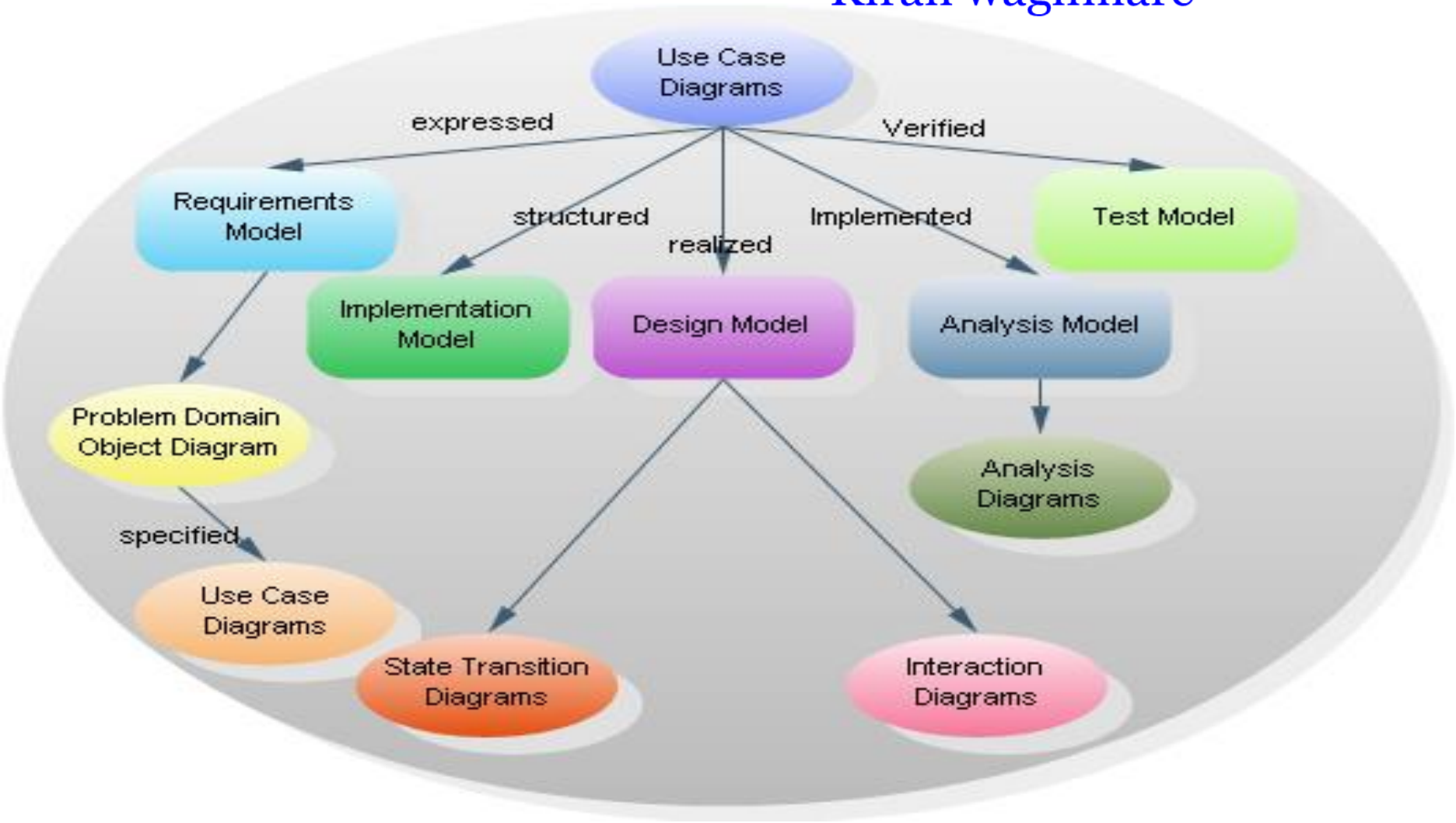
The SDLC Model

Session IV

Kiran waghmare



Spiral Model

Spiral Model

- In the 1980s; **Boehm** introduced - the spiral model.

- The spiral model comprises activities organized-
  - in a **spiral**, and has many **cycles**.

- This model combines the features of the **prototyping model** and **waterfall model** and is advantageous for **large, complex, and expensive projects.**

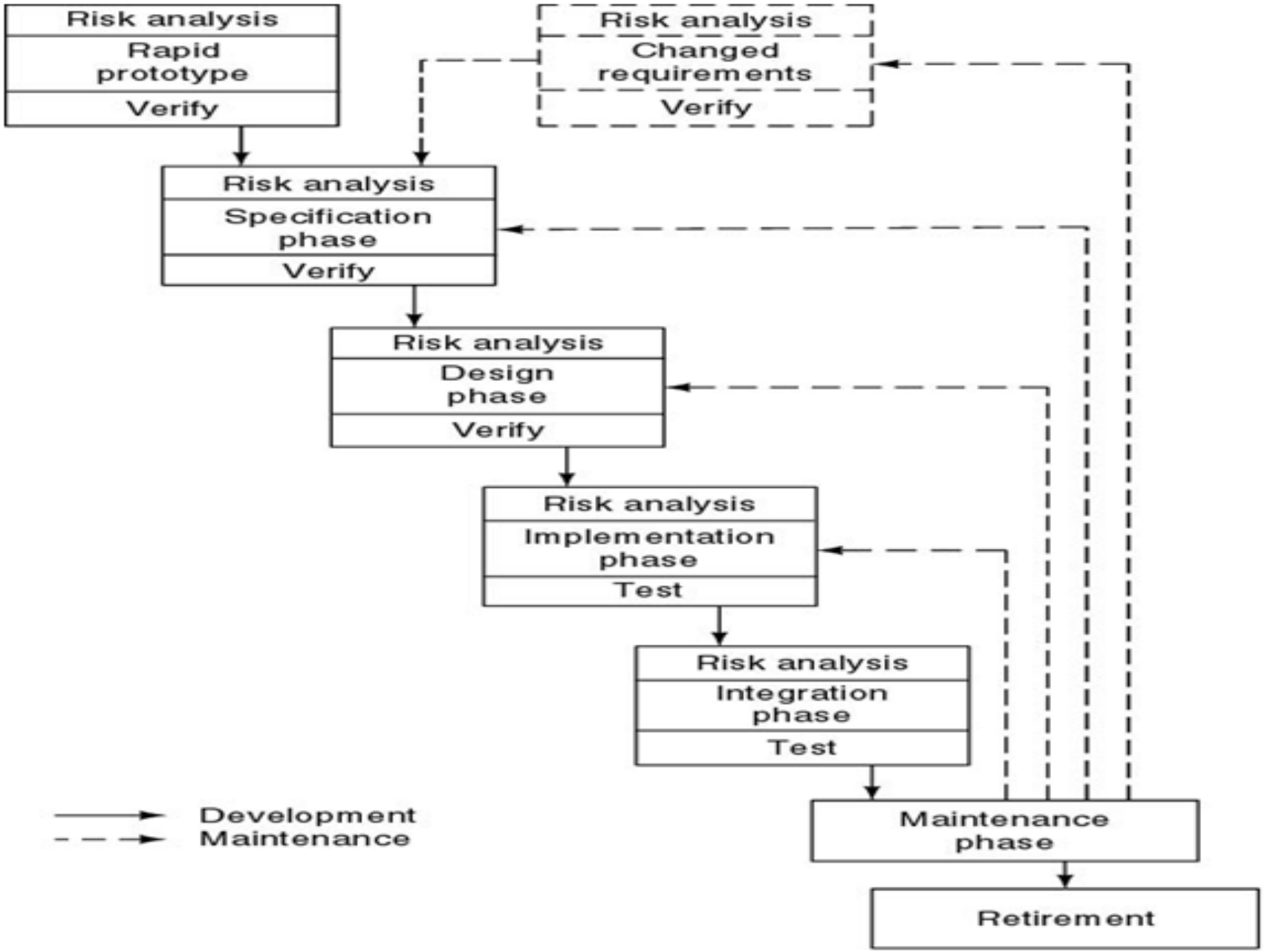
- It **determines requirements problems** in developing the prototypes.

- In addition, it guides and **measures the need of risk management** in each cycle of the spiral model.

