MAHARISHI INTERNATIONAL UNIVERSITY • USA

MS in Computer Science

COMPUTER PROFESSIONALS PROGRAM℠

Formerly Maharishi University of Management

CS425: Software Engineering

Lesson 1:

SOFTWARE DEVELOPMENT METHODOLOGIES

Wholeness

- A Software Development methodology is a process followed by a team in making a software product.
- Following a methodology is essential for successful execution of a software project because the process lays out a structured sequence of steps/activities that guide the team through each stage of development.
- *Science of consciousness: Life is structured in layers*

Methodologies

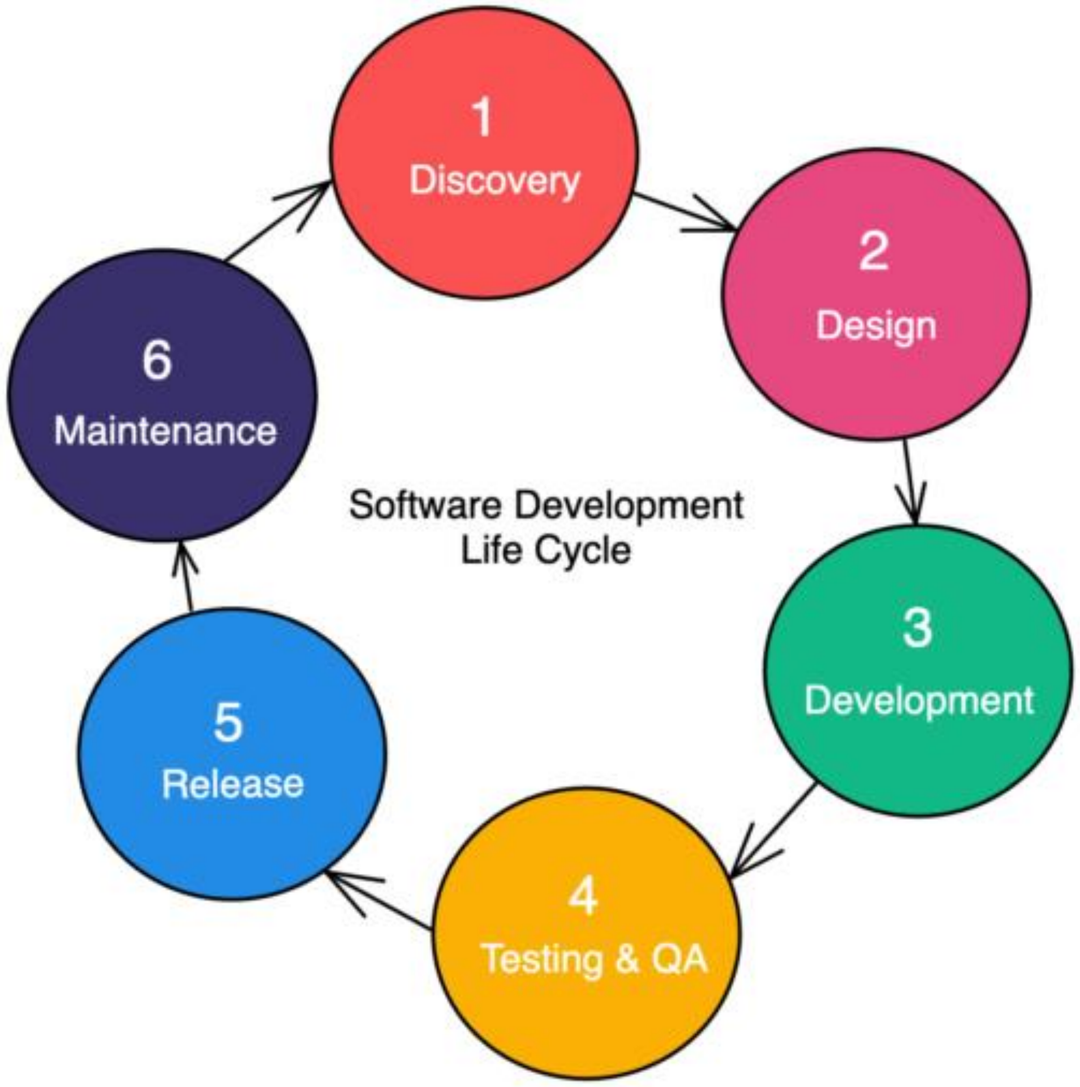
- There are many methodologies in existence. Notable ones: **Waterfall**, **RUP**, **Agile**, **DevOps** etc.
- For a successful project, the team must choose an appropriate methodology that will work best for the project.
- All have different strengths and weaknesses and serve different needs.

Software methodology

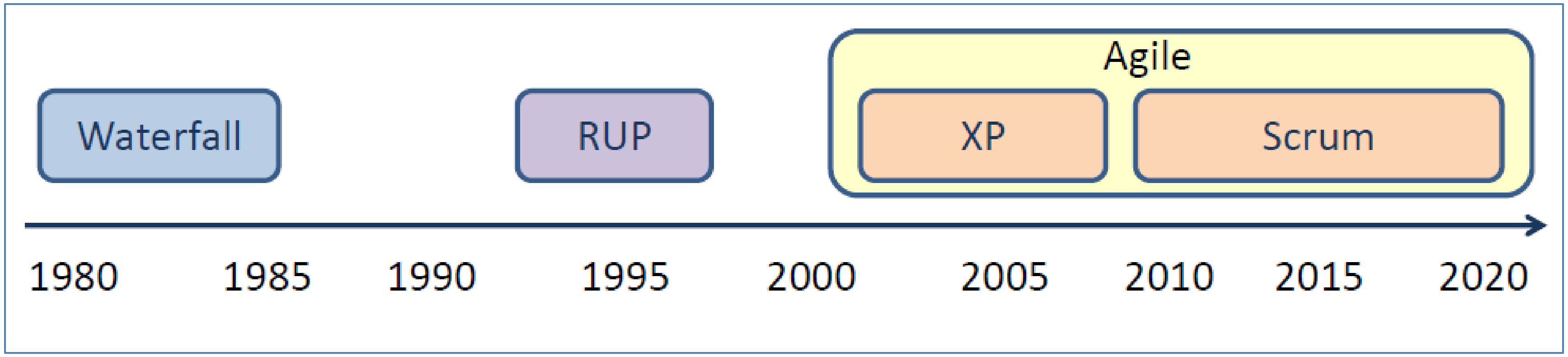
- Who
  - Roles of the people in the project
- What
  - What artifacts are created or used
- When
  - The order in which activities are done/performed
- How
  - What disciplines, activities, best practices etc.

Software Development Lifecycle

- Typically, a Software Development project will involve the following lifecycle of activities:

