CS608
Software Testing

Dr. Stephen Brown

Room Eolas 116

stephen.brown@mu.ie

CS608

Testing in the Software Process

(Essentials of Software Testing, Chapter 13)

Approaches to Testing

1. Test at end of product development
    1. Wait until all the code has been written and then to test the finished product all at once
2. Test during development
    - A more modern approach is to test the software while it is being developed

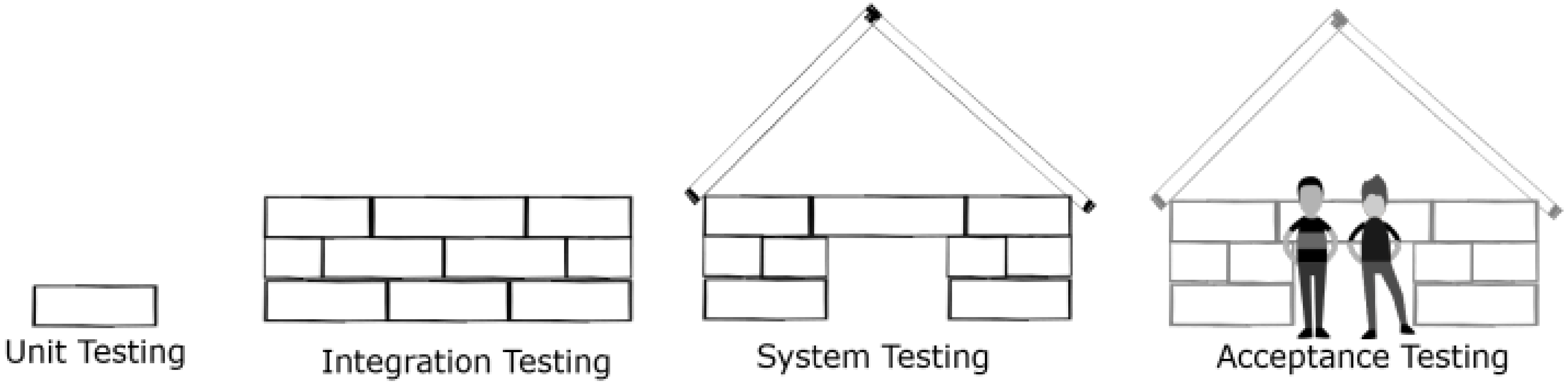
1. Test at end of Product Development

- “Big Bang" development
- Superficially attractive: testing activities do not hold back the progress
- Risky strategy: low likelihood final product works well (complex/large programs)
- If tests do reveal faults in the program, hard to identify their source in a large codebase

2. Test During Development

- “Incremental" development
- Individual modules, or software features, are tested as they are written
- Continues por additional software increments until product is complete
- May delay the release of the final product, but should produce higher quality, and provide tested interim versions
- Also, more responsive to changing user requirements

Stages of Incremental Testing



- **Unit Testing**: individual components are tested
- **Integration Testing**: the interaction between these components is tested
- **System Testing**: a complete system is tested against its specification
- **Acceptance Testing**: a user validates that the system meets their needs

Topics

- The activities required to **plan** software testing, and examines
- How software testing fits into different models of the software development **process**

Test Planning

- Typical contents:
  - Items to be tested
  - Tasks to be performed
  - Responsibilities
  - Schedules
  - Required resources
- Ref: IEEE Standard 29119

