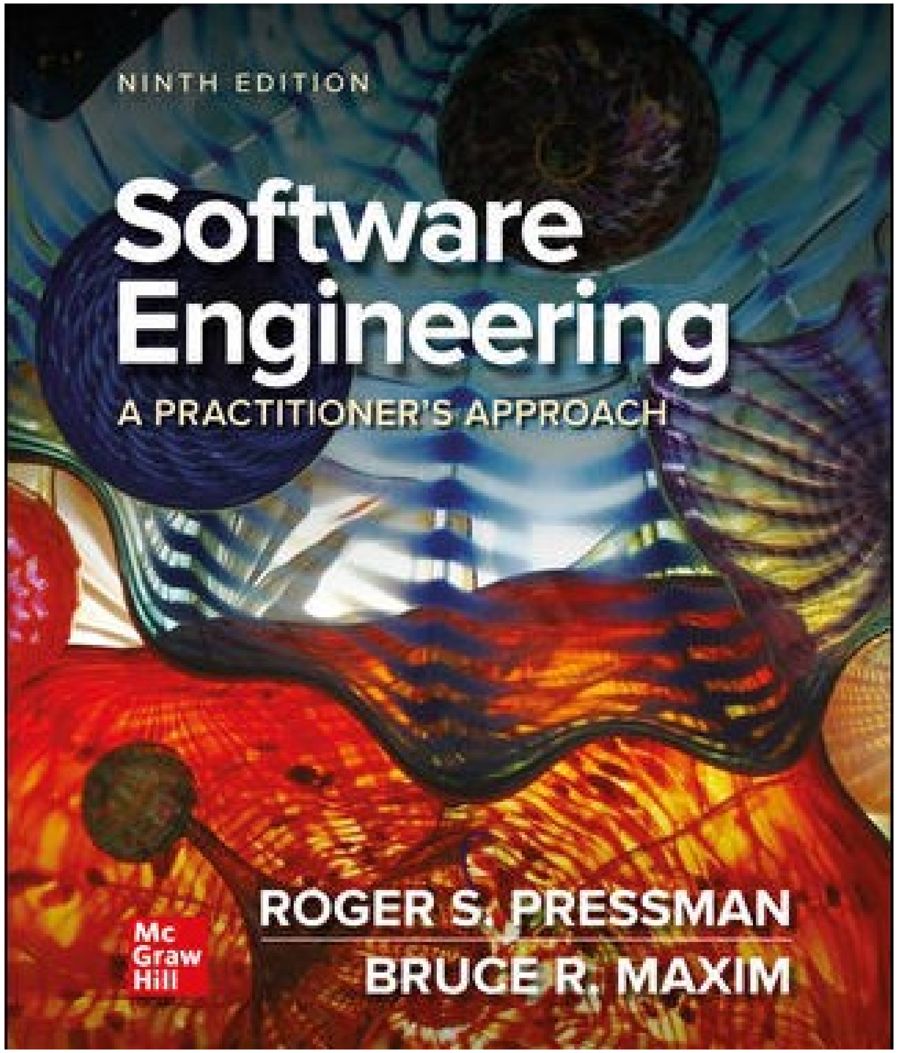
Because learning changes everything.®

Chapter 3

Agility and Process

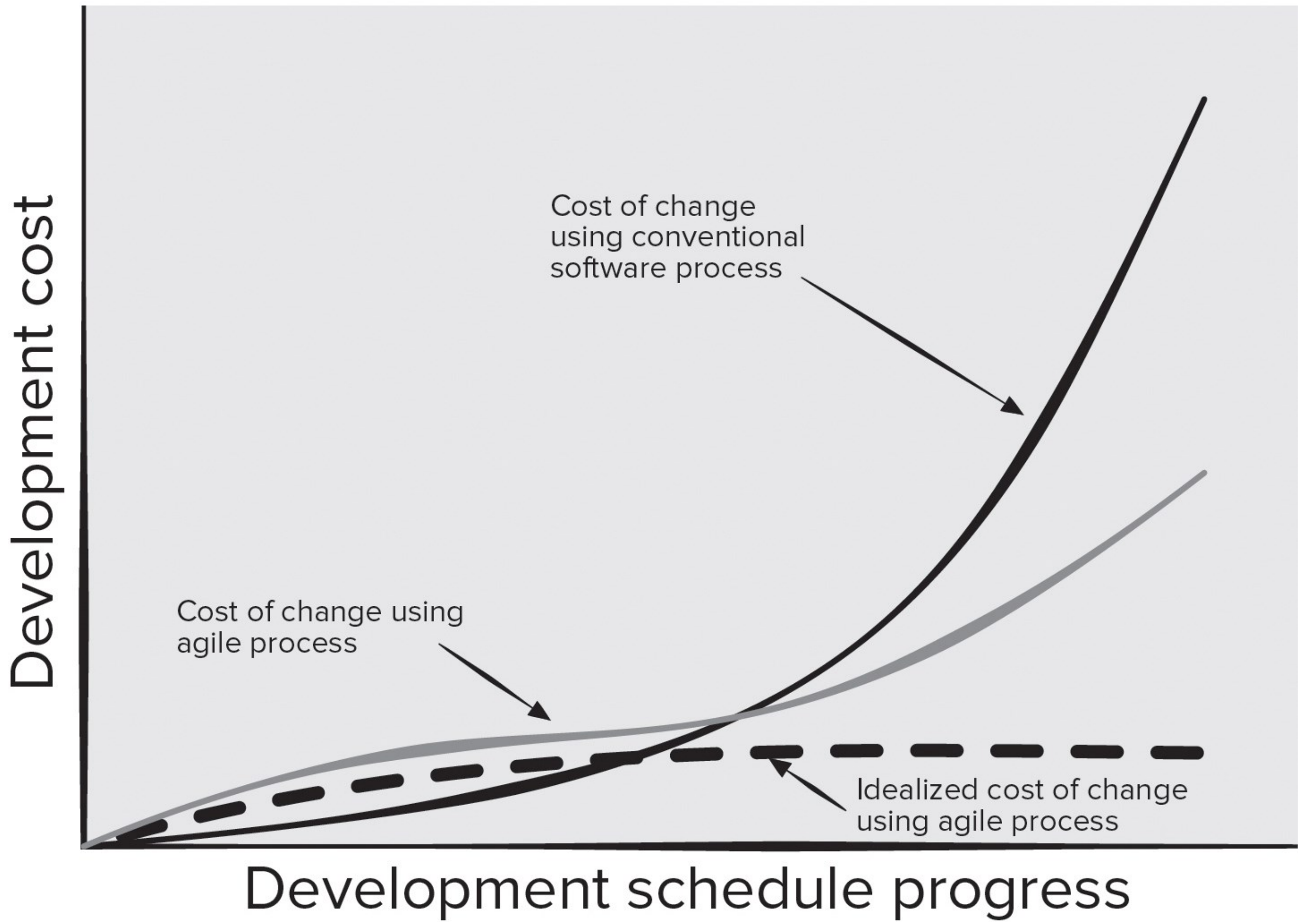
Part 1 - The Software Process



What is Agility?

- Effective (rapid and adaptive) response to change.
- Effective communication among all stakeholders.
- Drawing the customer onto the team.
- Organizing a team so that it is in control of the work performed.
- Rapid, incremental delivery of software.

Agility and Cost of Change



Access the text alternative for slide images.

What is an Agile Process?

- Driven by customer descriptions of what is required (scenarios).
- Customer feedback is frequent and acted on.
- Recognizes that plans are short-lived.
- Develops software iteratively with a heavy emphasis on construction activities.
- Delivers multiple ‘software increments’ as executable prototypes.
- Adapts as project or technical changes occur.

