THE UNIVERSITY of EDINBURGH
informatics

Supporting Criteria-Based Marking

Paul Anderson & Volker Seeker

<dcspaul@ed.ac.uk>

<volker.seeker@ed.ac.uk>

Software & Documentation

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

Overview

- A (little) bit about marking
- The PMark program by example
- INF1B Case Study
  Experiences with PMark and a large CS programming class

Marking

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 (Moskal2000)

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 markers holistic impression (Sadler 2009)

We have been experimenting with ...

- lots of small criteria with a simple evaluation
  for example: "no", '"not really", "sort-of", "yes"
- software to assist in the combination of the criteria and produce an overall result which is closer to the markers 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 tasks have to be adequate for a pass"
- relate the outcome directly to the objectives
  "you failed because you weren't good at bandaging the dog"
- but these are not so easy to evaluate ...
  for example, with a spreadsheet or traditional application

Motivation

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

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

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

counting: mark

[results]

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

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

Feedback

Default 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.

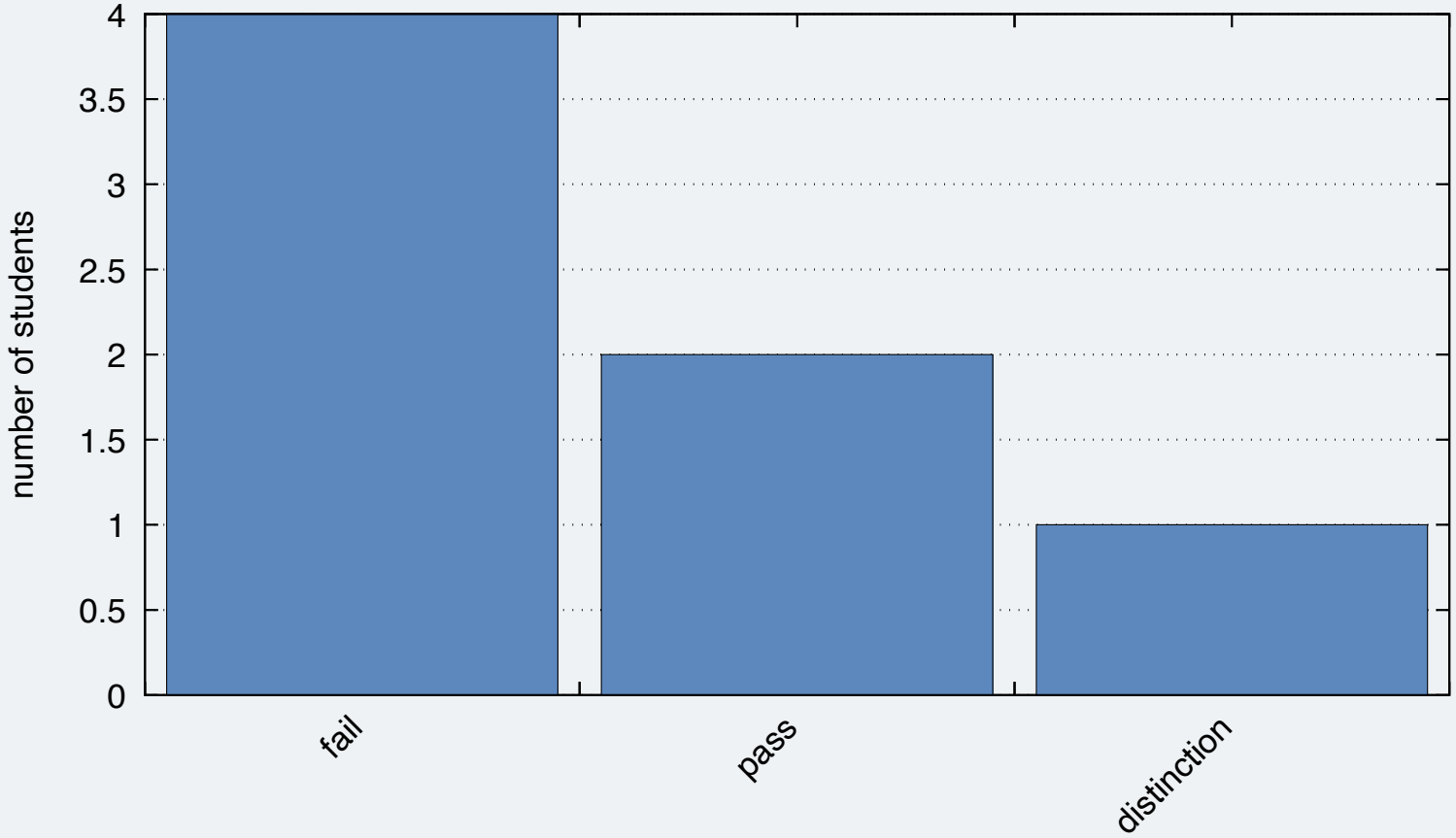
- a good for the counting attribute instead of a adequate.