IEEE standard 291191

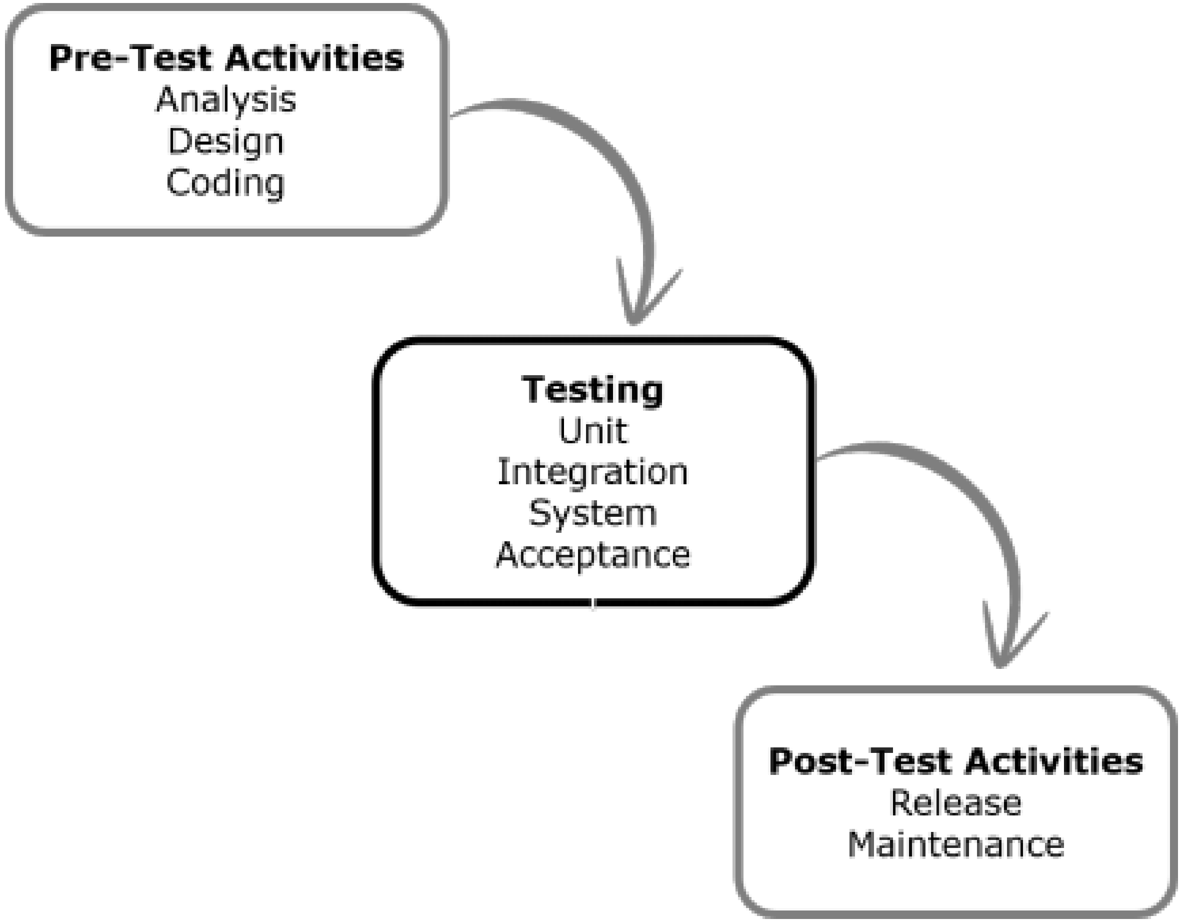
- Formal framework to prepare and execute a test plan
- Three levels of documentation
  1. **Organisational Test Documentation**
     - Defines organisational test policy and strategy
  2. **Test Management Documentation**
     - Defines pre-test and post-test management documents
     - Test Plans and Test Completion Reports
  3. **Dynamic Test Documentation (our focus)**
     - Prior to testing: defines the Test Environment and the Test Data
     - After testing: defines the Test Execution Documentation, including Incident Reports and Test Status Reports
- In reality, only very large/mission-critical projects use the full standard
- Most projects use tailored subset

The Software Lifecycle

- A structured plan for organizing the development of a software product
- Several models for such processes: describe tasks or activities
- Need for such planning: growth in size and complexity of software projects
- Unstructured approaches:
  - cost overruns, late deliveries and uncertain code quality
- Planned development:
  - Repeatable and predictable software development process
  - Automatically improve productivity and quality
- We’ll consider the standard lifecycle models from a testing viewpoint
  - Different models, different emphasis on testing

The Waterfall Model

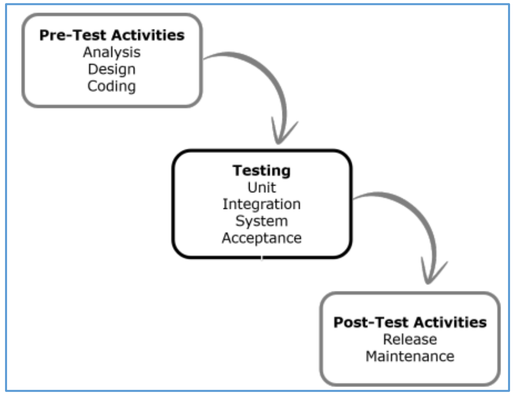
- Linear sequence
- Focus early to ensure requirements & design are correct, saving time and effort later
- No wasted effort from incorrect requirements of design
- Emphasis on documentation
- Simplified model centered on testing activity