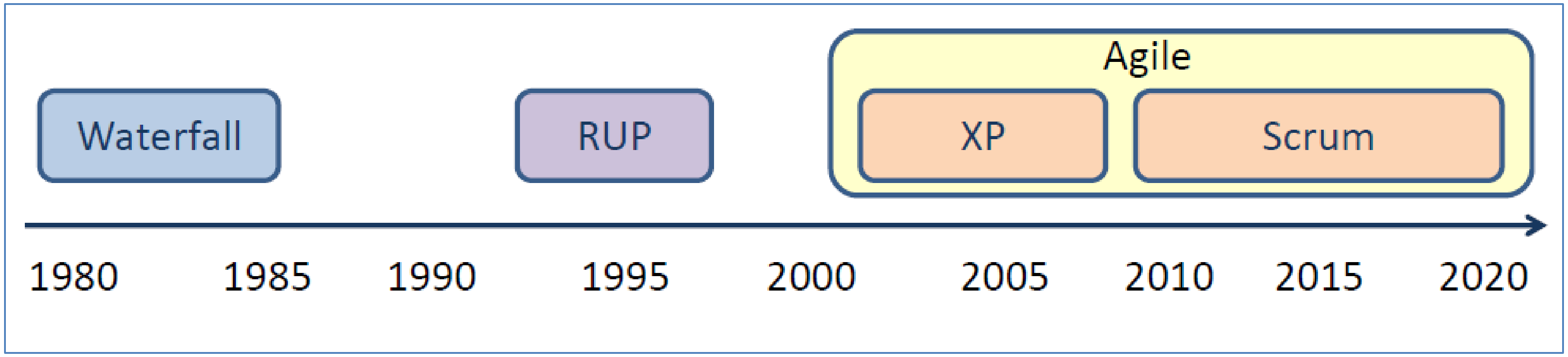


Evolution of the software development methodologies



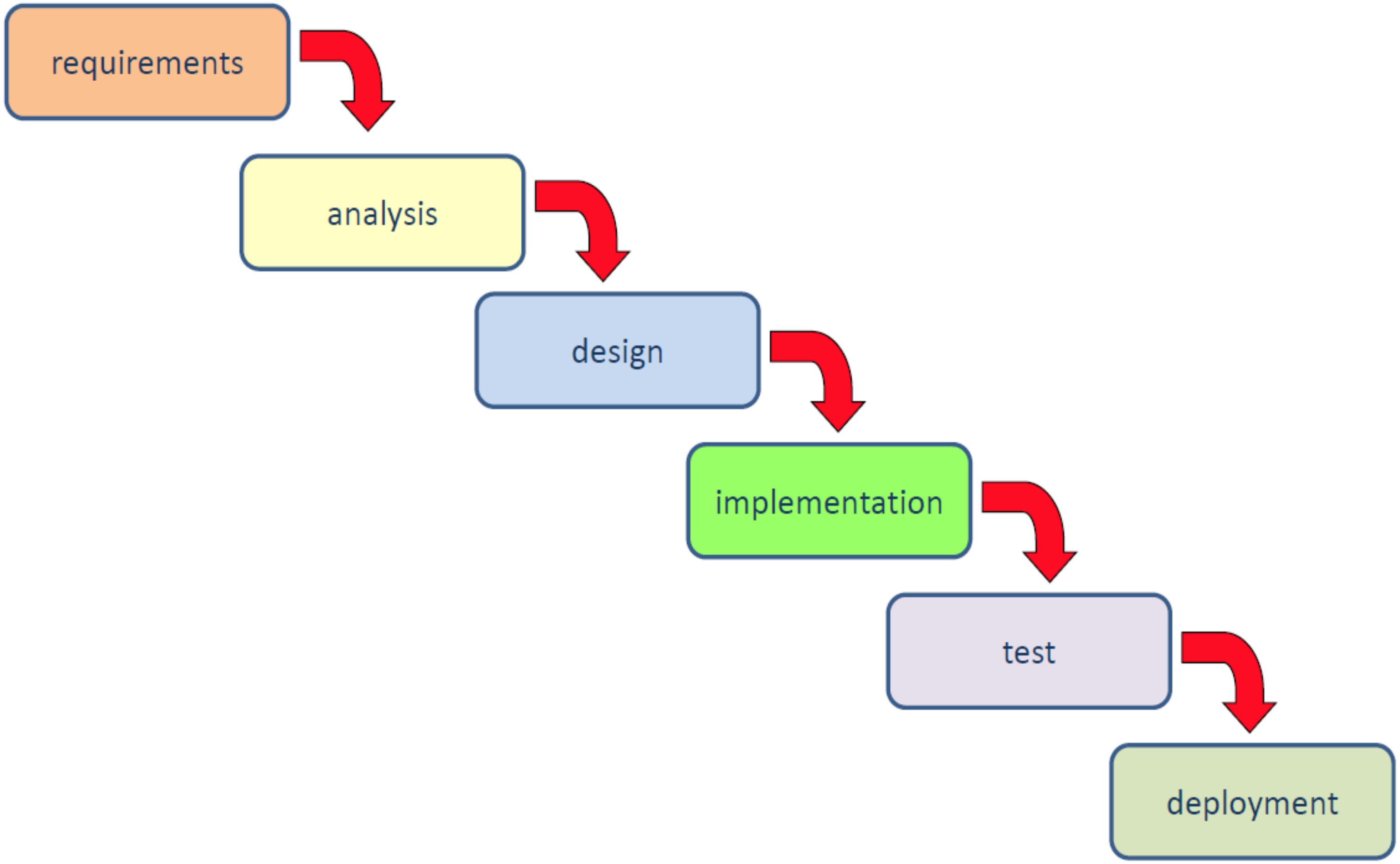
Waterfall methodology

Software development methodologies: Waterfall



- Linear
- Different roles
- Document driven
- Customer is outside the project
- Large projects(time, nr. of people)
- Req. statements

Waterfall methodology



Real-World Contexts

1. E-commerce Website	**Requirements**: Customers can browse products, add them to a cart, and check out.	**Analysis**: Define entities like Products, Users, and their relationships. Create use cases for adding items to a cart or processing payments.	**Design**: Define APIs for GET /products, POST /cart, and POST /checkout. Design database tables for Products, Users, and Orders.
2. Mobile Banking App	**Requirements**: Users can transfer money, check their balance, and view transaction history.	**Analysis**: Create a sequence diagram for transferring money (e.g., authentication -> select account -> confirm transaction).	**Design**: Design UI screens for transfer and balance checks. Create database schemas for Accounts and Transactions.
3. House Construction	**Requirements**: The house must have 3 bedrooms, 2 bathrooms, and a kitchen.	**Analysis**: Create a floor plan showing the layout of rooms and dimensions.	**Design**: Choose construction materials, plumbing and electrical layouts, and architectural drawings.

Real-World Contexts

1. E-commerce Website	**Implementation**: Code shopping cart and APIs, integrate payment gateway.	**Testing**: Test cart functionality, simulate user traffic	**Deployment**: Deploy to AWS, monitor traffic, enable CI/CD.
2. Mobile Banking App	**Implementation**: Build UI, secure APIs, integrate with core systems.	**Testing**: Validate transactions, test app security.	**Deployment**: Publish to app stores, monitor usage.
3. House Construction	**Implementation**: Build foundation, install plumbing and wiring	**Testing**: Inspect structure, test plumbing and wiring.	**Deployment**: Clean up, hand over keys, ensure utility setup.

Req vs Analysis vs Design

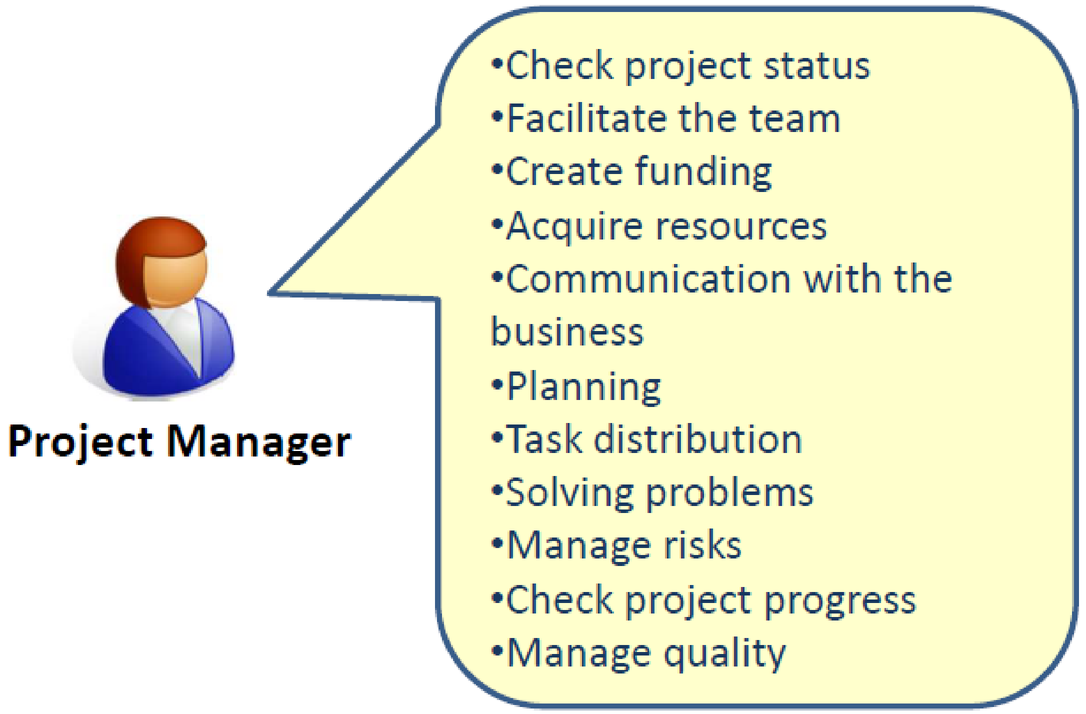
Requirements define **what** the system must do, focusing on stakeholder needs.

Analysis determines **how** to meet these requirements conceptually and ensures feasibility.

Design specifies the **technical implementation** of these plans, creating detailed blueprints for development or construction.

Core Roles

- Project Manager
- Analyst
- Developer
- Tester
- Architect



•Check project status
•Facilitate the team
•Create funding
•Acquire resources
•Communication with the business
•Planning
•Task distribution
•Solving problems
•Manage risks
•Check project progress
•Manage quality

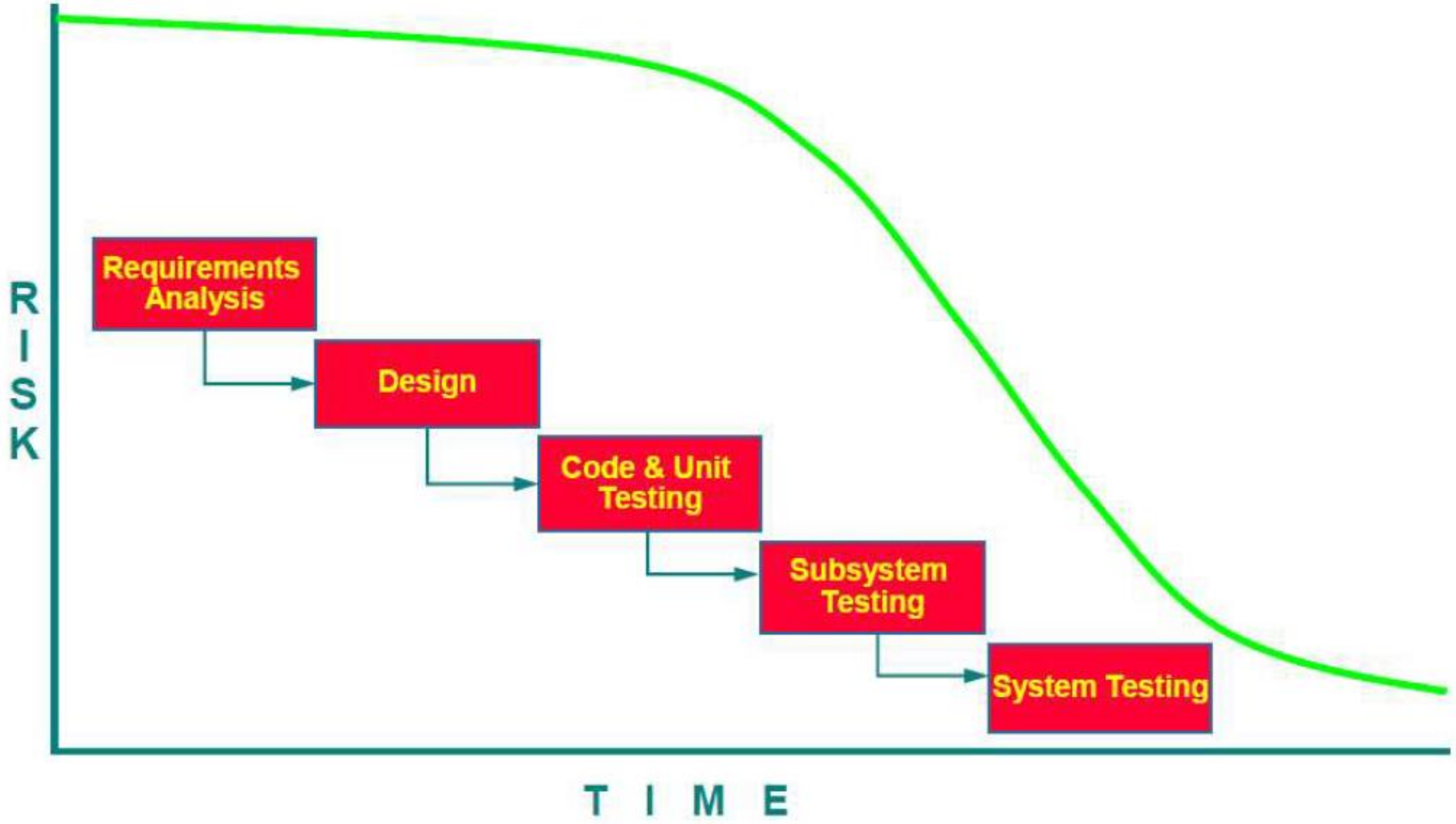
Core Artifacts

- Planning document
- Requirements document
- Architecture document
- Design document
- Code
- Test plan

