Testing in the Software Process

Spring, 2023

Yi Xiang xiangyi@scut.edu.cn

Contents

1. Waterfall Model

- All the planning is done at the beginning, and once created it is not to be changed.
- There is no overlap between any of the subsequent phases.
- Often anyone’s first chance to “see” the program is at the very end once the testing is complete.

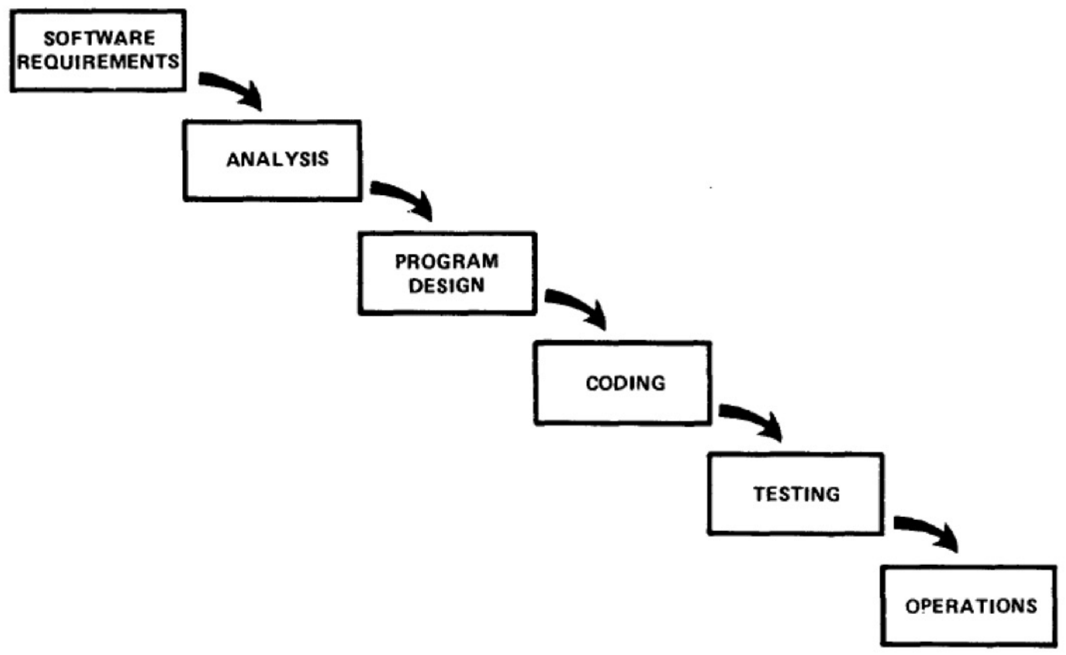
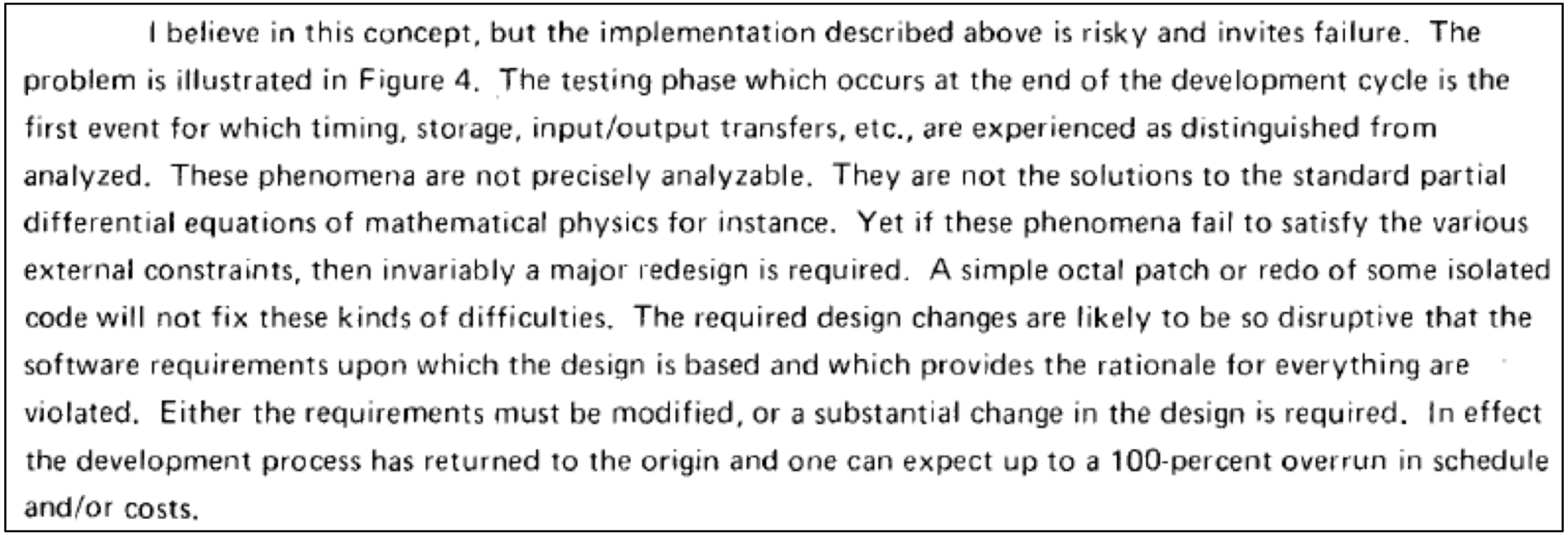


Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

Royce, W. Managing the Development of Large Software Systems. IEEE WESCON, 1970.

Waterfall Model – Strength& Weakness

The caption immediately below that figure, in the original paper, is:



I believe in this concept, but the implementation described above is risky and invites failure. The problem is illustrated in Figure 4. The testing phase which occurs at the end of the development cycle is the first event for which timing, storage, input/output transfers, etc., are experienced as distinguished from analyzed. These phenomena are not precisely analyzable. They are not the solutions to the standard partial differential equations of mathematical physics for instance. Yet if these phenomena fail to satisfy the various external constraints, then invariably a major redesign is required. A simple octal patch or redo of some isolated code will not fix these kinds of difficulties. The required design changes are likely to be so disruptive that the software requirements upon which the design is based and which provides the rationale for everything are violated. Either the requirements must be modified, or a substantial change in the design is required. In effect the development process has returned to the origin and one can expect up to a 100-percent overrun in schedule and/or costs.

