alok.giri@ncit.edu.np

Introduction to Agile Planning

- Agile planning is iterative and incremental, focusing on delivering value early and often.
- It involves frequent re-prioritization and adaptive

planning as the project evolves.

- Agile planning typically includes:
    - Release Planning: High-level planning over several iterations.
    - Sprint Planning: Detailed planning for the upcoming sprint.

User Stories

- **Definition**: A user story is a brief, simple description of a feature told from the perspective of the end user.
- Purpose: Captures what a user needs and why, helping teams understand user value.

- Example: *As a user, I want to view my transaction history so that I can track my expenses.*

Standard user story template:

As a [type of user], I want [some goal] so that [some reason].

This format helps ensure clarity and alignment with user needs.

INVEST Criteria

Independent: Can be developed and delivered independently.

Negotiable: Not a contract; open to discussion. Valuable:

Delivers value to the customer.

Estimable: Can be estimated for effort.

Small: Can be completed within a sprint.

Testable: Has clear acceptance criteria to validate completion.

Acceptance Criteria

- **Definition**: Specific conditions that must be met for a user story to be considered complete.
- Format: Often written using **Given-When-Then** syntax.
  - Given [context],
  - When [action],
  - Then [expected outcome].
- Example:

- Given the user is logged in,
- When they click on 'Logout',
  - Then they should be redirected to the login page.

Real-Life Examples of User Stories

1. *As a blogger, I want to schedule posts so that I can manage publishing even when I am offline.* 2. *As a student, I want to receive email reminders for assignment deadlines so that I don't miss them.* 3. *As an admin, I want to deactivate inactive accounts so that I can maintain security and performance.*

Story Points

- **Definition**: A unit of measure to estimate the overall effort needed to implement a user story.

- Story points are **relative**, not tied to specific hours or days.
- Purpose: Helps measure team velocity and plan sprints accordingly.

Relative Sizing

- Instead of estimating in absolute time, teams compare user stories against each other.
- Helps maintain consistency and reduces estimation errors.
- Techniques include analogy-based comparison and using previously completed stories as reference.

Estimation Factors

When estimating story points, consider:

● **Complexity**: Technical difficulty involved.

● **Effort**: Amount of work required.

● **Risk**: Uncertainty or potential issues. ● **Uncertainty**: Ambiguity in scope or requirements.

Fibonacci Sequence in Estimation

● Common story point values: **1, 2, 3, 5, 8, 13, 21, etc.**

● Larger numbers reflect greater uncertainty. ● Benefits: Encourages teams to differentiate between simpler and more complex tasks.

Planning Poker

● A collaborative estimation technique where:

1. A story is presented.
2. Team members independently select a story point value.
3. All estimates are revealed simultaneously. 4. Discussions follow, especially if there are wide discrepancies.
5. Process repeats until consensus is achieved.

● Promotes team alignment and inclusive decision-making.

Release Planning

- **Definition**: Long-term planning outlining when and what features will be released.
- Involves high-level prioritization of features and estimating how many sprints are needed.
- Aligns business goals with development capacity.

Roadmap Planning

- Strategic plan outlining **product development direction over time**.
- Helps stakeholders visualize the progression of product features.
- Typically divided into **phases** or **quarters**.

Sprint Planning

- Conducted at the beginning of each sprint.
- Teams decide **which stories they will commit to delivering** based on priority and capacity.
- Sprint Backlog is created, and each task is discussed.

Release Planning vs Sprint Planning

Feature	**Release Planning**	**Sprint Planning**
Focus	Long-term goals	Short-term goals
Timeframe	Weeks or months	1-4 weeks
Participants	Product Owner, Stakeholders	Development Team

Output Product roadmap Sprint backlog

Minimum Viable Product (MVP)

- **Definition**: A version of a product with just enough features to satisfy early users and gather feedback.
- Purpose: Quickly validate assumptions and minimize development costs.
- Example: Dropbox's MVP was a video demo explaining the idea before building the actual product.

Iterative Development

Definition:

- Iterative development is an approach where the project is broken down into small parts and built through repeated cycles (iterations).
- Each iteration involves planning, designing, coding, and testing.

Benefits:

- Continuous feedback from stakeholders.
- Early detection and correction of defects.
- Adaptability to changes.
- Progressive development towards the final product.

Examples:

- Building an e-learning platform: Start with login + dashboard, then add quizzes, forums, and analytics in future iterations. ● Agile frameworks like Scrum follow iterative development with **sprints** as iterations.

2. Incremental Delivery

Definition:

- Delivering the product in **increments** or working pieces that add functionality over time.
- Each increment is a **usable** and **potentially shippable** product.

