AIAI Seminar: 6th July 2020

THE UNIVERSITY *of* EDINBURGH
informatics

# Supporting Criteria-Based Marking

**Paul Anderson**

<dcspaul@ed.ac.uk>

**Software & Documentation**

http://homepages.inf.ed.ac.uk/dcspaul/pmark

## Overview

- Motivation
- A (little) bit about marking
- Computing grades with PMark (by example)
- A (little) bit about algorithms (suggestions welcome?)
- Generating feedback with PMark

there is a video on the web page of
another talk which includes a description of
how this was used for INFIB

# Motivation

## The Common Marking Scheme

- specifies an explicit rubric for the various mark ranges
- for example, marks over 80 require:
  "*demonstrates that the student is actively extending their knowledge and capacity well beyond required materials and making new connections independently*"

## Authenticity

- "adding up" the marks may produce an overall result which doesn't correspond with the desired intuitive assessment of the work
- attempts to adjust this numerically are usually extremely arbitrary

## Variability of marks

- especially with large classes and multiple markers

## Feedback

- the relationship between the feedback and mark is not clear

# Approaches to marking

## "Divergent" tasks

- "real" programming is a "divergent" task ...

  *"intended to provide opportunities for students to demonstrate sophisticated cognitive abilities, integration of knowledge, complex problem solving, critical opinion, lateral thinking and innovative action" (Sadler 2009)*

## A "holistic" scheme

- a single descriptive scale

  there may be a list of criteria, but it is up to the marker how these are combined and weighted to yield an overall mark

## An "analytic" scheme

- separate criteria for different aspects (attributes)
- results combined (in some way) to generate overall mark
- there is some debate about how effective this is in capturing the marker's holistic impression

# Combining marks

## Additive marking

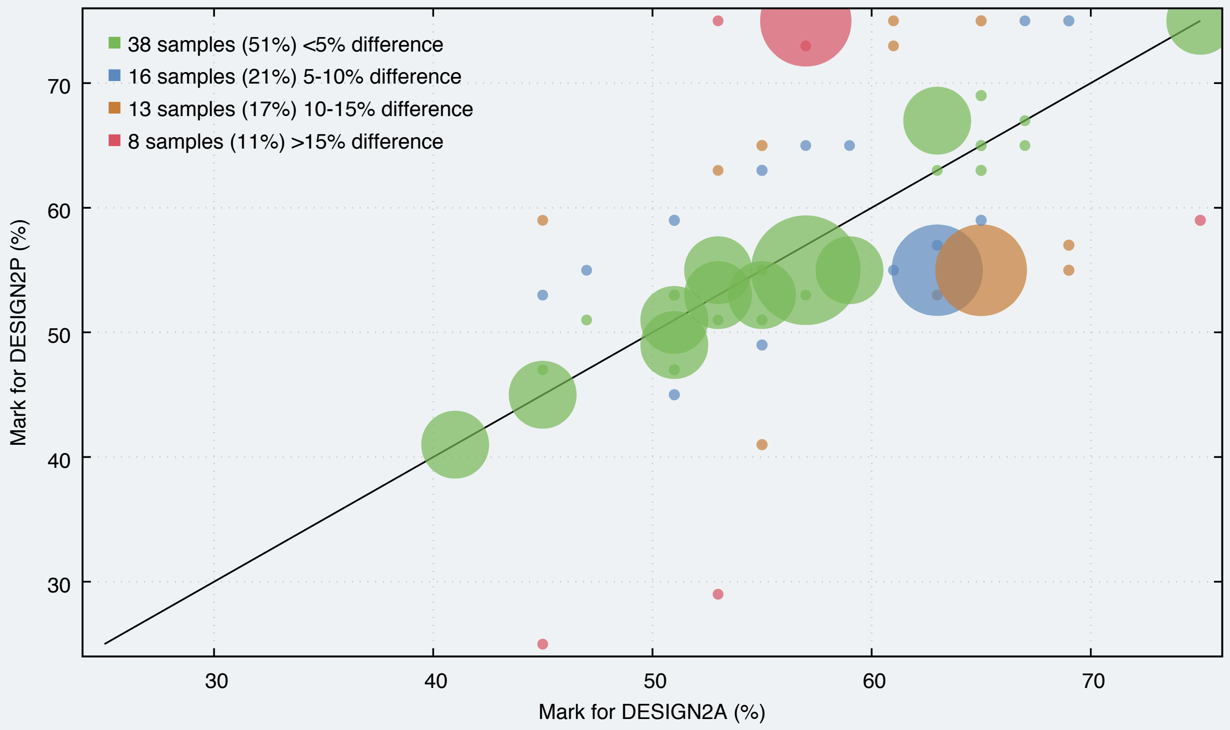
- we could assign a numeric score to the attributes and sum them
- this is "compensatory"
  good marks on some attributes compensate for bad marks elsewhere
  weighting schemes do not solve this problem
- "*grade cutoff scores are not directly linked to mastery of a specific subject matter or skill - the pattern of strengths and weaknesses is lost entirely*" (Sadler 2005)