Custom feedback

- the mark scheme can be annotated to add custom feedback for individual rules and attributes

Graphs

- PMark can produce graphs of the overall results, or individual attributes




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

THE UNIVERSITY of EDINBURGH
informatics

Supporting Criteria-Based Marking

Inf1B Case Study

Paul Anderson & Volker Seeker
<dcspaul@ed.ac.uk>
<volker.seeker@ed.ac.uk>

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

Inf1B - Course Overview

- First year programming course with large cohort (400+ students)
- Assessment via assignments only

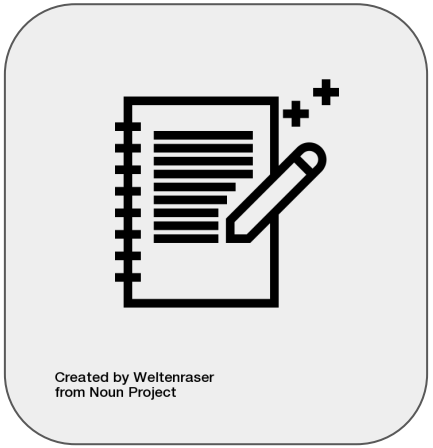
Inf1B - Course Overview

- First year programming course with large cohort (400+ students)
- Assessment via assignments only

cw1

programming

cw2



essay

cw3

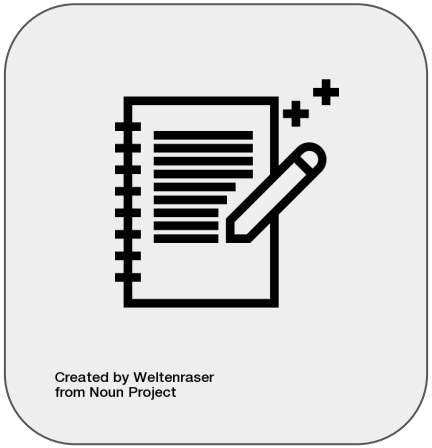
programming

Inf1B - Course Overview

- First year programming course with large cohort (400+ students)
- Assessment via assignments only

cw1

cw2



cw3

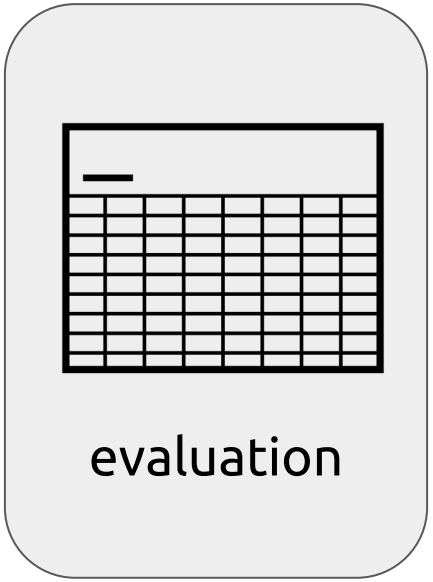
for credit

Marking Workflow

- Mark cw2 and cw3 as a unit based on expected criteria
- Split marking work among a team of markers