Pre-Test Activities
Analysis
Design
Coding

Testing
Unit
Integration
System
Acceptance

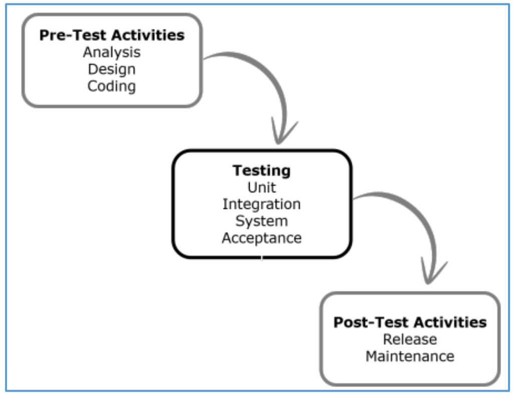
Post-Test Activities
Release
Maintenance

Activities



- From a testing viewpoint:
  - **Pre-Test** Activities consist of Analysis of the user requirements, followed by the Design of the system, and Coding
  - **Testing** consists of Unit Testing, Integration Testing, System Testing and Acceptance Testing of the system
  - **Post-Test Activities** consist of Release of the software/product, and the subsequent Maintenance after the software has been deployed
- Planning first, not responsive to changes
  - User requirements
  - Design enhancements apparent during coding

Waterfall and Testing

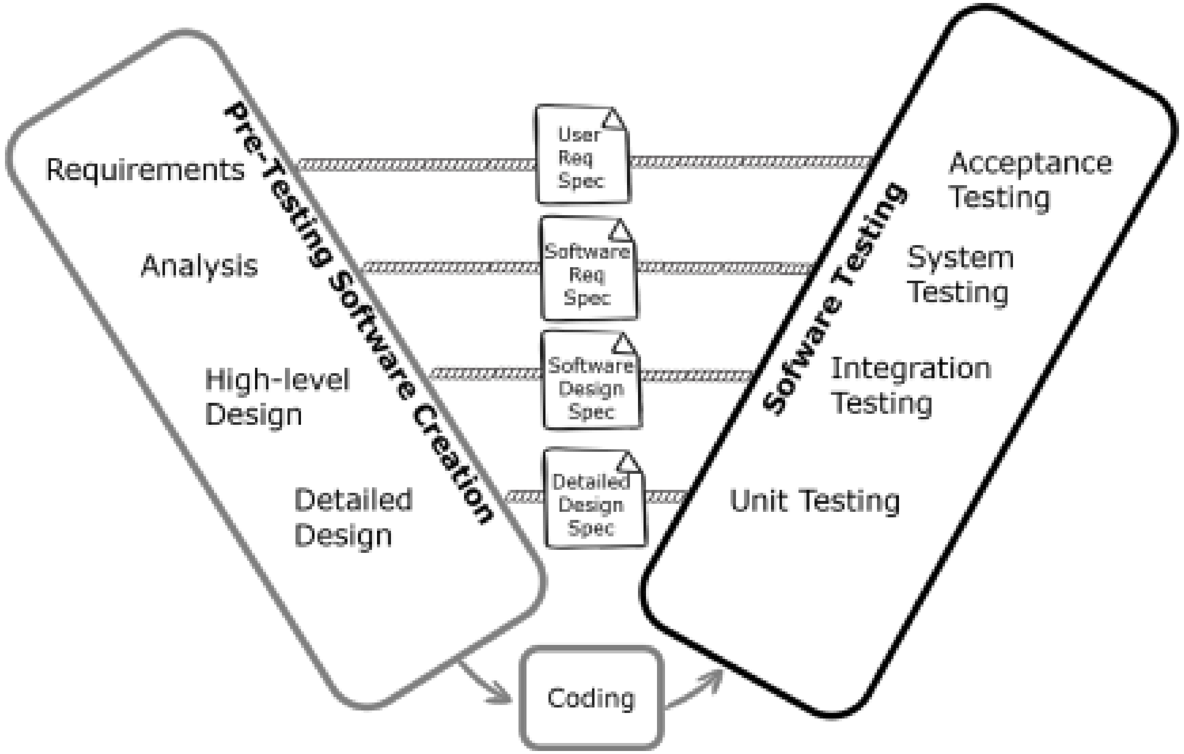


- All the tests are carried out once the software is completed
- Problems:

