SEOC Deliverable 1
Feedback

Massimo Felici

School of informatics

Overview

- Collective Feedback to the Different Parts of Deliverable 1
- Review of Marks

Software Requirements Specification

- Some functional requirements need to be refined - they often capture different requirements or subtle non-functional requirements together.
- Non-functional requirements are underspecified or described by general features (e.g., security, reliability, etc.). This makes difficult to assess them. Moreover, it makes their interpretations to be subjective.
- Lack of traceability links to Use Cases.
- Difficulties in applying Requirements Engineering practices (e.g., numbering requirements).

Use Cases

- Missing Actors.
- Lack of generalisation between actors.
- Lack of analysing different viewpoints (i.e., D, P and O) consistently. Different viewpoints identify different system functionalities, which might relate to different system parts or components. However, at the use case level, it is necessary to analyse the system as a whole and its interaction with 'external' actors.
- Identifying system parts mixes requirements with design aspects. This is a critical mistake/pitfall in requirements engineering practices.

- Lack of relationships between Use Cases.
- The use cases fail to capture system functionalities. They capture general activities, which are too complex for being system functionalities. This could be due to a lack of analysis (of system requirements) or a high level of granularity (that is, use cases should have been refined).
- lack of understanding of `<includes>` and `<extends>` relationships between use cases. These relationships have been mistakenly used in order to stress procedural and workflow aspects rather than relationships among different system use cases, hence, functionalities.
- Use cases capturing work flows.
- Use cases that are just special 'cases', which might be specified in the use case descriptions, rather than different ones.

- Missing conditions for extensions or variations.

Class Diagrams

- Missing classes.
- Some classes or relationships between them seem to be too complex. This might affect any maintainability – changes might affect most of the classes in the design. This is usually due to tight-coupling design.
- Lack of understanding the differences between aggregation and composition.
- Although design patterns have not been explicitly required, some of them could have been useful.
- Lack of validation (by CRC card games) makes difficult to understand/assess how a class diagram realises specific functional requirements.

