Unit 2: Study: Why Projects Fail and Gathering Requirements Exercise

Required Reading

Behave (2020) Behaviour Driven Development: The Gherkin Language.

Behavior-driven development (BDD) is an agile software development technique that promotes collaboration among developers, QA, and business stakeholders. It was introduced by Dan North in 2003 as an evolution of test-driven development (TDD). BDD emphasizes understanding software behavior through discussions with stakeholders and writing test cases in natural language, making them accessible to non-programmers. Key practices include establishing stakeholder goals, using examples to describe application behavior, automating tests, and questioning software responsibilities. BDD is driven by business value and often utilizes the Gherkin language for defining scenarios. It encourages the use of mocks to simplify collaboration and maintain loosely coupled classes.
The Gherkin language is a structured, natural language used in behavior-driven development (BDD) to define requirements and scenarios in a clear and understandable format. It employs keywords such as "Given," "When," and "Then" to outline the context, actions, and expected outcomes of a scenario. For example, a Gherkin scenario might describe the behavior of a retail application regarding item returns. This language facilitates collaboration among technical and non-technical stakeholders by providing a common framework for discussing application behavior. Scenarios written in Gherkin can also be automated for testing purposes, enhancing the development process.

Additional Reading

Andriole, J. (2017) The Death of Big Software. Communications of the ACM 60(12): 29-32.

The article "The Death of Big Software" by Stephen J. Andriole discusses the decline of large, monolithic software architectures in favor of smaller, cloud-based solutions. It highlights the high failure rates of big software projects, driven by issues such as loss of control, governance challenges, and the complexity of implementation. The rise of microservice-based architectures and cloud delivery is presented as a more flexible and cost-effective alternative, enabling continuous digital transformation. The article concludes that traditional big software design is obsolete, and vendors are adapting by offering smaller, cloud-based solutions, though they have yet to fundamentally rearchitect their applications.