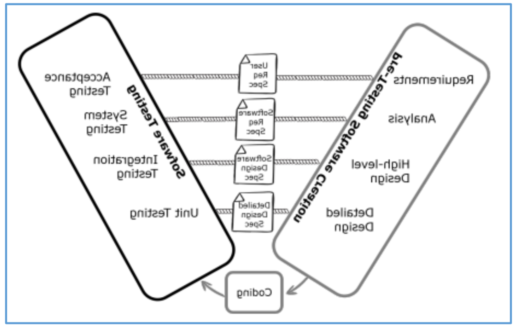
1. Budgetary or time pressure near end can result in insufficient or incomplete testing
    - Focus on testing program as a whole rather than systematically progressing from unit testing to application/system testing
2. If testing exposes design faults in the program, it is too late to do a redesign in this process, and the only option is to try and fix the problems in the code, which is followed by more testing
    - If some faults are difficult to trace, it could result in many iterations back and forth between fixing and testing
3. Lastly, customers may request changes once they have received the product, which may lead to a long maintenance phase

The V-Model

- V&V (Verification & Validation)
- Software development as a relationship between specification and associated testing activities
- Increase the focus on testing
- Activities have two outputs:
    - specification of the next activity
    - criteria for testing the activity has been correctly executed later in the process

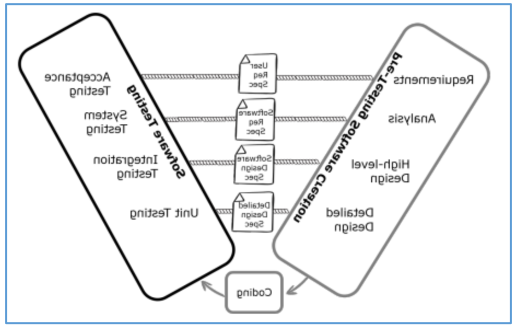


Matching Documents and Activities



- Requirements gathering produces User Requirements Specification
  - Input to Analysis
  - Basis for Acceptance Testing
- Analysis produces the Software Requirements Specification
  - Input to High-level Design
  - Basis for System Testing
- High-level Design produces Software Design Specification
  - Input to Detailed Design
  - Basis for Integration Testing
- Detailed Design produces the Detailed Design Specification
  - Used to write the code
  - Basis for Unit Testing

V-Model and Testing



- Simple and easy to manage due to the rigidity of the model
- Encourages verification and validation at all phases
- Each phase has specific deliverables and a review process
- Unlike the waterfall model, it gives equal weight to testing

Incremental and Agile Development

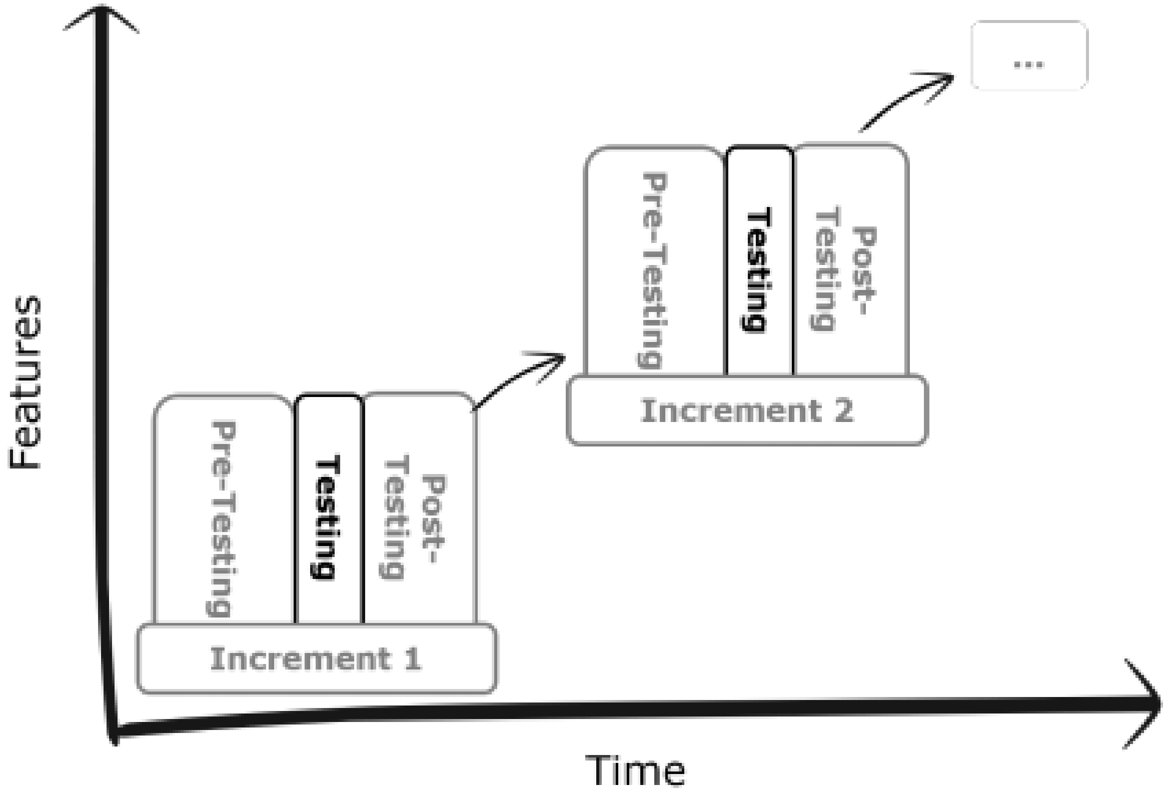
- In many projects impossible to arrive at stable, consistent user requirements at the start
- Waterfall & V Models too inflexible
- Incremental or Agile approach provide better results
- Agile Manifesto:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan

Testing Guidelines in Agile

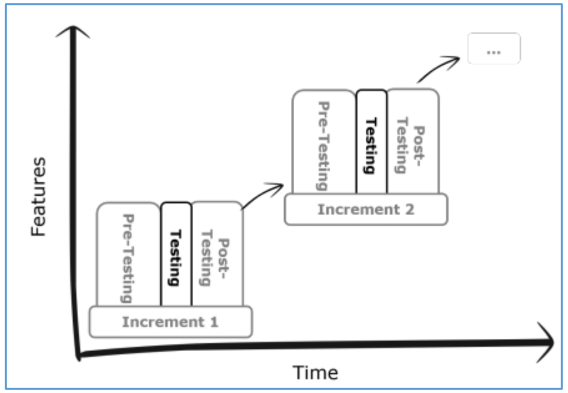
- Testers should participate in the requirement “negotiation" process that usually occurs between developers and customers
- This includes asking questions, identifying if requirements are untestable, and other issues
- Testers should immediately translate requirements into test cases serving as documentation for the upcoming iteration
- Testers and developers should collaborate in automating those test cases
- Testers should be informed immediately when requirements change so that they can modify their test cases

Incremental Development

- Begins with simple implementation of a part of the software system
- Each increment enhances product until the final version is reached
- From a testing viewpoint, each increment has three phases: PreTesting, Testing, and Post-Testing