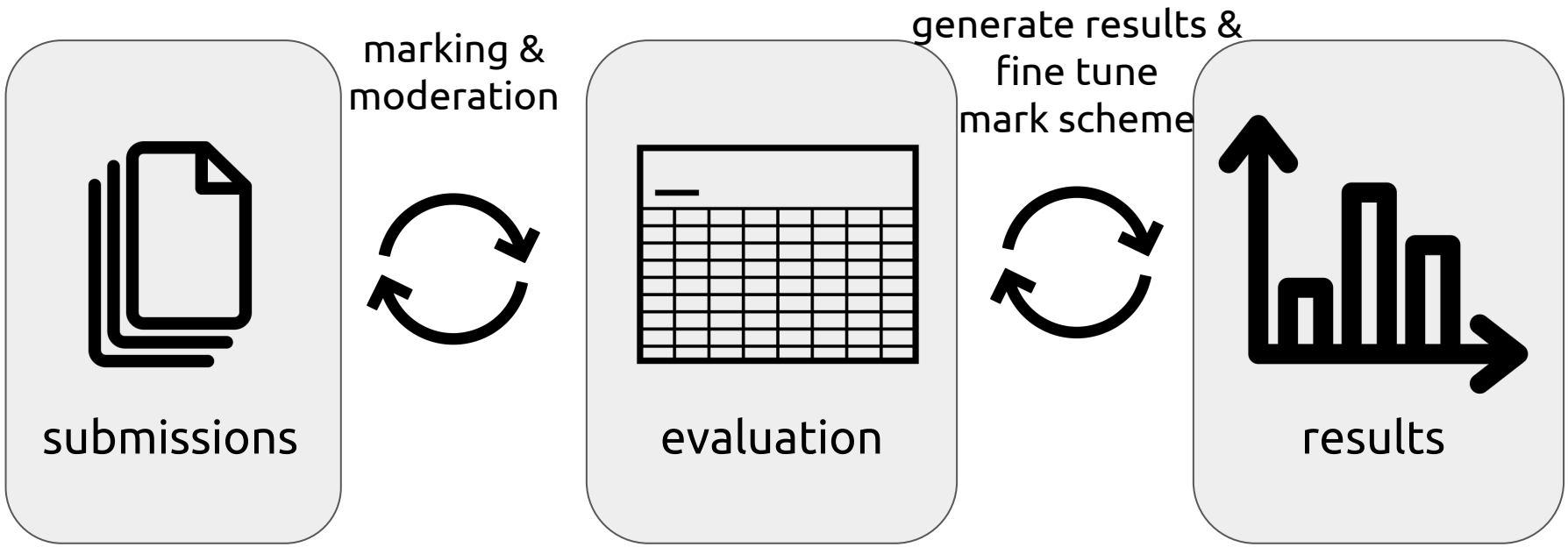
Marking Workflow

- Mark cw2 and cw3 as a unit based on expected criteria
- Split marking work among a team of markers



