Agile Software Development

UNIT IV:

- ✧ Planning: Vision, Release Planning, The Planning Game,
- ✧ Risk Management, Iteration Planning, Slack, Stories, Estimating.
- ✧ Developing: Incremental requirements,
- ✧ Customer Tests, Test-Driven Development,
- ✧ Refactoring, Simple Design ,
- ✧ Incremental Design and Architecture,
- ✧ Spike Solutions, Performance Optimization,
- ✧ Exploratory Testing

Planning

- The larger your project becomes, the harder it is to plan everything in advance.
- The more chaotic your environment, the more likely it is that your plans will be thrown off by some unexpected event.
- Yet in this chaos lies opportunity.

The Agile approach to Planning

- 8 practices that allow you to control the chaos of endless possibility
  - Vision
  - Release Planning
  - The Planning Game
  - Risk Management
  - Iteration Planning
  - Slack
  - Stories
  - Estimating

The Agile approach to Planning

- ✧ Vision
  - Reveals where the project is going and why it’s going there.
- ✧ Release Planning
  - Provides a roadmap for reaching your destination.
- ✧ The Planning Game
  - Combines the expertise of the whole team to create achievable plans.
- ✧ Risk Management
  - Allows the team to make and meet long-term commitments.

The Agile approach to Planning

- ✧ Iteration Planning
  - Provides structure to the team’s daily activities.
- ✧ Slack
  - Allows the team to reliably deliver results every iteration.
- ✧ Stories
  - Form the line items in the team’s plan.
- ✧ Estimating
  - Enables the team to predict how long its work will take.

Vision

- ✧ We know why our work is important and how we will be successful
  - Product Vision
  - Where Visions Come From
  - Identifying the Vision
  - Documenting the Vision
  - Promoting the Vision

Vision

- ✧ Sometimes the vision for a project strikes as a single, compelling idea
- ✧ One person gets a bright idea and gets approval to pursue it. This person is a ***visionary***
- ✧ More often, the vision isn’t so clear. There are multiple visionaries, each with their own unique idea of what the project should deliver
- ✧ Either way, the project needs a single vision

Vision

- ✧ Someone must unify, communicate, and promote the vision to the team and to stakeholders
- ✧ That someone is the ***product manager***

Vision

- *The vision statement*
  - It is a clear and simple way of describing why the project deserves to exist.
  - It documents three things
    - What the project should accomplish
    - Why it is valuable
    - The project’s success criteria
  - It’s not a roadmap
    - That’s the purpose of *release planning*

Release Planning

- ✧ Work out the details of how to achieve the vision
- ✧ Create your own plans
- ✧ Set your own release dates

Release Planning

- ✧ One Project at a Time
- ✧ Release Early, Release Often
- ✧ How to Release Frequently
- ✧ Adapt Your Plans
- ✧ How to Create a Release Plan
- ✧ Adaptive Planning and Organizational Culture