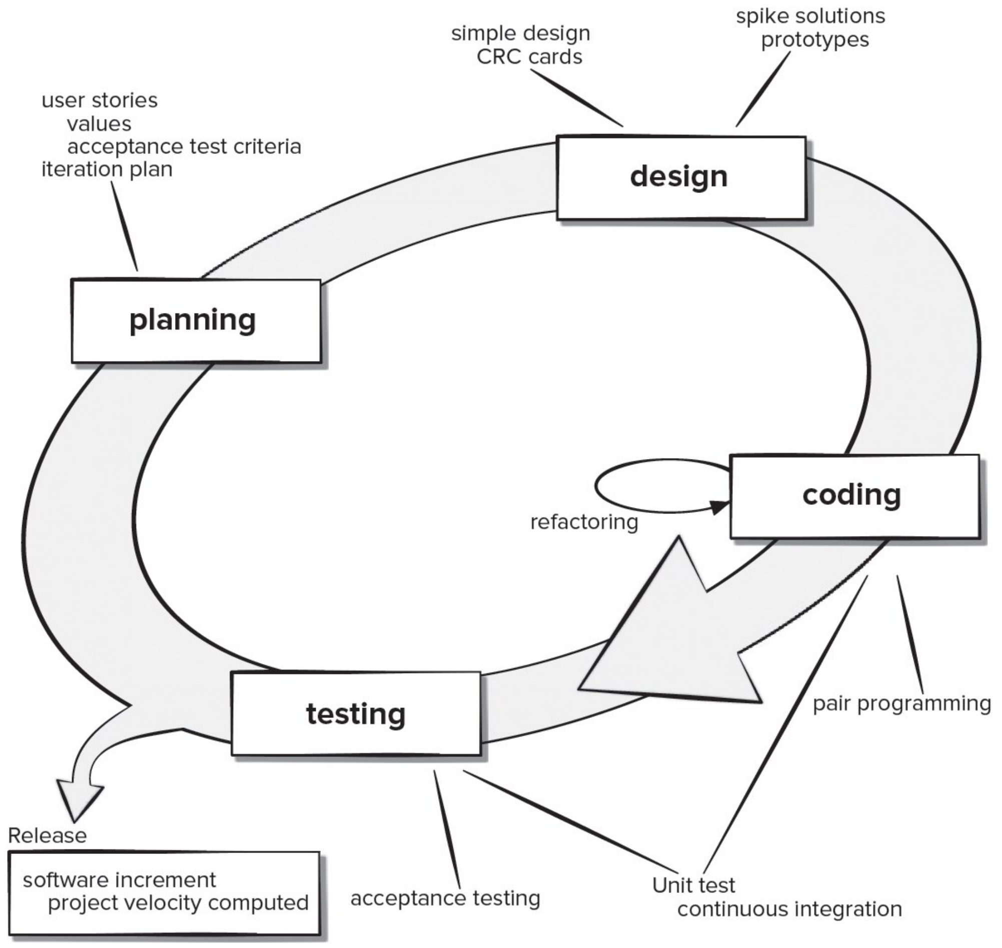
Agility Principles 1

- Customer satisfaction is achieved by providing value through software that is delivered to the customer as rapidly as possible.
- Develop recognize that requirements will change and welcome changes.
- Deliver software increments frequently (weeks not months) to stakeholders to ensure feedback on their deliveries is meaningful.
- Agile team populated by motivated individuals using face-to-face communication to convey information.
- The Team follows a process that encourages technical excellence, good design, simplicity, and avoids unnecessary work.

Agility Principles 2

- Working software that meets customer needs is the primary goal.
- Pace and direction of the team’s work must be “sustainable,” enabling them to work effectively for long periods of time.
- An agile team is a “self-organizing team”—one that can be trusted to develop well-structured architectures that lead to solid designs and customer satisfaction.
- Part of the team culture is to consider its work introspectively with the intent of improving how to become more effective its primary goal (customer satisfaction).

Extreme Programming (XP) Framework



Access the text alternative for slide images.

XP Framework - Planning

- The planning activity (also called the planning game) begins with a requirements activity called listening
- Listening leads to the creation of a set of “stories” that describe required output, features, and functionality for software to be built.
- Each user story is written by the customer and is placed on an index card.
- The customer assigns a value (i.e., a priority) to the story based on the overall business value of the feature or function
- Members of the XP team then assess each story and assign a cost—measured in development weeks—to it.

XP Framework - Planning

- The planning activity (also called the planning game) begins with a requirements activity called listening
- Listening leads to the creation of a set of “stories” that describe required output, features, and functionality for software to be built.
- Each user story is written by the customer and is placed on an index card.
- The customer assigns a value (i.e., a priority) to the story based on the overall business value of the feature or function
- Members of the XP team then assess each story and assign a cost—measured in development weeks—to it.

XP Framework – Planning II

- Customers and developers work together to decide how to group stories into the next release.
- Once a commitment is made for a release XP team orders the stories that will be developed in one of three ways
  - All stories will be implemented immediately (within a few weeks)
  - The stories with highest value will be moved up in the schedule and implemented first
  - The riskiest stories will be moved up in the schedule and implemented first.
- After the first project release, the XP team computes ***project velocity*** (the number of customer stories implemented during the first release). Project velocity can then be used to help estimate delivery dates and schedule for subsequent releases.