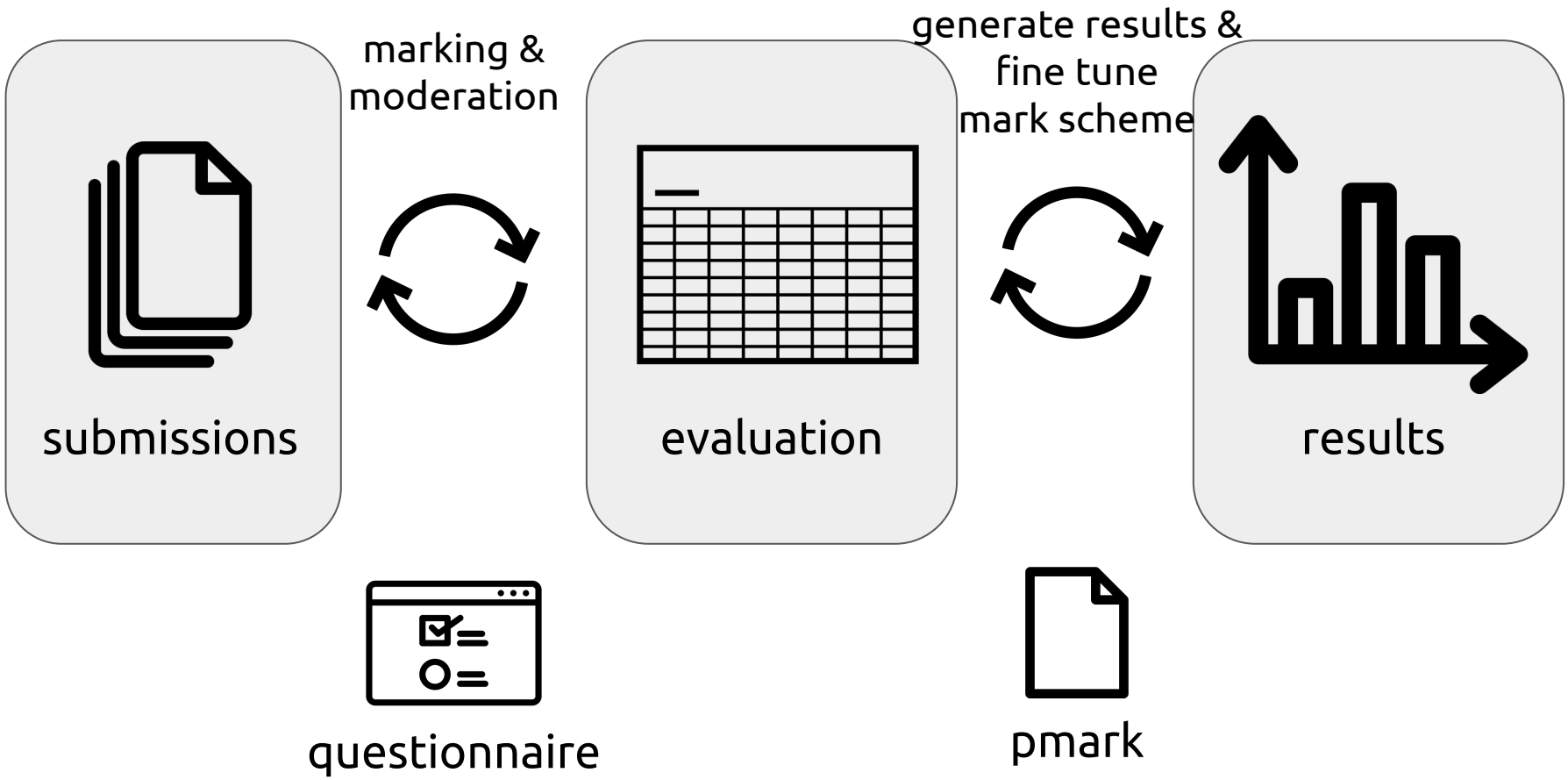
Marking Workflow

- Mark cw2 and cw3 as a unit based on expected criteria
- Split marking work among a team of markers



Marking Workflow

- Mark cw2 and cw3 as a unit based on expected criteria
- Split marking work among a team of markers



Mark Scheme: attribute types

```
[types]
// no submission, no, not-really, sort-of, yes
lickert: +/- [0,1,2,3,4]

...

inf1b-cms: [ 0..100
       P1 = pass = 40
       P2 = 45
       G = good = 50
       VG = very-good = 60
       D = distinction = 70
       E = exceptional = 80
       E2 = 85
       O = outstanding = 90
      ]
```

Mark Scheme: attributes

```
[attributes]
// cw2
DOCUMENTATION_CW2: lickert #qcw2
STRUCTURE_CW2: lickert     #qcw2
LANGUAGE_CW2: lickert      #qcw2

// cw3
PLAUSIBLE: lickert

BASIC_T1: lickert    #fundB #basic
ADVANCED_T1: lickert #fundA #advanced
...
```

Questionnaire

```
[attributes]
// cw2
DOCUMENTATION_CW2
STRUCTURE_CW2
LANGUAGE_CW2

// cw3
PLAUSIBLE

BASIC_T1
ADVANCED_T1
...
```

Assignment Part II

Does the answer to the Code Documentation question provide actionable steps to improve corresponding code which are specific, justified and kind?

