

Software Testing

17-316/616 Fall 2025

AI Tools for Software Development

<https://ai-developer-tools.github.io>

Austin Henley and Andrew Begel

Today

- Read this paper: <https://arxiv.org/abs/2510.17164>
 - Maria Deolinda Santana, Cleyton Magalhaes, Ronnie de Souza Santos. **Software Testing with Large Language Models: An Interview Study with Practitioners.** Alware 2025.
- Overview on testing
- Mobbing
- Sign up for reflections
- Logistics: New deadlines

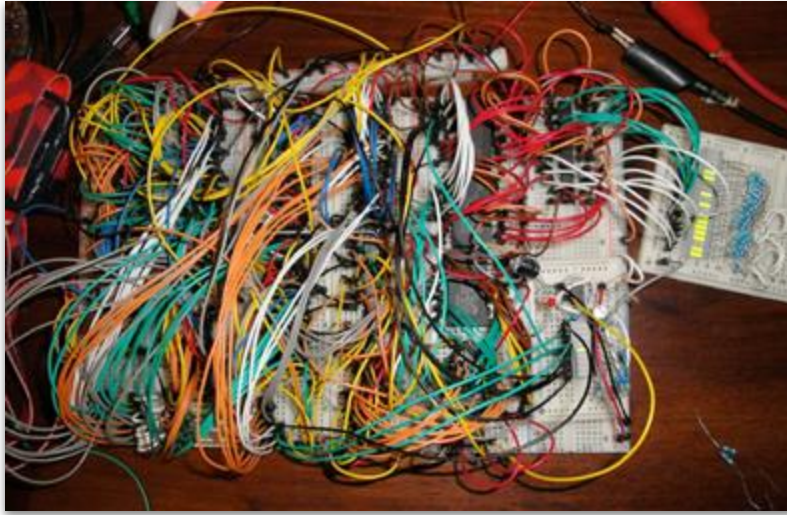
How Testers Use LLMs

- Generating test cases
- Automating test execution through scripting
- Documenting tests and writing test specifications
- Learning about best practices in testing
- Generating clear and structured acceptance criteria
- Examining differences in test output from multiple runs

Software Quality



Internal Quality



- Is the code well structured?
- Is the code understandable?
- How well documented?

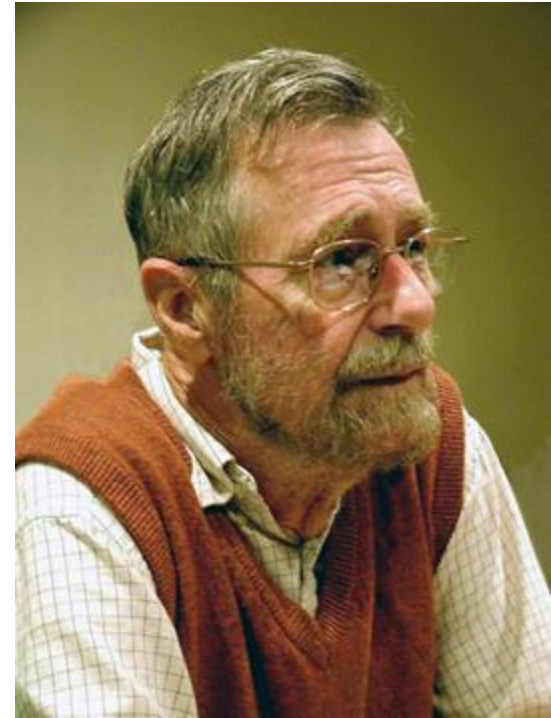
External Quality



- Does the software crash?
- Does it meet the requirements?
- Is the UI well designed?

Avoid the *absence of defects* fallacy

- Testing shows the presence of defects
- Testing does not show the absence of defects!
- “no test team can achieve 100% defect detection effectiveness”