Spiral Model

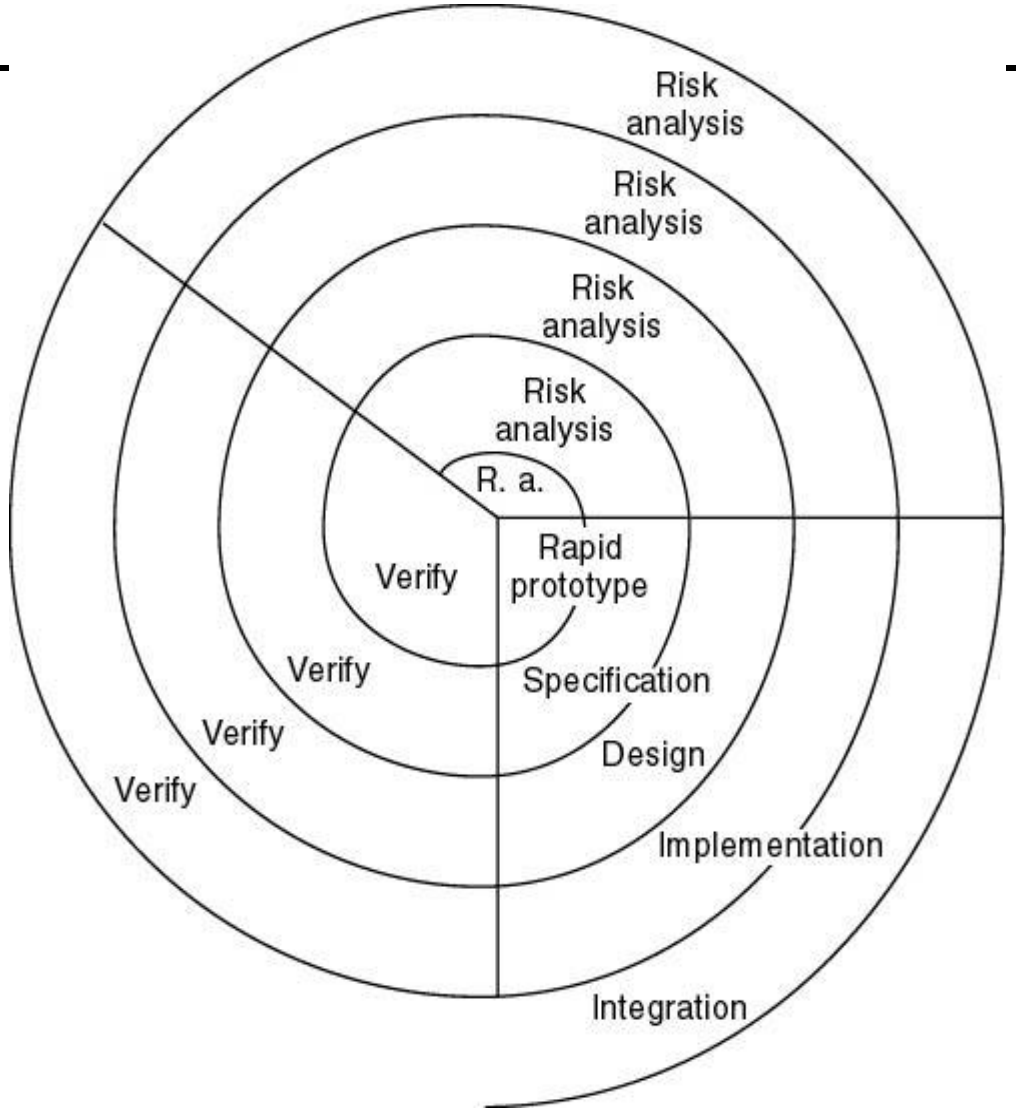
- IEEE defines the spiral model as
  - a model of the software development process in which
    - the **constituent activities**,
    - typical **requirements analysis,**
    - preliminary and detailed· **design,**
    - **coding,**
    - **integration**, and
    - **testing**, are performed iteratively until the software is complete.

- The objective of the spiral model is
  - to emphasize management **to evaluate and resolve risks** in the software project.

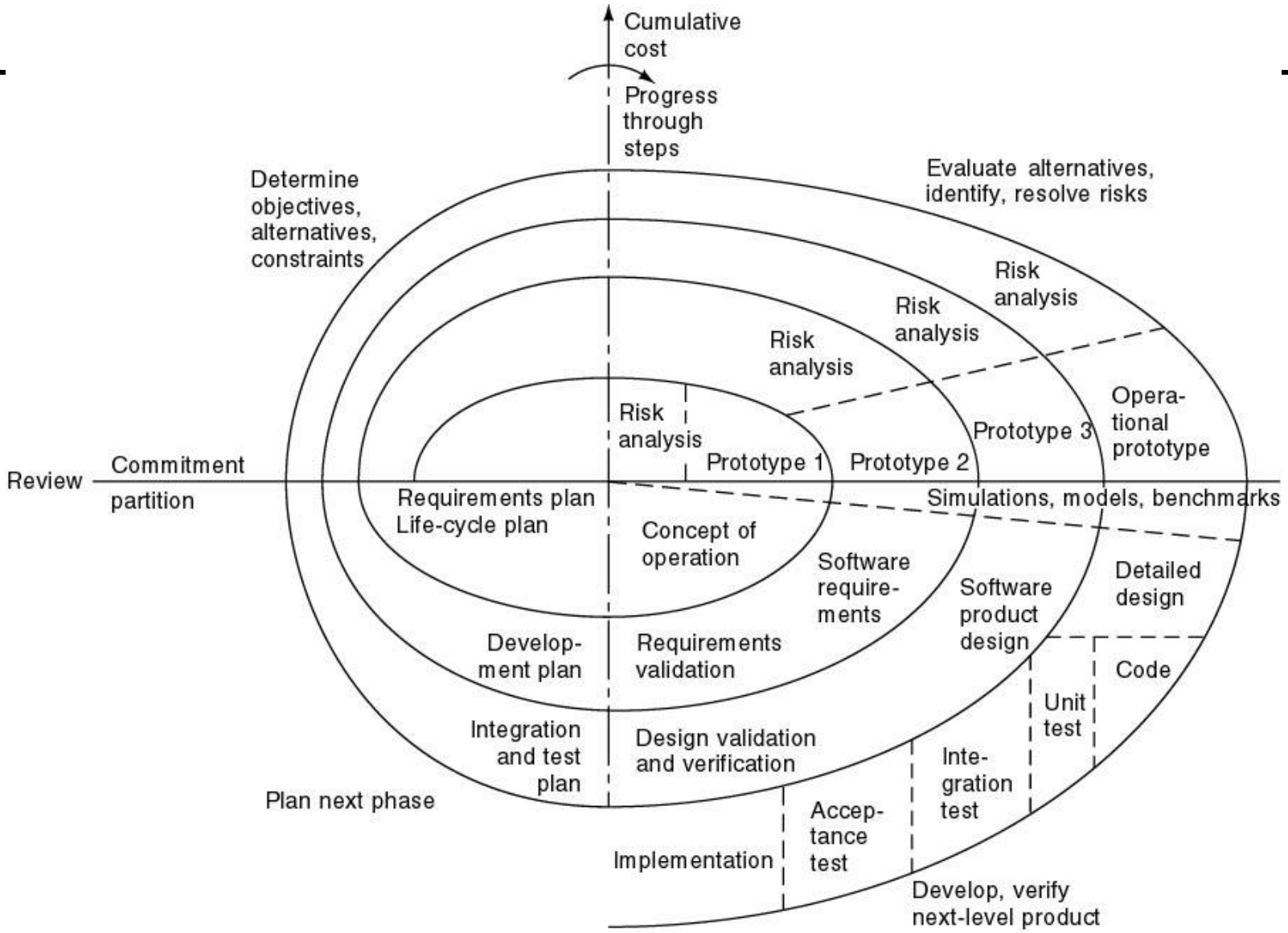

Risk analysis
Rapid prototype
Verify
Risk analysis
Changed requirements
Verify
Risk analysis
Specification phase
Verify
Risk analysis
Design phase
Verify
Risk analysis
Implementation phase
Test
Risk analysis
Integration phase
Test
Maintenance phase
Retirement
Development
Maintenance