XP Framework – Design

- XP design rigorously follows the KIS (keep it simple) principle.
- The design of extra functionality (because the developer assumes it will be required later) is discouraged.
- XP encourages the use of CRC cards as an effective mechanism for thinking about the software in an object-oriented context.
- CRC cards identify and organize the object-oriented classes that are relevant to the current software increment.
- CRC cards are the only design work product produced as part of the XP process.
- If a difficult design problem is encountered as part of the design of a story, XP recommends the immediate creation of an operational prototype

XP Framework – Coding

- After user stories are developed and preliminary design work is done, the team does not move to code, but rather develops a series of unit tests that will exercise each of the stories that is to be included in the current release.
- Once the unit test has been created, the developer is better able to focus on what must be implemented to pass the test.
- A key concept during the coding activity (and one of the most talked-about aspects of XP) is *pair programming*.
- XP recommends that two people work together at one computer to create code for a story. This provides a mechanism for real-time problem solving and real-time quality assurance.
- As pair programmers complete their work, the code they develop is integrated with the work of others. This “continuous integration” strategy helps uncover compatibility and interfacing errors early.

XP Framework – Testing

- The unit tests that are created should be implemented using a framework that enables them to be automated.
- XP *acceptance tests* (customer tests), are specified by the customer and focus on overall system features and functionality that are visible and reviewable by the customer.

XP Summary

- **XP Planning** – Begins with user stories, team estimates cost, stories grouped into increments, commitment made on delivery date, computer project velocity.
- **XP Design** – Follows KIS principle, encourages use of CRC cards, design prototypes, and refactoring.
- **XP Coding** – construct unit tests before coding, uses pair.
- **XP Testing** – unit tests executed daily, acceptance tests define by customer.