Artifacts of each phase

Phase	Artifacts
Requirements	SRS (Software Requirement Specification), Use Case Diagrams, Stakeholder Requirements, Constraints Document
Analysis	Feasibility Report, DFD (Data Flow Diagram), Risk Analysis Document
Design	System Architecture, Database Schema, ERD (Entity Relationship Diagram), Component Diagrams, UI Mockups
Implementation	Source Code, Version Control Logs, Build Scripts, Code Documentation
Test	Test Cases, Test Plan, Bug Reports, Test Results Report
Deployment	Deployment Plan, Release Notes, Installation Scripts, Monitoring Configuration

Risk

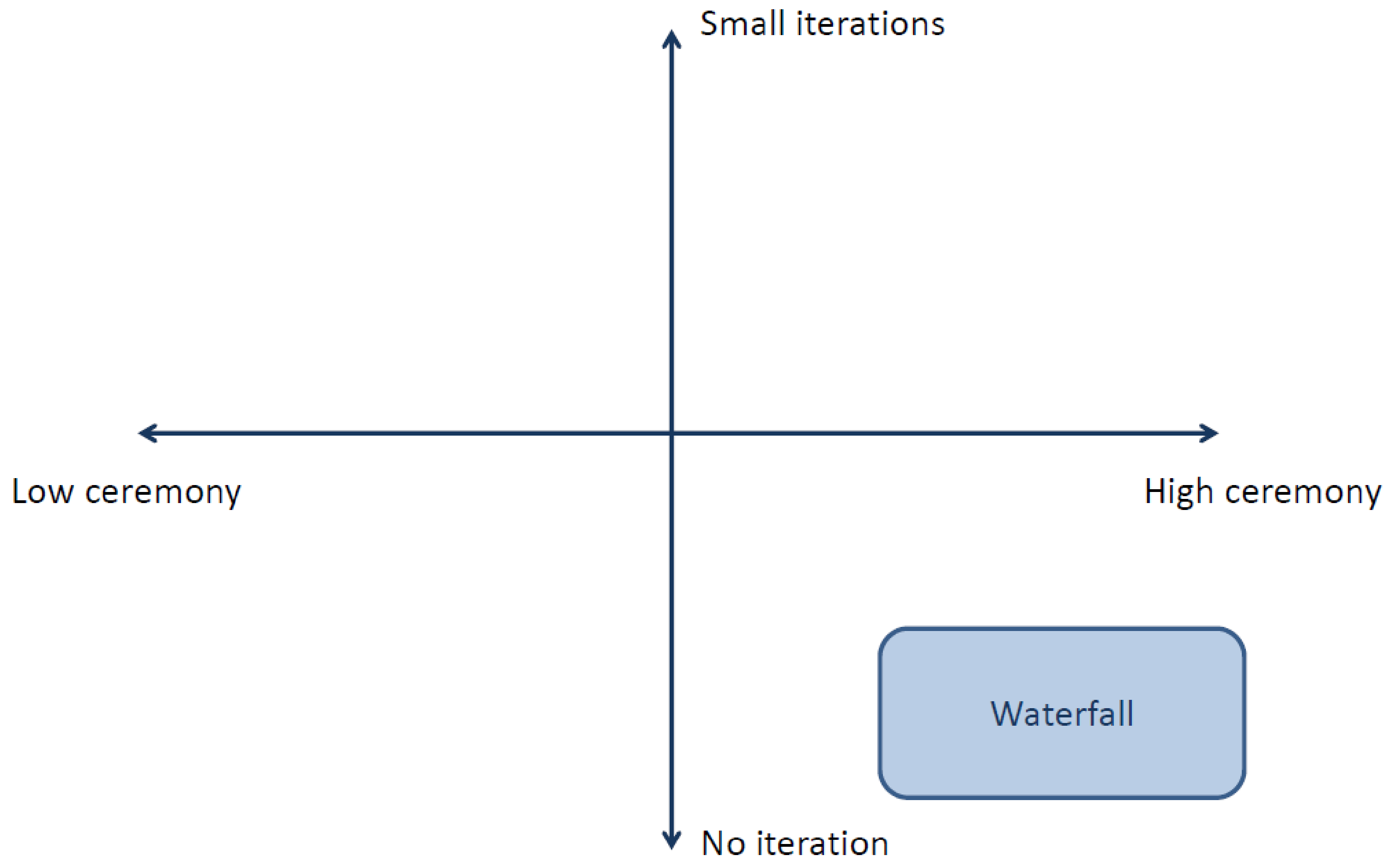

R I S K
Requirements Analysis
Design
Code & Unit Testing
Subsystem Testing
System Testing
T I M E

Characteristics of Waterfall

Document driven	The customer is involved only at the beginning of the project
Risks are found late in the project	Lot of different roles
Requirements are frozen	Software can be used only at the end of the project
Throw artefacts over the wall	Not much possibilities for reflection and improvement
Project status is not clear	No feedback
No possibilities to learn other disciplines	There is no time left for testing

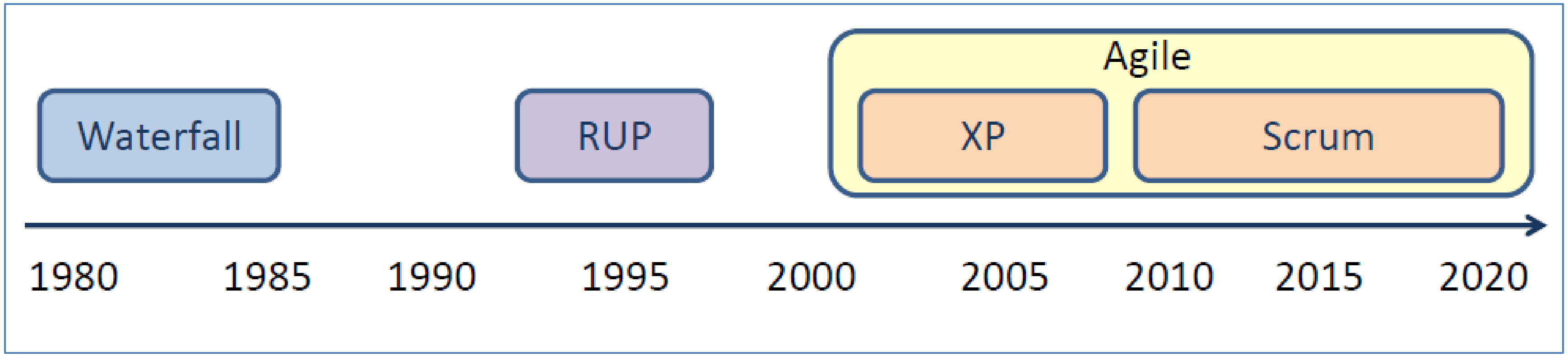
- Waterfall is highly risky, inefficient and static
- It works for the Project Manager but not for the dev team

