Software Process

The 2nd Most Important SWE Tool

What is a Software Process?

Software Process = Recipe for Building Software

- **Set of activities** organized to produce software
- **Defines WHO does WHAT, WHEN, and HOW**
- **Ensures quality and predictability**
- **Coordinates team efforts**

Think of it as: *A cookbook for software development teams*

Why Do We Need Software Processes?

Without Process:

- Everyone works differently
- Chaos and confusion
- Missed deadlines
- Poor quality
- Team conflicts

With Process:

- Clear expectations
- Coordinated teamwork
- Predictable results
- Better quality

The Agile Process

The Agile process is the most widely used software process, valued for its ability to deliver better, faster, and more adaptable software products

- Frequent updates ensure software products remain current and evolving.
- Improved software quality through iterative development and continuous testing.
- Rapid patching of known bugs, minimizing downtime and user impact.
- Effective incorporation of user feedback and requests, enhancing user satisfaction and product relevance.

Traditional vs. Agile: Two Different Philosophies

Traditional (Waterfall)	Agile
Plan everything upfront	Plan as you learn
Big design, then build	Small designs, build quickly
Documentation heavy	Working software focused
Sequential phases	Iterative cycles
Resist changes	Embrace changes

Traditional Waterfall: Building a Bridge

Like civil engineering:

- Plan every detail before construction
- Can't change the foundation later
- Expensive to modify once built
- Works well for predictable projects

Waterfall Phases:

1. Requirements → 2. Design → 3. Implementation → 4. Testing → 5. Deployment

Agile: Building a Startup

Like entrepreneurship:

- Start with a basic idea
- Build a minimum viable product (MVP)
- Get customer feedback
- Adapt and improve quickly
- Learn by doing

Perfect for: Software projects with changing requirements

The Agile Manifesto (2001)

Four Core Values:

1. **Individuals and interactions** over processes and tools
2. **Working software** over comprehensive documentation
3. **Customer collaboration** over contract negotiation
4. **Responding to change** over following a plan

"While there is value in the items on the right, we value the items on the left more."

Agile Principles in Practice

Customer Satisfaction:

- Deliver working software frequently (every 1-4 weeks)
- Welcome changing requirements
- Daily collaboration between business and developers

Team Empowerment:

- Face-to-face conversation
- Self-organizing teams
- Regular reflection and adaptation

Real-World Example: Netflix vs. Blockbuster

Blockbuster (Traditional):

- Planned years ahead
- Heavy investment in physical stores
- Couldn't adapt to streaming quickly
- Went bankrupt

Netflix (Agile):

- Started with DVD by mail
- Quickly pivoted to streaming
- Continuously adapted to user feedback
- Became global leader

Agile Methodologies

Method	Focus	Team Size	Duration
Scrum	Sprints & ceremonies	5-9 people	1-4 weeks
Kanban	Continuous flow	Any size	Ongoing
XP	Engineering practices	Small teams	1-2 weeks
Lean	Eliminate waste	Any size	Variable

Most Popular: Scrum (used by 75% of agile teams)

Scrum: The Most Popular Agile Framework

Scrum = Rugby strategy applied to software development

Key Idea: Small team works together in short bursts (sprints) to achieve goals

Scrum Framework Overview

Three Roles:

- **Product Owner** → What to build
- **Scrum Master** → How to work together
- **Development Team** → Build the software

Three Artifacts:

- **Product Backlog** → List of features to build
- **Sprint Backlog** → Work for current sprint
- **Product Increment** → Working software

Scrum Events (Ceremonies)

Event	Duration	Purpose	Participants
Sprint Planning	2-8 hours	Plan sprint work	Entire Scrum team
Daily Standup	15 minutes	Sync and coordinate	Development team
Sprint Review	1-4 hours	Demo completed work	Team + stakeholders
Sprint Retrospective	1-3 hours	Improve process	Entire Scrum team

Sprint: The Heart of Scrum

Sprint = Time-boxed iteration (1-4 weeks)

Sprint Goal: Complete specific features and deliver working software

Sprint Structure:

- **Week 1:** Plan and start development
- **Week 2:** Build and test features
- **Week 3:** Complete and integrate
- **Week 4:** Review, demo, and retrospective

Real Scrum Example: E-commerce Website

Sprint 1 Goal: Basic user registration and login

Sprint Planning:

- Product Owner: "We need users to create accounts."
- Team estimates: User registration (5 points), Login (3 points)
- Sprint commitment: 8 story points

Daily Standups:

- Developer A: "Working on registration form"
- Developer B: "Testing login functionality"
- Developer C: "Setting up user database"

Sprint Review:

- Demo working registration and login
- Product Owner accepts the features
- Stakeholders provide feedback

Sprint Retrospective:

- What went well: Good team communication
- What to improve: Need a better testing environment
- Action items: Set up staging server

Scrum Roles in Detail

1. Product Owner

- **Represents customers and stakeholders**
- **Defines what features to build (product backlog)**
- **Prioritizes work based on business value**
- **Accepts or rejects completed work**

Think of as: *The voice of the customer*

2. Scrum Master

- Facilitates Scrum events
- Removes obstacles (impediments)
- Coaches team on Scrum practices
- Protects team from external distractions

Think of as: *The team coach and facilitator*

3. Development Team

- Cross-functional (has all skills needed)
- Self-organizing (decides how to work)
- 5-9 people (optimal size)
- Collectively responsible for delivery

Think of as: *The builders and craftspeople*

Scrum Artifacts Explained

1. Product Backlog

This is an example; a team can choose what backlogs to make to solve the problem most efficiently.

```
Product Backlog (E-commerce Example):
1. User Registration (Priority: High, Points: 5)
2. User Login (Priority: High, Points: 3)
3. Product Catalog (Priority: High, Points: 8)
4. Shopping Cart (Priority: Medium, Points: 13)
5. Payment Processing (Priority: Medium, Points: 21)
6. Order History (Priority: Low, Points: 5)
7. Product Reviews (Priority: Low, Points: 8)
```

Maintained by: Product Owner

Format: User stories with priorities and estimates

2. Sprint Backlog

This is an example; a team can choose what should be kept in the backlog to solve the problem most efficiently.

```
Sprint 1 Backlog:
□ User Registration
  ├── Design registration form (2h)
  ├── Create user database schema (4h)
  └── Write registration tests (2h)

□ User Login
  ├── Design login form (1h)
  ├── Implement login API (4h)
  └── Write login tests (2h)
```

Owned by: Development Team

Updated: Daily during sprint

3. **Product Increment**

This is an example; a team can choose what should be delivered as a product increment to solve the problem most efficiently.

- **Working software at the end of the sprint**
- **Must be "Done" (tested, documented, deployable)**
- **Potentially shippable**
- **Cumulative (includes all previous increments)**

Goal: Customer can use the software!

Daily Standup: Problem-Solving, Not Status Reporting

"Use daily meetings to solve problems together, not show off progress"

Wrong Approach:

- Reporting what you accomplished (show off)
- Proving how hard you're working
- Individual status updates to manager
- **Result:** Boring meetings, no collaboration, stressful when no progress is made

Right Approach:

- Sharing problems and asking for help
- Collaborative problem-solving
- Team coordination and support
- **Result:** Problems solved faster, stronger team, making software development a team play game.

Standup Format: Focus on Problems

Traditional 3 Questions (Often Misused):

1. What did I do yesterday? *(Status report - boring!)*
2. What will I do today? *(Individual plan - not collaborative)*
3. What's blocking me? *(Finally useful - but often skipped)*

Problem-Focused Approach:

1. **What problems am I facing?** *(Ask for help)*
2. **How can I help others?** *(Offer assistance)*
3. **What did we learn that affects the team?** *(Share knowledge)*

Agile Advantages

1. Customer Satisfaction

- Frequent delivery of working software
- Early and continuous customer feedback
- Ability to change requirements

2. Quality Software

- Continuous testing and integration
- Regular code reviews
- Focus on working software

3. Team Benefits

- Improved communication
- Higher motivation
- Shared ownership

4. **Business Benefits**

- Faster time to market
- Reduced risk
- Better return on investment
- Competitive advantage

5. **Flexibility**

- Adapt to changing market conditions
- Pivot when needed
- Learn from failures quickly

Agile Disadvantages

1. **Requires an Experienced Team**
    - Self-organization needs mature developers
    - Less guidance than traditional methods
    - Requires strong communication skills
2. **Customer Commitment**
    - Needs active customer involvement
    - Regular feedback sessions are required
    - May not work with distant customers

3. Documentation Challenges

- Less comprehensive documentation
- Knowledge may be lost when people leave
- Harder to onboard new team members

4. Scope Creep Risk

- Flexible requirements can lead to endless changes
- Harder to estimate total project cost
- May never reach "completion"