## Decision rules

- specify explicit requirements for each grade
  *"all of the criteria have to be adequate for a pass"*
- relate the outcome directly to the objectives
  *"you failed because you did not demonstrate this learning outcome .."*
- but these are not so easy to evaluate automatically
- and it is not obvious how to generate a numerical mark

# Marker variation

*"we mislead students that there is something fixed, accessible and rational that they can use to guide their work" (Bloxham 2011)*



*"assessment decisions at this level are so complex, intuitive and tacit that variability is inevitable" (Bloxham 2016)*

# Lots of small rules

## We have been experimenting with ...

- lots of small "atomic" criteria with a simple evaluation: for example: "no", '"not really", "sort-of", "yes"
  - our markers have found these easier and more reliable to mark
  - variation seems more likely to average out
  - the results provide explicit feedback and reasoning
  - avoids multiple implicit sub-criteria
  - avoids agonising over whether something is a 13/20 or 14/20

## The literature is wary of this ...

- because of the difficulty of composing these into a meaningful holistic result

## But ...

- we have been using software to support the composition
- the rules can be adjusted incrementally to achieve an authentic result

## An example

### Software Readability:

- "no", "not really", "sort-of", or "yes" ?
  - is the code properly indented?
  - do the large-scope variables have meaningful names?
  - are there sufficient comments?
  - are there unnecessary comments?
  - is there redundant commented-out code?
  - are there any methods which are too large?
  - etc ... ?

### We might also ask the marker ...

- do you think this is exceptionally good for some reason (explain) ?
- do you think there is something else about the software which makes it particularly readable (or not) which is not captured by these criteria (explain) ?

# Tool requirements

## We would like …

- to have a correct and repeatable evaluation of decision rules with an explicit and transparent mark scheme
- to support potentially large numbers of small rules to mitigate marking variation, and to clearly relate the marking to the objectives
- to be able to develop the mark scheme iteratively (and retrospectively) so that the result really reflects what we want to assess
- to be lenient in the interpretation of the rules, and allow for some degree of marker variation, while still being strict in those cases where it is appropriate
- to be able to discriminate between students who just meet the requirements for a grade, and those who meet the requirements well
- to have clear and explicit feedback about the results and an explanation of how they relate to the rules and attributes

## PMark

### Freely available program

- currently runs on Mac or Linux (in perl)

### Takes ...

- a CSV file
  with textual or numeric values for each "attribute" for each student
- a plain-text "marking scheme"
  describing how to compute the results from the attributes

### Produces ...

- a CSV file
  with textual or numeric results for each student
- a text (or HTML) file
  with descriptive feedback for each student
- various graphs and statistics

## An example

### 5 practical tasks

- hand-washing
- cat-shaving
- dog-bandaging
- hamster-injecting
- pill-counting

### Assessed on a 4-point lickert scale

- "no"
- "almost"
- "adequate"
- "good"

### Results as

- pass/fail
- percentage (common marking scheme)

## CSV Input file (attributes)

```
id, washing, shaving, bandaging, injecting, counting

Sarah, no, no, no, no, no
Dylan, adequate, good, good, almost, good
Max, adequate, adequate, adequate, good, adequate
John, good, almost, adequate, good, adequate
Victoria, adequate, no, almost, adequate, adequate
Lucy, good, good, good, good, adequate
Leo, almost, good, adequate, almost, good
```

