What are Software development methodologies

- Development Methodologies are the various processes which can be selected for the development of the project depending on the project's goals
- The selection of a specific methodology has a very high impact on the testing that is carried out
- It will define the what, where, and when of our testing
- It will influence various types of testing
- Selection of a specific methodology will also help us determine which testing techniques to use
- Development and Testing are carried out in various ways depending on the methodology chosen

Development methodologies

- Waterfall Methodology
- V-Model
- Incremental Methodology
- RAD Methodology
- Agile Methodology
- Iterative Methodology
- Spiral Methodology
- Prototype Methodology

Waterfall Methodology

- Each phase of the SDLC must be completed before the next phase can begin

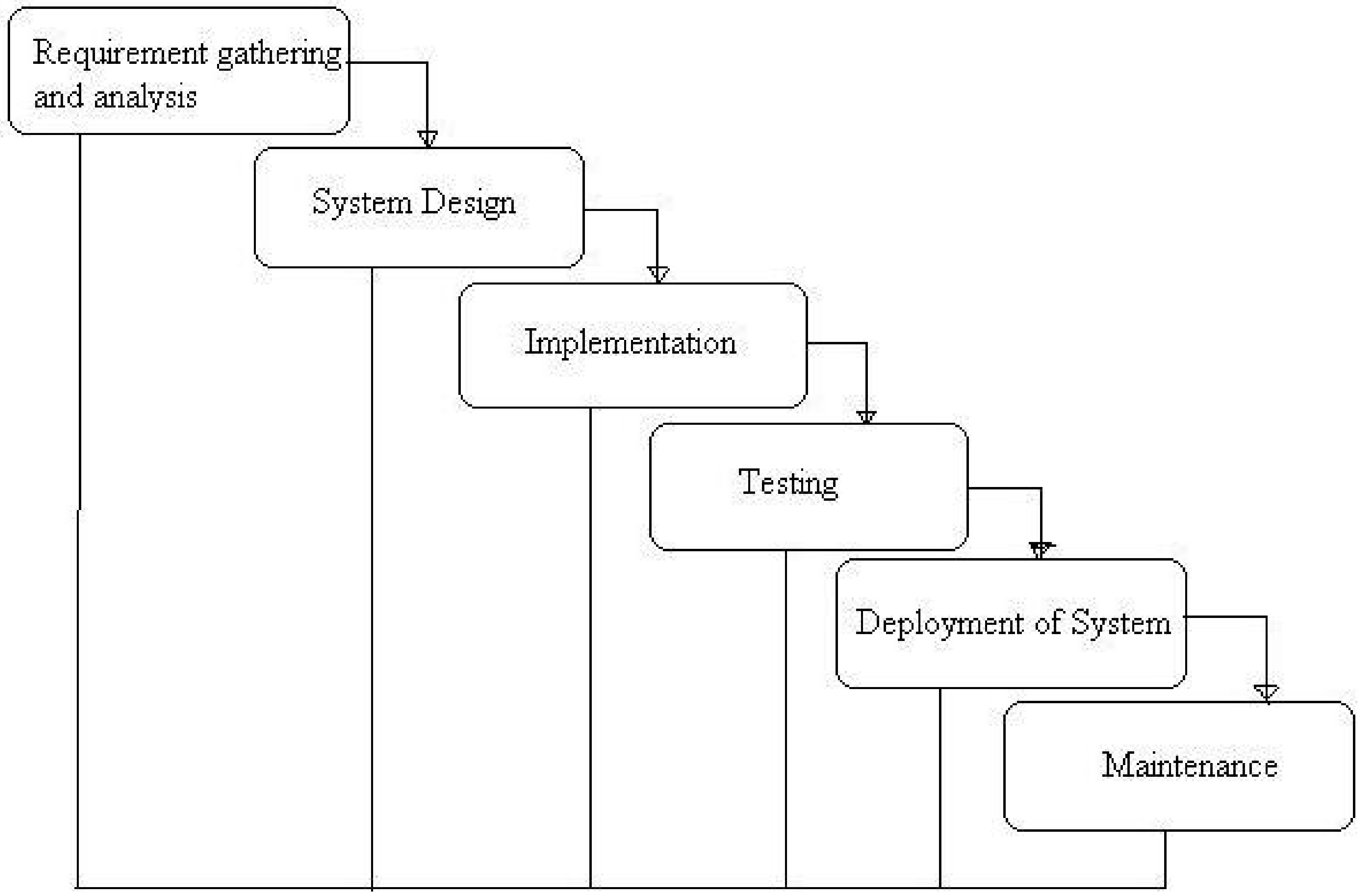
- Waterfall is good for these types of project:
  - Small projects
  - Projects where all requirements are known and are not likely to change