Incremental vs Iterative

Feature Iterative Incremental

Approach Repeating refinement

Focus Improvement and feedback

Example Revise UI design multiple times
Adding new functional parts

Delivery of usable functionalities

Deliver dashboard -> Profile -> settings

Benefits:

- Early value delivery to customers.
- Helps prioritize features based on user needs.
- Reduces risk and simplifies testing.

Tracking Progress

Importance:

- Ensures transparency.
- Allows teams to spot issues early.
- Helps in adapting plans based on actual progress.

Key Tools:

- **Task Boards**: Shows tasks in To Do, In Progress, Done.
- **Kanban Boards**: Visual workflow for continuous delivery.
- **Dashboards**: Central view of metrics, status,

blockers, etc.

Visual Tools

Task Boards:

- Columns: Backlog → To Do → In Progress → Review → Done.
- Great for daily stand-ups and sprint planning.

Kanban Boards:

- Focus on **Work In Progress (WIP) limits**.
- Used for continuous flow rather than time-boxed sprints.
- Excellent for maintenance/support teams.

Dashboards:

- Integrated view of metrics:
  - Velocity
  - Defect count
  - Sprint goal progress
  - Team capacity

Burndown and Burnup Charts

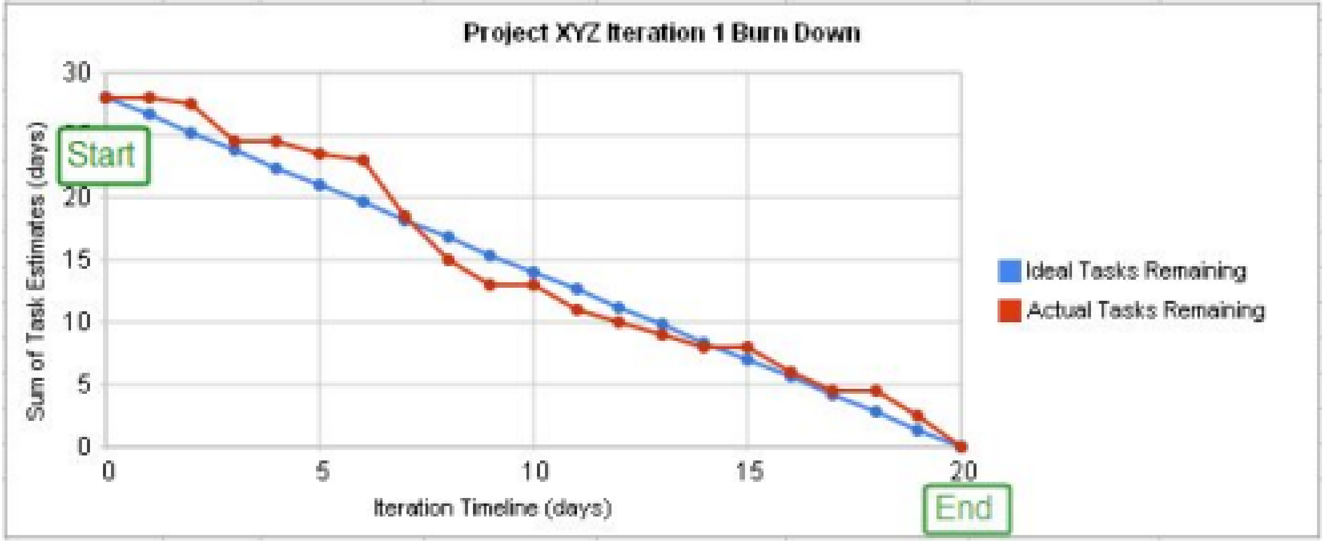
Burndown Chart:

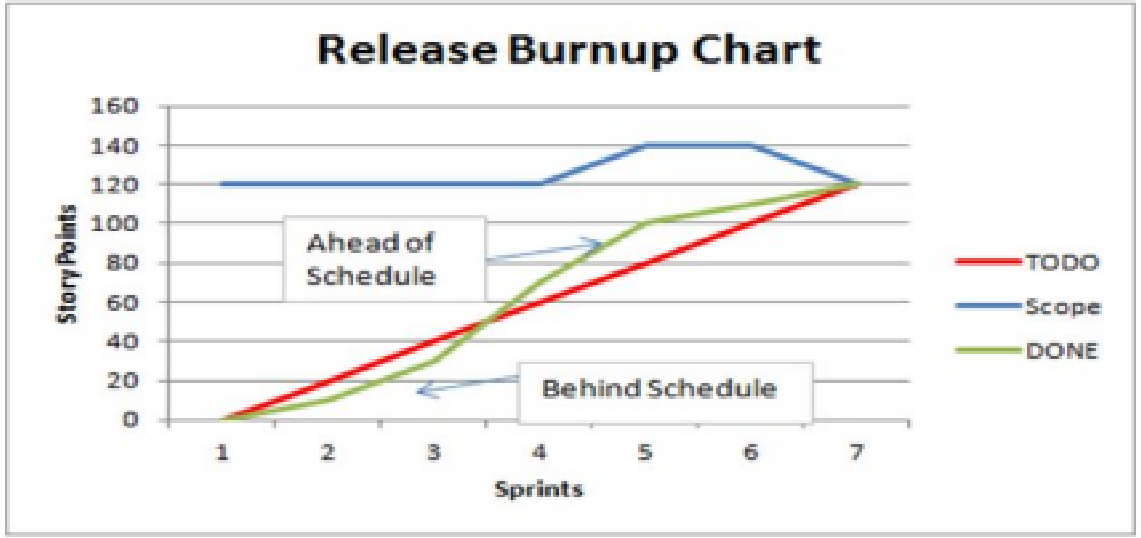
- Tracks remaining work over time.
- X-axis: Time (e.g., sprint days), Y-axis: Remaining effort/story points.