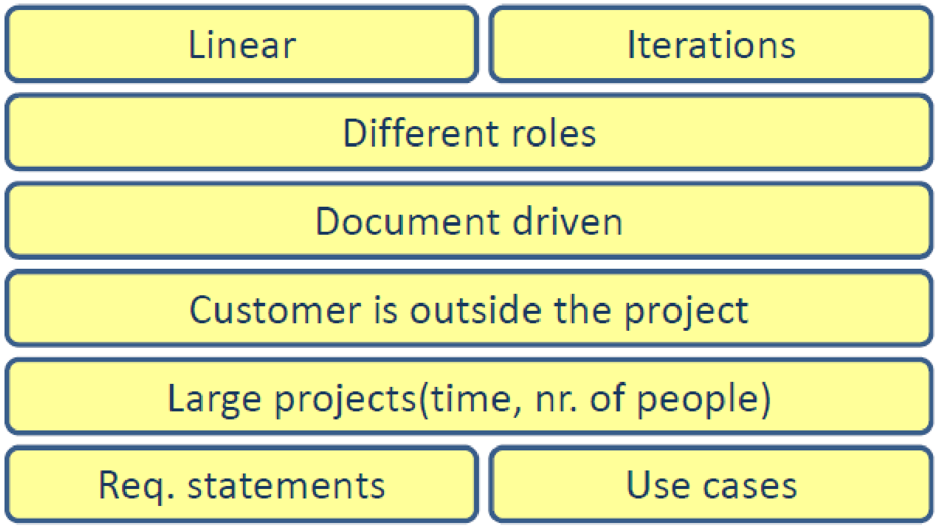
Software development methods



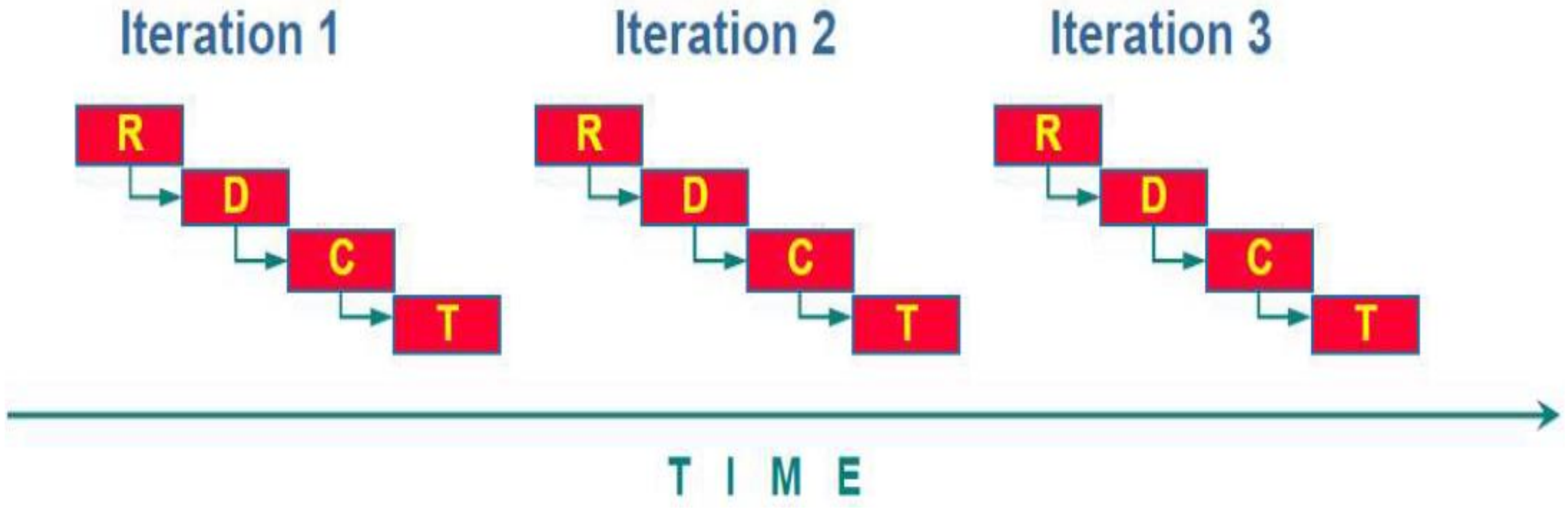
Rational Unified Process (RUP) methodology

Software development methodologies: RUP (Rational Unified Process)