- At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project

- SDLC phases do not overlap each other

- **Testing starts after development has been completed**

General Overview of "Waterfall Model"



Advantages of waterfall

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model phases are processed and completed one at a time. Phases do not overlap.
- Waterfall model works well for smaller projects where requirements are very well understood

Disadvantages of waterfall

- Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.

When to use waterfall

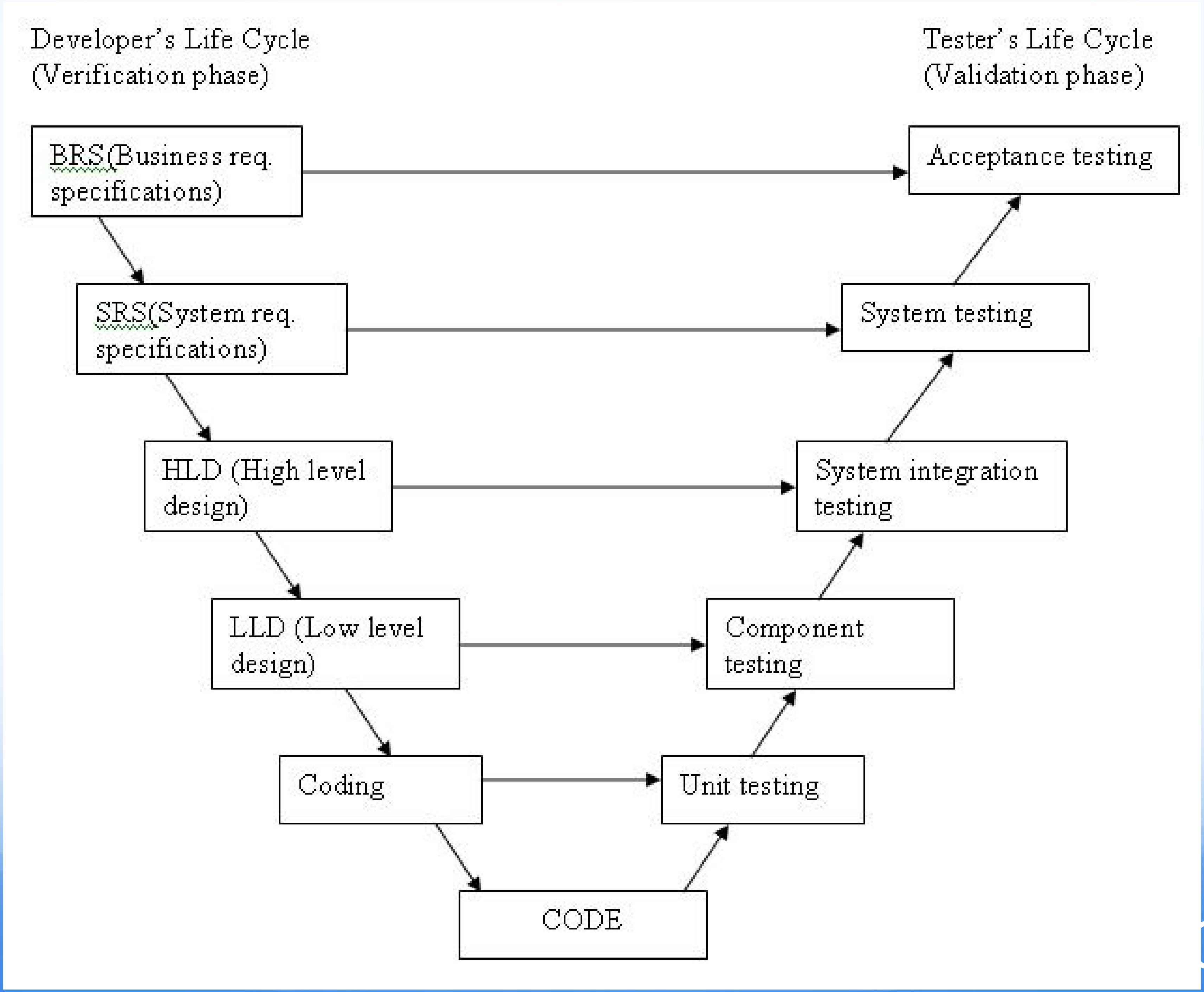
- This model is used only when the requirements are very well known, clear and fixed.
- Product definition is stable (not changing)
- Technology is understood
- There are no ambiguous requirements
- Ample resources with required expertise are available freely
- The project is short