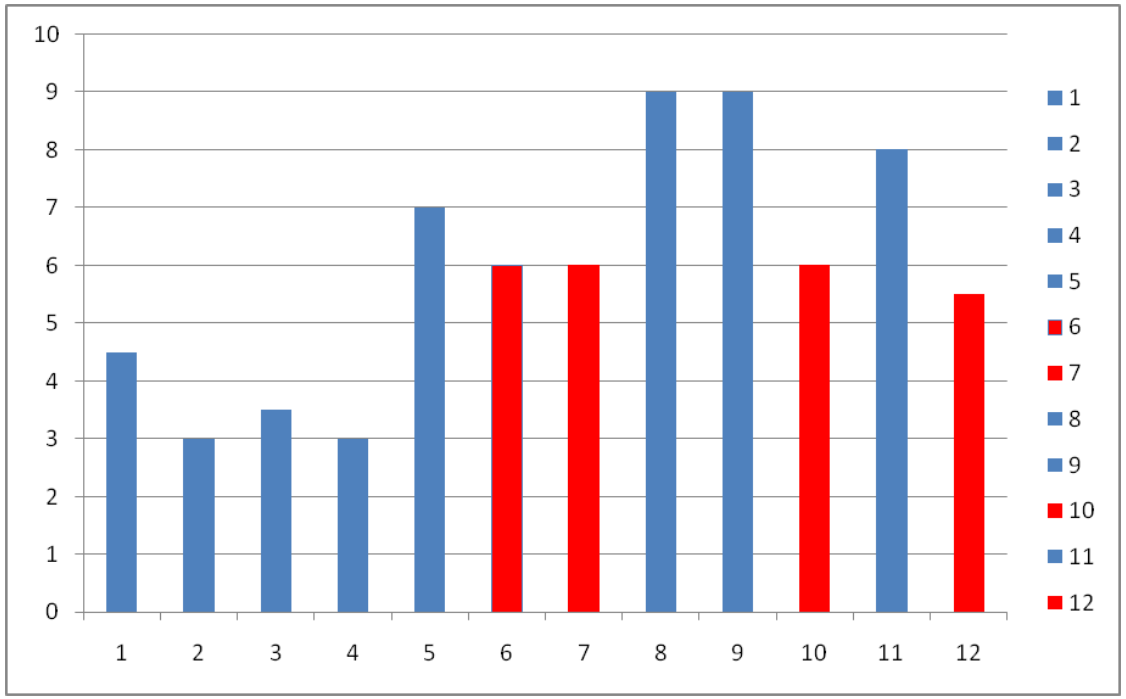
Validation of Class Model

- Missing or incomplete CRC cards. Note that there should be one CRC card for each class in your design (i.e., Class diagram).
- Underspecified, unclear or wrong CRC games.
- Inconsistencies between CRC cards and class diagram.

Deliverable Assessment

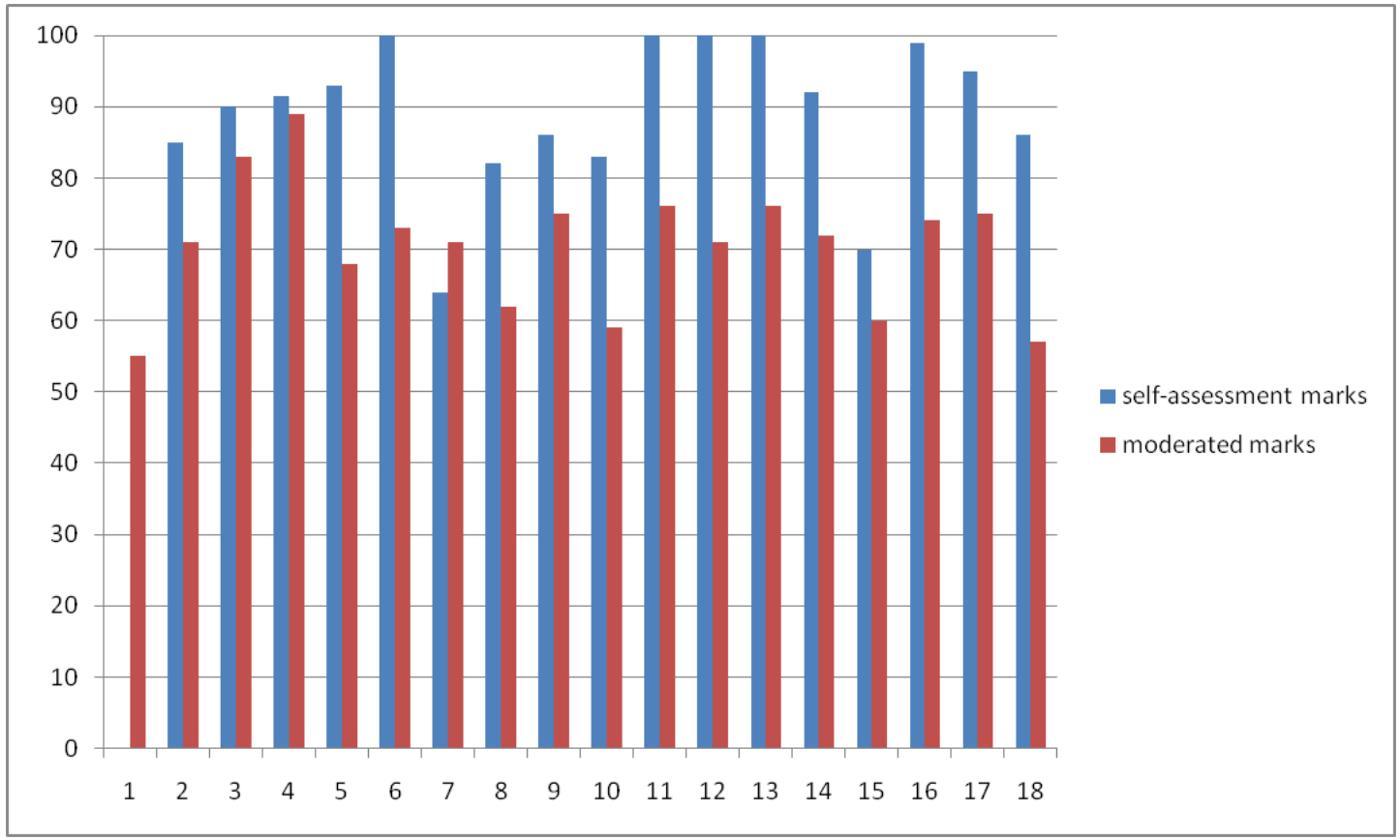
- Missing (parts of the) assessment form

