Software metrics

CS-370: Software Design and Development

Robert Bruce

What are software metrics?

- Software metrics are characteristic which describe a finished software project such as “lines of code written, bugs fixed, milestones missed”.
- In a post-project analysis, software metrics can reveal how productive and efficient (or unproductive and inefficient) a project was. Identifying the cause(s) for key issues impacting a former project can be a huge influence on improving development productivity and efficiency on future software projects.
- For bugs or software defects, it is vital to ask the following questions during a post-project assessment:
  - “How could you have avoided the defect in the first place?”
  - “How could you have detected the defect sooner?”
  - “For customer-discovered defects, how could you have found the defect before the customer did?”

Source: pages 372-374, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Measurable software metrics

- **Cost**: “Money spent on the project for hardware, software, development tools, networking services, paper, training, and so forth.”
- **Effort**: A measure in person-hours for how much time it takes to finish a project.
- **Defect Rates**: “The number of defects measured over time.”
- **Lines of Code (LoC)**: “The number of lines of code produced per day.”
- **Pages of Documentation**: This metric can be divided into project documentation (for developers) and user-documentation (for customers).
- **Functionality**: “How well does the application do what it is supposed to do?”
- **Quality**: “Do customers perceive this application as being high quality?
- **Complexity**: “How complex is the project?” This is difficult to measure. Hint: examine product documentation to determine software product complexity. Extensive documentation may indicate a complex project.
- **Efficiency**: “How efficient is the application?” This is difficult to measure.
- **Reliability**: “How reliable is the application?” Hint: count the number of times the software product fails under various scenarios such as network outtage.
- **Maintainability**: “How easy will it be to maintain the application in the long term?” Hint: examine amount of project documentation, software comments, code complexity, etc.

Source: pages 385-386, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Ishikawa Diagrams

- Ishikawa diagrams are a graphical tool for identifying the root cause of a software defect.
- Ishikawa diagrams are also called:
  - Fish-bone diagrams
  - Cause-and-effect diagrams
- Ishikawa diagrams are named after the Japanese organizational theorist, Kaoru Ishikawa.

Source: page 377, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Example: Ishikawa Diagram

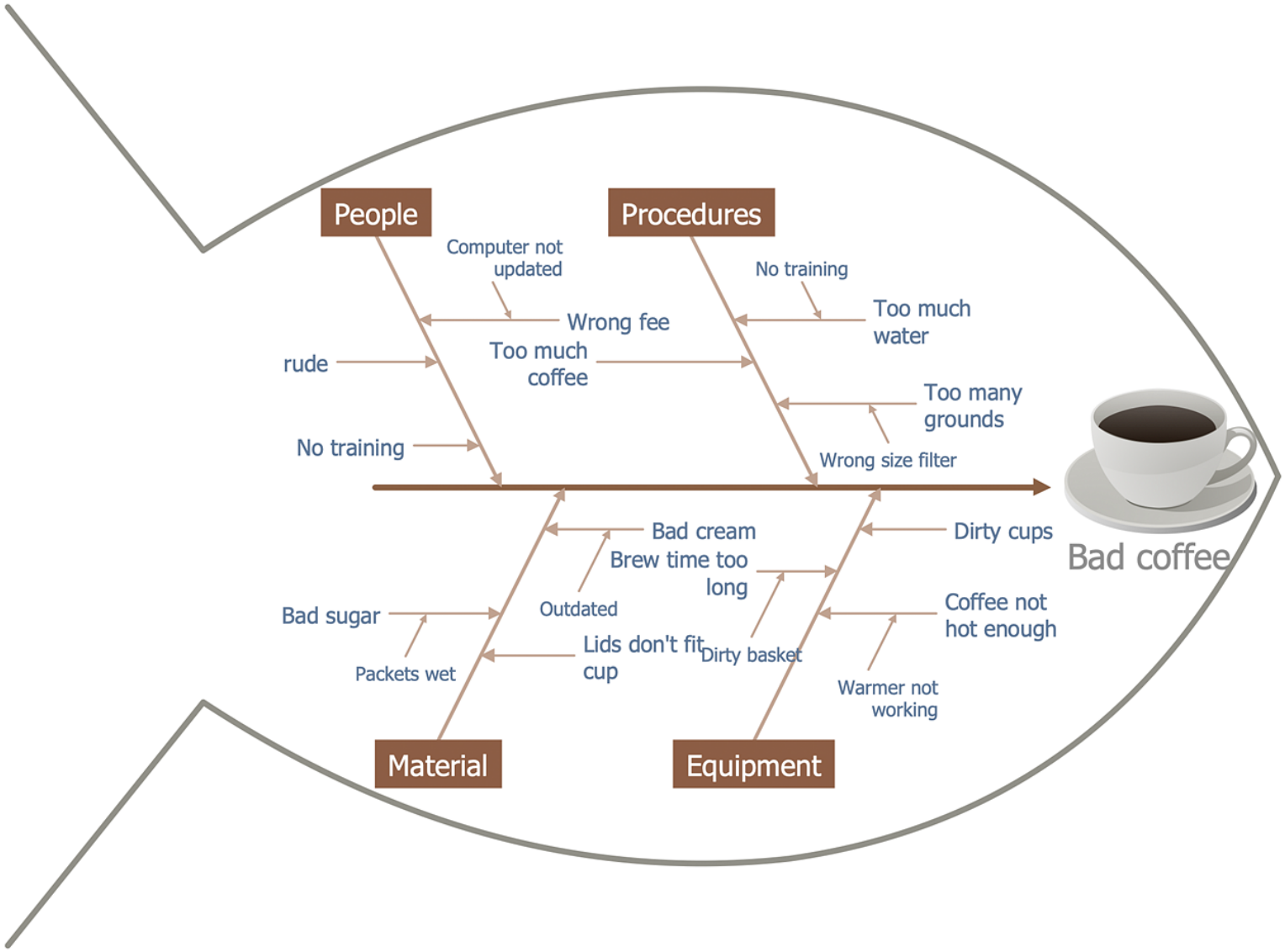

Image source: https://www.conceptdraw.com/How-To-Guide/picture/ishikawa-diagram-sample-new.png