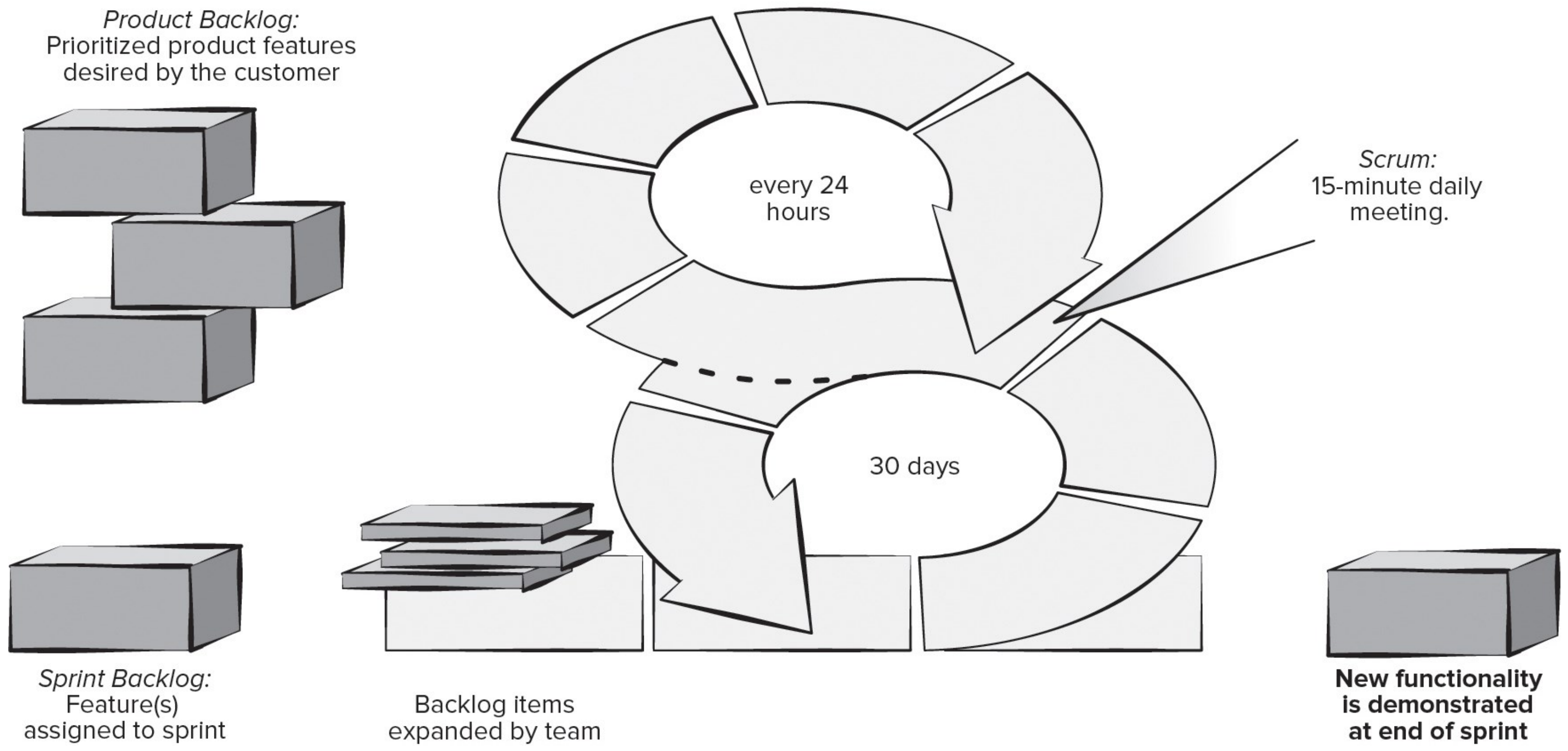
Pros

- Emphasizes customer involvement.
- Establishes rational plans and schedules.
- High developer commitment to the project.
- Reduced likelihood of product rejection.

Cons

- Temptation to “ship” a prototype.
- Requires frequent meetings about increasing costs.
- Allows for excessive changes.
- Depends on highly skilled team members.

Scrum Framework


Product Backlog: Prioritized product features desired by the customer
every 24 hours
Scrum: 15-minute daily meeting.
30 days
Sprint Backlog: Feature(s) assigned to sprint
Backlog items expanded by team
New functionality is demonstrated at end of sprint

Access the text alternative for slide images.

Scrum Details

- **Backlog Refinement Meeting** Developers work with stakeholders to create product backlog.
- **Sprint Planning Meeting** Backlog partitioned into "sprints" derived from backlog and next sprint defined.
- **Daily Scrum Meeting** Team members synchronize their activities and plan work day (15 minutes max).
- **Sprint Review** Prototype "demos" are delivered to the stakeholders for approval or rejection.
- **Sprint Retrospective** After sprint is complete, team considers what went well and what needs improvement.