Key Sentence

I believe in this concept, but the implementation described above is risky and invites failure.

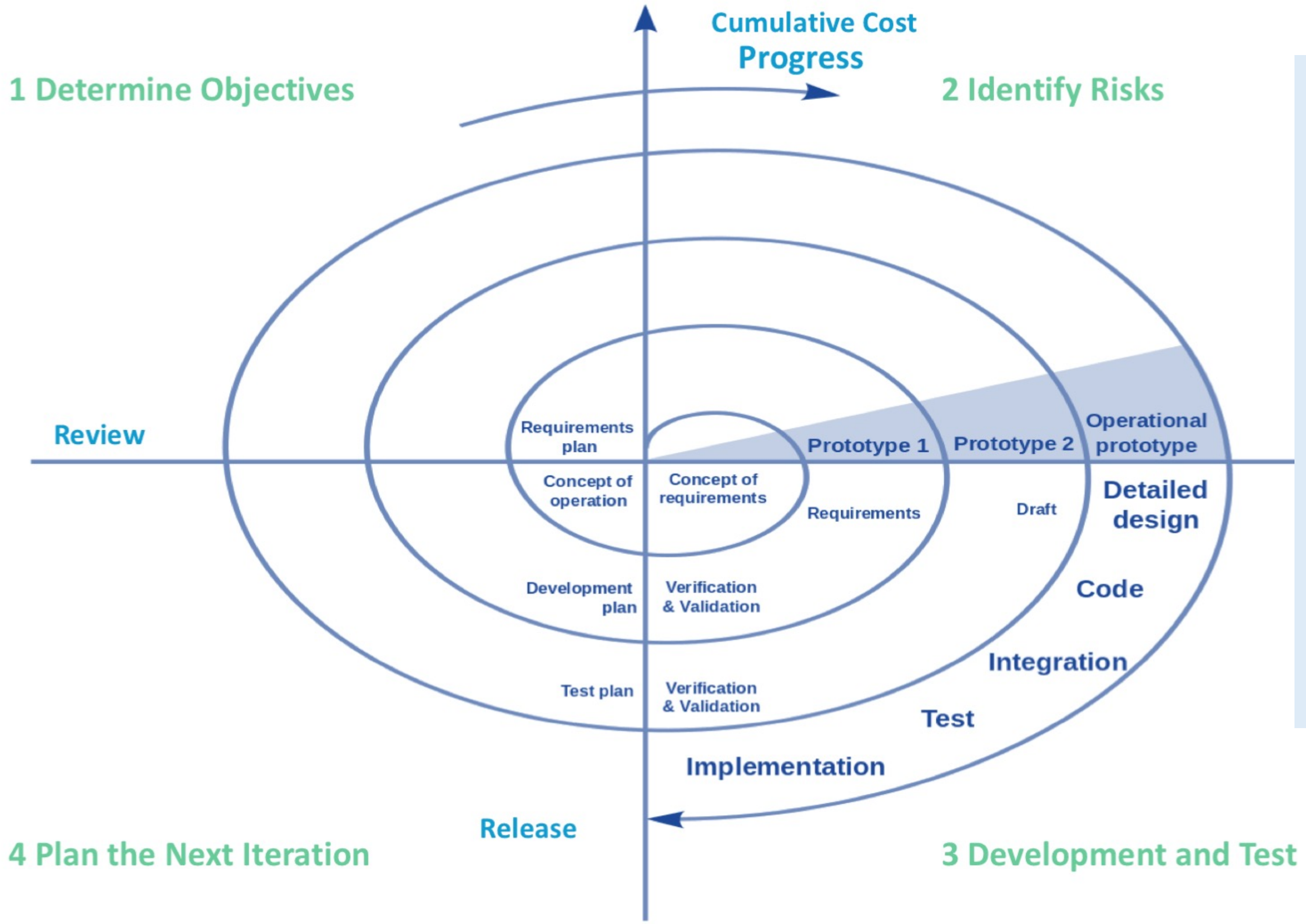
Waterfall Model – Strength& Weakness

- Strength:
  1) If time is spent early on making sure that the requirements and design are absolutely correct, then this will save much time and effort later.
  2) There is an emphasis on documentation which keeps all knowledge in a central repository and can be referenced easily by new members joining the team.

Waterfall Model – Strength& Weakness

- Weakness:
  1) Few visible signs of progress until the end of the project
  2) It is not flexible to changes
  3) Time-consuming to produce all the documentation
  4) Tests are only carried out at the end – this could mean a compromise if time or budgetary constraints exist
  5) Having to test the program as a whole could result in incomplete testing
  6) If testing does identify a fault that suggests a redesign it may be ignored because of the trouble involved
  7) If the customer is unhappy it may incur a long maintenance phase resolving their issues