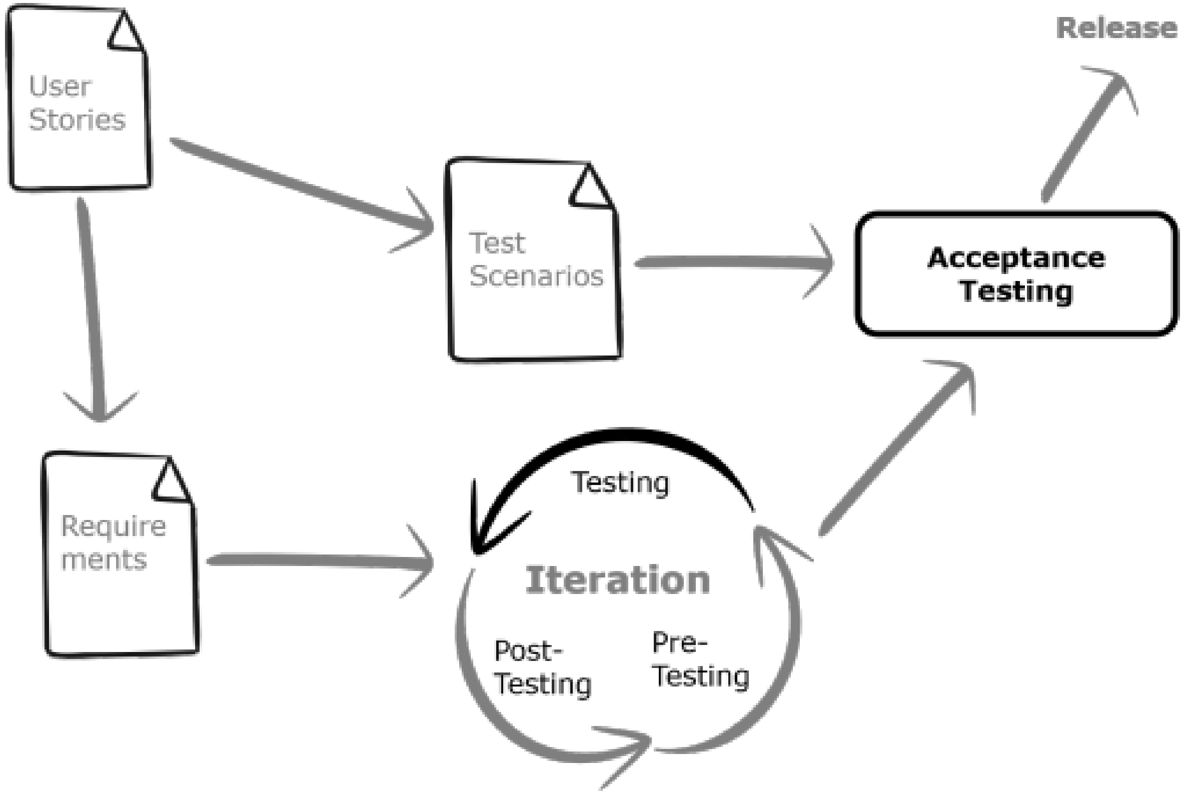
Incremental Dev. with Testing



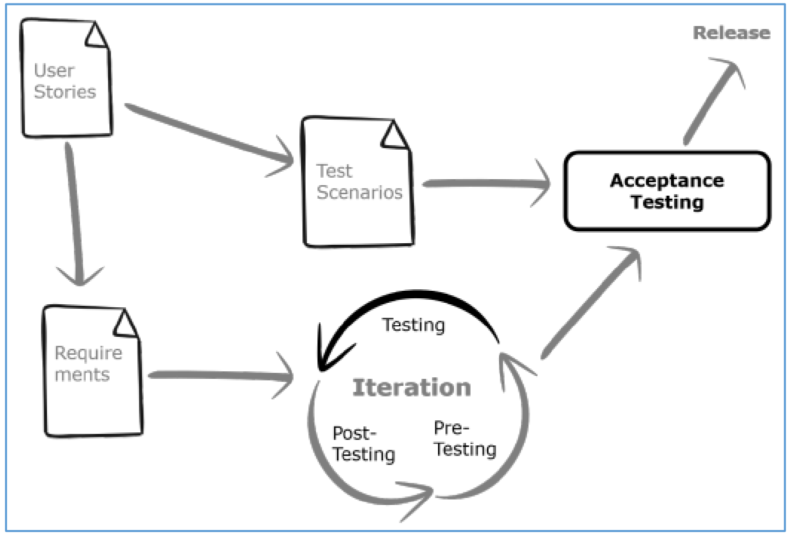
- Testing for each increment
  - Regression testing (to ensure the new increment has not broken any previously working software)
  - Testing of the new features added
- The progressive release of tested software increments means that interim versions of the software become available much earlier in the development process
- The quality of the final product is expected to be higher, due to increased testing and the opportunity for a customer to view early releases of the product

XP

- Continuously communicate with customers and fellow programmers
- Keep design simple and clean
- Get feedback from early software testing
- Deliver a system to customers as early as possible and implement changes as suggested, so developers respond with courage to changing requirements and technology
- Unit tests developed before code