- Goal: Reach zero remaining work by end of sprint.

Burnup Chart:

- Tracks completed work over time **and** total scope.
- Two lines:
    - Work Done (grows upward)
    - Total Scope (may grow with changes)
- Highlights scope creep clearly.


Project XYZ Iteration 1 Burn Down
Sum of Task Estimates (days)
30
20
15
10
5
0
Start
0
5
10
15
20
Iteration Timeline (days)
End
Ideal Tasks Remaining
Actual Tasks Remaining



Ideal vs Actual Lines

- **Ideal Line**:
  - Represents perfect progress if work is done evenly.
- **Actual Line**:

    - Represents real progress; deviations help spot issues.
- A gap between lines shows over/underestimation, delays, or blockers.

Sprint Burndown vs Release Burndown

Type	Sprint Burndown	Release Burndown
Focus	Single sprint progress	Multiple sprints (entire release)
Use Case	Daily team sync-ups	Stakeholder reporting
Timeframe	Short (1-4 weeks)	Medium to long term (2-6

months)

When to Prefer Burnup Over Burndown

Prefer Burnup When:

- Scope is changing frequently.
- You want to show how far you've come **and** how much is left.
- Stakeholders need clear visibility on scope creep or adjustments.

Prefer Burndown When:

- Scope is relatively stable.
- You need a simple, fast view of what’s left to do.

- Used mostly in team-focused sprint tracking.

Identifying and Mitigating Risk

Risk identification in Agile is not a one-time event (like in traditional waterfall) — it happens **continuously** at every sprint, every meeting, and every conversation.

Agile teams **proactively** look for risks and build **mitigation strategies** early before the risks become issues.

Mitigation strategies include:

- Changing backlog priorities
- Adjusting sprint goals
- Improving team communication

- Enhancing technical practices (like better testing, integration)

Definition and Types of Risk

Risk: A potential problem or opportunity that may impact project outcomes — in either a positive (opportunity) or negative (threat) way.

Types:

- **Technical Risk**: Related to technology challenges (e.g., new frameworks, integration issues)
- **Business Risk**: Misalignment with user needs, market changes
- **Resource Risk**: Team availability, lack of expertise, tools shortage
- **Operational Risk**: Daily workflow problems, miscommunications
- **External Risk**: Outside changes — vendors failing, legal regulations, pandemics

Difference in Risk Handling

Aspect	Traditional Projects	Agile Projects
When risk is handled	Upfront during planning	Continuously throughout
Who manages risk	Project Manager only	Whole team collectively
Risk plan updates	Rarely updated once finalized	Updated sprint by sprint
Response to change	Slow due to formal processes	Fast and reliable

Agile doesn’t eliminate risk, it embraces it and reacts faster.

Sources of Risk in Agile Env.

Changing Requirements: As customers understand more, their expectations evolve.

Dependencies: Agile relies heavily on cooperation with other teams, tools, external parties.

Team Skills: Agile needs strong technical skills, team maturity, self-management.

Velocity Fluctuation: Teams may not always deliver the same speed due to complexity or team dynamics.

Risk Identification Methods

Sprint Planning: Discuss risks for selected backlog items

Backlog Grooming: Surface risks while refining backlog items

Daily Standups: Identify small blockers turning into bigger risks

Retrospectives: Reflect and find repeating patterns of risk

Risk Brainstorming: Special session where team discusses: “What could go wrong?”

Sprint Planning & Backlog Grooming

Sprint Planning:

- Break down backlog items.
- Discuss technical or resource risks for each item.
- Adjust scope if risk is too high.

Backlog Grooming:

- Identify risky stories (too large? too vague? external dependencies?)
- Refine and split stories to lower risk.