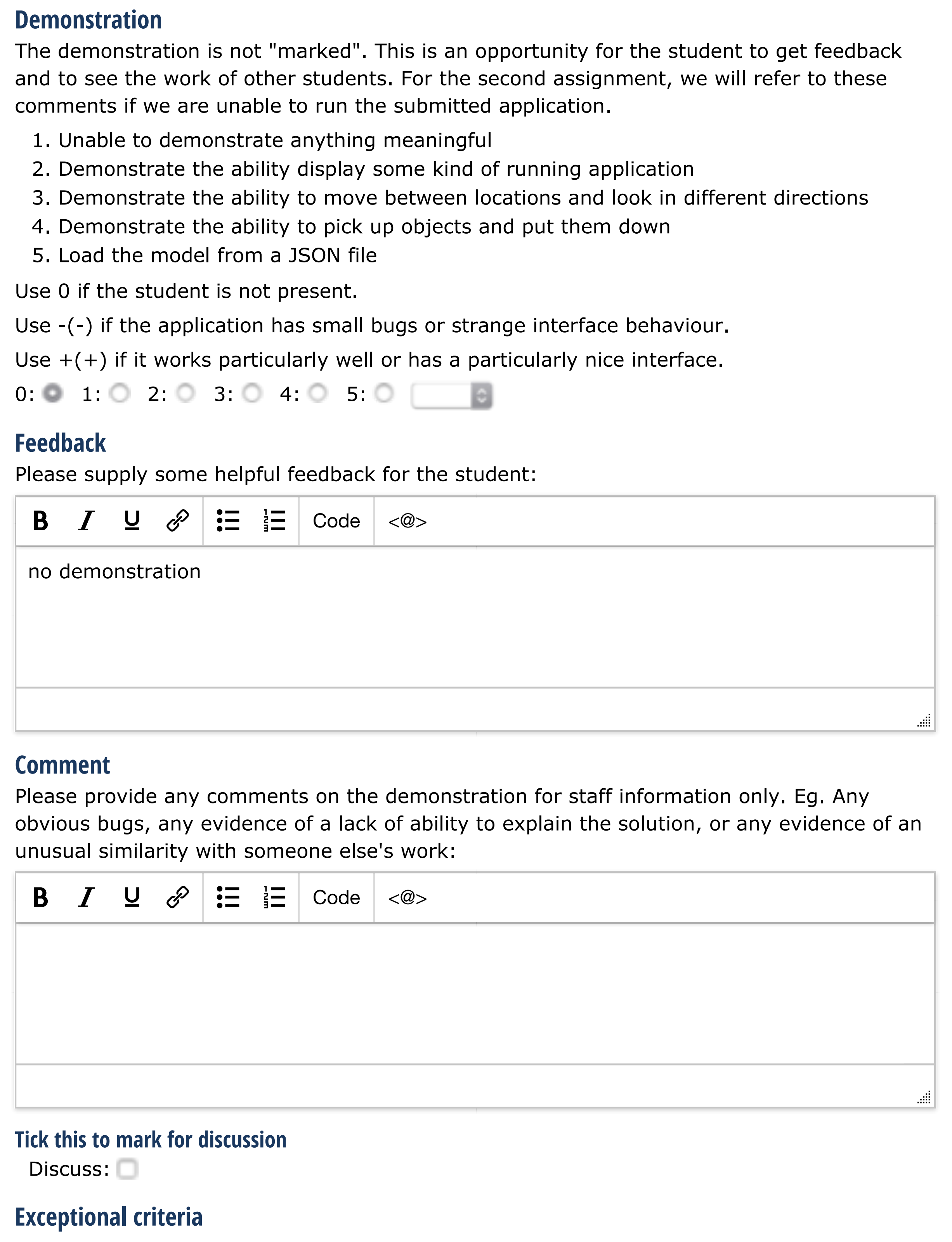
# Marking forms

## IPPO

*this is not part of PMark, but I created online marking forms for IPPO which generate the CSV fields for input to PMark*

## INF1B

*used a similar approach (with a different form implementation)*



### Demonstration

The demonstration is not "marked". This is an opportunity for the student to get feedback and to see the work of other students. For the second assignment, we will refer to these comments if we are unable to run the submitted application.

1. Unable to demonstrate anything meaningful
2. Demonstrate the ability display some kind of running application
3. Demonstrate the ability to move between locations and look in different directions
4. Demonstrate the ability to pick up objects and put them down
5. Load the model from a JSON file

Use 0 if the student is not present.

Use -(-) if the application has small bugs or strange interface behaviour.

Use +(+) if it works particularly well or has a particularly nice interface.

[✓ 0:] 1: 2: 3: 4: 5:

### Feedback

Please supply some helpful feedback for the student:

no demonstration

### Comment

Please provide any comments on the demonstration for staff information only. Eg. Any obvious bugs, any evidence of a lack of ability to explain the solution, or any evidence of an unusual similarity with someone else's work:

### Tick this to mark for discussion

Discuss: ☐

### Exceptional criteria

## Mark scheme: attributes

```
[attributes]
washing
shaving
bandaging
injecting
counting
```

the attribute names must match
the column headings in the CSV file

# Mark scheme: attribute type

```
[types]
mark: [no,almost,adequate,good]

[attributes]
washing:   mark
shaving:   mark
bandaging: mark
injecting: mark
counting:  mark
```