Simplified Spiral Model

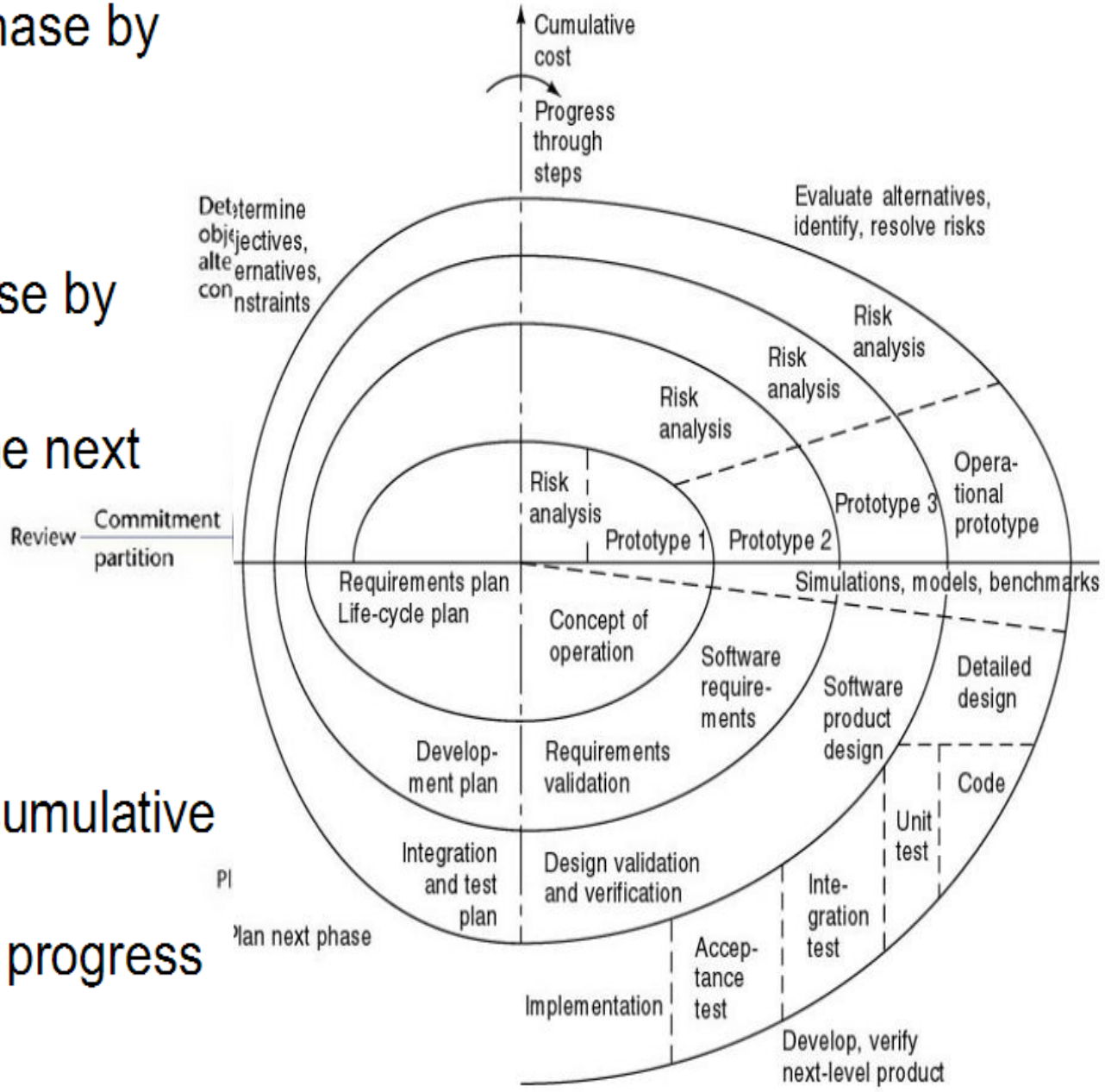
- ~~If risks cannot~~ be resolved, project is immediately terminated



Full Spiral Model (contd)



- Precede each phase by
  - Alternatives
  - Risk analysis
- Follow each phase by
  - Evaluation
  - Planning of the next phase



Radial dimension: cumulative costs to date
Angular dimension: progress through the spiral.

Advantages of Spiral model:

1. **Easy to judge** how much to test
2. **No distinction** between development, maintenance
3. **Provides a working model** to the user early in the process, enabling early assessment and increasing user's confidence.
4. **Helps in reducing risks** associated with the software.
5. **Changing requirements** can be accommodated.
6. **Allows for extensive use** of prototypes.
7. Requirements can be captured **more accurately.**
8. Development can be **divided into smaller parts and more risky parts can be developed earlier** which helps better risk management.

Disadvantages of Spiral model:

1. **For large-scale software** only

---

2. For internal **(in-house) software** only

3. If the user is not satisfied by the developed prototype, then a new prototype is developed. This process goes on until a perfect prototype is developed. Thus, this **model is time consuming and expensive**.

4. **End of project may not be known** early.

5. **Not suitable for small or low risk projects** and could be expensive for small projects.

6. Process is **complex**. Spiral may **go indefinitely.**

7. **Large number of intermediate stages requires** excessive documentation.

Spiral Model

- Strengths
  - The emphasis on alternatives and constraints supports the reuse of existing software and the incorporation of software quality as a specific objective.
  - There is essentially no distinction between development and maintenance since maintenance is another cycle of the spiral.

- Weaknesses
  - For large-scale software only
  - For internal (in-house) software only

5

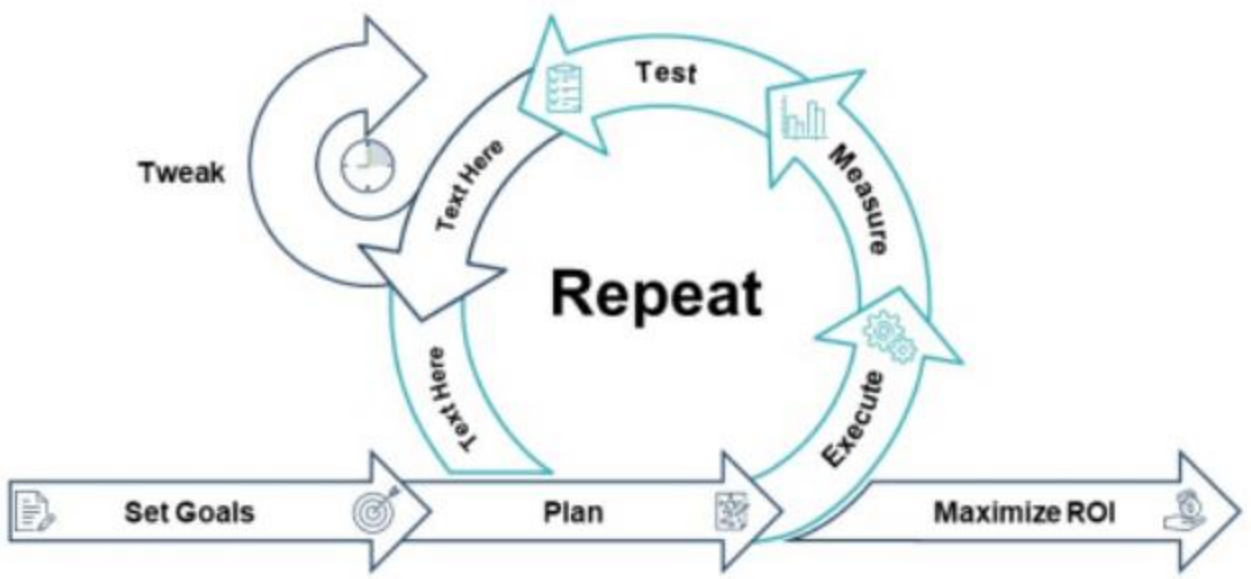
When to use Spiral model:

1. When **costs and risk evaluation is important**
2. For medium to **high-risk projects**
3. **Long-term project commitment** unwise because of potential changes to economic priorities
4. Users are **unsure of their needs**
5. Requirements are **complex**
6. **New product line**
7. **Significant changes are expected** (research and exploration)

Comparison of Life Cycle Models

Life-Cycle Model	Strengths	Weaknesses
Waterfall life-cycle model	▪Document driven ▪Easier maintenance	▪Delivered product may not meet client's needs
Iterative-and-Incremental life-cycle model	▪Closely models real world software production ▪Easier to test and debug	▪Each phase of an iteration is rigid and do not overlap each other.
Evolution-Tree Model	▪Closely models real world software production. Equivalent to Iterative and Incremental	
Code-and-fix life cycle model	▪Use for small scale programs that require no maintenance	▪Should not be used on large nontrivial programs
Rapid-prototyping life-cycle model	▪Ensured that the delivered product is to client's specification	▪Not yet proven beyond all doubt
Spiral life-cycle model	▪Risk Driven	▪Best used for large scale products ▪Developers must be competent in risk analysis

Agile Model: XP



Agile Model: XP

- Agile SDLC model is a combination **of iterative and incremental process models**
- Agile Methods break the product into **small incremental builds.**

- These builds are provided **in iterations.**
- Each iteration typically lasts from about **one to three weeks**.
- Every iteration involves **cross functional teams** working simultaneously on various areas like
  - planning, requirements analysis, design, coding, unit testing, and acceptance testing.