More on waterfall

- Customer interaction is minimal during the development of the product
- The customer is shown the project only once it is ready to be demonstrated to the end users
- Once the product is developed, if there are any failures then the cost of fixing such issues are very high
  - This is because we need to update EVERYTHING beginning with the requirements

V-Model

- Means Verification and Validation Model
- The V-Shaped lifecycle is a sequential path of execution
- Each phase must be completed before the next phase begins
- Testing of the product is **planned** in parallel with a corresponding phase of development


Developer's Life Cycle
(Verification phase)
Tester's Life Cycle
(Validation phase)
BRS(Business req. specifications)
Acceptance testing
SRS(System req. specifications)
System testing
HLD (High level design)
System integration testing
LLD (Low level design)
Component testing
Coding
Unit testing
CODE

V-model phases

- Requirements (BRD & SRS) begin the V-model lifecycle (similar to Waterfall)
- HOWEVER a System Test Plan is generated PRIOR to Development
  - This test plan focuses on meeting the functionality specified in the requirements
- High-Level Design Phase – focuses on system architecture and design
  - Provides an overview of the solution, platform, system, product, and processes
  - An Integration Test Plan is generated in this phase in order to test how the different pieces of the application will work once put together
- Low-Level Design Phase – Actual software components are designed
  - Defines the actual logic for each component of the system
  - Component Tests are created in this phase
- Implementation Phase – (AKA Coding/Development Phase) All development or coding takes place in this phase
  - Once Implementation (Coding) is completed, the path of execution continues up the right side of the V-Model where the test plans created earlier are now put to us

Advantages of v-model

- Simple and easy to use.
- Testing activities like planning and test designing happen well before coding.
    - This saves a lot of time.
    - Hence higher chance of success than the waterfall model.
- Proactive defect tracking – that is defects are found at an early stage
- Avoids the downward flow of the defects
    - With Waterfall, a defect (failure) might be present in a requirement, but won't be detected until later in the testing phase
- Works well for small projects where requirements are easily understood.

Disadvantages of v-model

- Very rigid and least flexible of all methodologies
- Software is developed during the implementation phase, so no early prototypes of the software are produced.
- If any changes happen midway through the project, then the test documents along with requirement documents have to be updated

When to use v-model

- V-Model should be used for small to medium sized projects where requirements are clearly defined and fixed.
- The V-Model should be chosen when ample technical resources are available with needed technical expertise.