N/A

- No + - Not really + - Sort-of + - Yes +

Does the answer to the Code Structure question provide actionable steps to improve corresponding code which are specific, justified and kind?

N/A

- No + - Not really + - Sort-of + - Yes +

Does the answer to the Use of the Java Language question provide steps to improve corresponding code which are specific, justified and kind?

N/A

- No + - Not really + - Sort-of + - Yes +

Assignment Part III — Completion, Correctness and Robustness

Was some plausible code submitted for a significant part of the fundamental questions even if it does not compile?

N/A

- No + - Not really + - Sort-of + - Yes +

Are all basic features implemented for task 1 (BookEntry)?

Data Table

```
[attributes]
// cw2
DOCUMENTATION_CW2
STRUCTURE_CW2
LANGUAGE_CW2

// cw3
PLAUSIBLE

BASIC_T1
ADVANCED_T1
...
```

id	name	DOCUMENTATION_CW2	STRUCTURE_CW2	LANGUAGE_CW2	PLAUSIBLE	BASIC_T1
s1234567	Name 1	4	4	4	4	4
s1234568	Name 2	3+	3	3	1-	4
s1234569	Name 3	0	0	0	0	0
s1234570	Name 4	4-	3+	4-	4	4
s1234571	Name 5	4	4	4	4	4
s1234572	Name 6	4	3+	3	1	0
s1234573	Name 7	4	4	4	4	4
s1234574	Name 8	3+	4-	4-	1-	4
s1234575	Name 9	3+	4-	4	1-	4
s1234576	Name 10	4	1	4	0	4-
s1234577	Name 11	4	4-	3+	4	4
s1234578	Name 12	3	4	4	0	4
s1234579	Name 13	4	4	4	4	4
s1234580	Name 14	4	4	4	4	4
s1234581	Name 15	4	4	4	4	4
s1234582	Name 16	4-	3-	3-	4	4
s1234583	Name 17	0	0	0	0	0
s1234584	Name 18	4-	3+	4-	4	4
s1234585	Name 19	3+	3-	0	1-	4
s1234586	Name 20	3+	3	3	4	4
s1234587	Name 21	0	0	0	4	4
s1234588	Name 22	4	4-	4	0	4
s1234589	Name 23	4	4	4	4	4
s1234590	Name 24	4-	4-	4-	1-	4
s1234591	Name 25	4-	4	4-	4	4
s1234592	Name 26	4	4	4	4	4
s1234593	Name 27	4	4-	4	4	4
s1234594	Name 28	4	4	4	2	1
s1234595	Name 29	4	4	4	4	4

Mark Scheme: Rules

```
[rules]
// PASS
plausible-code: PLAUSIBLE=3

// P2
min-fundamental-basic: most #fundB=2
cw2-attempt: one #qcw2=2

...
```

Mark Scheme: Rules

```
[rules]
// PASS
plausible-code: PLAUSIBLE=3

// P2
min-fundamental-basic: most #fundB=2
cw2-attempt: one #qcw2=2

...

// ------------- Grade Rules -----------

P1: plausible-code

P2: all { min-fundamental-basic, cw2-attempt }
```