- At the end of the iteration a **working product is displayed to the customer** and important stakeholders.

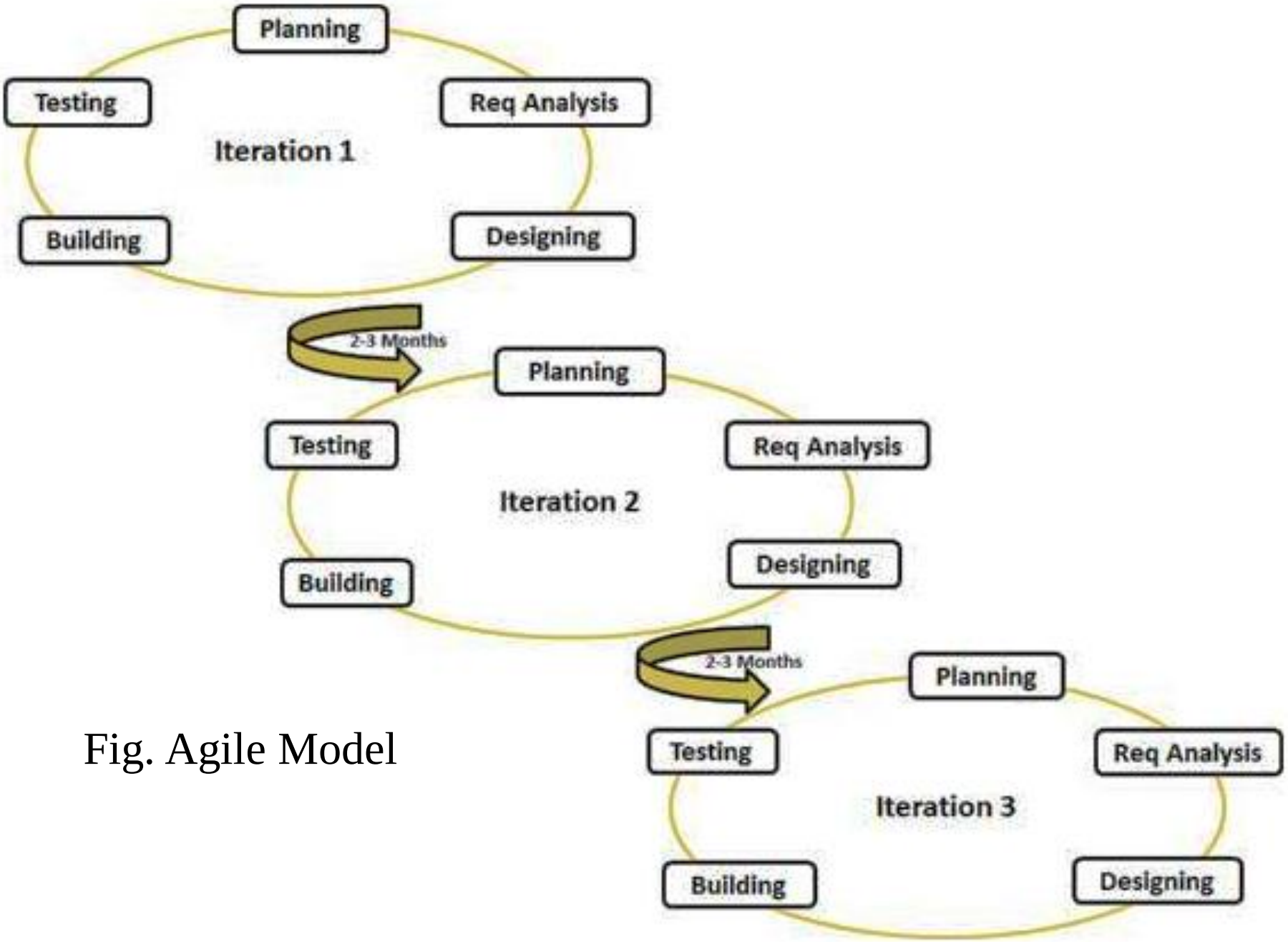


Fig. Agile Model

Merits	Limitations
• Agile methodology is adaptive and hence it can adapt well with changing requirements. • Dedicated time and effort is not required because customer requirement may change • Customer can give continuous inputs and have face to face interactions with the team. •Agile method does not give room for guesswork, documentation is exhaustive but crisp • Customer satisfaction is ensured	• Difficult to scale when projects are large where documentation is needed • Agile teams need to have experience and skills • In some deliverables it is problematic to evaluate as required at beginning of SDLC • Deign is not much emphasized • The project can be easily distracted when the customer is not clear on outcomes • Testing and test construction is difficult and needs specialized skills

Extreme Programming

- Extreme Programming is a somewhat **controversial**
  - new approach to software development based on the **iterative-and-incremental model**.

- The software development team determines
  - the **various features** (stories) the client would like the product of support.

Extreme Programming

- Pair Programming
  - The programmer **works together with a partner on one screen** to implements the task ad ensure that all the test cases work correctly.
- Continuous integration of tasks since
  - a number of **pairs implement tasks in parallel.**
- The TDD test cases used for
  - the task are **retained and utilized** in all further integration testing.