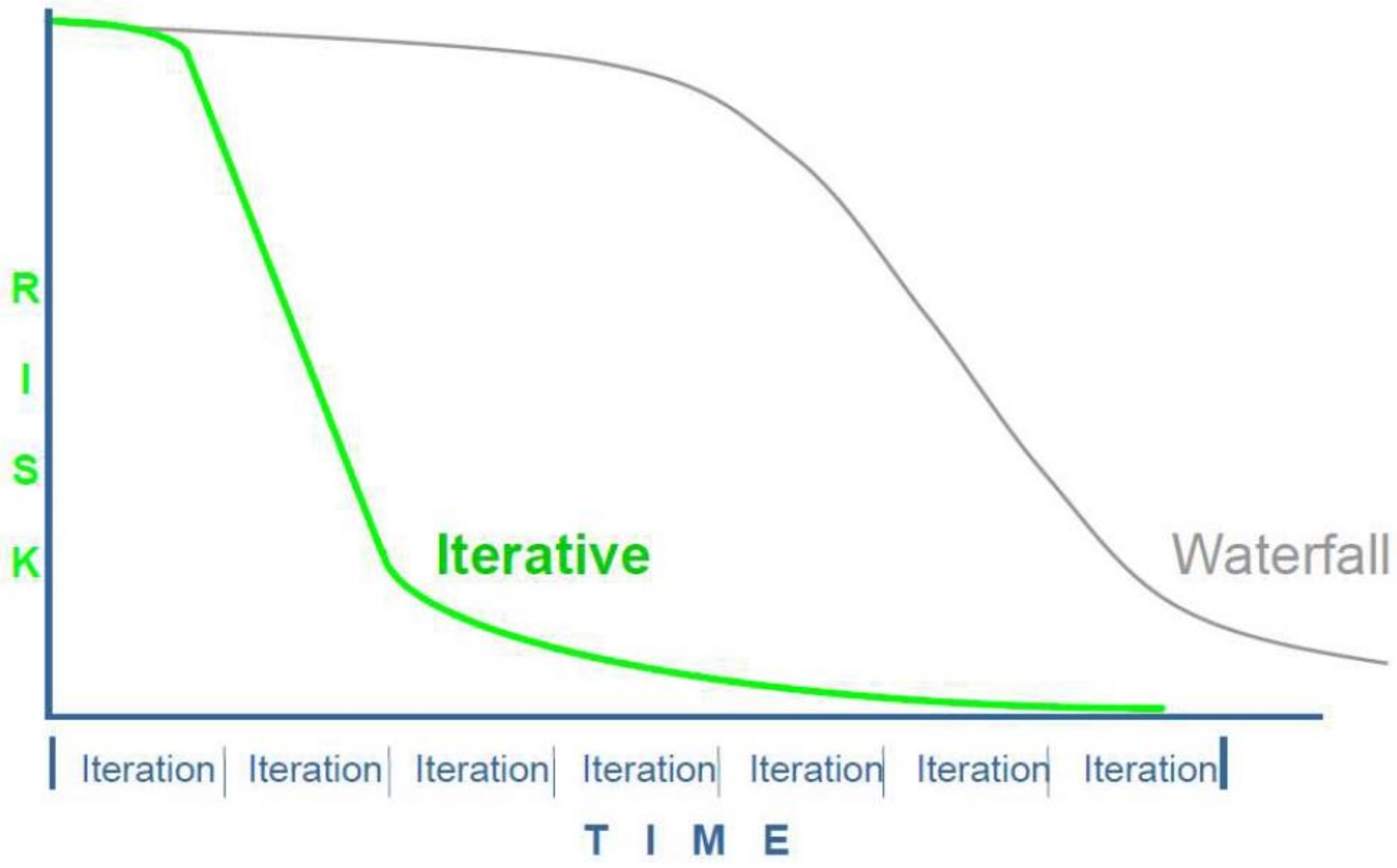




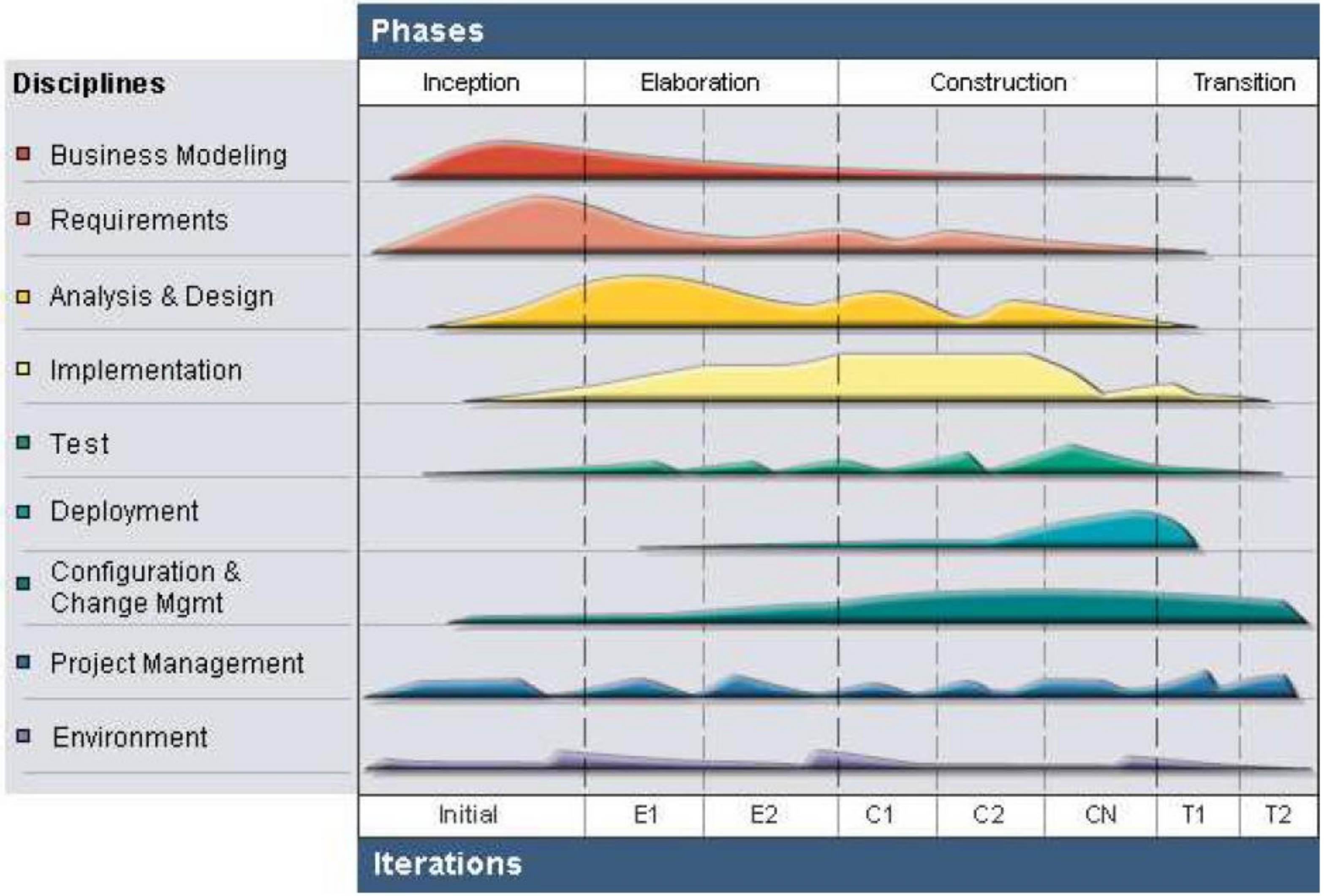
Iterations



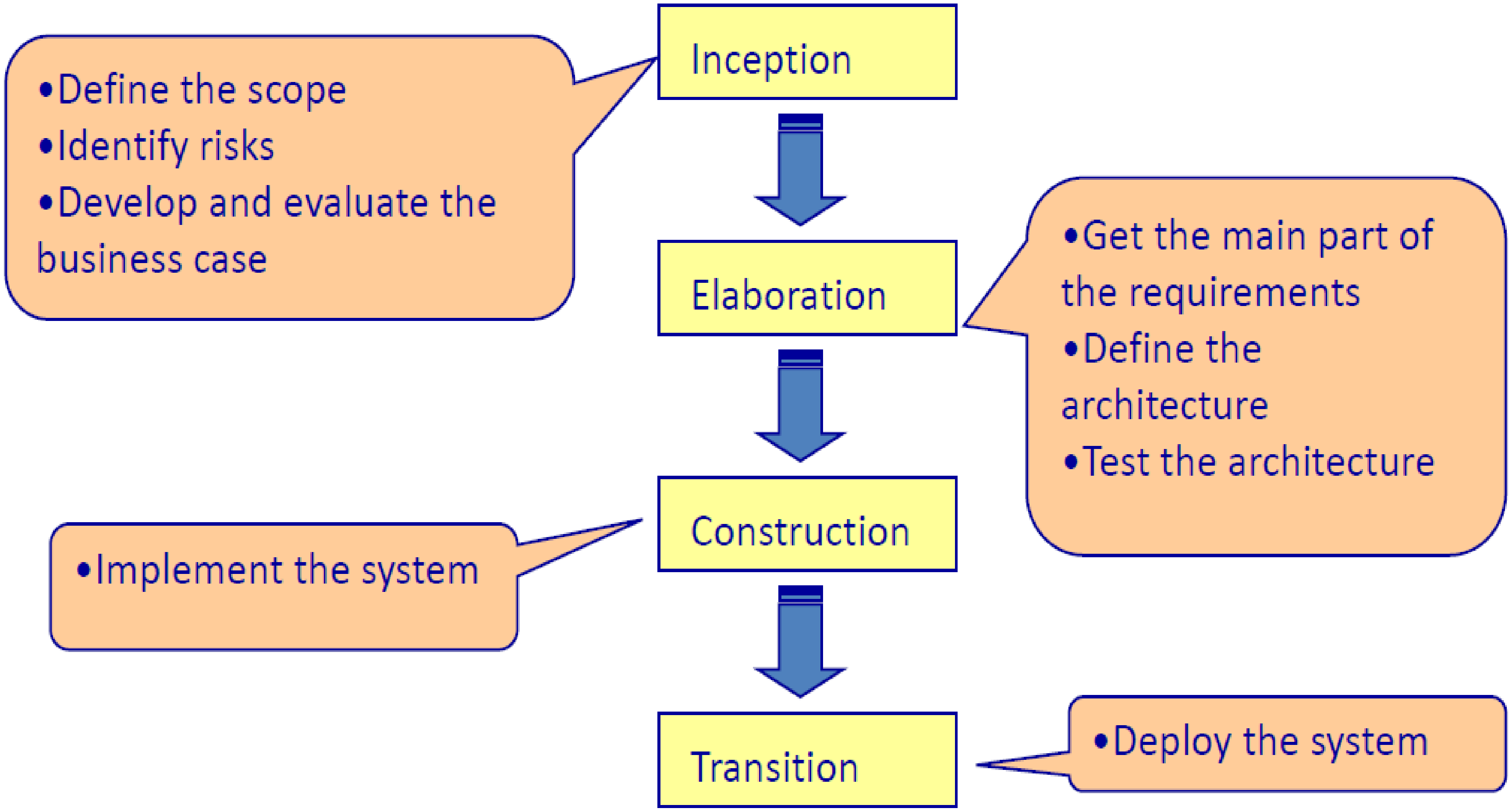
Risk


R I S K
Iterative
Waterfall
Iteration
Iteration
Iteration
Iteration
Iteration
Iteration
Iteration
T I M E

Rational Unified Process (RUP)

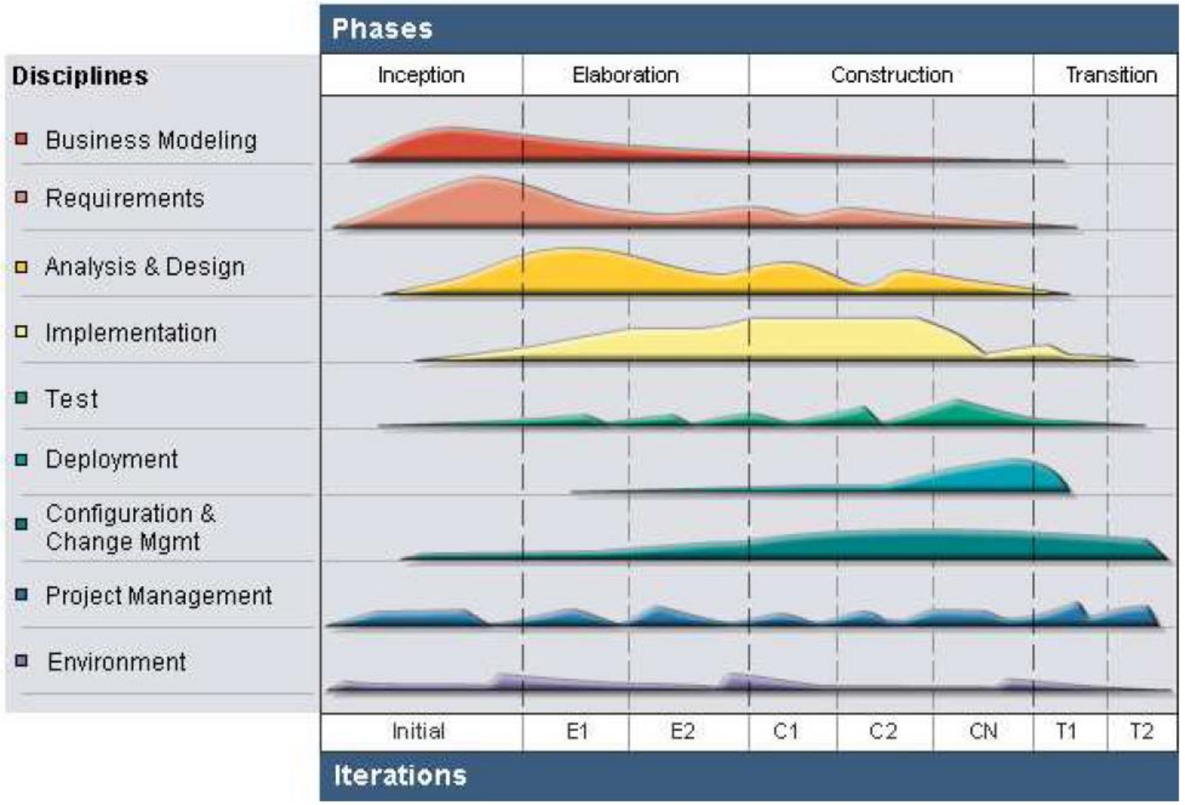


RUP Phases


Inception
Elaboration
Construction
Transition
•Define the scope
•Identify risks
•Develop and evaluate the business case
•Get the main part of the requirements
•Define the architecture
•Test the architecture
•Implement the system
•Deploy the system

RUP characteristics

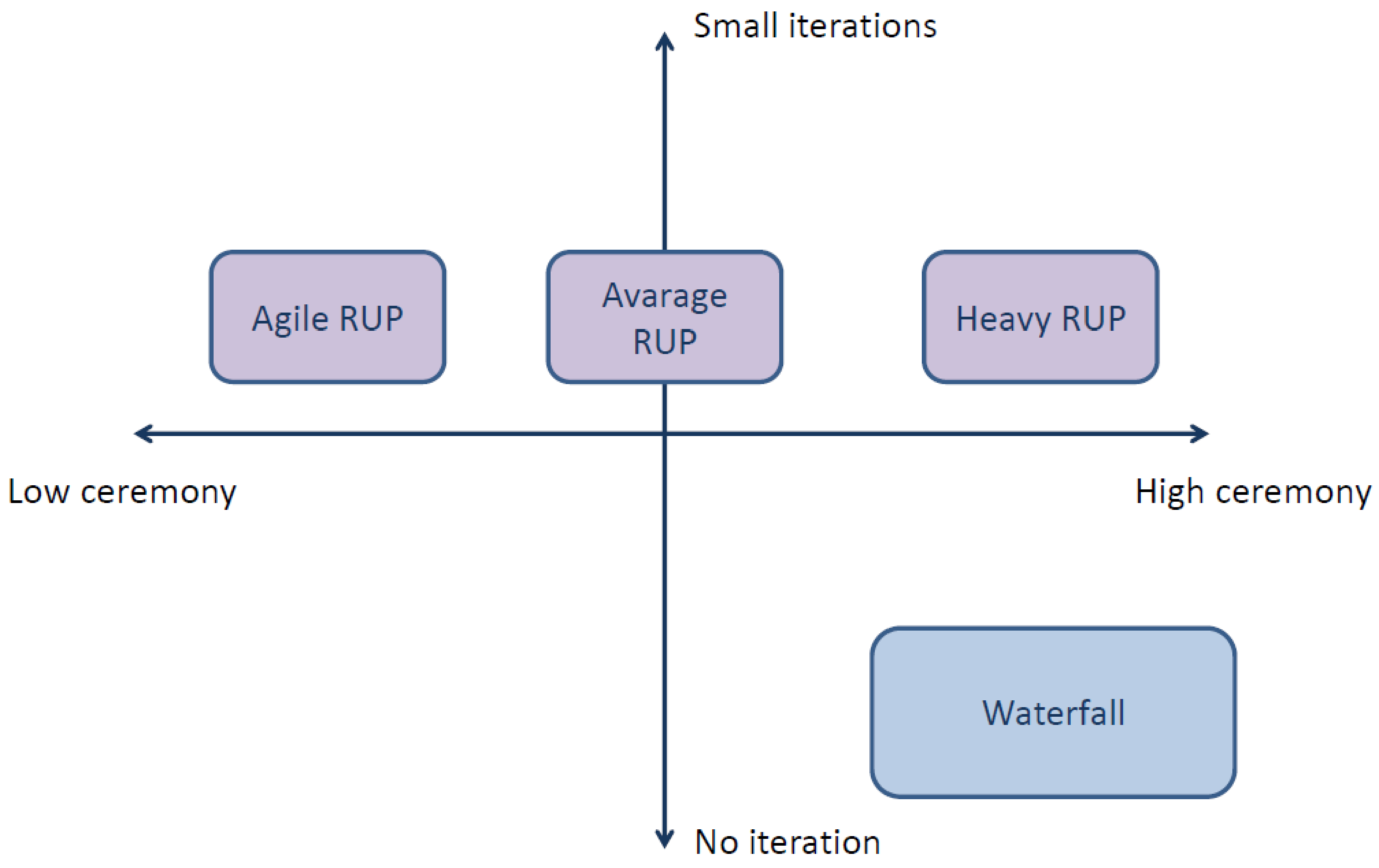
- Iterations
- Use-case driven
- Visual modeling: UML
- Architecture centric
- Test everything
- Manage changes