# Mark scheme: attribute type

```
[types]
mark: [no,almost,adequate,good]
```

```
[attributes]
washing:   mark
shaving:   mark
bandaging: mark
injecting: mark
counting:  mark
```

there is nothing special about the values **no, almost, adequate & good** they can be arbitrary names or integers and there can be any number of them

but the order is important!

there is nothing special about the name **mark** - this just connects the attribute to the collection of possible values

# Mark scheme: result

```
[types]
mark: [no,almost,adequate,good]


[attributes]
washing: mark
shaving: mark
bandaging: mark
injecting: mark
counting: mark
```

```
[results]
result
```

# Mark scheme: result type

```
[types]
mark: [no,almost,adequate,good]
grade: [fail,pass]

[attributes]
washing: mark
shaving: mark
bandaging: mark
injecting: mark                                   [results]
counting: mark                                    result: grade
```

# Mark scheme: rules

```
[types]
mark: [no,almost,adequate,good]
grade: [fail,pass]

[attributes]
washing: mark
shaving: mark
bandaging: mark
injecting: mark
counting: mark
```

```
[rules]
pass: all of {
    washing = adequate
    shaving = adequate
    bandaging = adequate
    injecting = adequate
    counting = adequate }

[results]
result: grade
```

# Final mark scheme

```
[types]
mark: [no,almost,adequate,good]
grade: [fail,pass]

[attributes]
washing: mark
shaving: mark
bandaging: mark
injecting: mark
counting: mark
```

```
[rules]
pass: all of {
    washing = adequate
    shaving = adequate
    bandaging = adequate
    injecting = adequate
    counting = adequate }

[results]
result: grade
```

# Running PMark

```
id, washing, shaving, bandaging, injecting, counting

Sarah, no, no, no, no, no
Dylan, adequate, good, good, almost, good
Max, adequate, adequate, adequate, good, adequate
John, good, almost, adequate, good, adequate
Victoria, adequate, no, almost, adequate, adequate
Lucy, good, good, good, good, adequate
Leo, almost, good, adequate, almost, good
```

```
id,result

Sarah,fail
Dylan,fail
Max,pass
John,fail
Victoria,fail
Lucy,pass
Leo,fail
```

```
pmark eval -m vets1.pmark vets.csv
```

# Hashtags

[types]
mark: [no,almost,adequate,good]
grade: [fail,pass]

[attributes]
washing: mark #task
shaving: mark #task
bandaging: mark #task
injecting: mark #task
counting: mark #task

[rules]
~~pass: all of {~~
~~washing = adequate~~
~~shaving = adequate~~
~~bandaging = adequate~~
~~injecting = adequate~~
~~counting = adequate }~~

pass: all #task = adequate

[results]
result: grade

# Being lenient

```
[types]
mark: [no,almost,adequate,good]
grade: [fail,pass]

[attributes]
washing: mark #task
shaving: mark #task
bandaging: mark #task
injecting: mark #task
counting: mark #task
```

```
[rules]
pass:
  all but one of
     #task = adequate
  and all of
     #task = almost
```

```
[results]
result: grade
```

# Lenient results

```
id, washing, shaving, bandaging, injecting, counting

Sarah, no, no, no, no, no
Dylan, adequate, good, good, almost, good
Max, adequate, adequate, adequate, good, adequate
John, good, almost, adequate, good, adequate
Victoria, adequate, no, almost, adequate, adequate
Lucy, good, good, good, good, adequate
Leo, almost, good, adequate, almost, good
```

```
id,result

Sarah,fail
Dylan,pass
Max,pass
John,pass
Victoria,fail
Lucy,pass
Leo,fail
```

```
pmark eval -m vets3.pmark vets.csv
```

## Adding more grades

```
[types]
mark: [no,almost,adequate,good]
grade: [fail,pass,distinction]

[attributes]
washing: mark #task
shaving: mark #task
bandaging: mark #task
injecting: mark #task
counting: mark #task
```

```
[rules]
pass:
  all but one #task = adequate
  and all #task = almost
distinction:
  all but one #task = good
  and all #task = adequate

[results]
result: grade
```

# Results with distinctions

```
id, washing, shaving, bandaging, injecting, counting

Sarah, no, no, no, no, no
Dylan, adequate, good, good, almost, good
Max, adequate, adequate, adequate, good, adequate
John, good, almost, adequate, good, adequate
Victoria, adequate, no, almost, adequate, adequate
Lucy, good, good, good, good, adequate
Leo, almost, good, adequate, almost, good
```

```
id,result

Sarah,fail
Dylan,pass
Max,pass
John,pass
Victoria,fail
Lucy,distinction
Leo,fail
```

```
pmark eval -m vets4.pmark vets.csv
```