2. Spiral Model

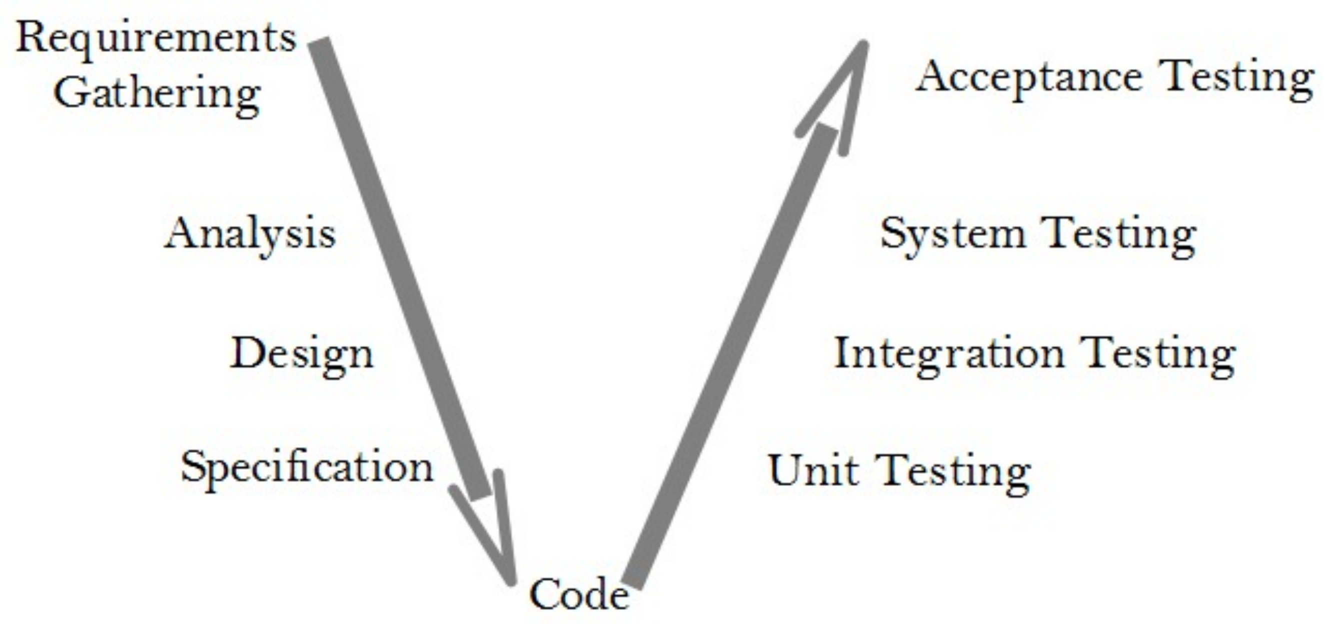


- Risk-driven development process
- Combination of waterfall model and Rapid Prototype Iteration model
- Begins with a design goal and ends with the client reviewing the progress.

Sprial Model – Strength& Weakness

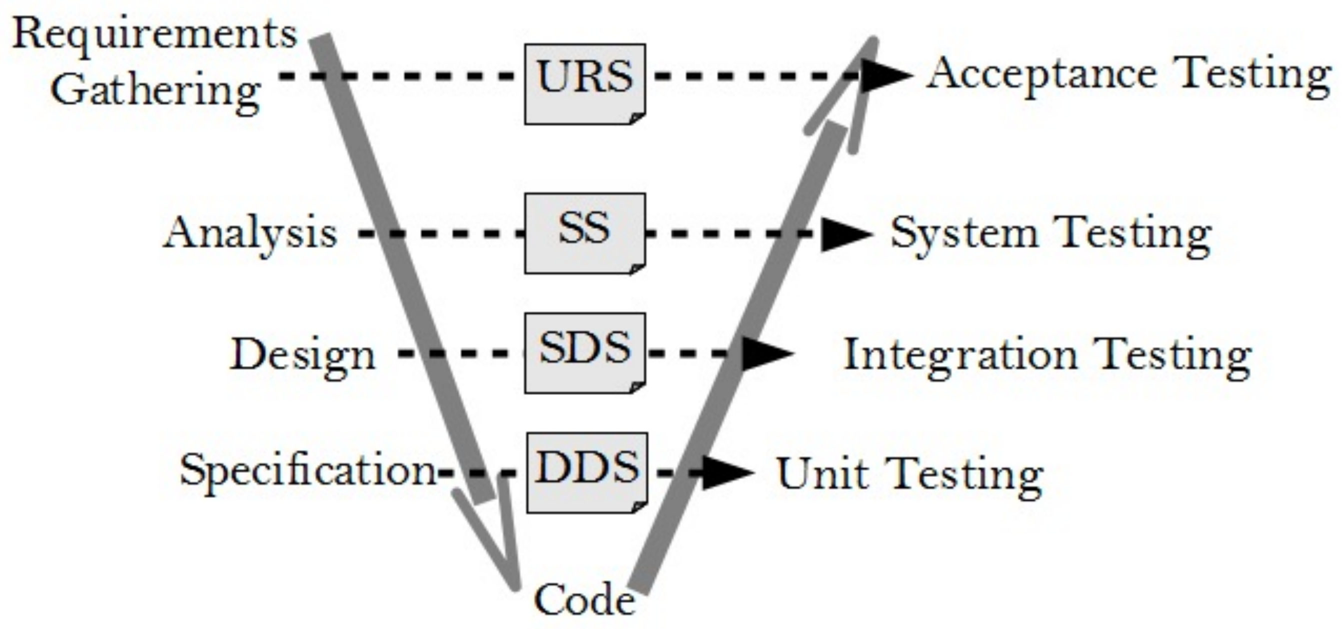
Strength	Weakness
Additional functionality or changes can be done at a later stage	Risk of not meeting the schedule or budget
Cost estimation becomes easy as the prototype building is done in small fragments	Spiral development works best for large projects only also demands risk assessment expertise
Continuous or repeated development helps in risk management	For its smooth operation spiral model protocol needs to be followed strictly
Development is fast and features are added in a systematic way in Spiral development	Documentation is more as it has intermediate phases
There is always a space for customer feedback	Spiral software development is not advisable for smaller project, it might cost them a lot

3. V Model



- Extension of the Waterfall model
  - emphasizes Verification & Validation by marking the relationships between each phase of the life cycle and testing activities
- Once the code implementation is finished the testing begins.
- Starts with unit testing, and moves up one test level at a time until the acceptance testing phase is completed