Mark Scheme: Results

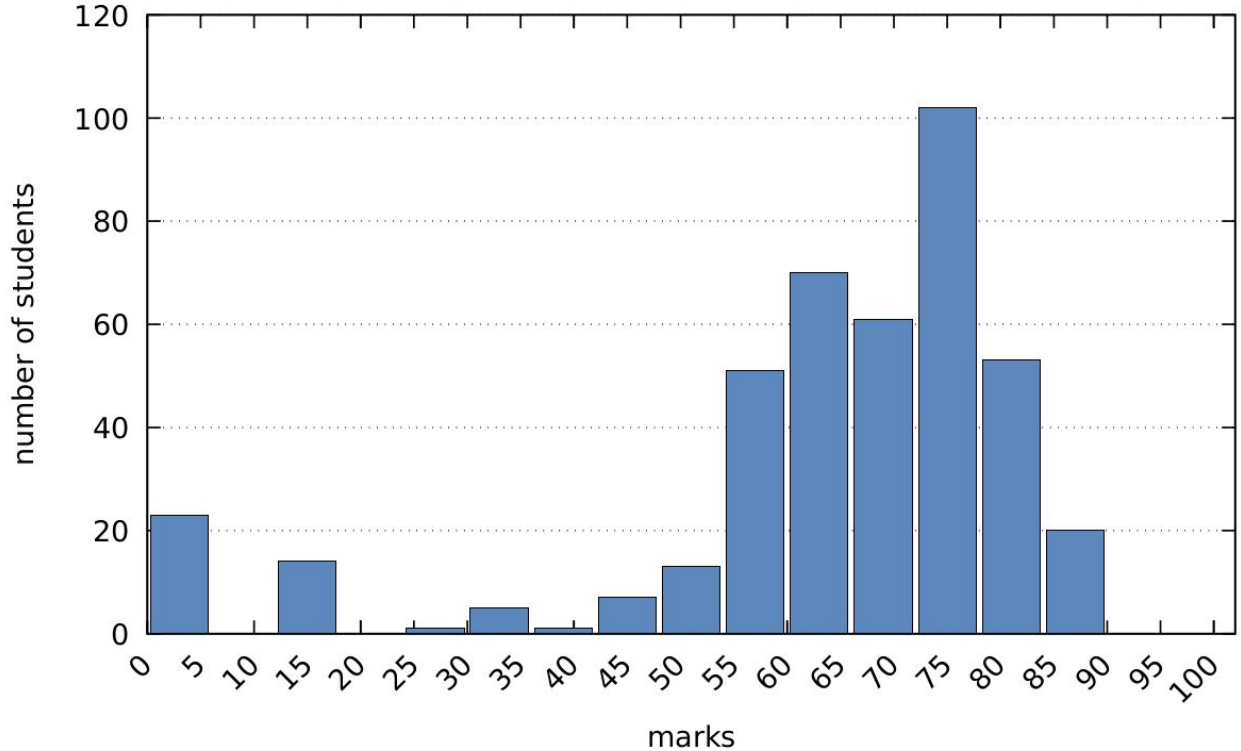
```
[results]
final-grade: inf1b-cms

[graphs]
Inf1b-results: final-grade (
    barwidth = 6
    xlabel = “marks”
)
```

