# A Checklist for Requirement Defects Discovery in an Academic Environment

Michael Baldwin

Department of Computer Science

Tennessee Technological University

## Introduction

- Utilizing a checklist while inspecting SRS documents aids in locating defects [1]

- Most checklists are too long and impractical for use in academic environments

# Defect Categories

- Correct
- Complete
- Unambiguous
- Verifiable

## Correct

- An SRS is correct if [2,3]:
  - every requirement represents something required of the system
  - is free of faults
  - meets user expectations

## Correct

1. **Are all of the requirements necessary to meet the system objectives?**
2. **Are there requirements that seem illogical?**
3. **Are the requirements free of syntactical and grammatical errors?**
4. **Are all requirements listed in appropriate sections of the SRS?**

## Correct

5. **For specific functions, are all specified inputs/outputs necessary and if so, are they described correctly?**
6. **Are contents, formats and constraints of all display outputs described?**
7. **Are there requirements that should really be considered as design?**

## Complete [4,5]

- Includes all expected functionality of the software
- Contains no missing functions, output, etc.
- No sections are marked TBD

## Complete

1. **Are all requirements adequate to meet objectives?**
2. **Are all requirements defined?**
3. **Is there any information missing from any individual requirement that is needed to effectively implement the system?**
4. **Is a standard format for requirements documentation with all essential parts present?**
5. **Is there any TBD (To Be Determined) section for any of the requirements?**

## Complete

6. Do the requirements provide an adequate basis for design?
7. Is all functionality related to individual requirements included?
8. Are all functions refined or elaborated to an appropriate level of detail?
9. Should the requirements be expressed in more or less detail in order to be effective?

## Unambiguous

- Every requirement stated had one and only one possible interpretation [7]

## Unambiguous

1. Is each individual requirement described in a discrete and unambiguous manner and with "one and only one" possible interpretation?
2. Do requirements consist of non-standard terms that are not defined or defined in a way that causes confusion?
3. Are the requirements distinct enough to generate test cases and detailed design specification?

## Verifiable

- If it is possible to develop a method to check a finite method of checking if each individual requirement can be satisfied by the system [6]