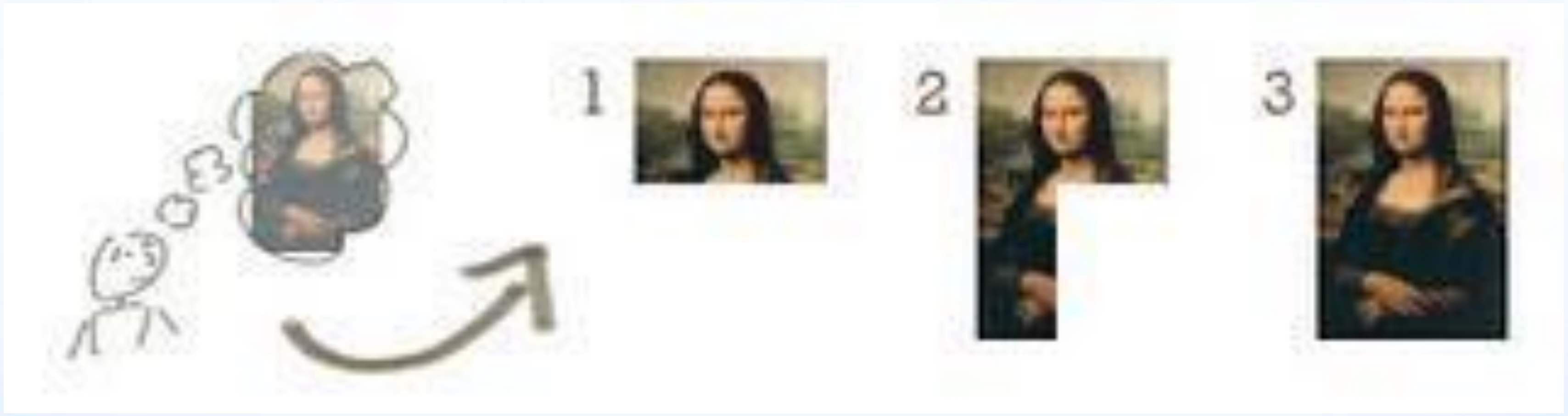
More on v-model

- Customer must have high confidence in the development team in order for V-Model to work
- Since no prototypes are produced, there is a high risk involved in meeting customer expectations

