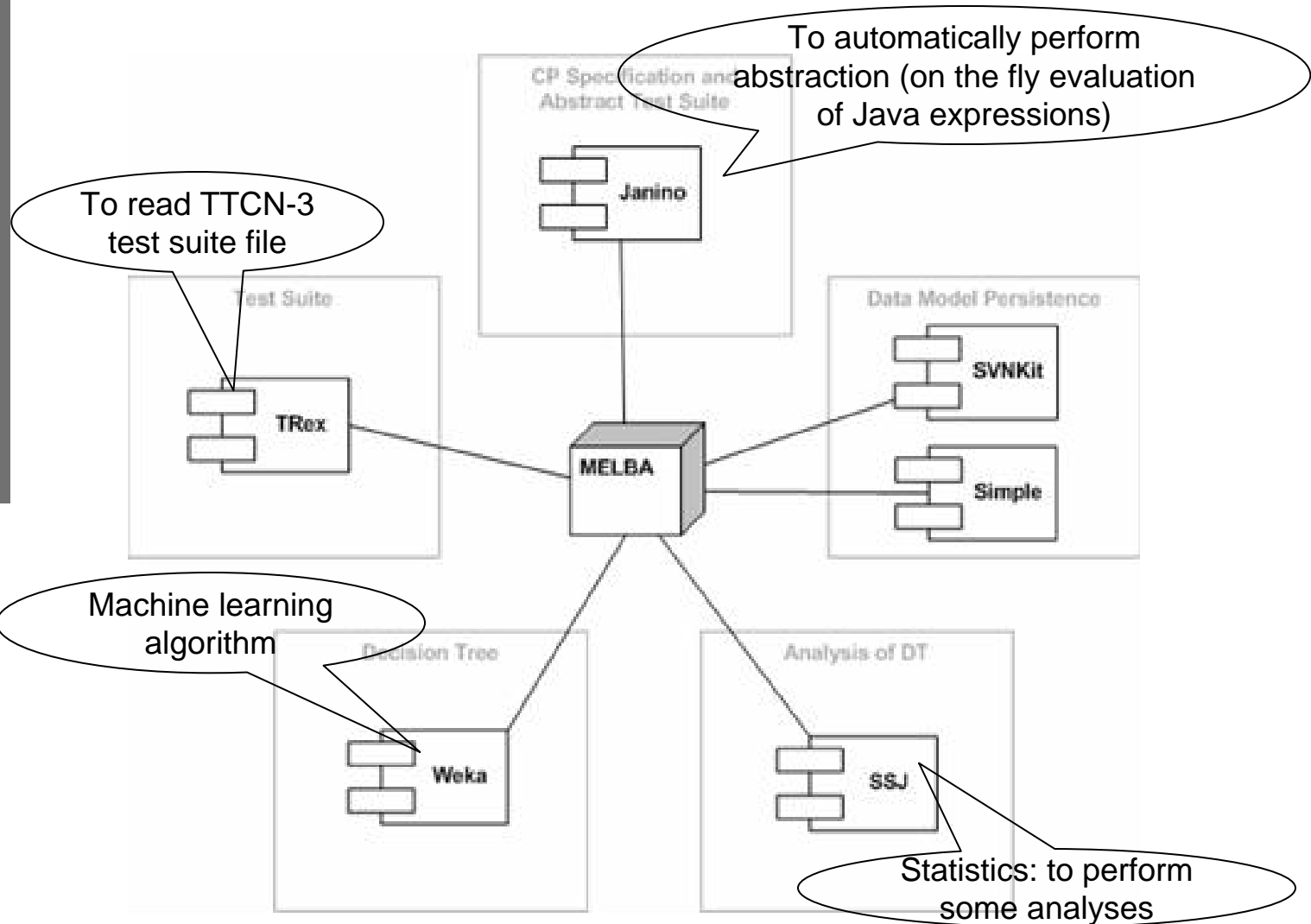
Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions



Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Melba Process—GUI

Specify categories and choices

Category name,
Choice name,
Choice
specification
Constraints

The screenshot shows the Melba Tool interface for a 'Taxi_Billing_System'. The window title is 'Melba Tool - Taxi_Billing_System'. The menu bar includes 'File', 'View', and 'Help'. Below the menu bar are two tabs: 'CP Specification - Input' and 'CP Specification - Output'. The main area displays 'Step 2 of 2 - Please Provide Categories and Choices for Input Params and Environment Vars'. A 'Parameter' dropdown is set to 'destination'. Under 'Category 1', the text 'Area of destination' is entered. The 'Choices' section includes a 'Check to be cleared choice?' checkbox, a 'Choice 1.1' label, and a text field with 'Please type a description'. Below this is an 'Operation' dropdown set to 'Select Operation' and an 'Expression' text field containing 'destination.equals("Paris") || destination.equals("Peripheral road")'. The 'Properties' section has a text field with 'Separate properties using comma' and a 'Selector' dropdown set to 'Use Java boolean expression (without if)'. There are checkboxes for 'Single' and 'Error' (checked). At the bottom are buttons for 'Add more choices', 'Clear selected', '< Back to previous category', 'Go to next category >', 'Add new category +', 'Save and validate categories', and 'Remove category'.



Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Melba Process—GUI

Specify output equivalence classes

Melba Tool - Taxi_Billing_System

File View Help

CP Specification - Input CP Specification - Output

Step 2 of 2 - Please Provide Equivalence Classes for Output Params

Output Equivalence Classes

Check to be cleared **Equivalence Class 8**
Total cost is between 0 and 5.80

Parameter	Operation	Between	and
totalCost	exclusive interval	0	5.80

< >

Add more Equivalence Classes Save and validate Clear selected

<< Back to parameters

Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Melba Process—GUI

Browsing through problems

Decision tree
(rules)

Decision Tree

```
J48 unpruned tree
-----
C 5 = Ch 5.2
| C 2 = Ch 2.2
| | C 3 = Ch 3.2
| | | C 4 = Ch 4.2: EC 1 (4.0/1.0) M1
| | | C 4 = Ch 4.1: EC 2 (2.0)
| | C 3 = Ch 3.3: EC 2 (6.0/1.0) M2
| | C 3 = Ch 3.1: EC 3 (2.0)
| C 2 = Ch 2.8: EC 2 (6.0)
| C 2 = Ch 2.3
| | C 1 = Ch 1.2
| | | C 4 = Ch 4.2: EC 1 (2.0)
| | | C 4 = Ch 4.1: EC 2 (2.0)
```

Additional data

```
=== Confusion Matrix ===
 a b c <-- classified as
 9 3 0 | a = EC 1
 2 34 0 | b = EC 2
 0 0 24 | c = EC 3

=== Detailed Accuracy By Class ===
```

Nonconformities identified

	All	Analyzed
Misclassifications	5	0
Unused CP Elements	1	0
Empty Leaves	2	0
Unused Combinations of Choices	135	0
Nonconformities Total	143	0

Confusion matrix

Summary of
problems

Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Melba Process—GUI

Detailed analysis of problems (rules with zero test case)

- Rule with zero test case
- But combination of choices identified as unfeasible
 - Nothing to do
- Rule with zero test case
- Combination is feasible
 - Investigate

The image shows two overlapping GUI windows from the Melba tool. The top window, titled "Melba Tool - Decision Tree Analysis", displays a decision tree on the left and a table of analysis results on the right. The table has columns for "Leaf with no instance", "Possible causes", and "Relevant information". Three rows are highlighted with red boxes, each containing a rule ID (E1, E2, E3) and a description of an "Unfeasible combination of choices". Red arrows point from the bullet points in the text to these rows. The bottom window, titled "Melba Tool - PackHexChar", shows a configuration screen for "Category 7" with a parameter "Length of s". It includes a "Choices" section with a "Choice 7.2" and an "Expression" field containing "s.length() < rien". Red arrows point from the text "Hyperlinks allow to go back and forth between views" to the hyperlinks in the top window and the configuration fields in the bottom window.



Hyperlinks allow to go back and forth between views

Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Melba Process—GUI

1. Melba has identified unused combinations of choices
2. Melba lists the unused combinations of choices (hyperlinks to the choice definitions)
3. Melba provides possible explanations
4. Melba can (sometimes) identify which explanation is more plausible
5. Melba allows the addition of a test case (hyperlink):
combination identified has been feasible

Combination	Explanation	Action
[(C 5,Ch 5.2)(C 2,Ch 2.2)(C 1,Ch 1.2)]	Already used in the ATS	Investigate
[(C 5,Ch 5.2)(C 2,Ch 2.8)(C 3,Ch 3.2)(C 4,Ch 4.2)]	Missing test cases CP Specification incomplete	Investigate Feasible combination not used in the ATS <small>Or Properties/Selectors missing in CP Spec Click to define new Test Case</small>
[(C 5,Ch 5.2)(C 2,Ch 2.8)(C 3,Ch 3.2)(C 4,Ch 4.1)]	Missing test cases CP Specification incomplete	Investigate Feasible combination not used in the ATS <small>Or Properties/Selectors missing in CP Spec</small>
[(C 5,Ch 5.2)(C 2,Ch 2.8)(C 3,Ch 3.3)]	Already used in the ATS	Investigate

Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Experiments

	PackHexChar	Space	Cab fee calculator
Description	Function of GhostScript: Compacting information	Specifies the configuration of an array of antennas	Fee pattern in Paris
Lines of code (without comments)	54 (Java)	5,900 (C)	159 (Java)
Category- Partition	11 categories 35 choices	83 categories 582 choices	5 categories 19 choices
Test suite size	19-31	3,585	72
Experiment	With students	Without students	Without students
Faults (seeded)	231	N/A	300
Melba	Manual	Manual	Tool



Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Experiment—PackHexChar

- Using
 - a student Category-Partition specification
 - a student test suite
- Using
 - an expert Category-Partition specification
 - A student test suite

Results

- A few iterations of Melba: typically 3
- Improvements to both Category-Partition and test suite
- Level of quality that would be achieved by an expert
 - Category-Partition specification equivalent to what the expert could devise.
- Improved test suite: fault detection capability improved



Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Experiment—Space

- Using the complete test suite
 - Known to have redundancy
- Using a reduced test suite

Results:

- Identification of problems: miss-classifications, rule with too many test cases (>1,000)
- Too expensive to apply Melba manually on such a large problem!



Context

Melba

- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Experiment—Cab

- Poor Category-Partition with test suite reaching good structural coverage
- Trying different test suite size...
 - Does this matter for Melba to converge to something “good”?
- Evaluation of different heuristics to add test cases

Results

- Improvements to Category-Partition
 - Equivalent to what an expert would generate
- Improvements to test suites
- Initial test suite size and quality (e.g., structural coverage) matters
 - Effects still under investigation
- Heuristics to add test cases have varying effects
 - Effects still under investigation



Context

Melba

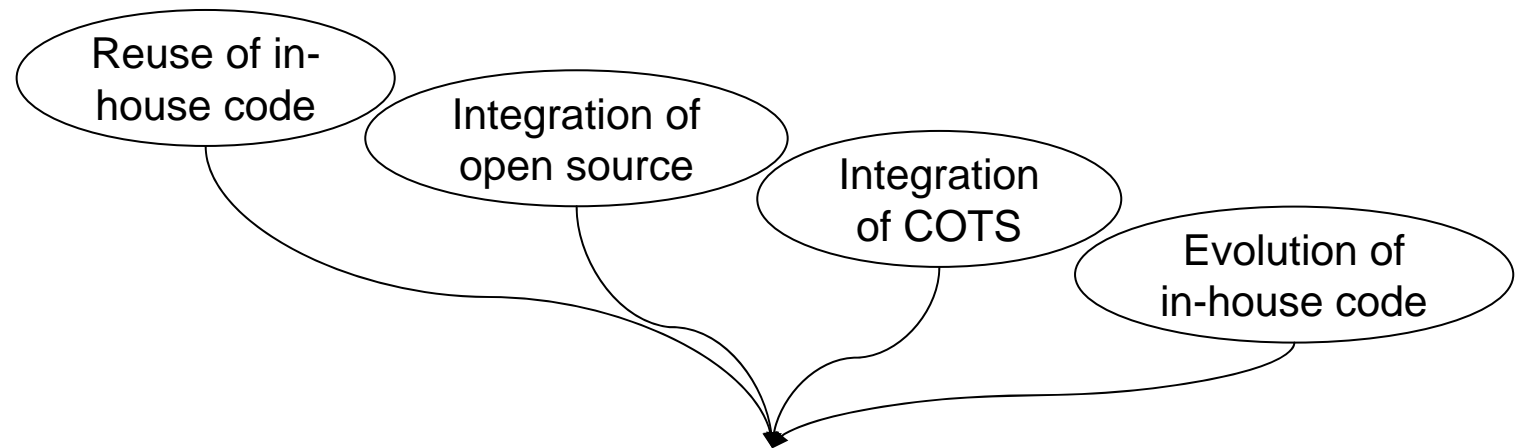
- Abstraction
- Learning
- Improvements

Tool overview

Experiments

Conclusions

Conclusions



Need to re-engineer test suite to improve dependability

- test suite information
- is extracted,
- abstracted,
- and used to make decisions (adding/removing test case)

Melba process and tool support

- All the automatable activities are automated
- Decision making is supported by heuristics
- Very promising results on case studies

Want to be part of it?
Want to evaluate your
own test suites?

Improvements to process and tool support through additional case studies

- Academic case studies
- Industrial case studies



Canada's Capital University



QUESTIONS ?