# Important tasks

```
[types]
mark: [no,almost,adequate,good]
grade: [fail,pass,distinction]

[attributes]
washing: mark #task
shaving: mark #task
bandaging: mark #task #imp
injecting: mark #task  #imp
counting: mark #task   #imp
```

```
[rules]
pass:
  all #imp = adequate
  and all #task = almost
distinction:
  all but one #task = good
  and all #task = adequate

[results]
result: grade
```

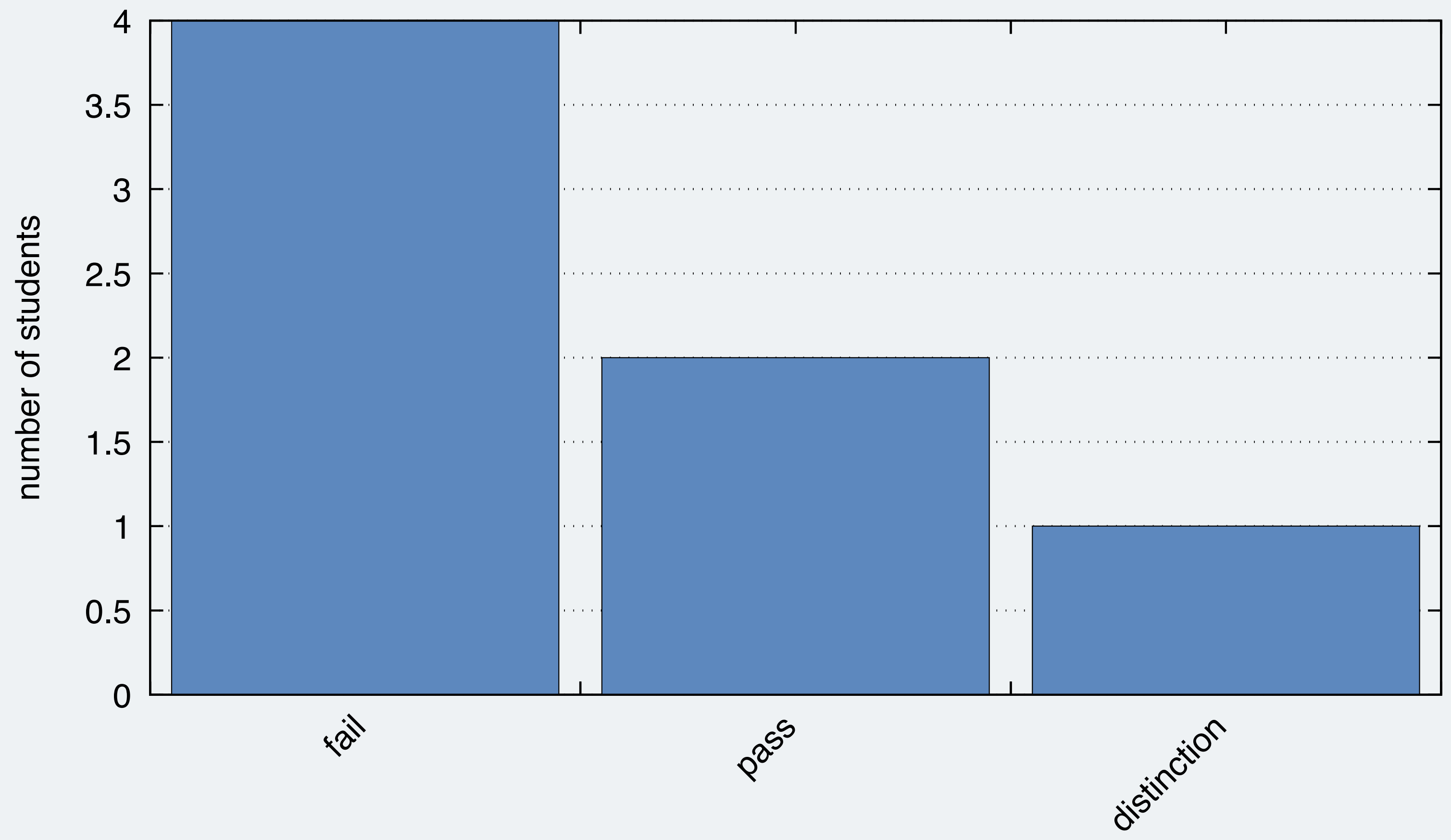
# Failing important tasks

```
id, washing, shaving, bandaging, injecting, counting

Sarah, no, no, no, no, no
Dylan, adequate, good, good, almost, good
Max, adequate, adequate, adequate, good, adequate
John, good, almost, adequate, good, adequate
Victoria, adequate, no, almost, adequate, adequate
Lucy, good, good, good, good, adequate
Leo, almost, good, adequate, almost, good
```

```
id,result

Sarah,fail
Dylan,fail
Max,pass
John,pass
Victoria,fail
Lucy,distinction
Leo,fail
```

```
pmark eval -m vets5.pmark vets.csv
```