Retrospectives and Daily Standups

Retrospective:

- Teams ask:

    - What risks materialized last sprint?
    - What risks did we manage well?
    - What can we change to handle risks better?

Daily Standups:

- If a blocker persists, treat it as a **risk escalation**.
- Action immediately to avoid sprint failure.

Risk Brainstorming

Format: 15–30 minutes session.

Ask questions like:

- "Where do we feel uncertain?"

● "What assumptions are we making?" ● "What external factors can affect us?"

Document Risks openly and assign owners.

Agile Practices that Inherently Reduce Risk

Short Iterations and Frequent Delivery

- Frequent delivery gives early feedback.
- Mistakes or misunderstandings are caught quickly.

Continuous Integration and Testing

- Integrating code daily and testing every build finds bugs early.

- Less integration held at the end of projects.

Cross-Functional Teams and Collaboration

- Teams with mixed skills (Dev, Test, UX) can resolve issues faster.
- Reduces bottlenecks when specific skills are needed.

Proactive Risk Response

- Teams adjust backlogs, priorities, or technical approaches before big problems hit.

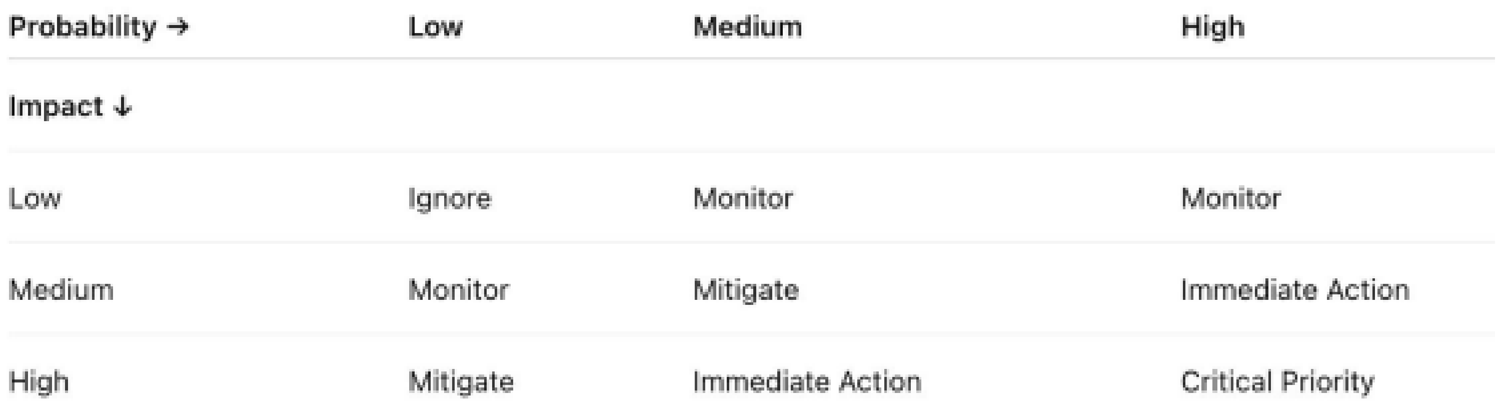
Prioritizing Risks Using Impact/Probability Matrix

Impact = Severity of risk if it happens.

Probability = Likelihood that the risk will happen.

Critical Priority: Act immediately.

Monitor: Keep an eye; act if worsens.



Probability →	Low	Medium	High
Impact ↓			
Low	Ignore	Monitor	Monitor
Medium	Monitor	Mitigate	Immediate Action
High	Mitigate	Immediate Action	Critical Priority

Agile Risk Management Cycle

Cycle repeats every sprint:

1. **Identify**: Spot potential risks.
2. **Assess**: Judge likelihood and impact.
3. **Mitigate**: Take steps to reduce risk severity/probability.
4. **Monitor**: Track status during standups and sprint reviews.
5. **Adapt**: Adjust plans, processes, team actions as needed.

Analogy: Think of risk management in Agile as a daily "weather forecast" — we predict, prepare, and adjust.

Responding to Risks Dynamically

Agile allows changing sprint goals mid-sprint if a serious risk appears.

Backlog priorities can be shifted based on newly identified risks.

Teams may even cancel or modify sprint goals if risks threaten sprint success.

Dynamic response is supported through:

- Quick decision-making (team autonomy)
- Lightweight documentation (user stories, not heavy specs)
- Close customer collaboration (instant feedback)

Agile’s flexibility is a risk defense mechanism, not a weakness.

In Agile, **risk management is everyone’s job, every day**.

Short feedback loops (sprint, review, retrospective) reduce big disasters.

Dynamic reaction to risk makes Agile robust in complex, changing environments.