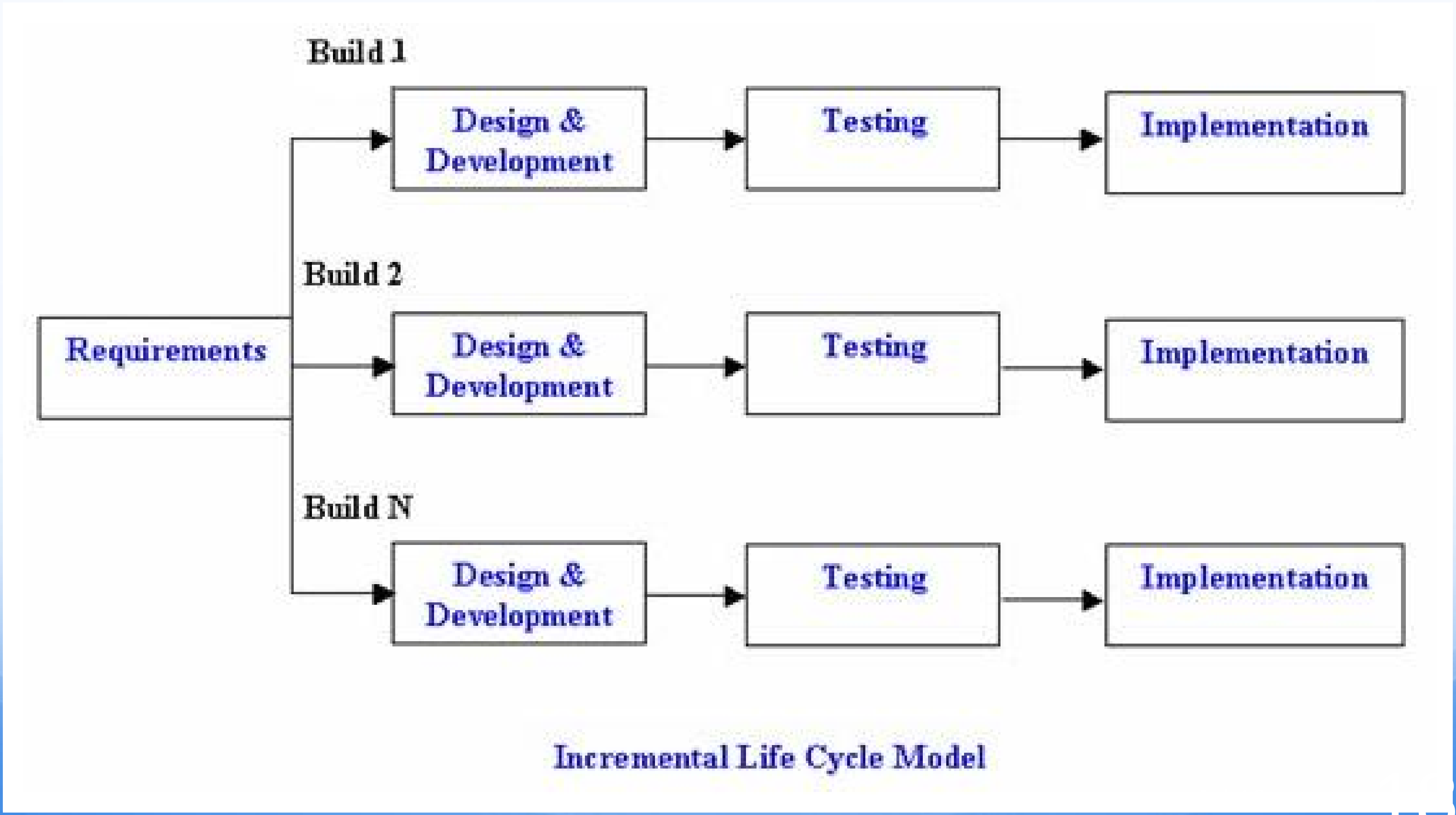
Incremental Methodology

- All of the requirements are split up into several builds (versions of the application)
- Multiple development cycles take place
  - ”Multi-Waterfall” cycle
- Cycles are divided up into smaller, more easily managed modules
- Each module passes through the SDLC phases
  - Requirements Analysis ➔ Design ➔ Development ➔ Testing ➔ Production
- A working version of the software is produced during the 1st module
- Each subsequent release of a module adds functionality to the entire application
- Process continues until the application is completed

Example of Incremental Methodology



Realistic visualization of incremental model



Incremental Life Cycle Model

Advantages of incremental model

- Generates working software quickly and early during the software life cycle.
- This model is more flexible – less costly to change scope and requirements.
- It is easier to test and debug during a smaller iteration.
- In this model customer can provide feedback with each new build that is released
- Lowers initial delivery cost.
- Easier to manage risk because risky pieces are identified and handled during each iteration.

Disadvantages of incremental model

- Needs good planning and design.
- Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
- Total cost is higher than waterfall.

When to use incremental

- This model can be used when the requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some details can evolve with time.
- There is a need to get a product to the market early.
- A new technology is being used (indicates increased risk)
- Resources with needed skill set are not available
- There are some high risk features and goals.

Rapid application development

- **R**apid **A**pplication **D**evelopment (RAD)
- A type of Incremental Model
- The components or functions are developed in parallel as if they were mini projects
- Each development project is time-boxed, delivered, and then assembled into a working prototype
- This can quickly give the customer something to see and use and to provide feedback on