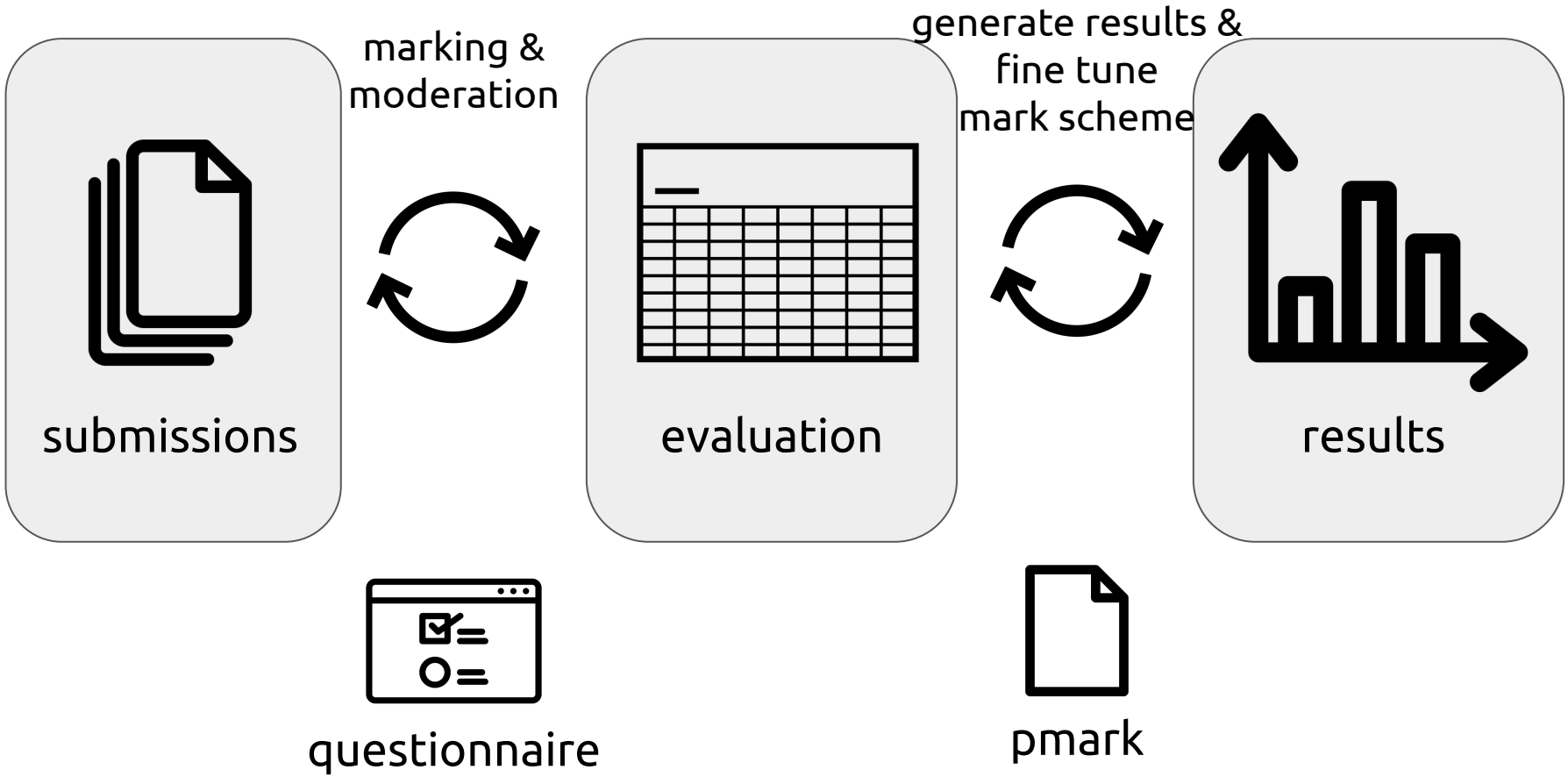
Mark Scheme: Results

```
[results]
final-grade: inf1b-cms

[graphs]
Inf1b-results: final-grade (
    barwidth = 6
    xlabel = “marks”
)
```



Fine Tuning the mark scheme



- What would be the minimum requirement for a pass?
- What if cw3 went really well but cw2 did not?
- What if some questions turned out to be much more difficult than expected?
- How can truly outstanding submissions be acknowledged?

Image References

- circle arrow by Tinashe Mugayi from the Noun Project
- histogram by Adnen Kadri from the Noun Project
- data table by Gene Stroman from the Noun Project
- stack of paper by amy morgan from the Noun Project
- questionnaire by LUTFI GANI AL ACHMAD from the Noun Project
- File by Galaxicon from the Noun Project

Where next?

Evaluation

- possible PTAS Project?
- Informatics MSc course
- interest in discussing or trying out PMark very welcome!

Software

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

Paul Anderson & Volker Seeker

<dcspaul@ed.ac.uk>

<volker.seeker@ed.ac.uk>

Software & Documentation

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