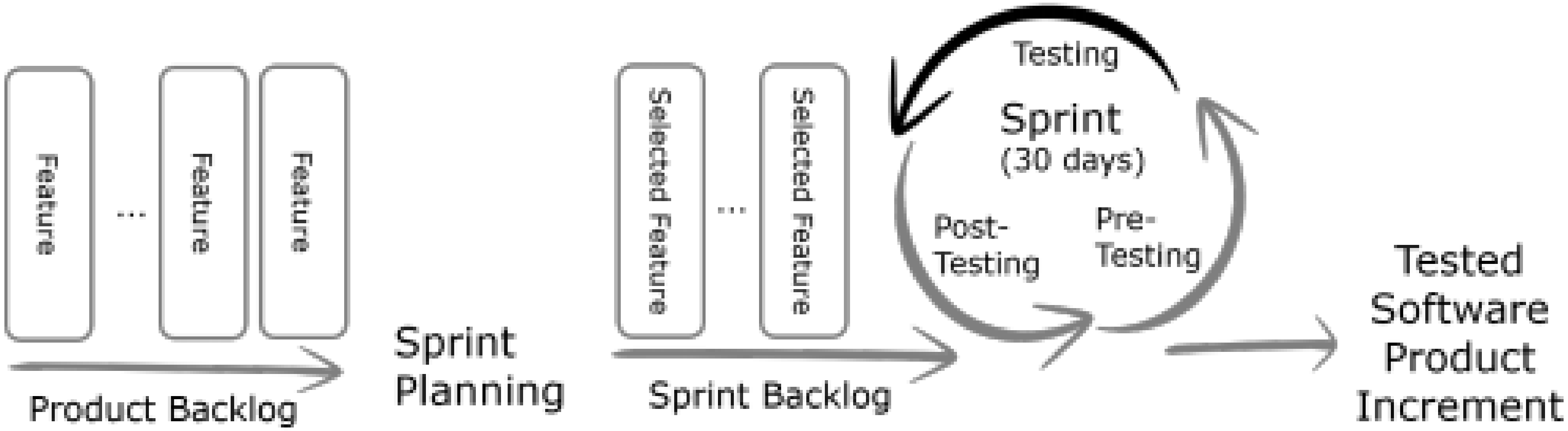


Testing and XP



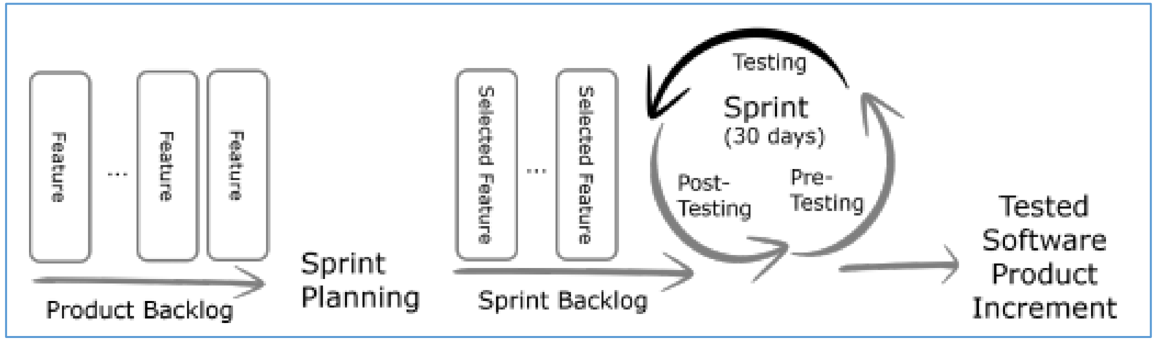
- User Stories are written by the customers
  - Providing the user requirements specifications
  - Used to plan releases
- Unit tests are implemented before the code is written
- Helps developer to think deeply about what they are doing
- Requirements are defined fully by the tests
- Code, influenced by the existing unit tests, is expected to be clearer to understand and easier to test
- New tests are created to ensure faults have been fixed correctly

Scrum



- Scrum teams work in iterations that are called Sprints
- These can last a little longer than XP iterations
- No changes to be introduced during the Sprints
- In Scrum there is more flexibility for additional stakeholders to influence the ordering of implementing features
- In Scrum it is up to the team to organize themselves and adopt the practices they feel work best for themselves

Scrum and Testing



- Product Backlog is prioritized list of required product features
- Sprint backlog: 30-day extract from product backlog
- Testing approach depends on the team – not strictly prescribed as in XP
  - Usually consist of unit testing, regression testing, etc.
- Selected features used to create acceptance tests
  - Verify tested software product increment for each Sprint
- This approach has proven successful, as the testing team collaborates closely with the developers from the start of the project

Process-Related Quality Standards & Models

- Testing as part of the Quality Assurance (QA) process
- key model is the ISO 9000/25000 series of standards
- The ISO 9000/25000 series of standards are a significant concern for companies developing software and systems for public tender
- They provide state bodies with an assurance that the software is being developed in a professional manner