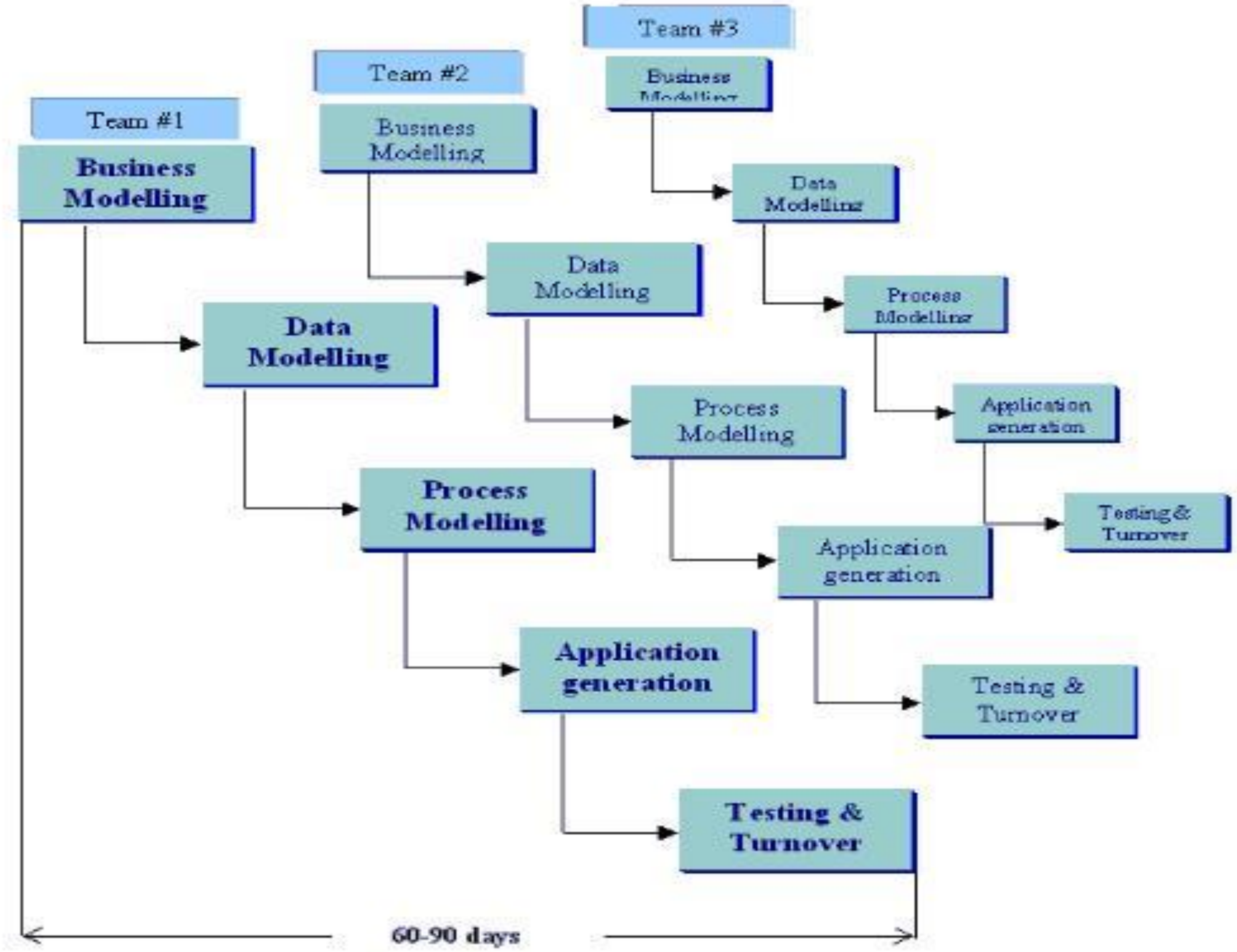


Figure 1.5 – RAD Model

Phases of rad

- **Business modeling:** The information flow is identified between various business functions.
- **Data modeling:** Information gathered from business modeling is used to define data objects that are needed for the business.
- **Process modeling:** Data objects defined in data modeling are converted to achieve the business information flow to achieve some specific business objective.
  - Descriptions are identified and created
- **Application generation:** Automated tools are used to convert process models into code and the actual system
- **Testing and turnover:** Test new components and all the interfaces.

Advantages of rad

- Reduced development time.
- Increases reusability of components
- Quick initial reviews occur
- Encourages customer feedback
- Integration from very beginning solves a lot of **integration issues**.