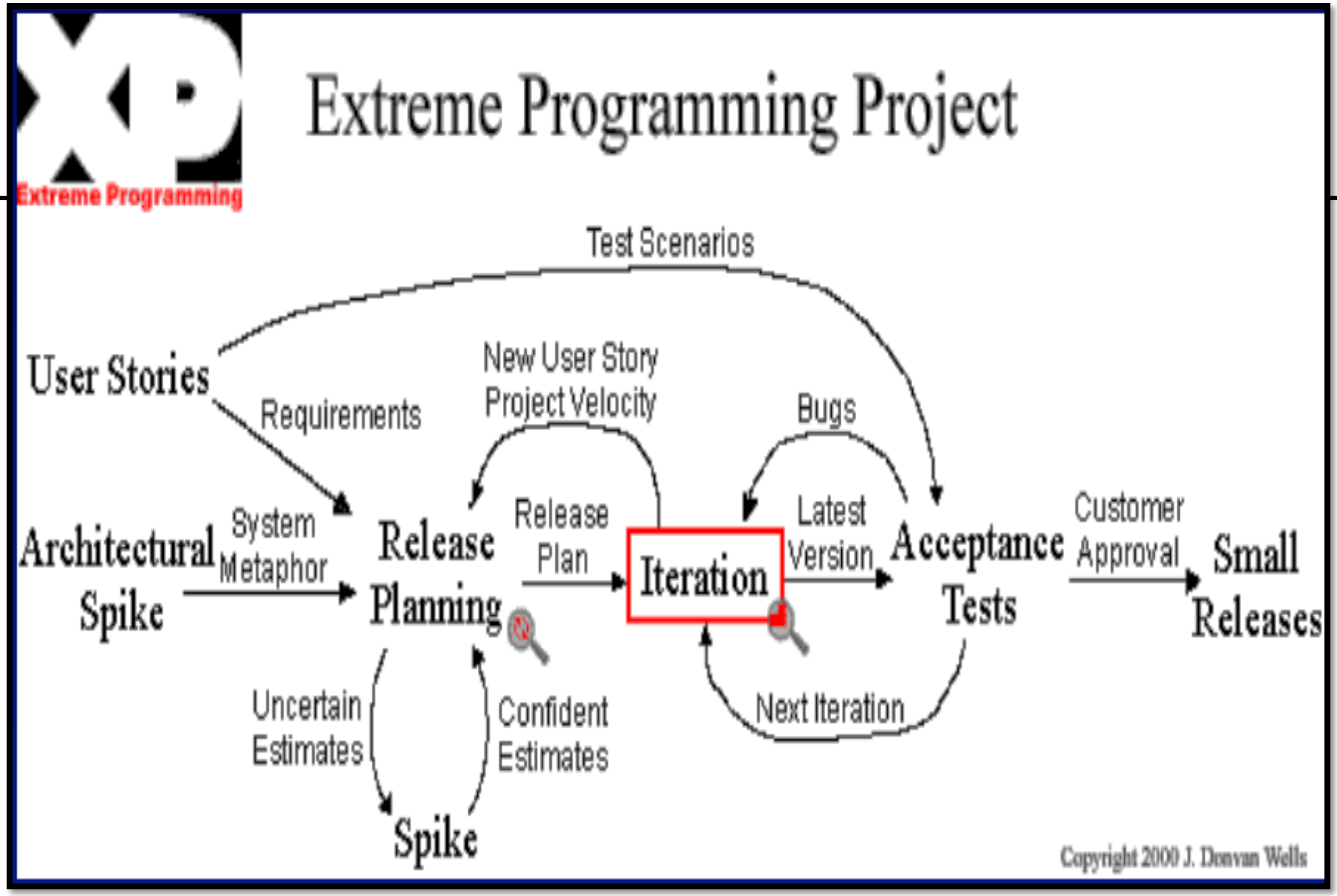


Fig. Extreme Programming

Features of Extreme Programming

- Computers of XP team are **set up in the center of a large room lined with small cubicles**.
- A **client representative** works with the XP team at all times.
- No individuals can work **overtime** for 2 successive weeks.
- There is **no specialization**.
- **Refactoring** is used.
- **Used** in no of small and medium size projects.

Advantages Extreme Programming

1. Useful when **requirements are vague or changing**
2. Emphasis on **teamwork** and communication
3. **Programmer estimates** before committing to a schedule
4. **Continuous measurement**; frequent, extensive testing

Disadvantages Extreme Programming

1. **Limited to small products and small teams –**

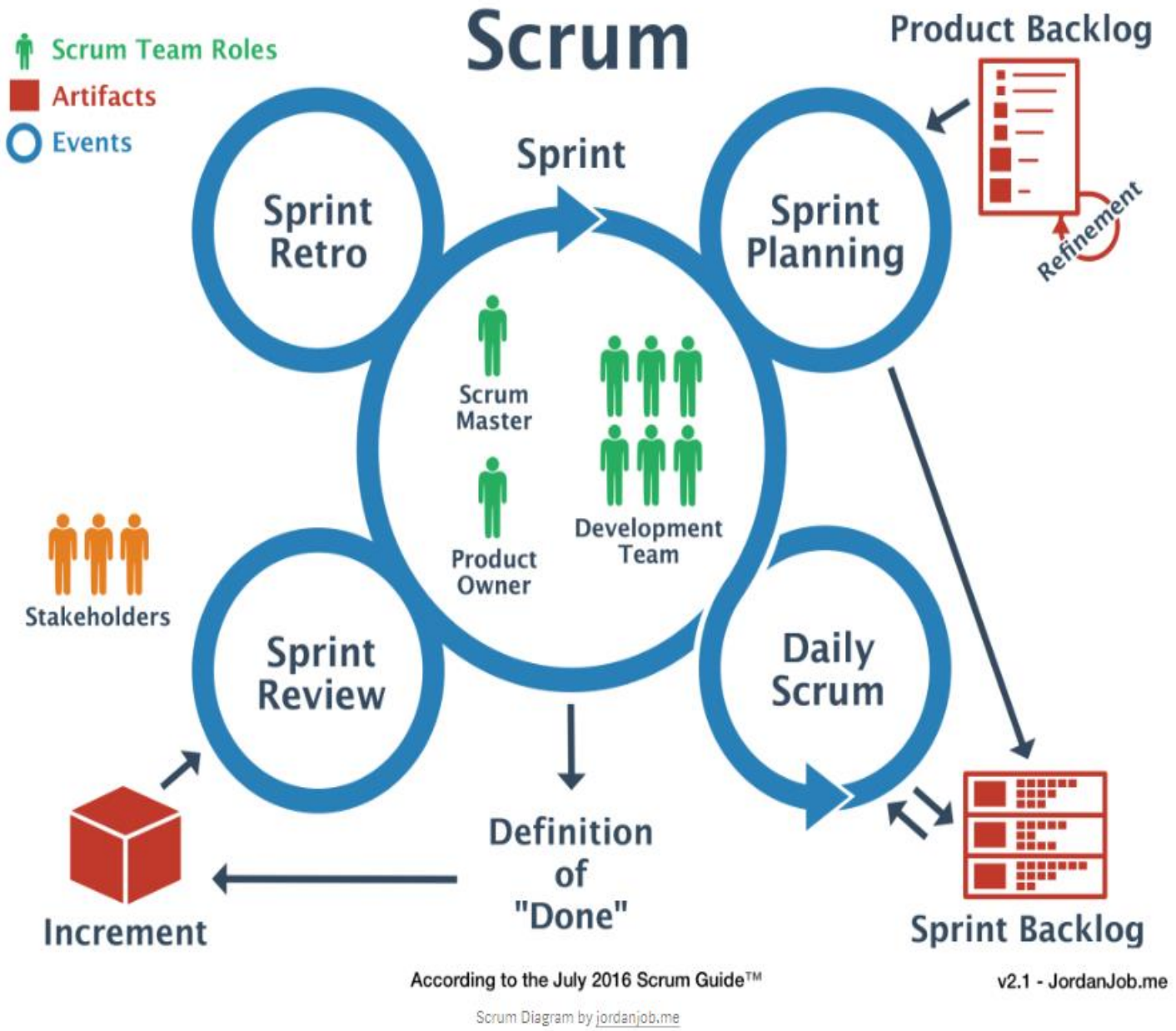
2. can be disastrous when **programs are larger than** a few thousand lines of code or

3. when the **work involves** more than a few people.