Scrum Advantages

1. **Clear Structure**
   - Well-defined roles and responsibilities
   - Regular ceremonies provide rhythm
   - Time-boxed iterations create urgency
2. **Transparency**
   - Everyone knows what everyone else is working on
   - Progress is visible to all stakeholders
   - Problems surface quickly
3. **Adaptability**
   - Can change direction every sprint

4. **Team Empowerment**
    - Self-organizing teams
    - Collective ownership
    - Protected from external interference
5. **Risk Reduction**
    - Short iterations reduce risk
    - Regular delivery provides early feedback
    - Fail fast, learn quickly

Scrum Disadvantages

1. **Learning Curve**
    - Requires training and coaching
    - Cultural change can be difficult
    - Takes time to see benefits
2. **Requires Commitment**
    - All team members must participate
    - Product Owner must be available
    - Daily meetings are mandatory
3. **Can Be Misapplied**
    - Often implemented incorrectly
    - "ScrumBut" - Scrum but we skip ceremonies
    - May become just daily status meetings

4. **Not Suitable for All Projects**
    - Large, distributed teams struggle
    - Projects with fixed requirements
    - Safety-critical systems
5. **Potential for Burnout**
    - Constant pressure to deliver
    - Continuous change can be exhausting
    - Short sprints create urgency

When to Use Agile/Scrum

Good Fit	Poor Fit
Changing requirements	Fixed, well-defined requirements
Innovative projects	Routine, predictable work
Small to medium teams	Very large teams (100+ people)
Customer available	Customer not accessible
Experienced developers	Junior developers only
Web/mobile applications	Safety-critical systems

Implementing Scrum: Getting Started

Step 1: Form the Team

- Identify Product Owner
- Choose Scrum Master
- Assemble Development Team (5-9 people)

Step 2: Create Product Backlog

- Write user stories
- Prioritize by business value
- Estimate effort (story points)

Step 3: Plan First Sprint

- Set sprint goal
- Select backlog items
- Break down into tasks

Step 4: Execute Sprint

- Daily standups
- Build and test features
- Update sprint backlog

Step 5: Review and Retrospect

- Demo completed work
- Gather feedback
- Improve process

Step 6: Repeat

- Plan next sprint
- Continuous improvement

Scrum Tools and Techniques (Examples)

Tool Category	Examples	Purpose
Backlog Management	Jira, Azure DevOps	Track user stories and tasks
Collaboration	Slack, Microsoft Teams	Team communication
Code Management	Git, GitHub	Version control
CI/CD	Jenkins, GitLab CI	Automated testing and deployment

Common Scrum Anti-Patterns

1. "ScrumBut"

- "We do Scrum, but we skip retrospectives."
- "We do Scrum, but we don't have a Product Owner."

2. Sprint as Mini-Waterfall

- Week 1: Requirements, Week 2: Design, Week 3: Code, Week 4: Test

3. No Definition of Done

- Features are "done" but not tested
- No clear completion criteria

4. **Scrum Master as Project Manager**

    - Assigning tasks to team members
    - Micromanaging daily work
    - Not facilitating, but controlling

5. **No Customer Involvement**

    - Product Owner doesn't talk to customers
    - Building features without feedback
    - Assumptions drive development

Success Metrics for Agile/Scrum

Team Metrics:

- **Velocity:** Story points completed per sprint
- **Burndown:** Work remaining over time
- **Cycle Time:** Time from start to deployment

Quality Metrics:

- **Defect Rate:** Bugs found per release
- **Code Coverage:** Percentage of code tested
- **Customer Satisfaction:** User feedback scores

Business Metrics:

- **Time to Market:** Feature delivery speed
- **Return on Investment:** Business value delivered
- **Customer Retention:** User satisfaction and loyalty

Real-World Success Stories

Spotify:

- Uses modified Scrum ("Spotify Model")
- Autonomous squads and tribes
- Continuous deployment
- Result: Global music streaming leader

Amazon:

- Two-pizza teams (small, autonomous)
- Continuous delivery
- Customer obsession
- Result: Cloud computing and e-commerce giant

Key Takeaways

Software Process is Essential:

- Coordinates team efforts
- Ensures quality and predictability
- Adapts to project needs

Agile > Traditional for Most Software:

- Faster feedback and adaptation
- Better customer satisfaction
- Higher team motivation

Scrum is Practical Agile:

- Clear structure and roles

Getting Started with Agile

Start Small:

- Try 2-week sprints
- Focus on basic ceremonies
- Get the team comfortable with the process

Invest in Learning:

- Scrum Master certification
- Team training
- Continuous coaching

Measure and Improve:

- Track team velocity