Roles

- Analysts
  - Business Architect
  - Business Designer
  - Business-Process Analyst
  - Requirements Specifier
  - Stakeholder
  - System Analyst
- Developers
  - Capsule Designer
  - Database Designer
  - Designer
  - Implementer
  - Integrator
  - Security Architect
  - Software Architect
  - User-Interface Designer
- General Roles
  - Review Coordinator
  - Reviewer
  - Stakeholder
  - Technical Reviewer
- Managers
  - Change Control Manager
  - Configuration Manager
  - Deployment Manager
  - Management Reviewer
  - Project Manager
  - System Administrator
  - Test Manager
- Production & Support
  - Course Developer
  - Graphic Artist
  - Process Engineer
  - System Administrator
  - Technical Writer
  - Tool Specialist
- Testers
  - Test Analyst
  - Test Designer
  - Test Manager
  - Tester

Software development methods

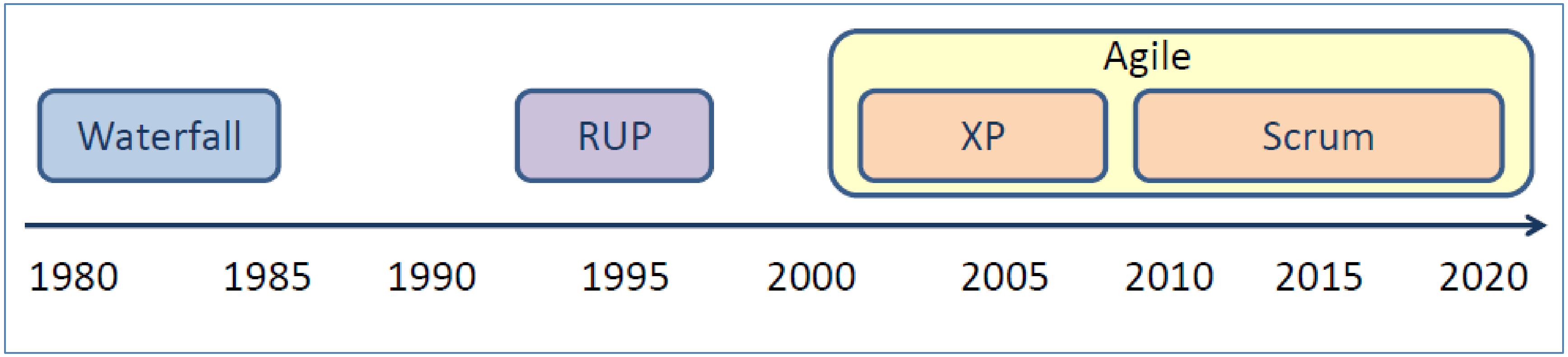


Agile Software Development methodology

What is the Agile methodology?

- Agile software development is a set of methods that results in fast and frequent delivery of value to the customer.
- It promotes well-planned, small iterations by highly collaborative, cross-functional team.
- Agile methods provide a better alternative to the linear/sequential development and long release cycles associated with the Waterfall approach.

Software development methodologies: Agile



Linear	Iterations	
Different roles		Cross-functional team
Document driven		Face-to-face
Customer is outside the project		Customer inside project
Large projects(time, nr. of people)		Small projects
Req. statements	Use cases	User stories

The 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

Agile principles

- Early and continuous delivery of valuable software.
- Welcome changing requirements.
- Business people and developers must work together daily.
- Give the team the environment and support they need, and trust them to get the job done.
- Prefer face-to-face conversation.
- Working software is the primary measure of progress.
- Continuous attention to technical excellence and good design
- Simplicity is essential.
- Self-organizing teams.

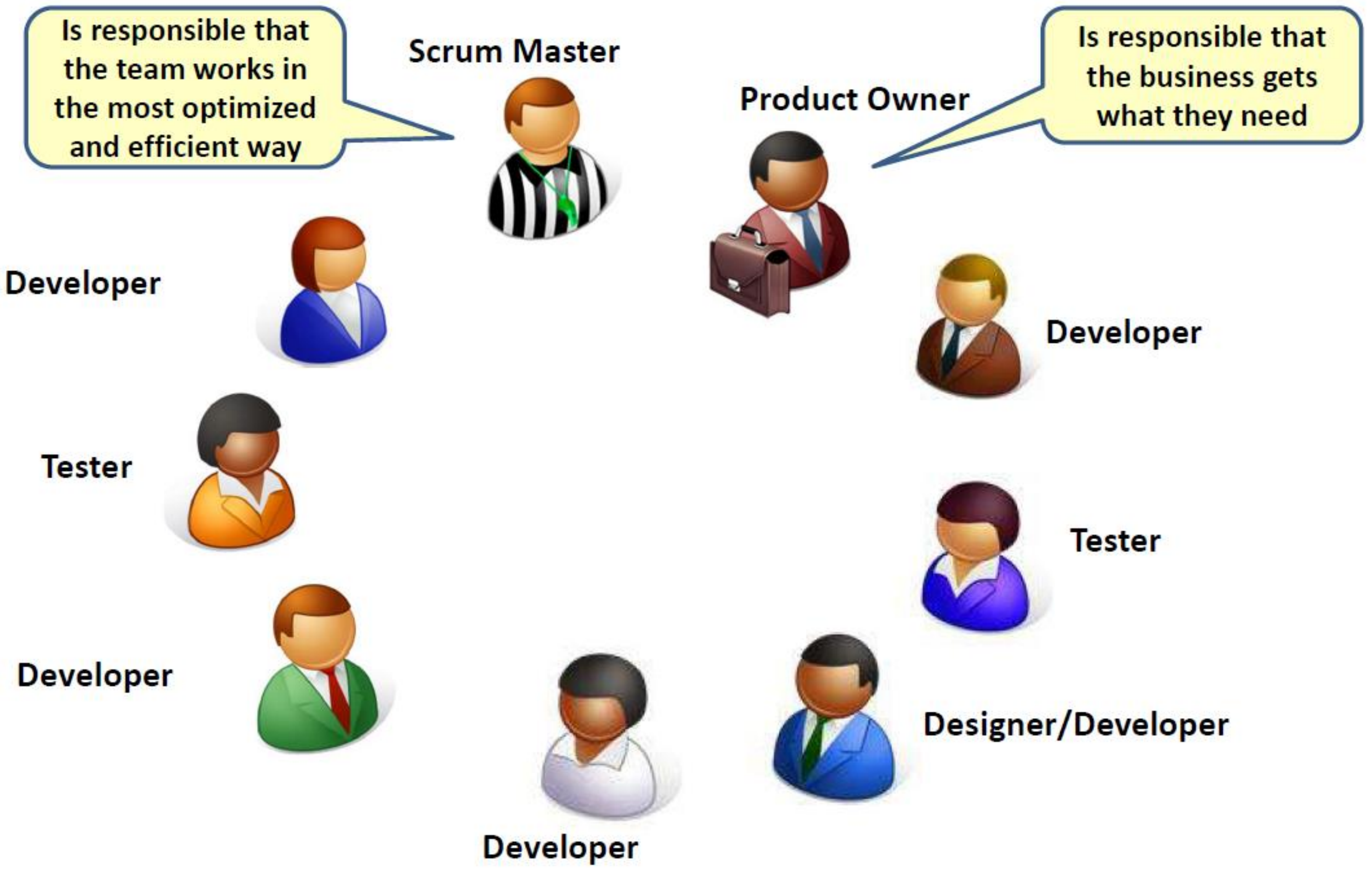
Scrum

Agile is the philosophy — the mindset. Scrum is an Agile project management framework that uses iterative cycles called sprints to deliver value by breaking large projects into smaller, manageable pieces