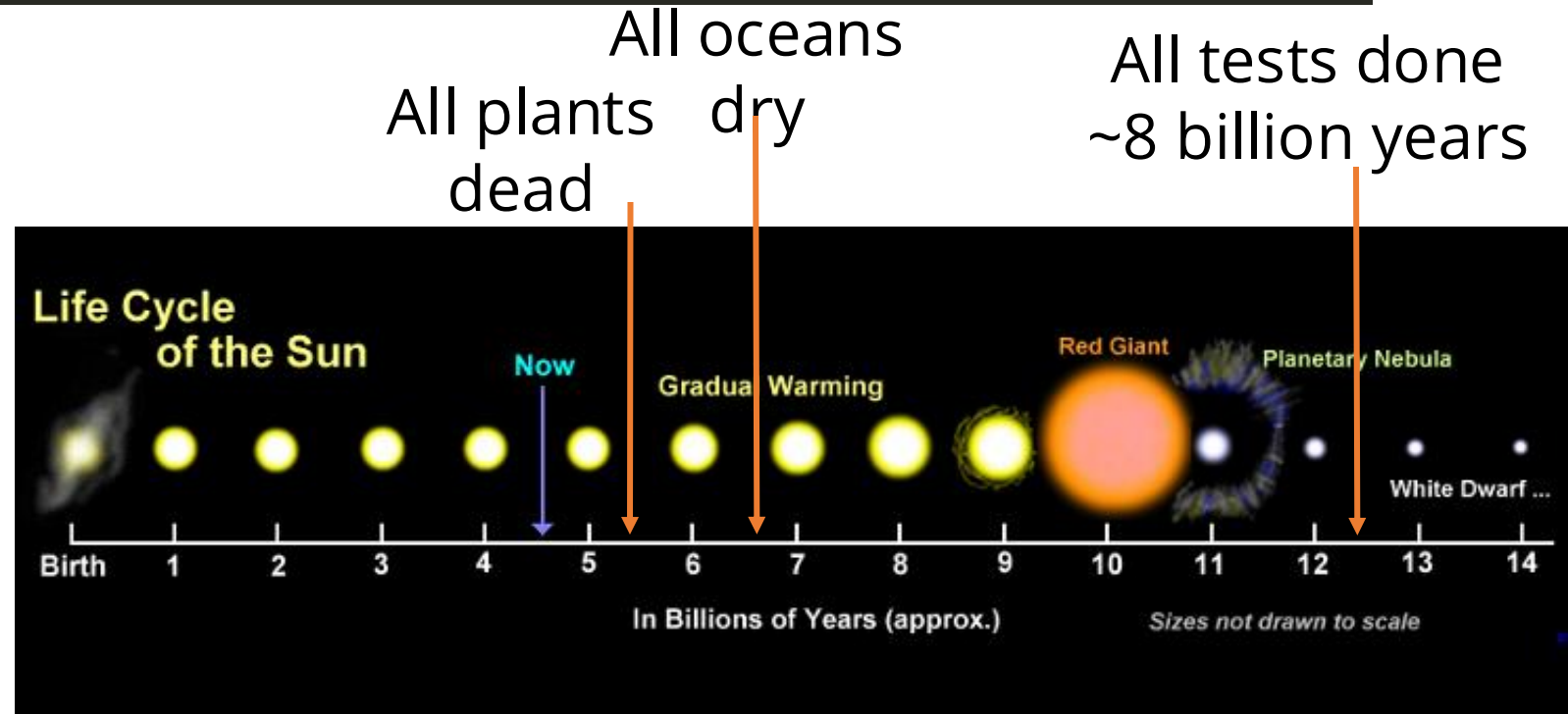


Effective Software Testing: A developer's guide. Maurizio Aniche

Exhaustive testing is impossible

```
1 def is_valid_email(email: str) -> bool:
2     ...
```

- A simple function, 1 input, string, max. 26 lowercase characters + symbols (@, ., _, -)
- Assume we can use 1 zettaFLOPS: 10^{21} tests per second