## A graph



```
pmark plot -m vets5.pmark vets.csv
```

## Interpolation

```
[types]
mark: [no,almost,adequate,good]
percentage: [
  0..100
   pass = 50,
   distinction = 70 ]

[attributes]
washing: mark #task
shaving: mark #task
bandaging: mark #task #imp
injecting: mark #task #imp
counting: mark #task #imp
```

```
[rules]
pass:
  all #imp = adequate
  and all #task = almost
distinction:
  all but one #task = good
  and all #task = adequate

[results]
result: percentage
```

# Percentage results

```
id, washing, shaving, bandaging, injecting, counting

Sarah, no, no, no, no, no
Dylan, adequate, good, good, almost, good
Max, adequate, adequate, adequate, good, adequate
John, good, almost, adequate, good, adequate
Victoria, adequate, no, almost, adequate, adequate
Lucy, good, good, good, good, adequate
Leo, almost, good, adequate, almost, good
```

```
id,result

Sarah,0
Dylan,42
Max,61
John,60
Victoria,26
Lucy,94
Leo,38
```

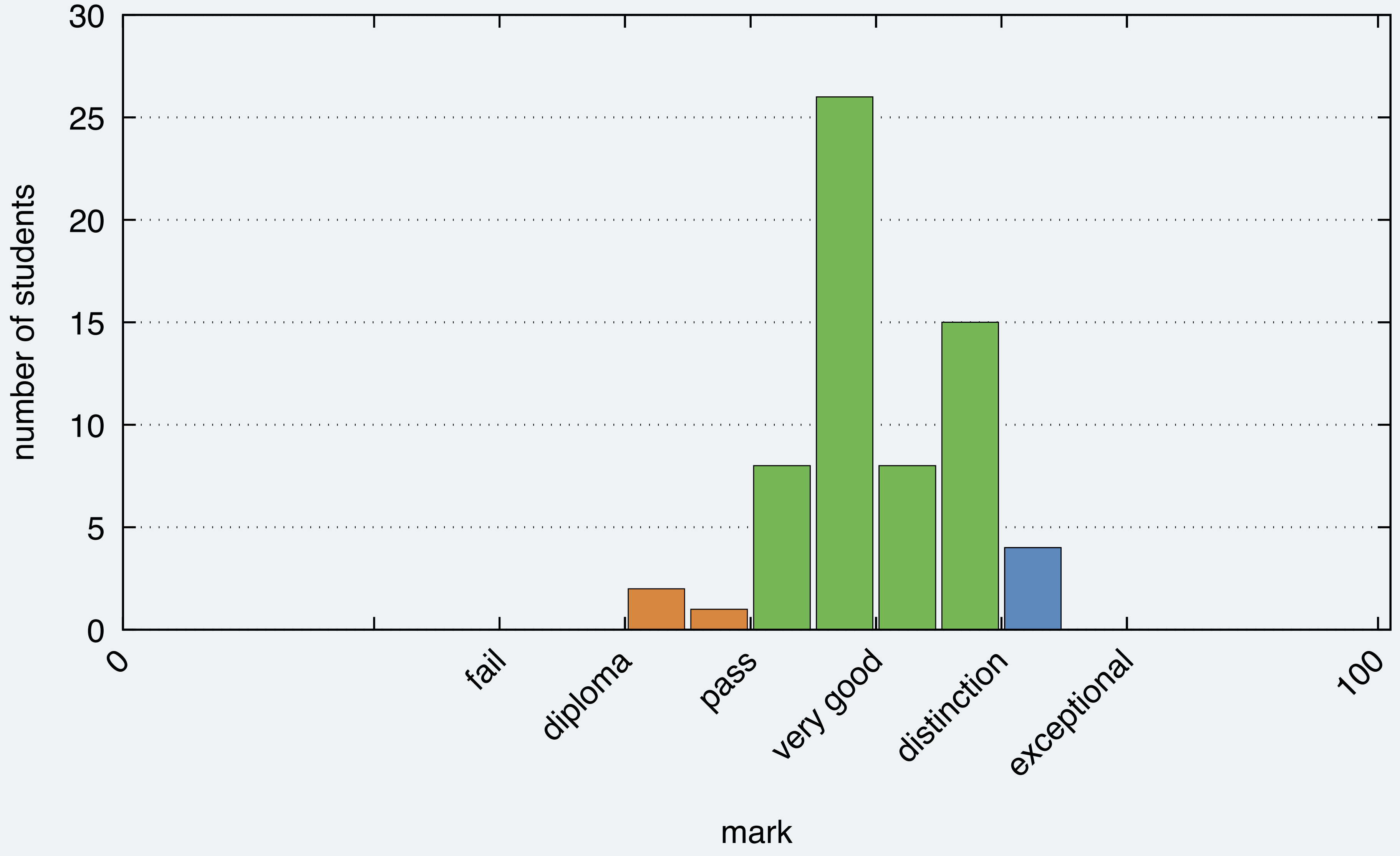
*Victoria and Leo both still fail*
*But Victoria is a "worse" fail than Leo*

```
pmark eval -m vets6.pmark vets.csv
```