Median Marks per Deliverable Aspect

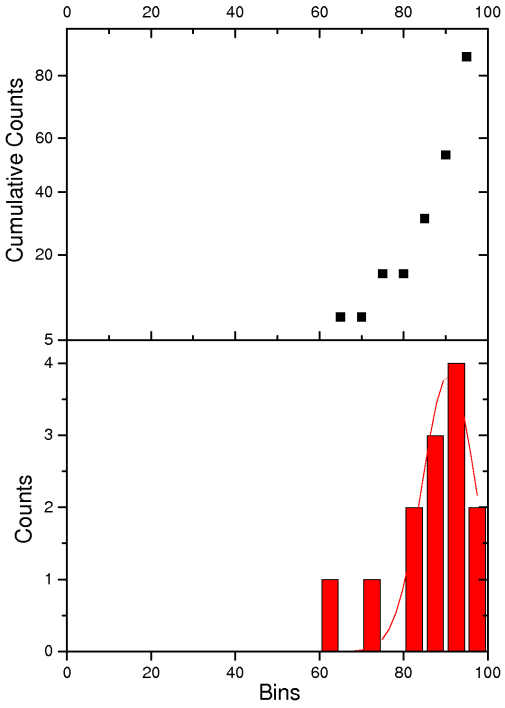


1. Software Requirements Specification
2. Types of Requirements
3. Requirements Completeness
4. Requirements Correctness
5. Use Cases
6. Use Case relationships
7. Use case templates
8. Class Diagram
9. Attributes and Methods
10. Class Relationships
11. CRC Cards
12. CRC Validation Games

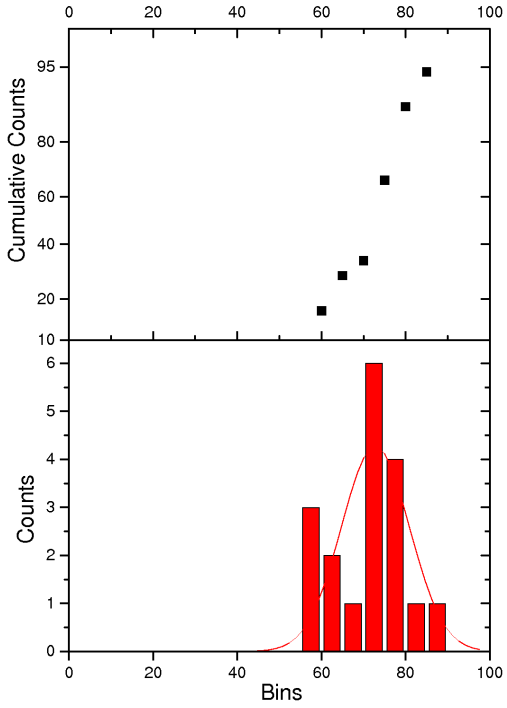
Marks


100
90
80
70
60
50
40
30
20
10
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
self-assessment marks
moderated marks

Mark Distributions



self-assessment distributions



moderated distributions