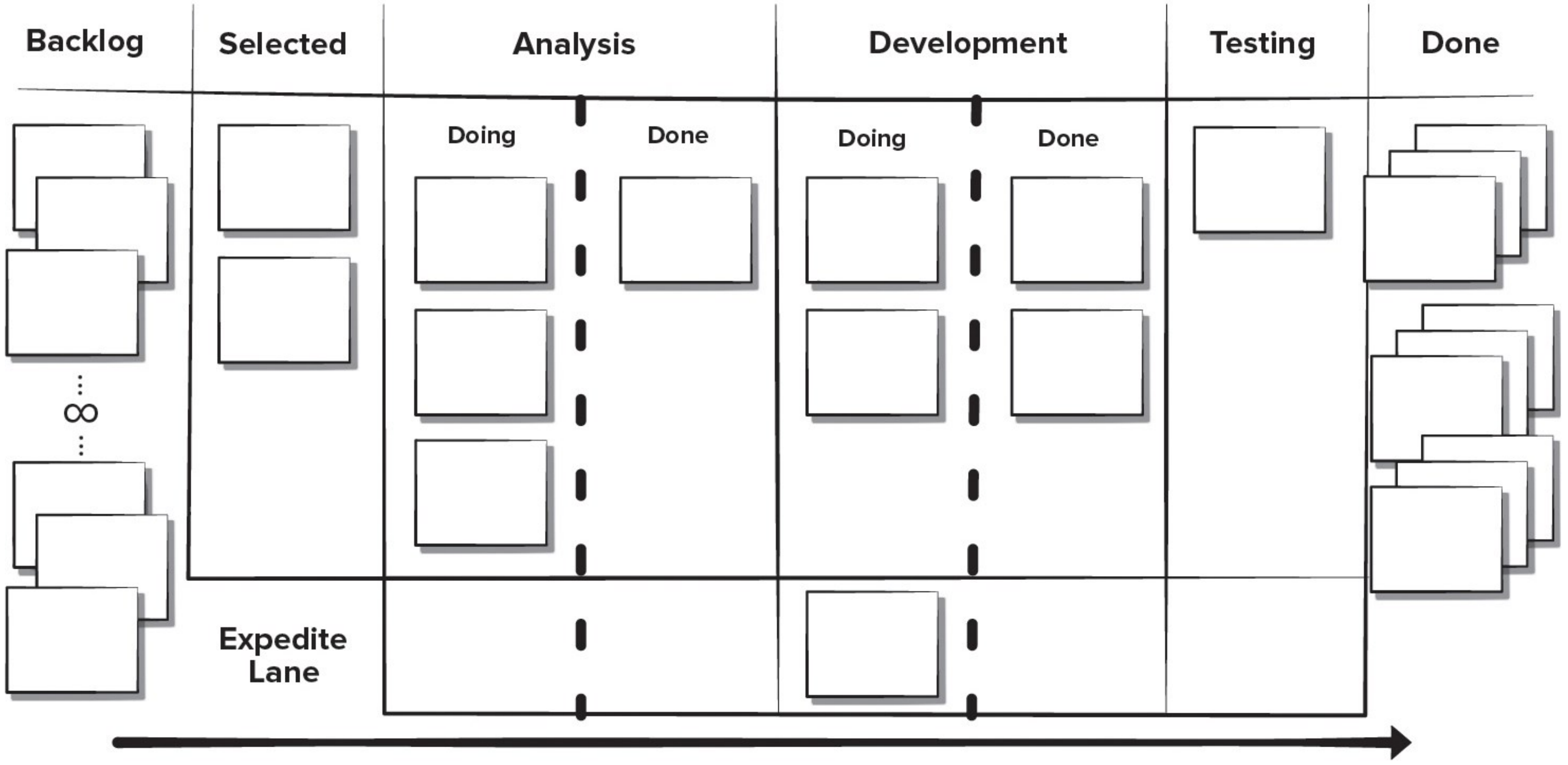
Pros

- Product owner sets priorities.
- Team owns decision making.
- Documentation is lightweight.
- Supports frequent updating.

Cons

- Difficult to control the cost of changes.
- May not be suitable for large teams.
- Requires expert team members.

Kanban Framework



Access the text alternative for slide images.

Kanban – Core Practices

- Kanban depends on six core practices:
- **Visualizing workflow using a Kanban board**. The Kanban board is organized into columns representing the development stage for each element of software functionality. The cards on the board might contain single user stories or recently discovered defects on sticky notes and the team would advance them from "to do," to "doing," and "done" as the project progresses.
- **Limiting the amount of work in progress (WIP)** at any given time. Developers are encouraged to complete their current task before starting another. This reduces lead time, improves work quality, and increases the team's ability to deliver software functionality frequently to their stakeholders

Kanban – Core Practices

- **Managing workflow to reduce waste** by understanding the current value flow, analyzing places where it is stalled, defining changes, and then implementing the changes.
- **Making process policies explicit** (e.g., write down your reasons for selecting items to work on and the criteria used to define "done").
- **Focusing on continuous improvement** by creating feedback loops where changes are introduced based on process data and the effects of the change on the process are measured after the changes are made.
- **Make process changes collaboratively** and involve all team members and other stakeholders as needed.

Kanban – Other Details

- The team meetings for Kanban are like those in the Scrum framework.
- The basis of the daily Kanban standup meeting is a task called “walking the board.” Leadership of this meeting rotates daily.
- The team members identify any items missing from the board that are being worked on and add them to the board.
- The team tries to advance any items they can to “done.” The goal is to advance the high business value items first.
- During the weekly retrospective meeting, process measurements are examined. The team considers where process improvements may be needed and proposes changes to be implemented.
- Kanban can easily be combined with other agile development practices to add a little more process discipline.

Kanban Summary

- Visualizing workflow using a Kanban board.
- Limiting the amount of *work in progress* at any given time.
- Managing workflow to reduce waste by understanding the current value flow.
- Making process policies explicit and the criteria used to define "done".
- Focusing on continuous improvement by creating feedback loops where changes are introduced.
- Make process changes collaboratively and involve all stakeholders as needed.

Pros

- Lower budget and time requirements.
- Allows early product delivery.
- Process policies written down.
- Continuous process improvement.