V Model

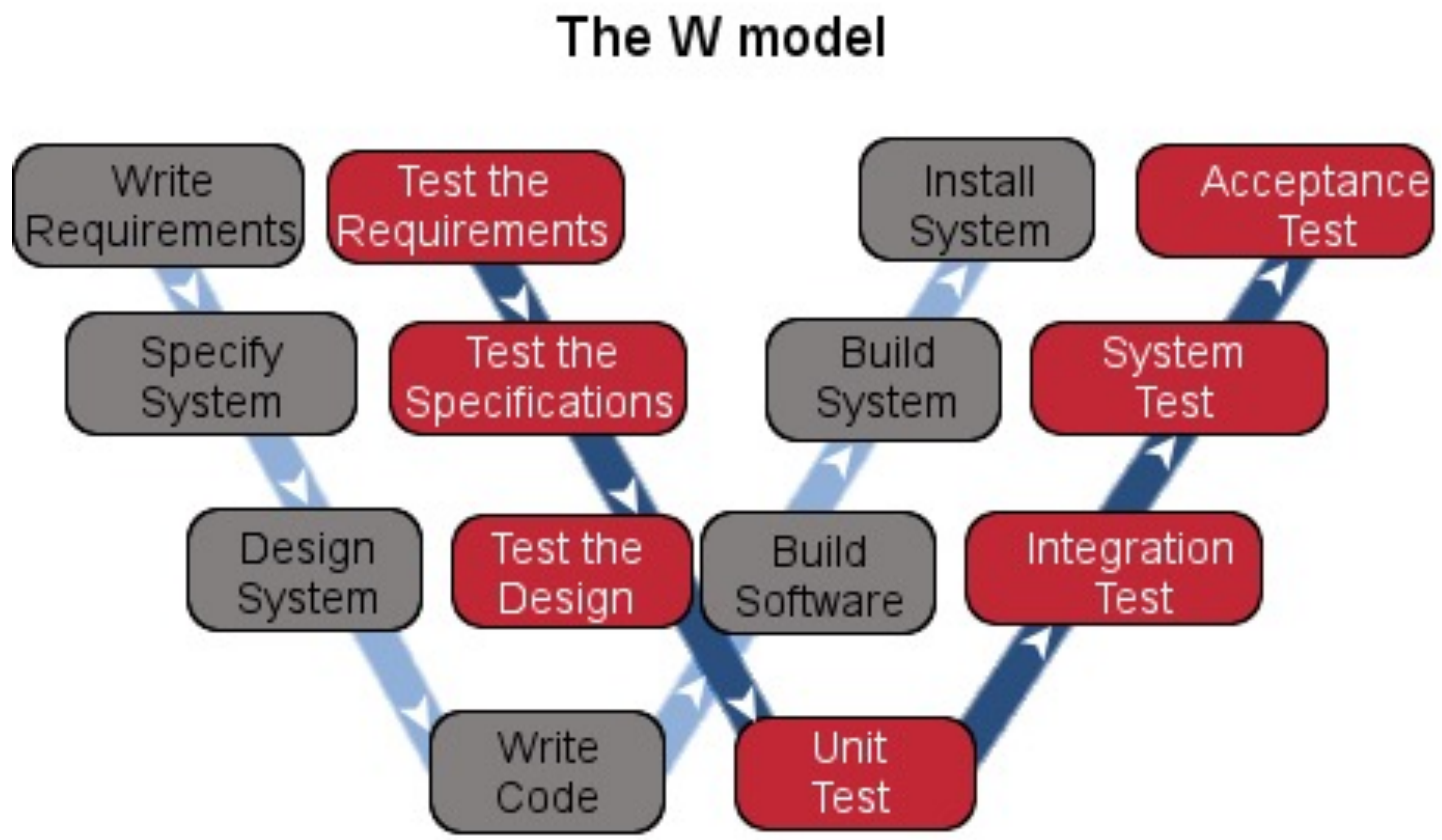


- Each document produced is associated with pairs of phases in the model.
  - – (a) the User Requirements Specification. URS
  - – (b) the System Requirements Specification, SRS
  - – (c) the System Design Specifications, SDS
  - – (d) Detailed Design Specifications, DDS

V Model – Strength& Weakness

Strength	Weakness
It is simple and easy to manage due to the rigidity of the model	Like the Waterfall model , there is no working software produced until late during the life cycle
It encourages Verification and Validation at all phases	It is unsuitable where the requirements are at a moderate to high risk of changing.
Each phase has specific deliverables and a review process.	It has been suggested too that it is a poor model for long, complex and object-oriented projects
It gives equal weight to testing alongside development rather than treating it as an afterthought at the end.	

4. W Model



- Extension of V Model/Both V
- Testing is not after the code implementation .
- Parallel to the development process, the test process is carried out.
- Co-operation between development and testing
- Testing is more than just construction, execution and evaluation of test cases.

5. Agile Model

- Agile methods share with other *incremental development* methods an emphasis on building releasable software in short time periods.

- However, Agile development differs from the other development models in that its time periods are measured in weeks rather than months and work is performed in a highly collaborative manner

Agile Model

For effective testing:

- – When the developers "negotiate" the *requirements* for the upcoming iteration with the customers, the testers must be *full participants* in those conversations.
- – The testers immediately translate the requirements that are agreed upon in those conversations into test cases.
- – When requirements change, testers are immediately involved because everyone knows that the test cases must be changed accordingly.

Agile Model

For effective testing:

- – When the developers “negotiate” the *requirements* for the upcoming iteration with the customers, the testers must be *full participants* in those conversations.
- – The testers immediately translate the requirements that are agreed upon in those conversations into test cases.
- – When requirements change, testers are immediately involved because everyone knows that the test cases must be changed accordingly.

Incremental Development

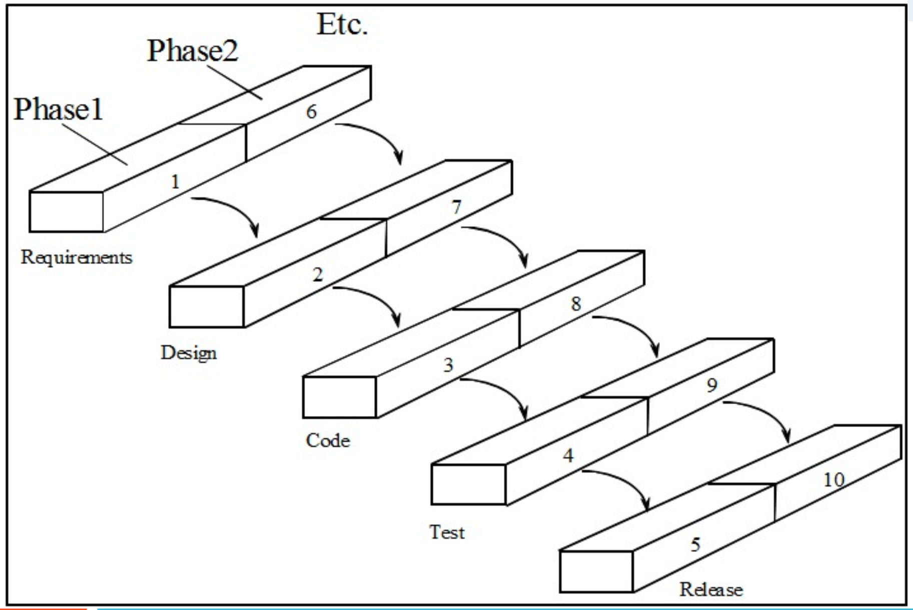
- The incremental model begins with a simple implementation of a part of the software system. With each increment the product evolves with enhancements being added every time until the final version is reached.

- Testing is an important part of the incremental model and is carried out at the end of each iteration. This means that testing begins earlier in the development process and that there is more of it overall.