4. **Lack of design** documentation

5. **Lack of a structured review** process

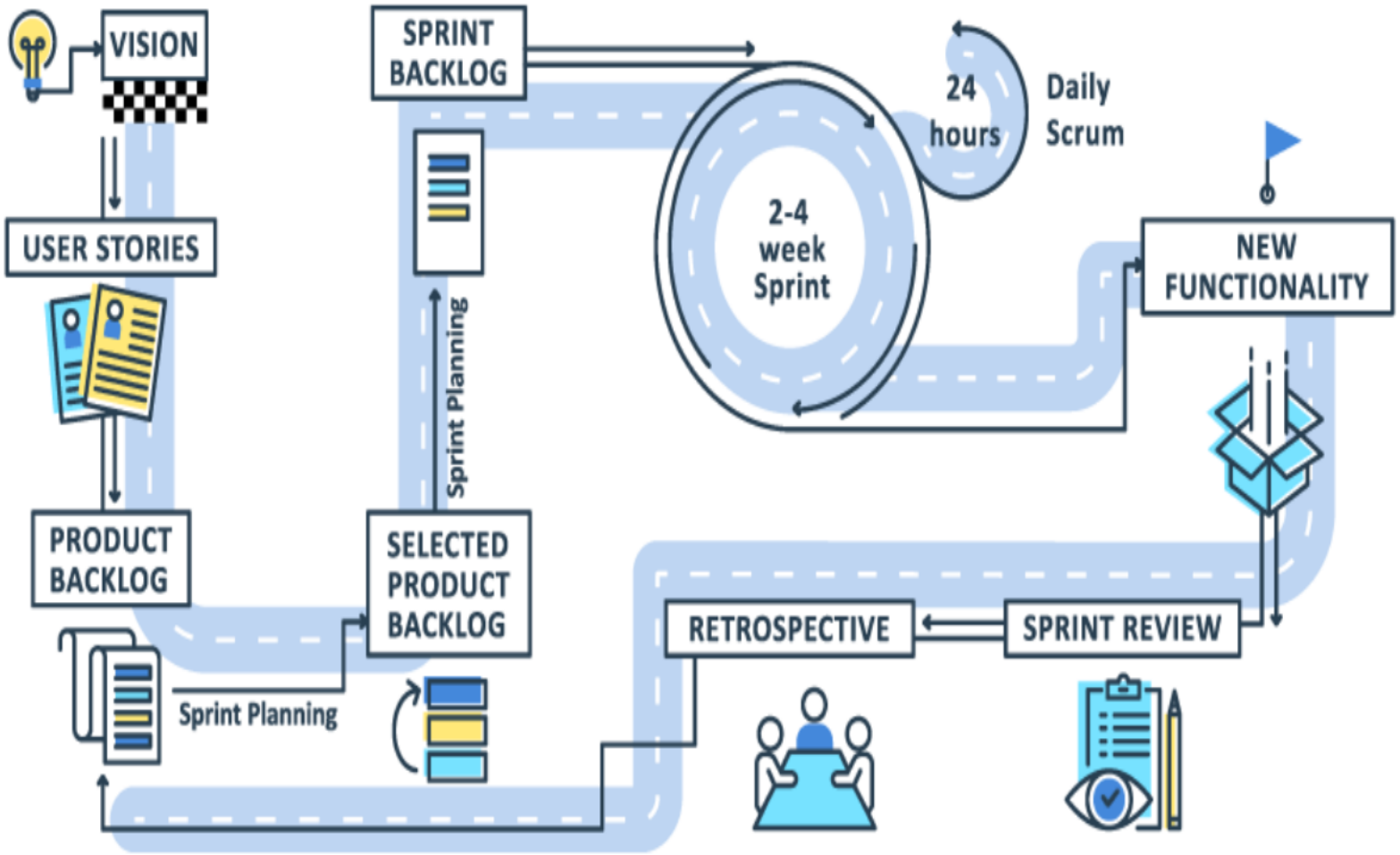
SCRUM


Scrum
Scrum Team Roles
Artifacts
Events
Product Backlog
Refinement
Sprint
Sprint Retro
Sprint Planning
Scrum Master
Product Owner
Development Team
Stakeholders
Sprint Review
Daily Scrum
Definition of "Done"
Increment
Sprint Backlog
According to the July 2016 Scrum Guide™
v2.1 - JordanJob.me
Scrum Diagram by jordanjob.me

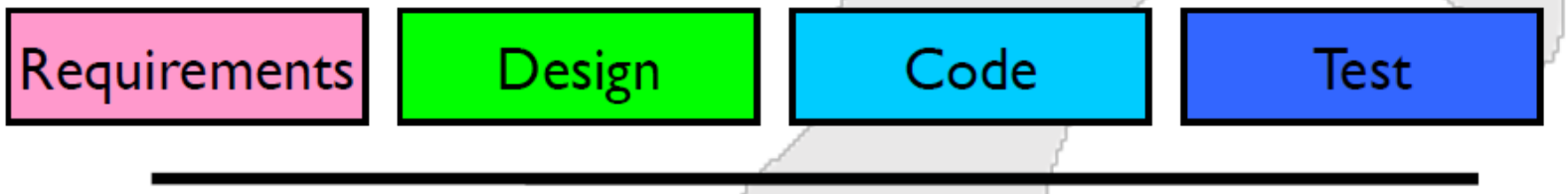
- **Dynamic Systems Development Method (DSDM)**
- DSDM is for practical people
- Pareto principle – 80% of an application can be delivered in 20% of the time needed to develop whole application.
  DSDM is for practical people.
- **Feature Driven Development (FDD)** is for large projects
- In FDD we decompose big problems into small features
- Features are small blocks of deliverable functionality, so their design and code are easier to inspect and implement.

- **Lean Software Development (LSD)** – get rid of unnecessary stuff
- **Agile Modeling (AM)** – introducing UML, and principles like: Model with purpose, Use multiple modes, Use models as travel light...

Types of Scrum Meetings and Scrum Best Practices

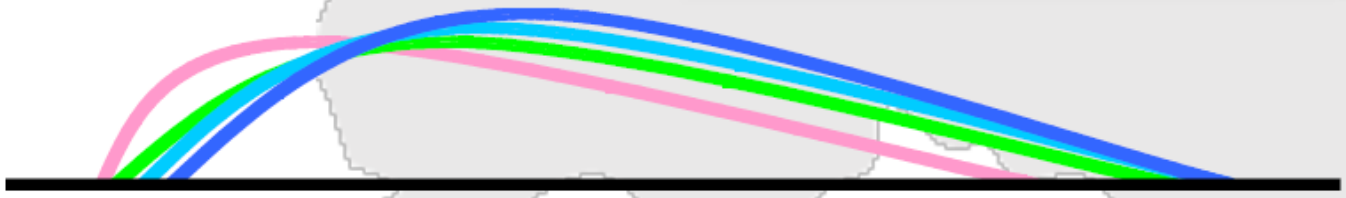


Sequential vs. overlapping development

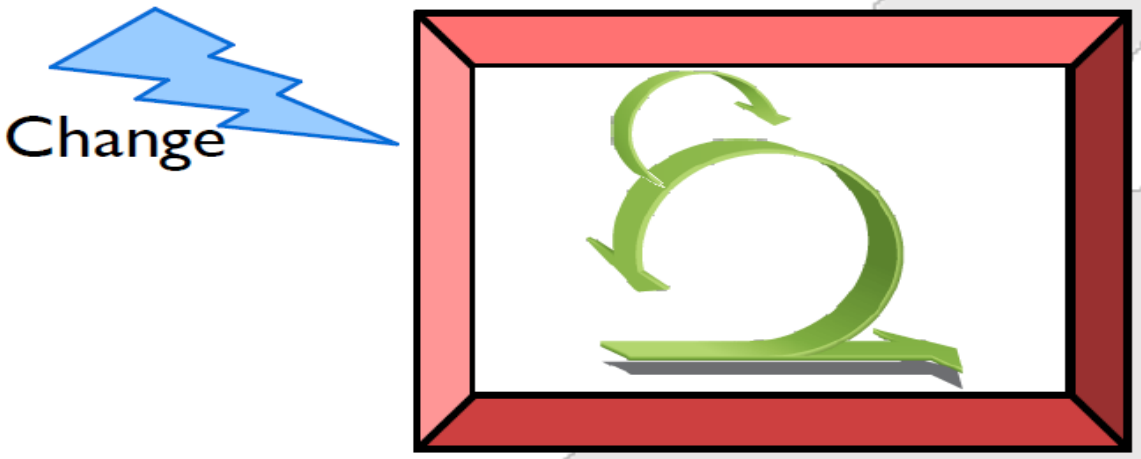


Rather than doing all of one thing at a time...

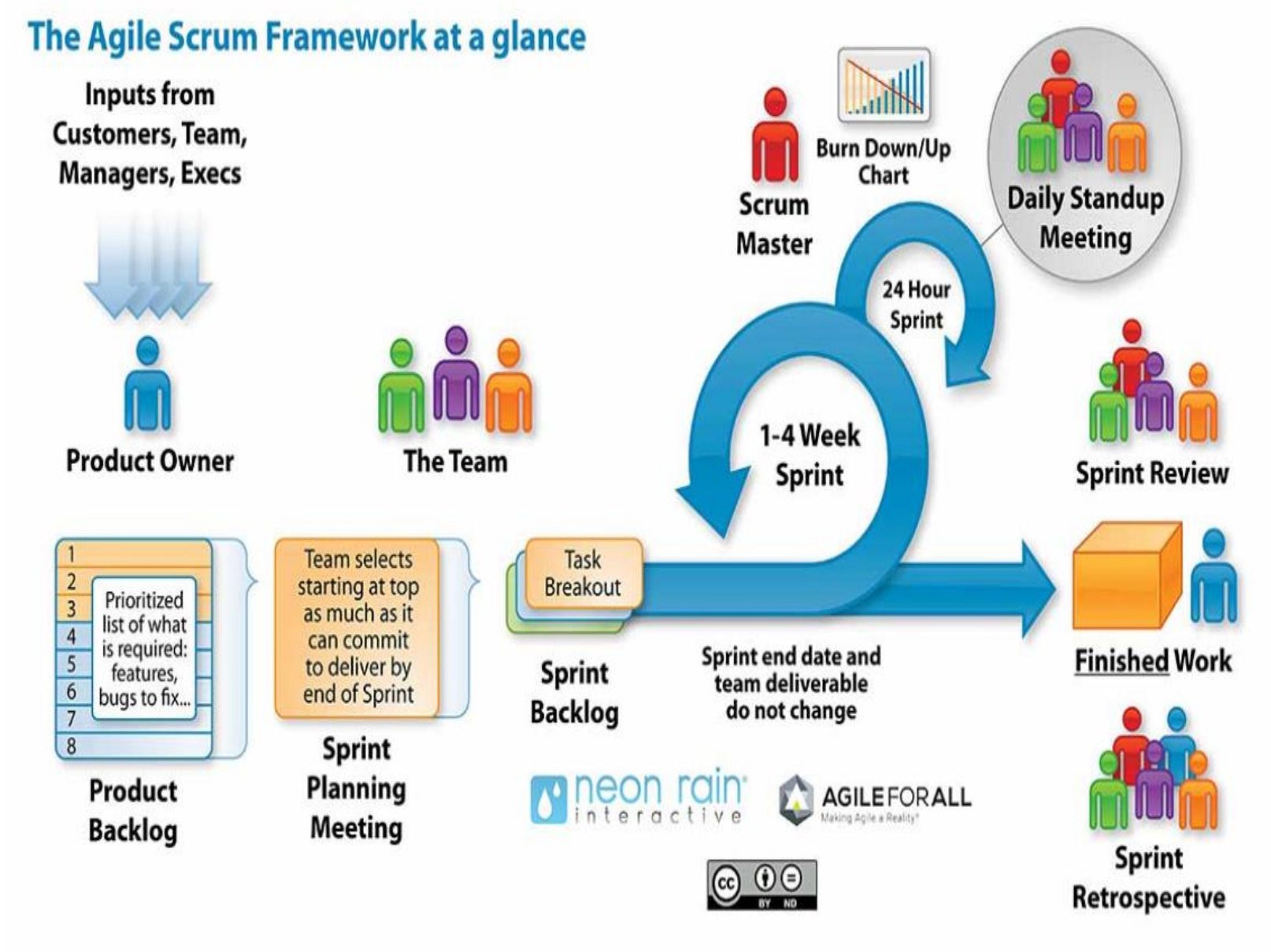
...Scrum teams do a little of everything all the time