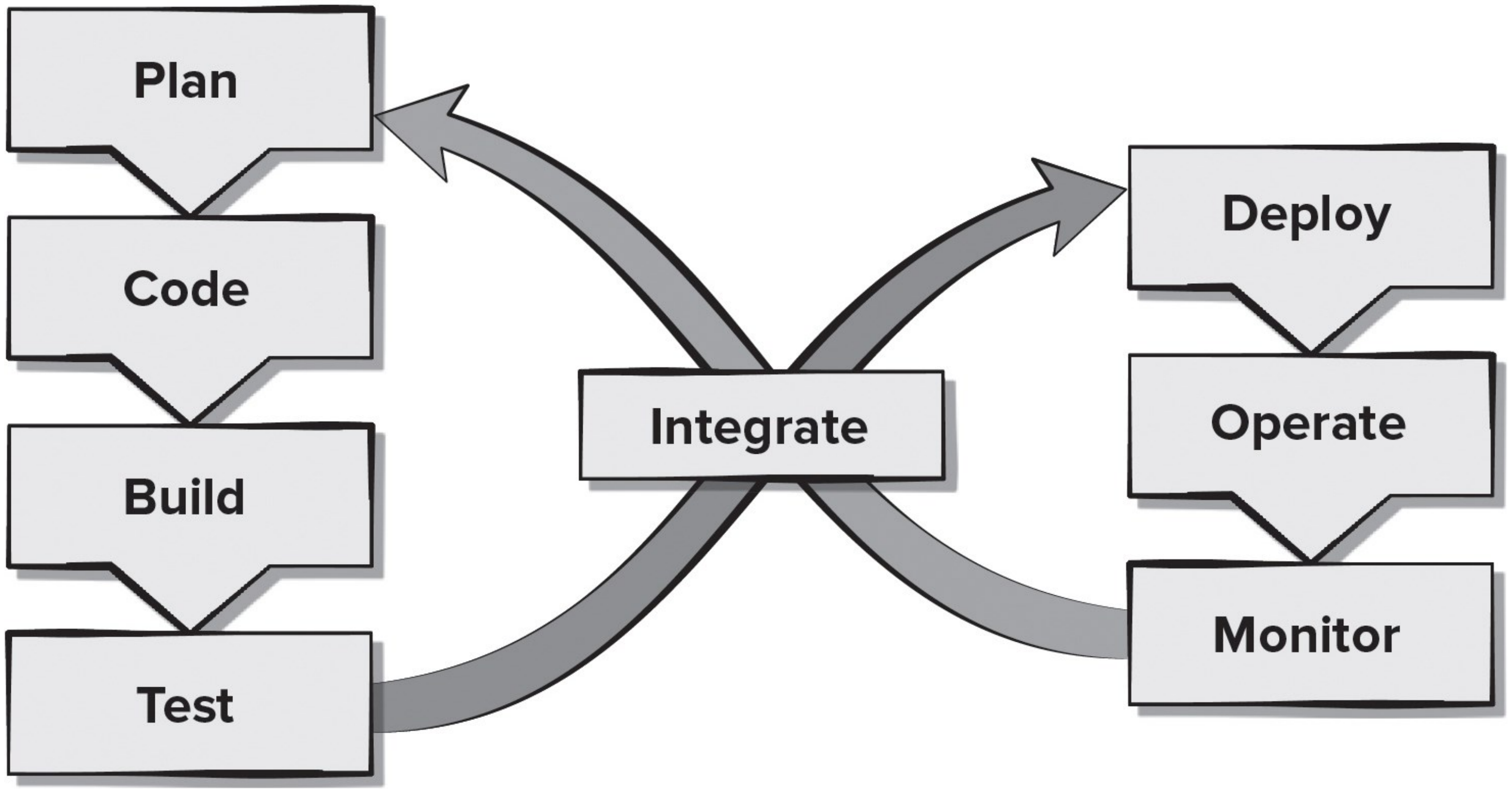
Cons

- Team collaboration skills determine success.
- Poor business analysis can doom the project.
- Flexibility can cause developers to lose focus.
- Developer reluctance to use measurement.