Disadvantages of rad

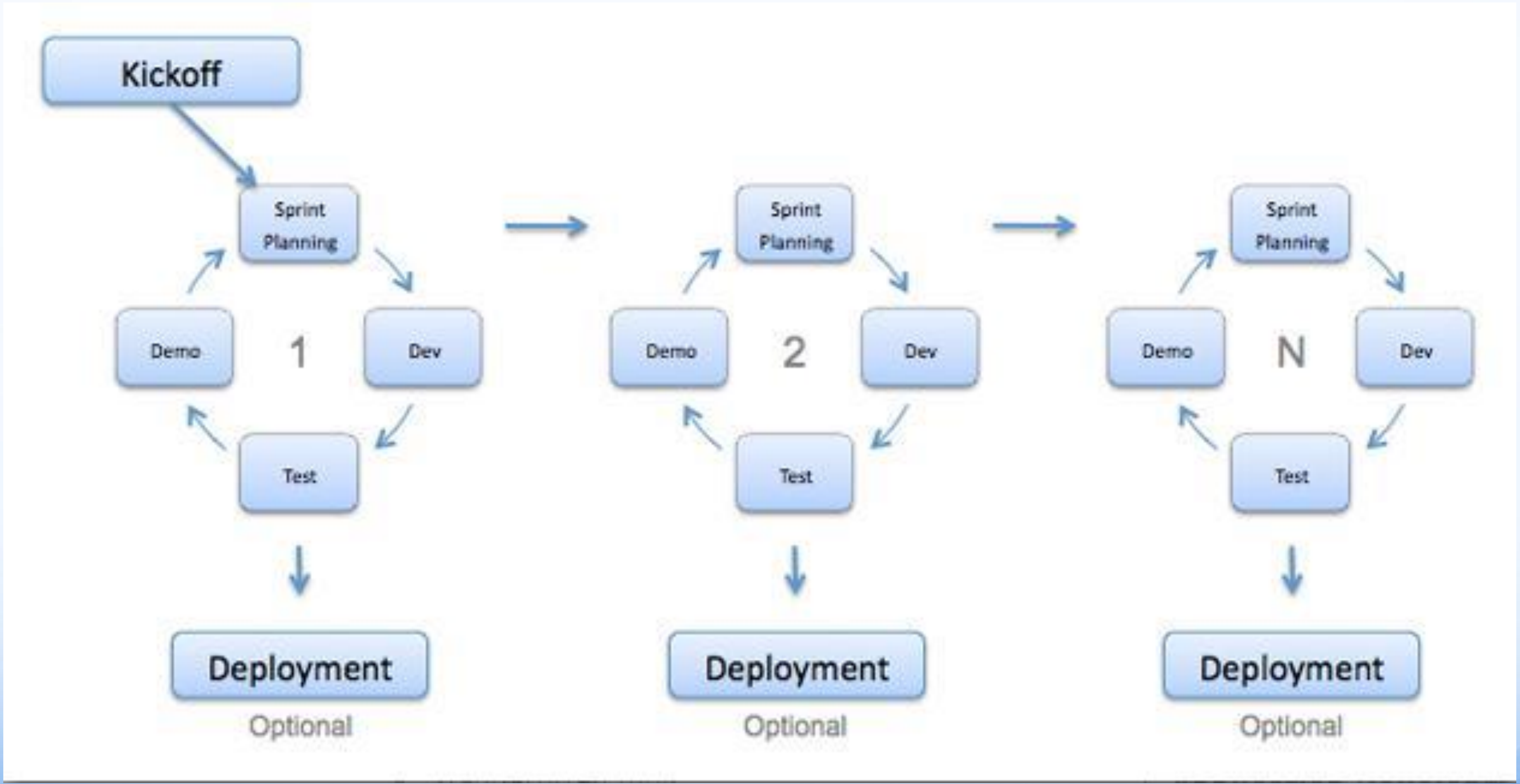
- Depends on strong team and individual performances for identifying business requirements.
- Only a system that can be modularized can be built using RAD
- Requires highly skilled developers/designers.
- High dependency on modeling skills
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

When to use rad

- RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
- It should be used if there's high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
- RAD SDLC model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

Agile methodology

- A type of Incremental Methodology
- Software is developed in incremental, rapid cycles
  - Each increment is called a **Sprint**
- Results in small incremental releases with each release building on previous functionality
- Each release is thoroughly tested to ensure software quality
- Used for time-critical applications
- SCRUM & Extreme Programming (XP) are 2 of the most commonly used Agile methodologies


Kickoff
Sprint Planning
Dev
Test
Demo
1
Deployment
Optional
Sprint Planning
Dev
Test
Demo
2
Deployment
Optional
Sprint Planning
Dev
Test
Demo
N
Deployment
Optional

Advantages of agile

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Close, daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed

Disadvantages of agile

- In the case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear on what final outcome that they want
  - Customer almost never knows what they want!
- Only senior programmers are capable of taking the kind of decisions required during the development process.
  - Hence it has no place for newbie programmers, unless combined with experienced resources.

When to use agile

- When new changes are needed to be implemented.
  - The freedom agile gives to change is very important.
  - New changes can be implemented at very little cost because of the frequency of new increments that are produced.
- To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
- Unlike the waterfall model, very limited planning is required with Agile in order to get started with the project.
  - Agile assumes that the end users' needs are ever changing in a dynamic business and IT world.
  - Changes can be discussed and features can be newly effected or removed based on feedback.
  - This effectively gives the customer the finished system they want or need.
- Both system developers and stakeholders alike, find they also get more freedom of time and options than if the software was developed in a more rigid sequential way.
  - Having options gives them the ability to leave important decisions until more or better data is available or even entire hosting programs are available
  - This means the project can continue to move forward without fear of reaching a sudden standstill.

Agile Manifesto

- “We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals & Interactions over processes & tools

Working Software over comprehensive documentation

Customer Collaboration over contract negotiation

Responding to Change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.”