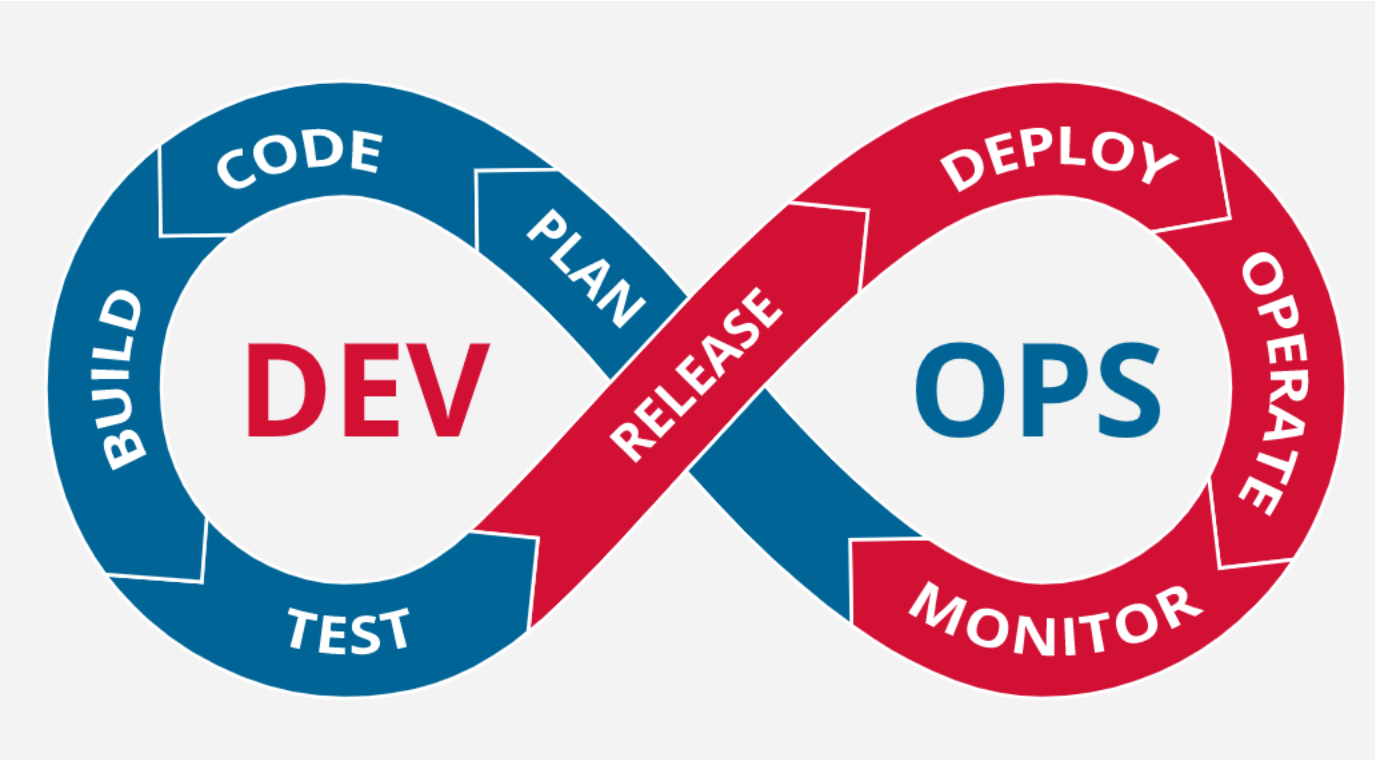
No changes during a sprint



- Plan sprint durations around how long you can commit to keeping change out of the sprint


The Agile Scrum Framework at a glance
Inputs from Customers, Team, Managers, Execs
Product Owner
The Team
Scrum Master
Burn Down/Up Chart
Daily Standup Meeting
24 Hour Sprint
1-4 Week Sprint
Sprint Review
1
2
3
4
5
6
7
8
Prioritized list of what is required: features, bugs to fix...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Finished Work
Sprint Retrospective
neon rain interactive
AGILEFORALL
Making Agile a Reality

Devops


CODE
PLAN
BUILD
TEST
DEV
RELEASE
DEPLOY
OPERATE
MONITOR
OPS

Devops

- DevOps (a combination of development and operations) is a software development method that stresses communication, collaboration and integration between software developers and information technology(IT) professionals thereby
  - Enable rapid evolution of products or services
  - Reduce risk, improve quality across portfolio, and reduce costs


IntelliPaat
CODE
PLAN
MONITOR
OPERATE
Dev
Ops
DEPLOY
RELEASE
TEST
BUILD

Tasks of DevOps Engineers

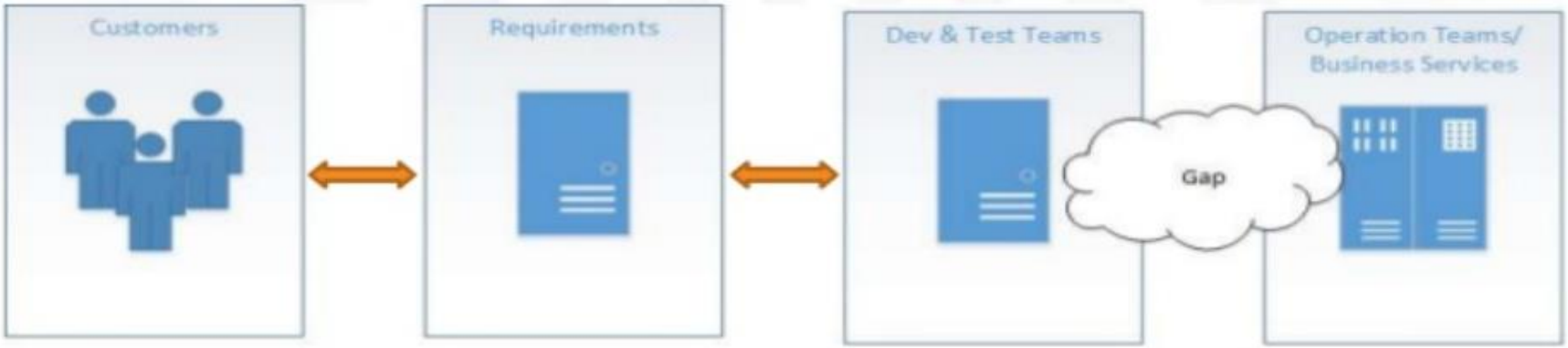
IntelliPaat

- Be an excellent Sysadmin
- Deploy Virtualization
- Hands-on experience in network and storage
- Knowledge of coding goes a long way
- Soft skills are a must-have
- Understanding the automation tools
- Knowledge of Testing
- Security aspects of the IT organization