DevOps

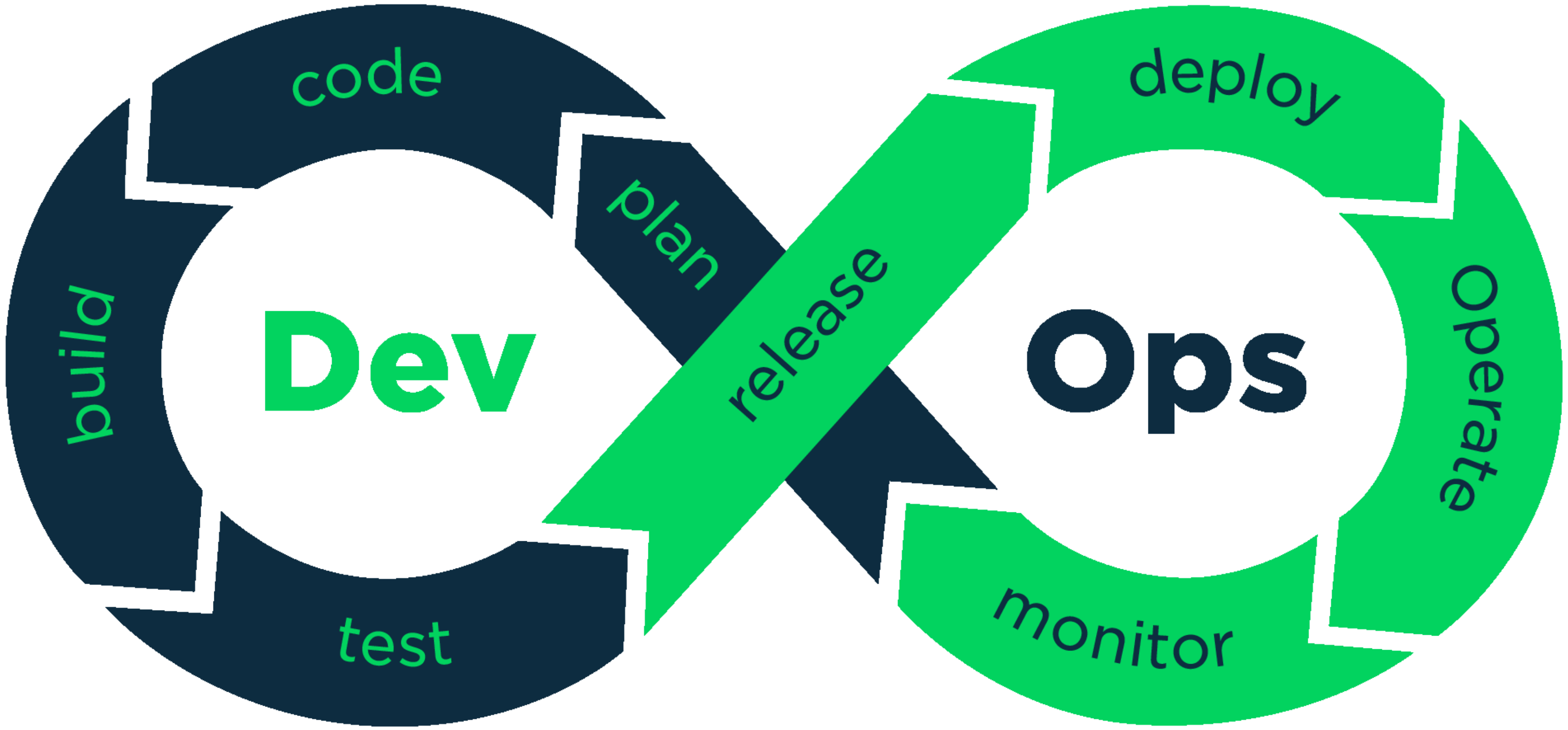
- DevOps was created by Patrick DeBois to combine Development and Operations.
- DevOps attempts to apply agile and lean development principles across the entire software supply chain.
- The DevOps approach involves several stages that loop continuously until the desired product exists

DevOps Workflow



Access the text alternative for slide images.

DevOps Workflow



DevOps

- **Continuous development**. Software deliverables are broken down and developed in multiple sprints with the increments delivered to the quality assurance members of the development team for testing
- **Continuous testing**. Automated testing tools are used to help team members test multiple code increments at the same time to ensure they are free of defects prior to integration.
- **Continuous integration**. The code pieces with new functionality are added to the existing code and to the run-time environment and then checked to ensure there are no errors after deployment.
- **Continuous deployment**. At this stage the integrated code is deployed (installed) to the production environment, which might include multiple sites globally that need to be prepared to receive the new functionality.
- **Continuous monitoring**. Operations staff who are members of the development team help to improve software quality by monitoring its performance in the production environment and proactively looking for possible problems before users find them.

DevOps Summary

- **Continuous development.** Software delivered in multiple sprints.
- **Continuous testing.** Automated testing tools used prior to integration.
- **Continuous integration.** Code pieces with new functionality added to existing code running code.
- **Continuous deployment.** Integrated code is deployed to the production environment.
- **Continuous monitoring.** Team operations staff members proactively monitor software performance in the production environment.

Pros

- Reduced time to code deployment.
- Team has developers and operations staff.
- Team has end-to-end project ownership.
- Proactive monitoring of deployed product.

Cons

- Pressure to work on both old and new code.
- Heavy reliance on automated tools to be effective.
- Deployment may affect the production environment.
- Requires an expert development team.

McGraw Hill Education

Because learning changes everything.®

www.mheducation.com