- Much of the testing is of the form of regression testing, and much re-use can be made of test cases and test data from earlier increments.

Incremental Development – Strength and Weakness

Strength	Weakness
The product is written and tested in smaller pieces, reducing risk and allowing for change to be included easily	It can be difficult to manage because of the lack of documentation in comparison to other models
The customer or users is/are involved from the beginning which means the system is more likely to meet their requirements and they themselves are more committed to the system	The continual change to the software can make it difficult to maintain as it grows in size.



Extreme Programming

- Extreme Programming (XP) is a subset of the philosophy of Agile software development.
- It emphasizes code reviews, continuous integration and automated testing, and very short iterations.
- It favours ongoing design refinement (or refactoring), in place of a large initial design phase, keeping the current implementation as simple as possible.
- It favours real-time communication, preferably face-to-face, over writing documents, and working software is seen as the primary measure of progress.
- The methodology also emphasizes team work. Managers, customers, and developers are all part of a team dedicated to delivering quality software.
- *Programmers* are responsible for testing their own work; *testers* are focused on helping the customer select and write functional tests, and on running these tests regularly.

Extreme Programming -Value

Communication:

- XP programmers communicate with their customers and fellow programmers

Simplicity

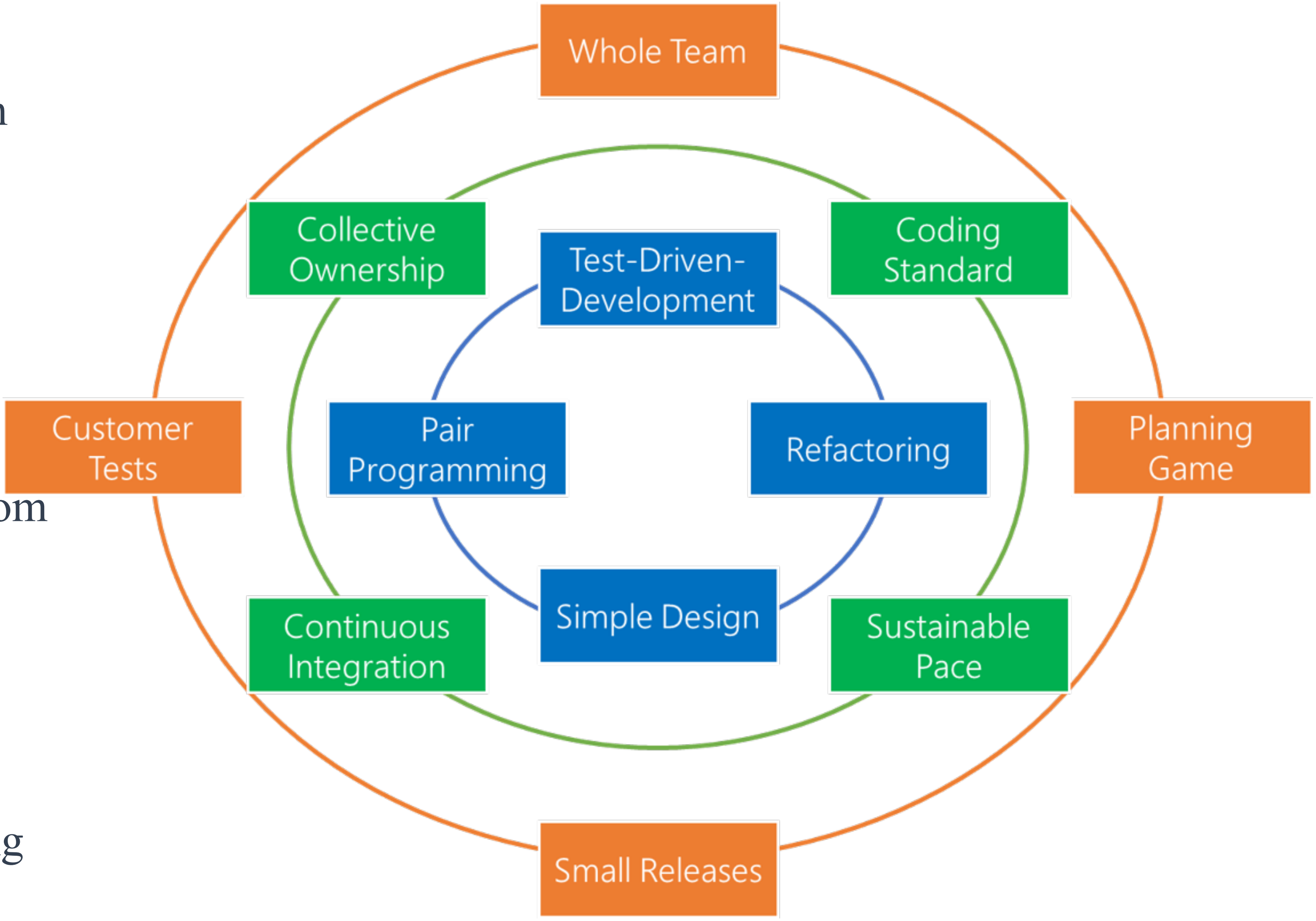
- they keep their design simple and clean

Feedback:

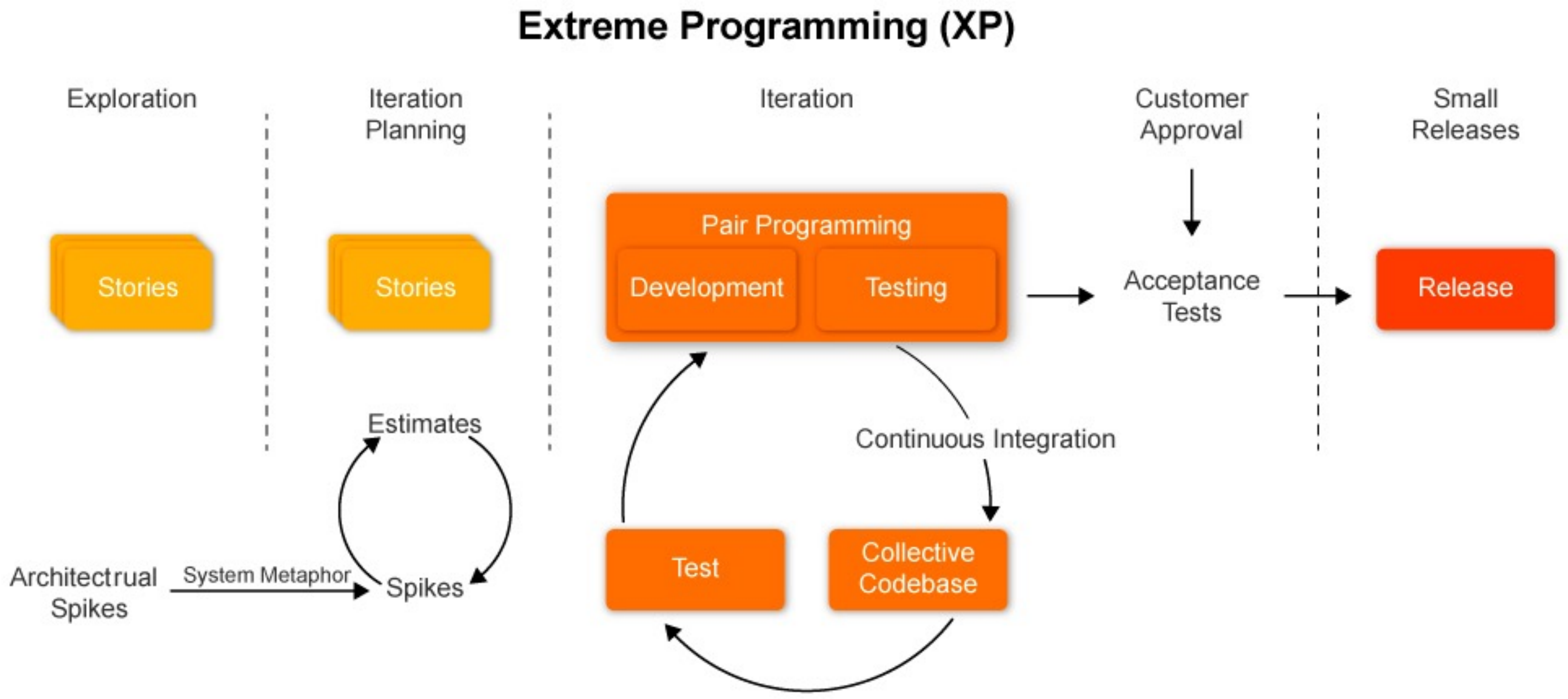
- Get feedback by software testing from the start

Courage:

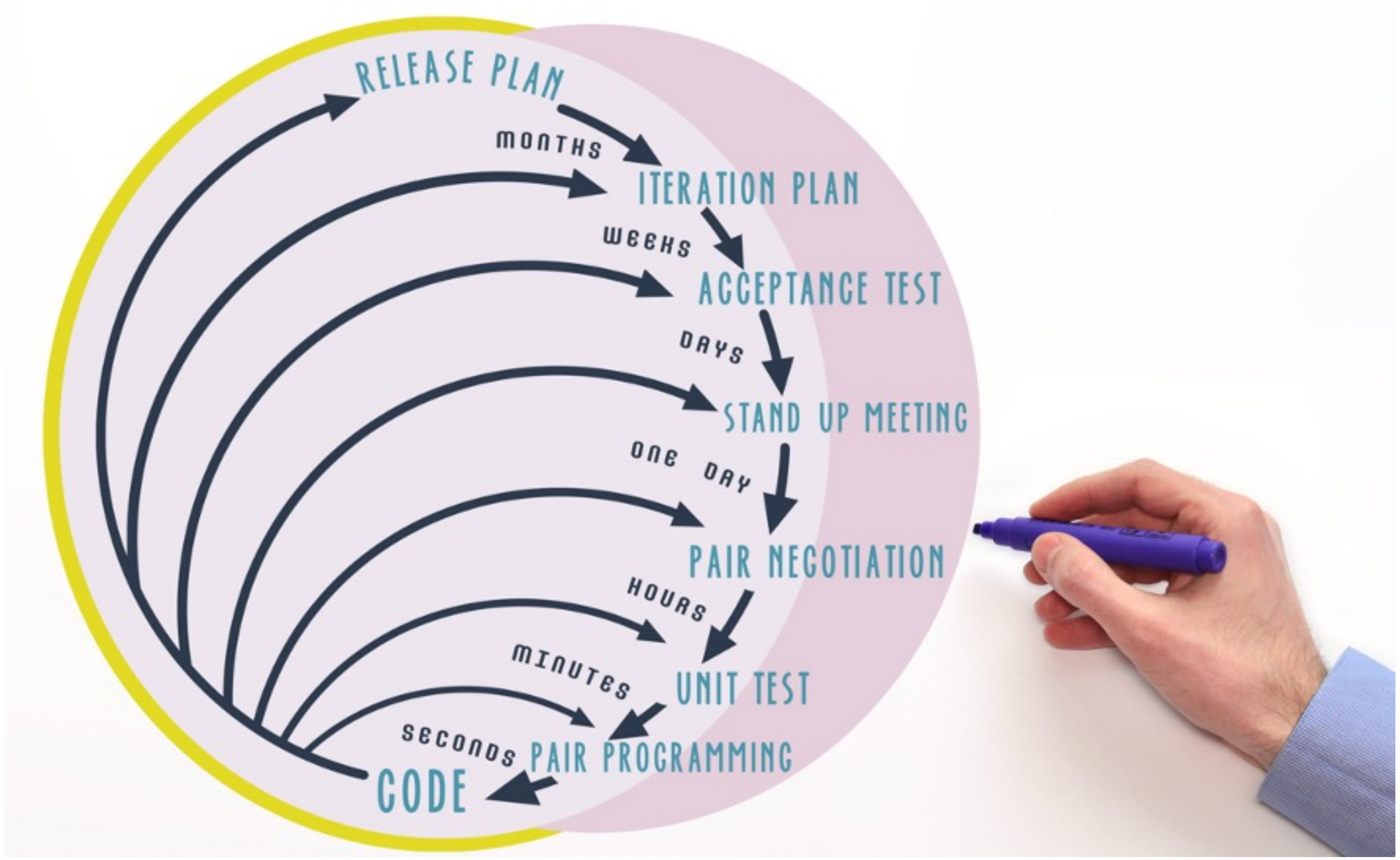
- Deliver the system to customers as early as possible
- Implement changes as suggested, responding with courage to changing requirements