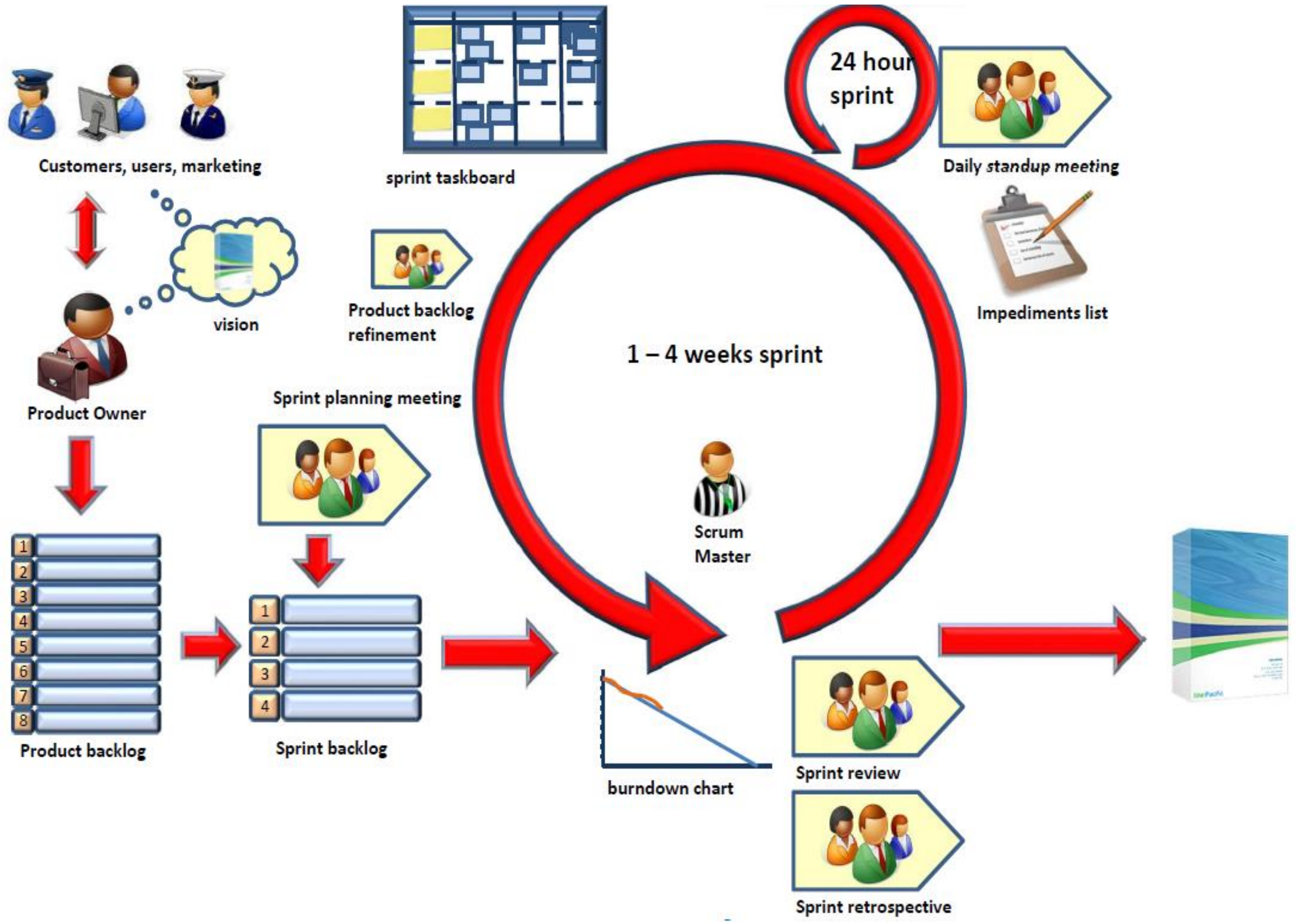
What is Scrum?

- A framework for project management
- Easy to learn
- Difficult to apply

Scrum team


Is responsible that the team works in the most optimized and efficient way
Scrum Master
Product Owner
Is responsible that the business gets what they need
Developer
Developer
Tester
Tester
Developer
Designer/Developer
Developer

Scrum in action



Product Owner

- Is responsible to ensure that the client/business gets what they need:
  - Discovers and defines the product features (requirements)
  - Updates these features and their priorities in every iteration (sprint)
  - Communicates these features to the team
  - Accepts the result created by the team

Scrum Master

- Is responsible hat the team works in the most optimized way:
  - Improves the team's productivity
    - Protects the team from disruptions
    - Facilitates team meetings
  - Creates the team (acting like a sports coach)
  - Makes sure the Agile principles are correctly understood and implemented
  - Removes impediments
  - Coaches the whole team
- Is NOT a project manager

Self-managing Team

- The team decides (not the manager)
- The team is responsible (not the manager)
- The team aims to improve continuously

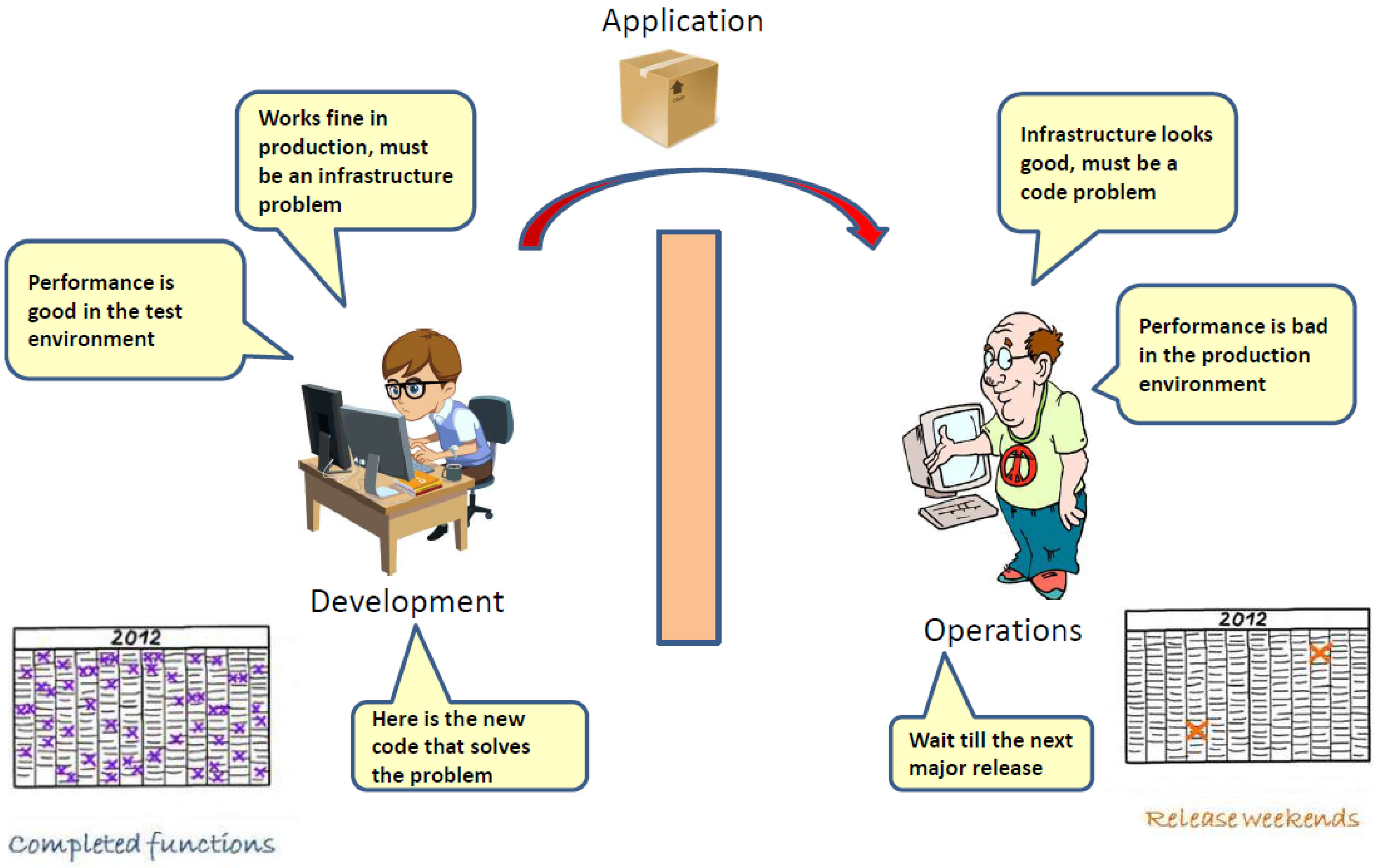
HEY PIG, I WAS THINKIN' WE SHOULD OPEN A RESTAURANT.
I DON'T KNOW. WHAT WOULD WE CALL IT?
Ham -n- Eggs
HOW ABOUT "HAM -N- EGGS."
NO THANKS. I'D BE COMMITTED, BUT YOU'D ONLY BE INVOLVED!
By Clark & Vizdos
© 2006 implementingscrum.com

Advantages of Agile/Scrum

- Increase productivity
- Improve project visibility
- Higher software quality
- Higher customer satisfaction
- Less risks
- Faster time-to-market
- Better alignment between IT & Business
- More enjoyable

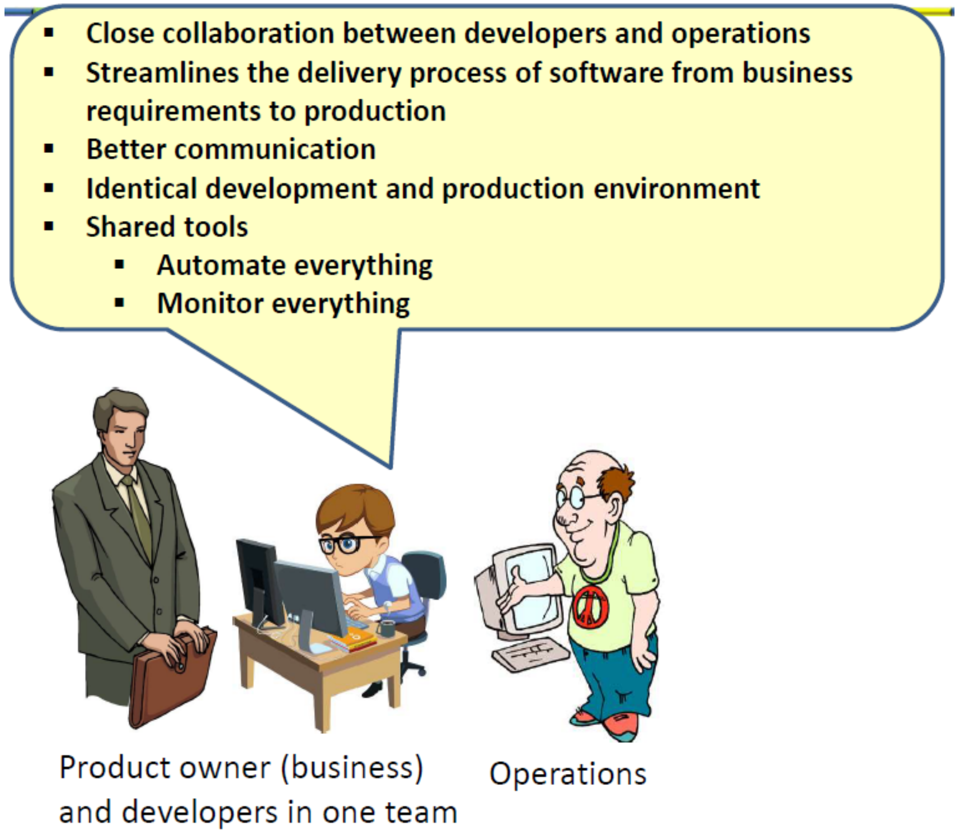
DevOps

Why DevOps?