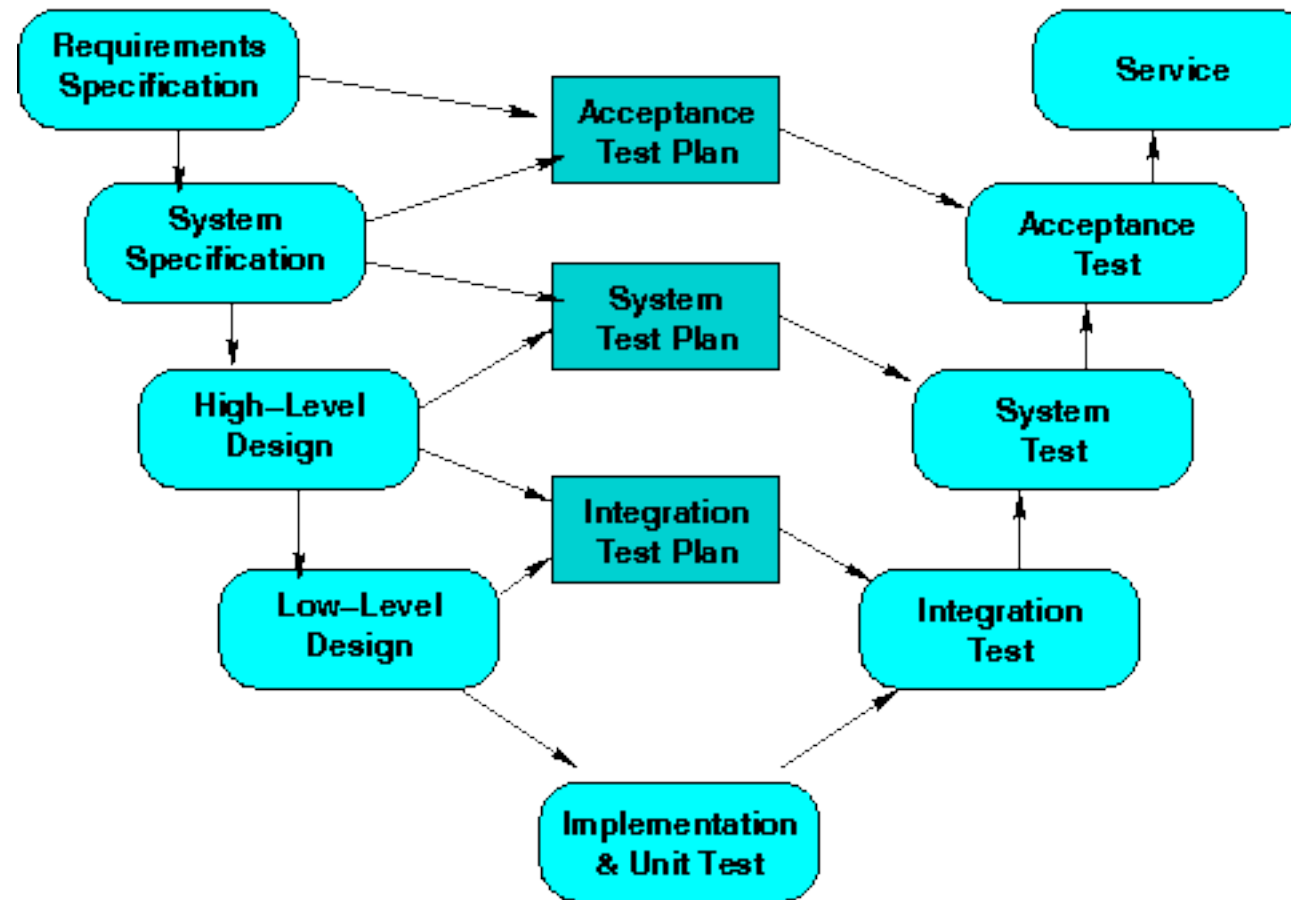


How to create tests?

Test design techniques

- **Opportunistic/exploratory testing:** Add some unit tests, without much planning
- **Specification-based testing ("black box"):** Derive test cases from specifications
 - Boundary value analysis
 - Equivalence classes
 - Combinatorial testing
 - Random testing
- **Structural testing ("white box"):** Derive test cases to cover implementation paths
 - Line coverage, branch coverage

Testing all the things



Unit tests



```
function validateEmail(email) {  
  if (typeof email !== 'string') return false;  
  
  const atIndex = email.indexOf('@');  
  if (atIndex === -1) return false;  
  
  const [localPart, domain] = email.split('@');  
  if (!localPart || !domain) return false;  
  if (domain.toLowerCase() !== 'gmail.com') return false;  
  
  return true;  
}
```



```
test('returns true for valid Gmail address', () => {  
  expect(validateEmail('student@gmail.com')).toBe(true);  
});  
  
test('returns false for non-Gmail address', () => {  
  expect(validateEmail('user@yahoo.com')).toBe(false);  
  expect(validateEmail('admin@outlook.com')).toBe(false);  
});  
  
test('returns false for emails with spaces', () => {  
  expect(validateEmail('my user@gmail.com')).toBe(false);  
});  
  
test('returns true for uppercase Gmail address', () => {  
  expect(validateEmail('USER@GMAIL.COM')).toBe(true);  
});  
  
test('returns false for non-string input', () => {  
  expect(validateEmail(null)).toBe(false);  
  expect(validateEmail(123)).toBe(false);  
  expect(validateEmail({})).toBe(false);  
});
```

Ask your AI to generate a unit test for your project

- Open up your project in Cursor
- Use Jest for unit testing in JavaScript
- Ask the LLM to generate one unit test for a specific function from a specific file of code
- Ask the LLM how to run your unit test, and run it

Mobbing: Unit testing

- Let's go to your Tinder or Instagram app
- Ask the LLM to add one test for a specific function of your choice
- Run it and review it
- Ask the LLM to add more tests for that function
- Run them and review it
- Ask the LLM to add tests for the entire file
- Run them and review it

Upcoming deadlines and reflection