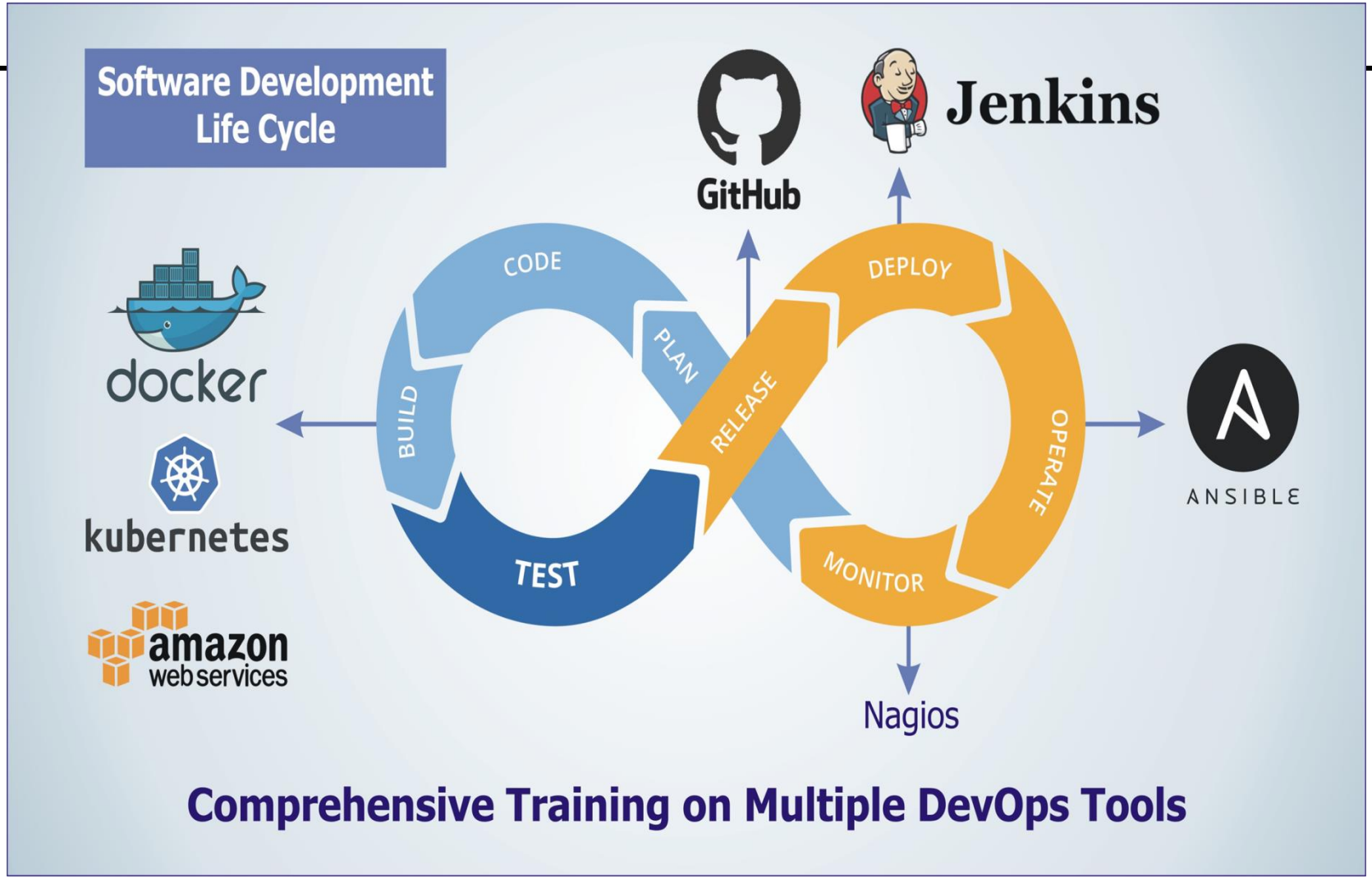
Devops

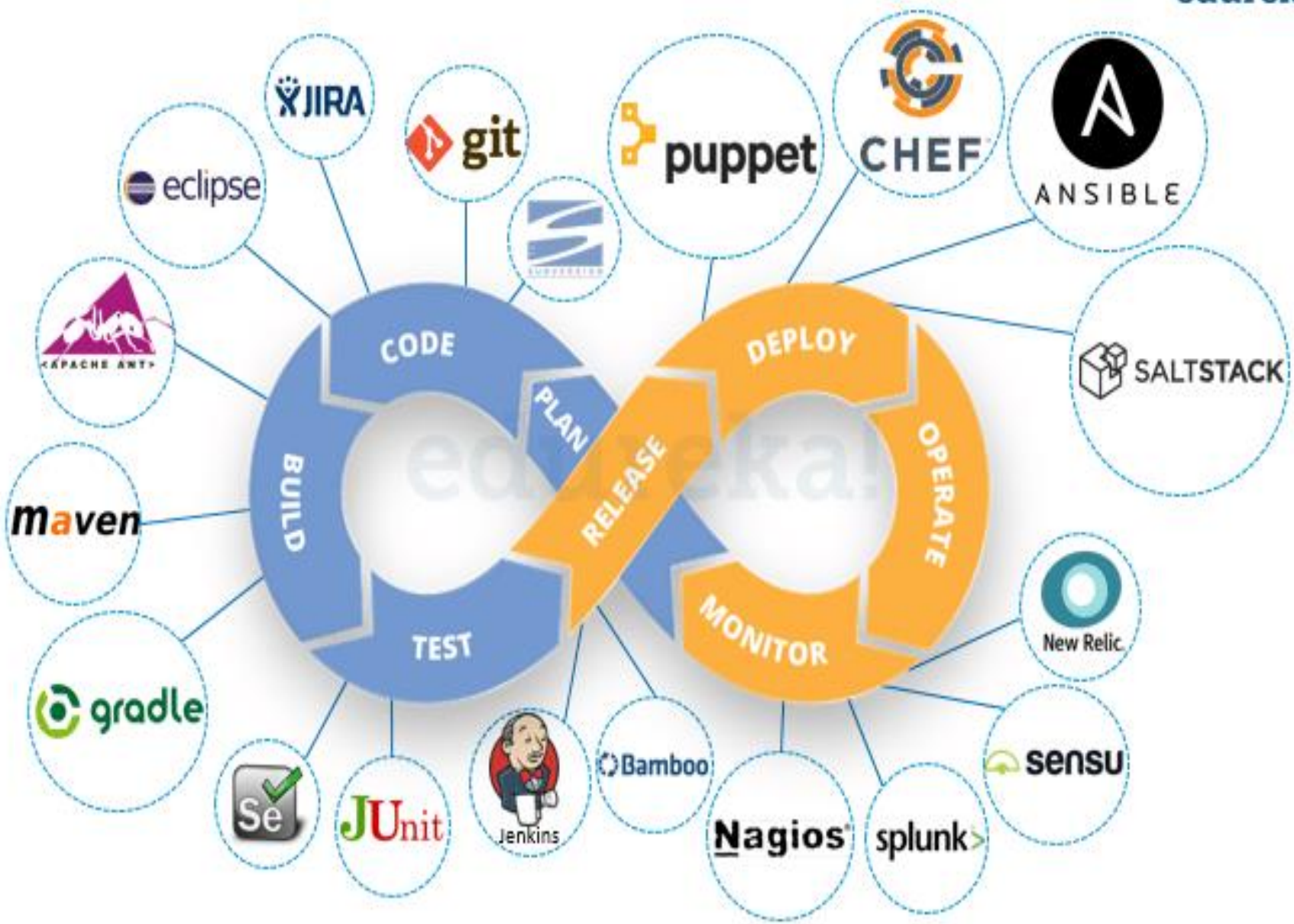
- Develop and test in an environment similar to production
- Deploy builds frequently
- Validate operation quality continuously

Why DevOps? – Delivery Challenges



Devops tools




edureka!
JIRA
git
eclipse
puppet
CHEF
ANSIBLE
SALTSTACK
CODE
PLAN
BUILD
TEST
RELEASE
DEPLOY
OPERATE
MONITOR
maven
gradle
New Relic
sensu
Bamboo
Jenkins
JUnit
Nagios
splunk>
edureka!

Some Tools Used by a DevOps

- **Git and GitHub:** Version control system and source code management
- **Jenkins:** Server automation and developing CI/CD pipelines
- **Selenium:** Automated testing
- **Kubernetes:** Container orchestration
- **Puppet:** Configuration management
- **Docker:** Software containerization
- **Nagios:** Continuous monitoring
- **Chef:** Configuration management
- **AWS:** Cloud platform integration