Extreme Programming - Process

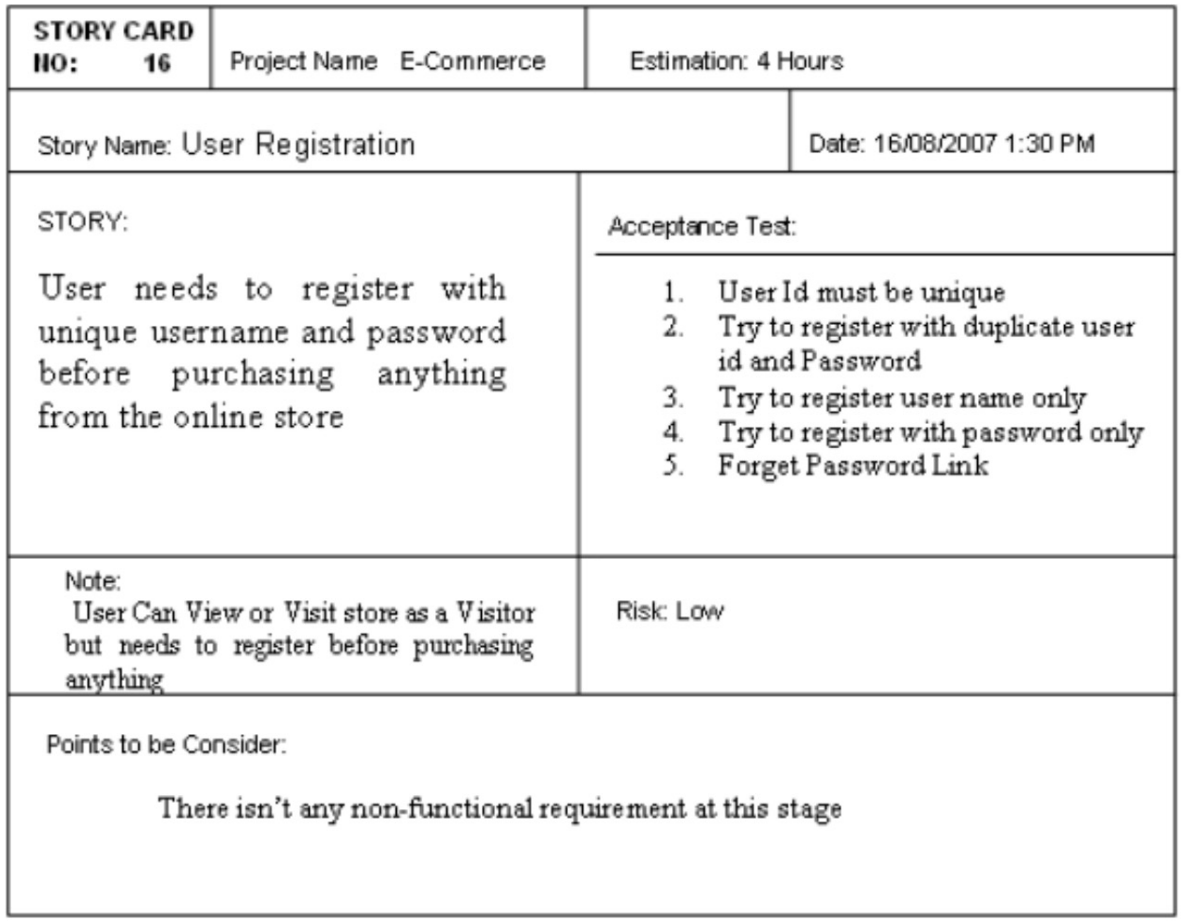


Extreme Programming - Process



User Stories and Story Card

- A User Story is one or more sentences in everyday language that captures one aspect of what the software system will need to do.
- These are usually written down on paper cards termed as Story cards
- The Story Cards are ordered to reflect the development of the system
- How should this be done? Prioritize the most difficult or components first? Or in the sequence of user actions?



STORY CARD NO: 16 | Project Name E-Commerce | Estimation: 4 Hours

Story Name: User Registration | Date: 16/08/2007 1:30 PM

STORY:

User needs to register with unique username and password before purchasing anything from the online store

Acceptance Test:

1. User Id must be unique
2. Try to register with duplicate user id and Password
3. Try to register user name only
4. Try to register with password only
5. Forget Password Link

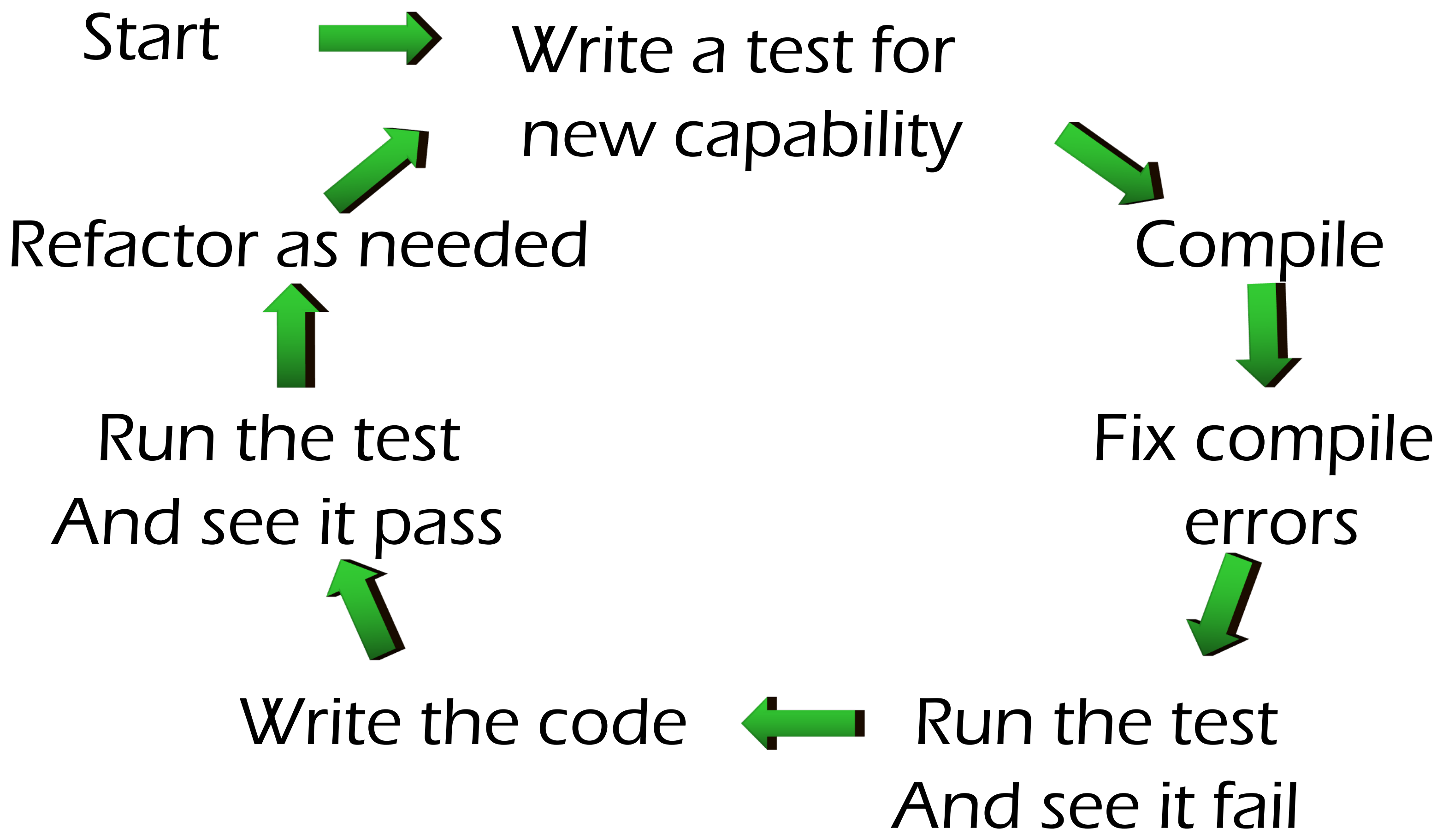
Note:
User Can View or Visit store as a Visitor but needs to register before purchasing anything