# A practical rule

```
// student must have submitted a draft design in order to pass
// really, we require a basic working application to pass
// but if the design is particularly good, we will accept some bugs
// we do require *some* sort of running implementation though
G1: P2 and all { DRAFT2A=1,
            some { good-design, working-app },
            some { RUN2B=1, DEMO2D=2 }}
// a reasonable collection of classes
// all of the MVC components must be reasonably explicit
// most of them must be very clear
good-design: all { CLASSES2A=4, all #mvc=3, most #mvc=4 }
// either the submitted code, or the demo must run
// no major bugs
working-app: some { RUN2B=2, DEMO2D=3 } and BUGS2B=2
```

# A customised plot



# Grade evaluation

## There are only two type of rule expression:

- *ATTRIBUTE = VALUE*
  true if the named *ATTRIBUTE* has (at least) the given *VALUE*
- *N of { $CRITERIA_1$, $CRITERIA_2$, ... $CRITERIA_M$ }*
  *true if (at least) N of the M criteria are true*
  *eg. "all but one of" => (M-1) of { $CRITERIA_1$ ... $CRITERIA_M$ }*

## The resulting grade ...

- is determined entirely by a simple boolean evaluation

## But ...

- the "scores" for the expressions are represented internally by values in the range [-1..+1] which are used to interpolate between the grades if required
- -1 = worst possible fail, 0 = minimal pass, 1 = maximal pass

## Interpolation

### ATTRIBUTE = VALUE

- if the criteria fails, the score [-1,0) is interpolated between the lowest possible value of the attribute, and the *VALUE*.
- if it passes, the interpolation [0,1] is between the the VALUE and the highest possible value

### N of { CRITERIA$_1$, CRITERIA$_2$, … CRITERIA$_M$ }

- the score is obtained by interpolating between the minimum and maximum possible values for the sum of the criteria
  (these will be different depending on whether the criteria passes of not)
- in practice, the algorithm is more complex because PMark allows allows weights to be assigned to the criteria

### The final result

- is obtained by using the score to interpolate between the passing grade and the next highest grade

## Interpolation

### Validity

- the interpolated value is only a heuristic
- however, this appears to produce a value which correlates well with an intuitive ranking of the results
- it is essentially equivalent to a (weighted) averaging of the scores for the grade
- PMark can provide a detailed audit of the interpolation, although this is usually too complex to be useful
  (the audit of the logical grade calculation is simpler and more useful)
- of course it is possible to use the interpolated values as a guide and assign the final values manually (as in the Informatics projects)