Agile Values – Individuals & Interactions

- Teams of people build software systems, and to do that they need to work together effectively
  - 5 developers working together VS. 5 hamburger flippers with a well defined process and the best tools – who could develop a better system?
- The most important factor to consider are the people and how they interact with each other
- Without a coherent team environment, the best tools and the most well-defined process won't be of any use

Agile Values – Working Software

- Would a user want a 50 page document describing what you intend to build for them or would they rather just have the software?
- So wouldn't it make sense that you produce software quickly and often – giving the customer what they actually want?
- Documentation has its place
  - When it's written properly, it can be a valuable guide for explaining how and why a system is developed the way it is
- Primary goal of **software** development is to develop **software**, not documentation

Agile Values – Customer Collaboration

- Only the customer can tell you what they want
  - They don’t have the technical skills to be as specific as you need
  - They won’t get it right the 1st time
  - They’ll likely change their minds
- Having a contract is important, but it doesn’t take the place of effective communication
- Successful development teams:
  - Work closely with their customers
  - Invest the effort to discover what their customers need
  - Educate their customers along the way

Agile Values – Responding to Change

- People change their priorities for many reason
- As work progresses on a system, your client's understanding of the business problem and your solution to it evolves
- Change is a reality of software development
  - Your software development process must reflect this
- There is nothing wrong with having a Project Plan
  - But it must be malleable (There must be room for change)

Agile Values – They're Agreeable, but hard to Implement!

- Almost everyone will instantly agree to these Agile Values but most will not usually adhere to them in practice!
- Senior management will always claim that its employees are the most important aspect of the organization, yet they will continue to treat their staff as replaceable assets
- Agile Development teams do what they say and say what they do!

Types of Agile

- Extreme Programming (XP)
- Lean & Kanban Development
- Crystal Methods
- Feature Driven Development
- Scrum

Extreme Programming (XP)

- Used in small, collected project teams up to 10 members (more is probable)
- Development is flexible and lightweight
- Based on 4 values:
  - Collaboration – Active participation of all stakeholders (customer is part of the dev team)
    - Looking at the task at hand rather than at the long-term
  - Feedback – It's effective if it's instantaneous
    - System feedback is achieved through Unit Testing
    - Customers give feedback on the existing model and requirements are changed through collaboration
  - Revision – XP focuses on current tasks and problems
    - Written code is often unstructured so requires constant updates (refactoring)
  - Respect – Programming in Pairs
    - Programmers split tasks – 1 does the coding (driver); the other is a problem solver

Lean Software Development

- Iterative Agile Methodology
- Focuses the team on delivering **value** to the customer, and on the efficiency of the "value stream"
- Main Principles:
  - Eliminating Waste
  - Amplifying Learning
  - Deciding as Late as Possible
  - Delivering as Fast as Possible
  - Empowering the Team
  - Building Integrity
  - Seeing the Whole

Kanban Method

- Used by organizations to manage the creation of products with an emphasis on Continual Delivery while not overburdening the development team
- It’s a process designed to help teams work together more effectively
- Based on 3 Principles:
  - Visualize What You Do Today (Workflow)
    - Seeing all the items in context of each other can be very informative
  - Limit The Amount of WIP
    - Helps balance the flow-based approach so teams don’t start and commit to too much work at once
  - Enhance Flow
    - When something is finished, the next highest thing from the *backlog* is pulled into play

Scrum

Overview of Scrum

- Original term comes from a move in Rugby
  - The team comes together and attempts to move the ball across the field as a team in a huddle
- Scrum is an Agile framework product owners can use to streamline their development process
- Goal is to deliver new software capability every 2-4 weeks

Who Uses Scrum

- Rugby Teams
- Software Development Teams
- Most popular Agile Methodology
- Over 70% of organizations are using Scrum or some form of it

Scrum & Agile Project Management

- Scrum is a sub-group of Agile
  - Agile is a set of values/principles that describe a group's day-to-day interactions and activities
  - Scrum Methodology follows values and principles of Agile

- Benefits of Scrum
  - Higher Productivity
  - Better-quality products
  - Reduced time-to-market
  - Improved Stakeholder satisfaction (customer satisfaction)
  - Better Team Dynamics
  - Happier Employees