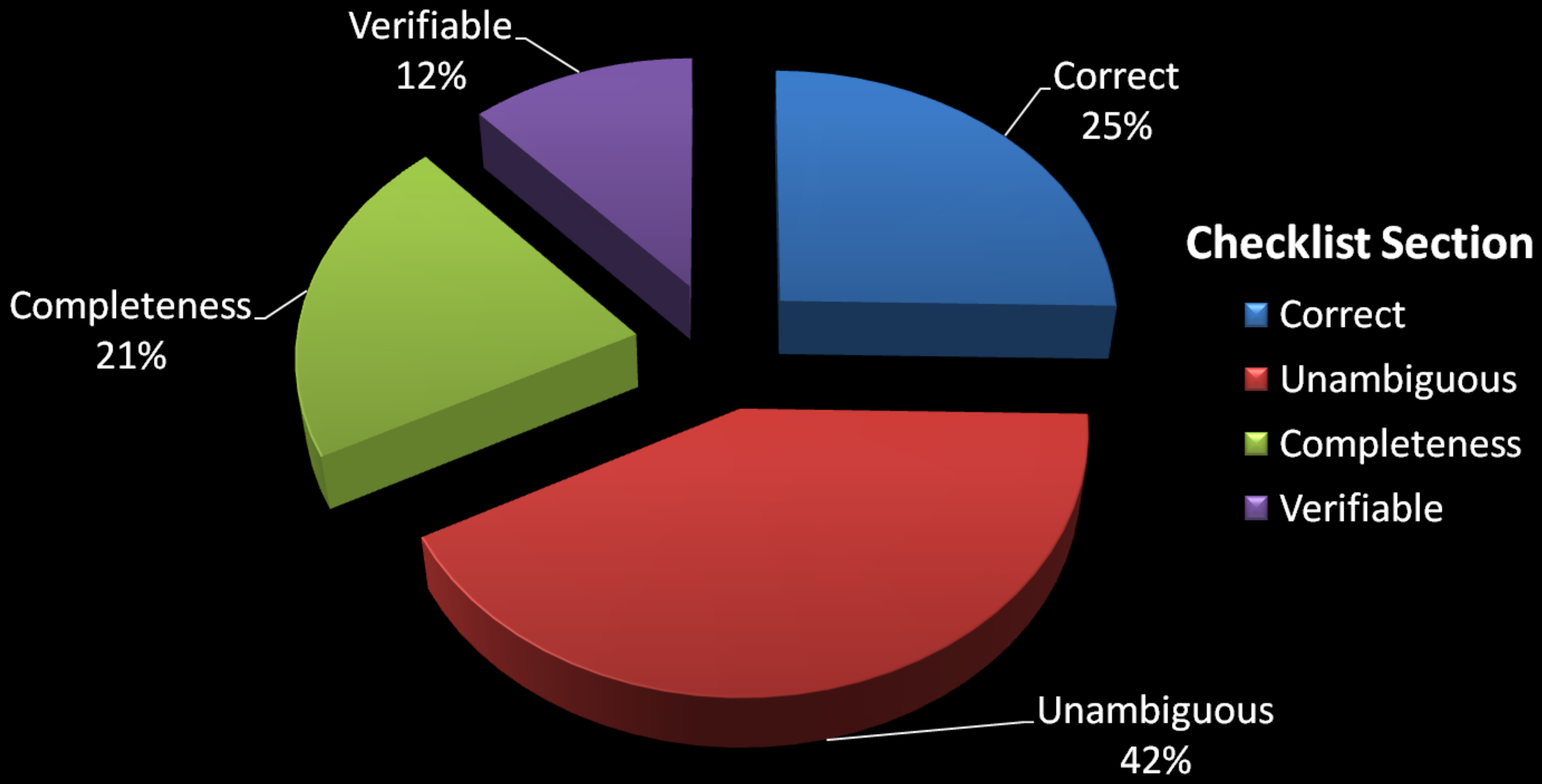
## Verifiable

1. **Are all objectives measurable, clear and reasonable?**

2. **Are there any requirements which cannot be verified due to multiple interpretations?**

3. **Are there requirements that cannot be verified by a process executed by man or machine?**

# Defect Distribution



## How the checklist was developed.

- A set of Software Requirement Specification (SRS) documents were analyzed using a checklist [7] to locate defects.

- Defects were recorded then the frequency of each category and rule was determined

## How the checklist was developed.

- The current checklist was then created by selecting the most frequently occurring items from the original checklist

## Current Use

- Currently being used in an undergraduate software engineering course.
- Given to students in both the 2007 spring & fall semesters.

## Conclusion

- A simple checklist, like this one, could greatly reduce the number of requirement defects
- A simplified checklist can be practical for academic use as a part of student projects

# References

[1] Boehm, B. Verifying and validating software requirements and design specifications. IEEE Software (1984), 7588.

[2] Davis, A. M. Software Requirements: Objects, Functions, and States. Prentice Hall, 1993.

[3] Davis et. al. Software Requirements Engineering, 2nd ed. IEEE Computer Society, 1997. Identifying and measuring quality in a software requirement specification, pp. 194206.

[4] Shull, F., Rus, I., and Basili, V. How perspective-based reading can improve requirements inspections. IEEE Computer 33(7) (2000), 7379.

[5] Siraj, A. A software inspection checklist based on IEEE recommended practice for software requirements specifications.

[6] The Institute of Electrical and Electronics Engineers (IEEE). Glossary of software engineering terminology, 1990.

[7] The Institute of Electrical and Electronics Engineers (IEEE). IEEE recommended practice for software requirements specifications, 1998.