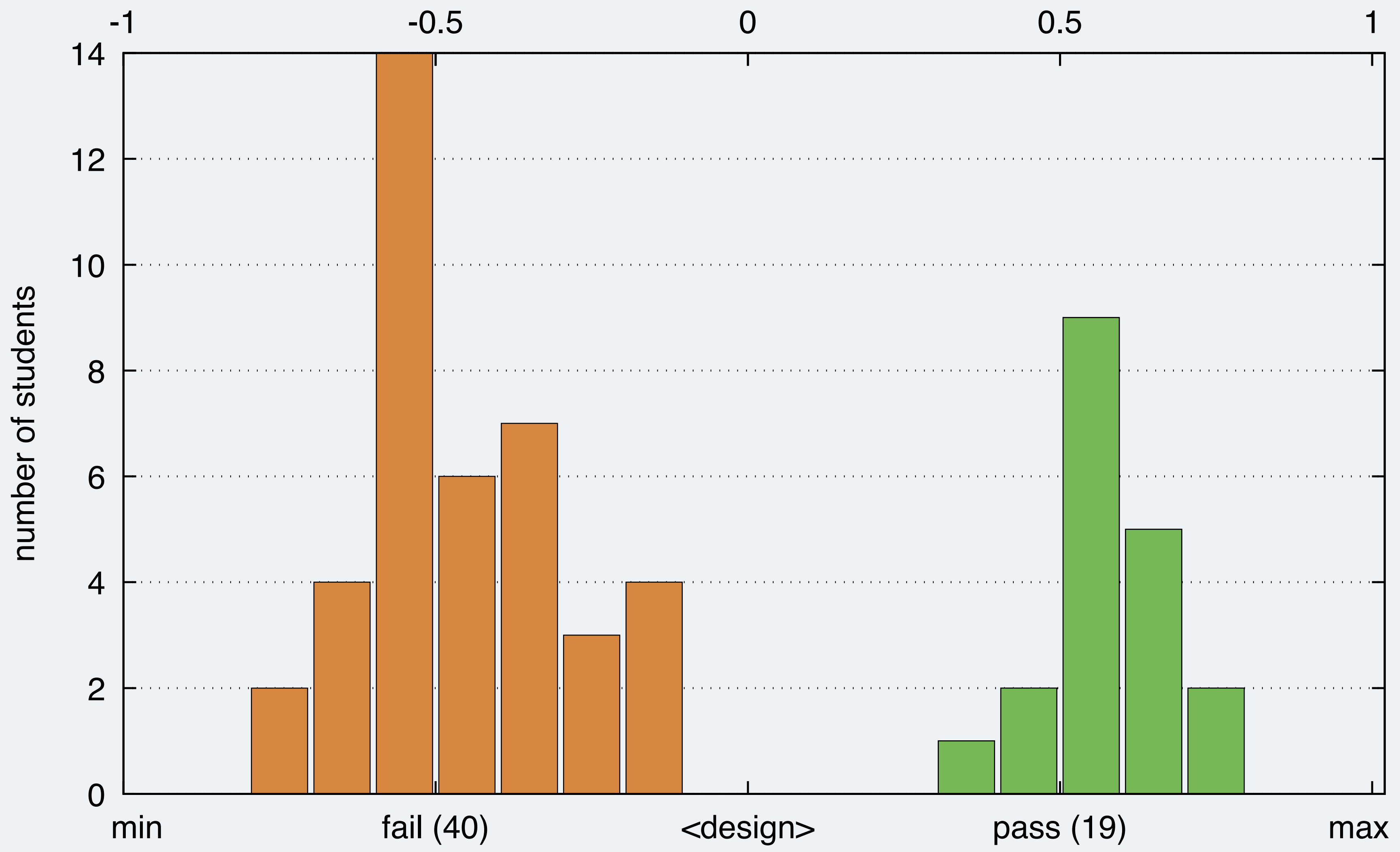
### Alternative algorithms ?

- welcome !

### Distribution

- the current algorithm produces a rather bimodal distribution which tends to clearly separate the "fails" from the "passes"

# Interpolation



## Feedback

### By default ...

- PMark generates some automatic text explaining what would be necessary to achieve the next grade:

Dylan (42) did not meet the requirements for any of the grades.

For a pass (50), we would like to have seen:

- a adequate for the injecting attribute instead of a almost.

John (60) achieved a pass (50) for the result.

For a distinction (70), we would like to have seen:

- a good for the shaving attribute instead of a almost.
- a good for the bandaging attribute instead of a adequate.
- a good for the counting attribute instead of a adequate.

# Custom feedback

## The mark scheme can be annotated

- to add custom feedback for individual rules and attributes

```
JSON2A: lickert4 (
  2 = "code to read the model from a JSON file"
  3 = "code which successfully reads the model from a JSON file"
  4 = "better designed code to read the model from a JSON file"
)
```

Clara (77) achieved a D2 (75) for the a2.
For a exceptional (80,E1), we would like to have seen:
- better designed code to read the model from a JSON file
- code with a view class which does not depend on JavaFx

## Feedback algorithm

### Logically ...

- it can be difficult to describe *precisely* what would be necessary to achieve the next grade
- in this example, if there were two tasks which were not "good", then meeting *either* of these would be sufficient to achieve the distinction

```
distinction:
  all but one #task = good
  and all #task = adequate
```

- PMark can display the full logical expression necessary to meet the requirement, but this is usually too awkward to be useful
- by default, it simply lists all of the individual criteria which might contribute to expression

## Where next?

### Evaluation

- PTAS Project
- IPPO (MSc course)
- INF1B (maybe)
- interest in discussing or trying out PMark very welcome

### Software

- potential interfaces (student projects)
  web or GUI?
  Learn integration?
- algorithm improvements
- suggestions?

**Paul Anderson**

<dcspaul@ed.ac.uk>

**Software & Documentation**

http://homepages.inf.ed.ac.uk/dcspaul/pmark