Risk: Low

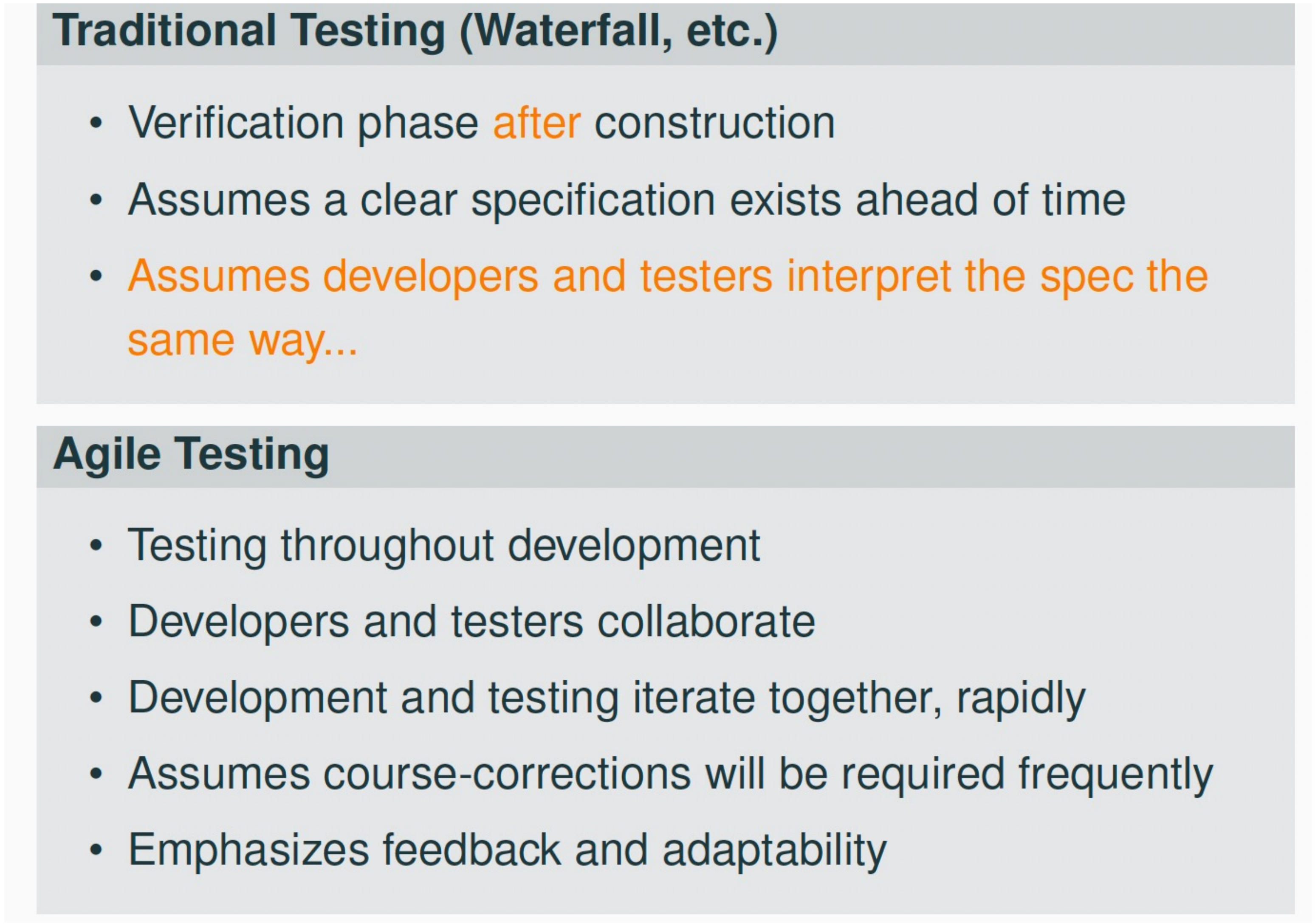
Points to be Consider:

There isn't any non-functional requirement at this stage

TDD - Test-Driven Development


Start
Write a test for new capability
Compile
Fix compile errors
Run the test And see it fail
Write the code
Run the test And see it pass
Refactor as needed

Two styles of testing



Traditional Testing (Waterfall, etc.)

- Verification phase after construction
- Assumes a clear specification exists ahead of time
- Assumes developers and testers interpret the spec the same way...

Agile Testing

- Testing throughout development
- Developers and testers collaborate
- Development and testing iterate together, rapidly
- Assumes course-corrections will be required frequently
- Emphasizes feedback and adaptability