Application
Works fine in production, must be an infrastructure problem
Performance is good in the test environment
Development
Here is the new code that solves the problem
2012
Completed functions
Infrastructure looks good, must be a code problem
Performance is bad in the production environment
Operations
Wait till the next major release
2012
Release weekends

What is DevOps?

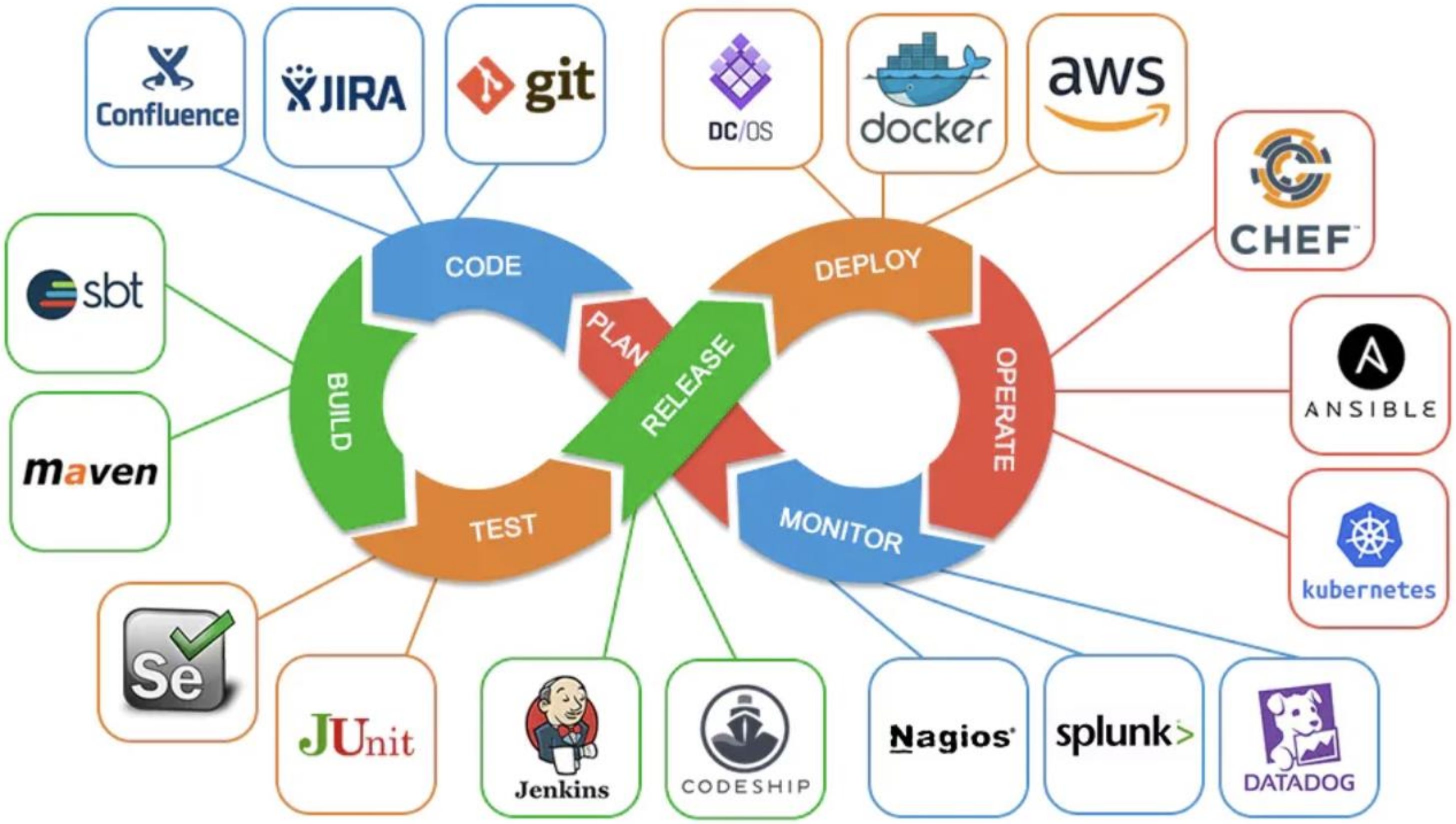
DevOps embodies a set of practices, tools, techniques and approaches aimed at speeding up the process by which software requirement goes from development to deployment in a production environment where it can provide value to the customer



- Close collaboration between developers and operations
- Streamlines the delivery process of software from business requirements to production
- Better communication
- Identical development and production environment
- Shared tools
  - Automate everything
  - Monitor everything

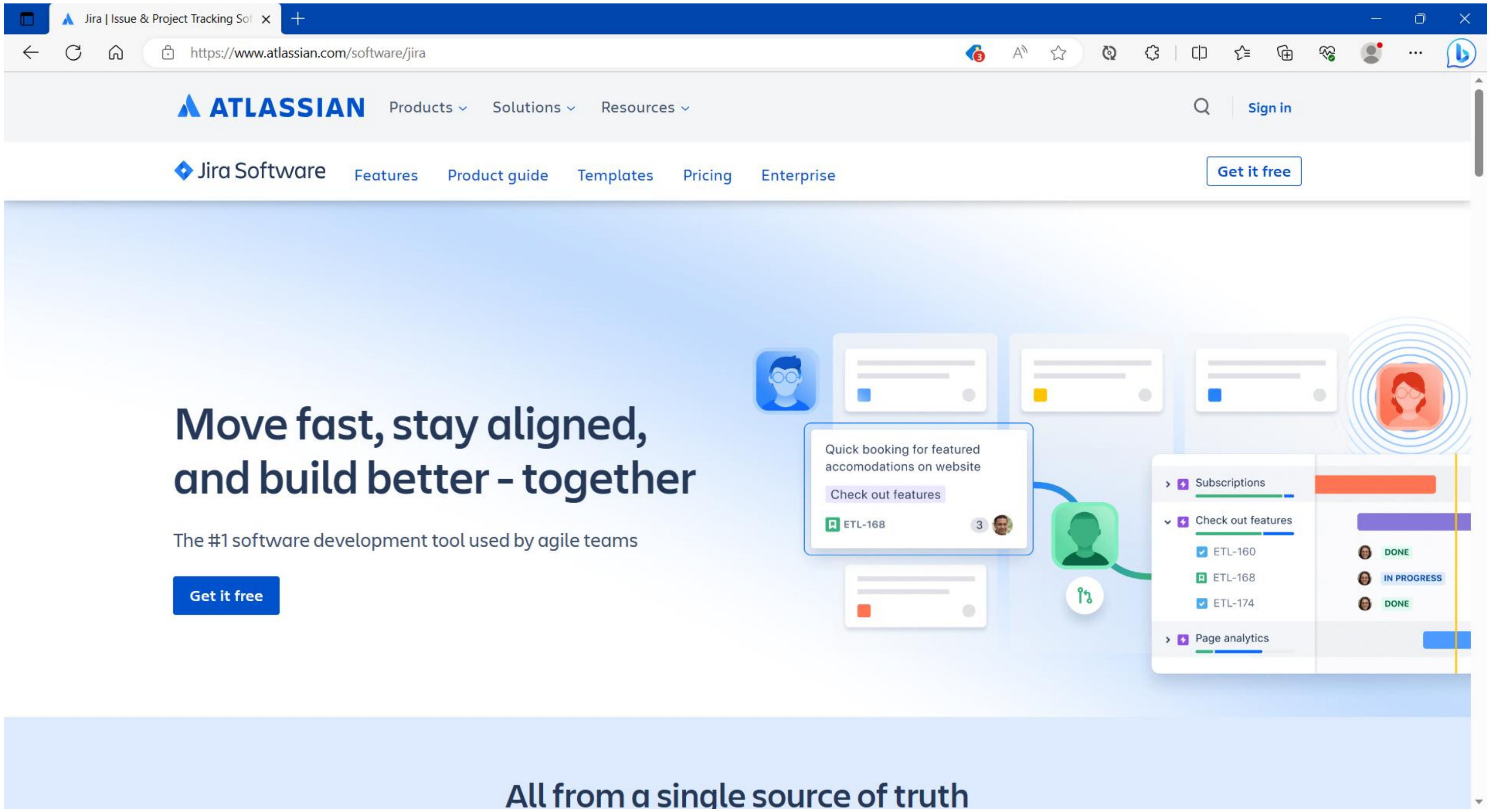
Product owner (business) and developers in one team
Operations

DevOps Lifecycle with Tools



What are DevOps Deployment Tools? (Source: Internet)

Software project management with Atlassian JIRA



Move fast, stay aligned, and build better - together

The #1 software development tool used by agile teams

Get it free

All from a single source of truth