One Project at a Time

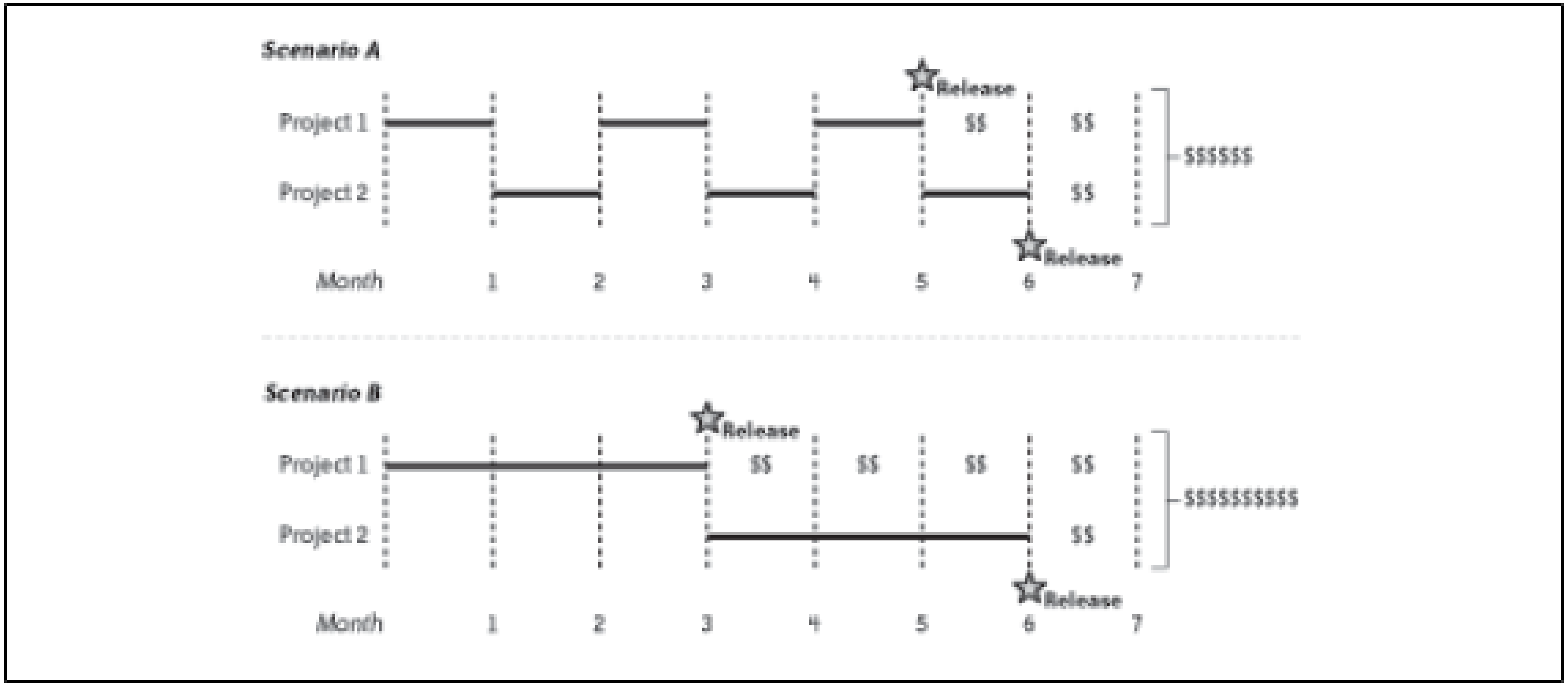


Figure 8-1. Effects of multitasking on value

Release Early, Release Often

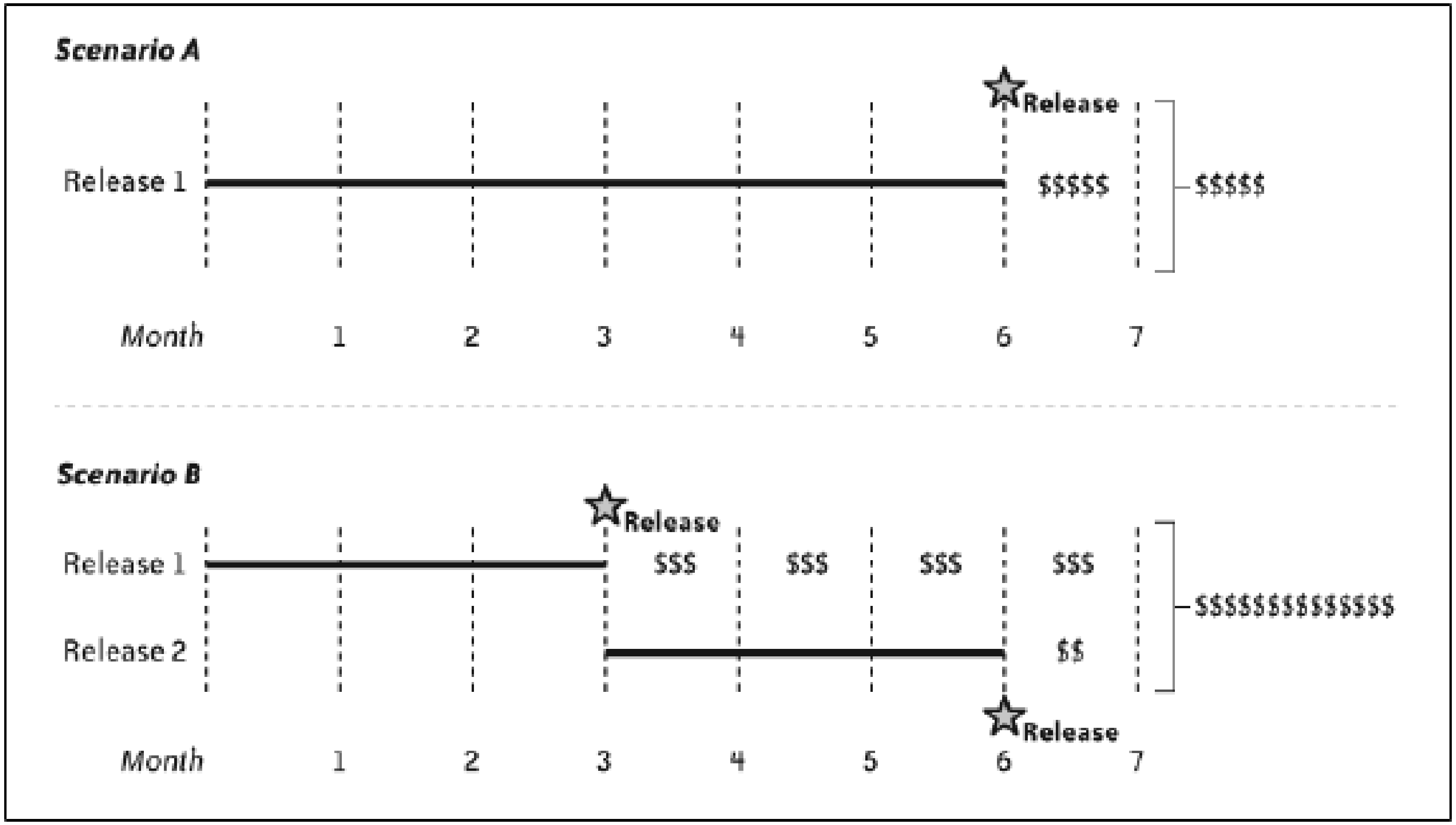
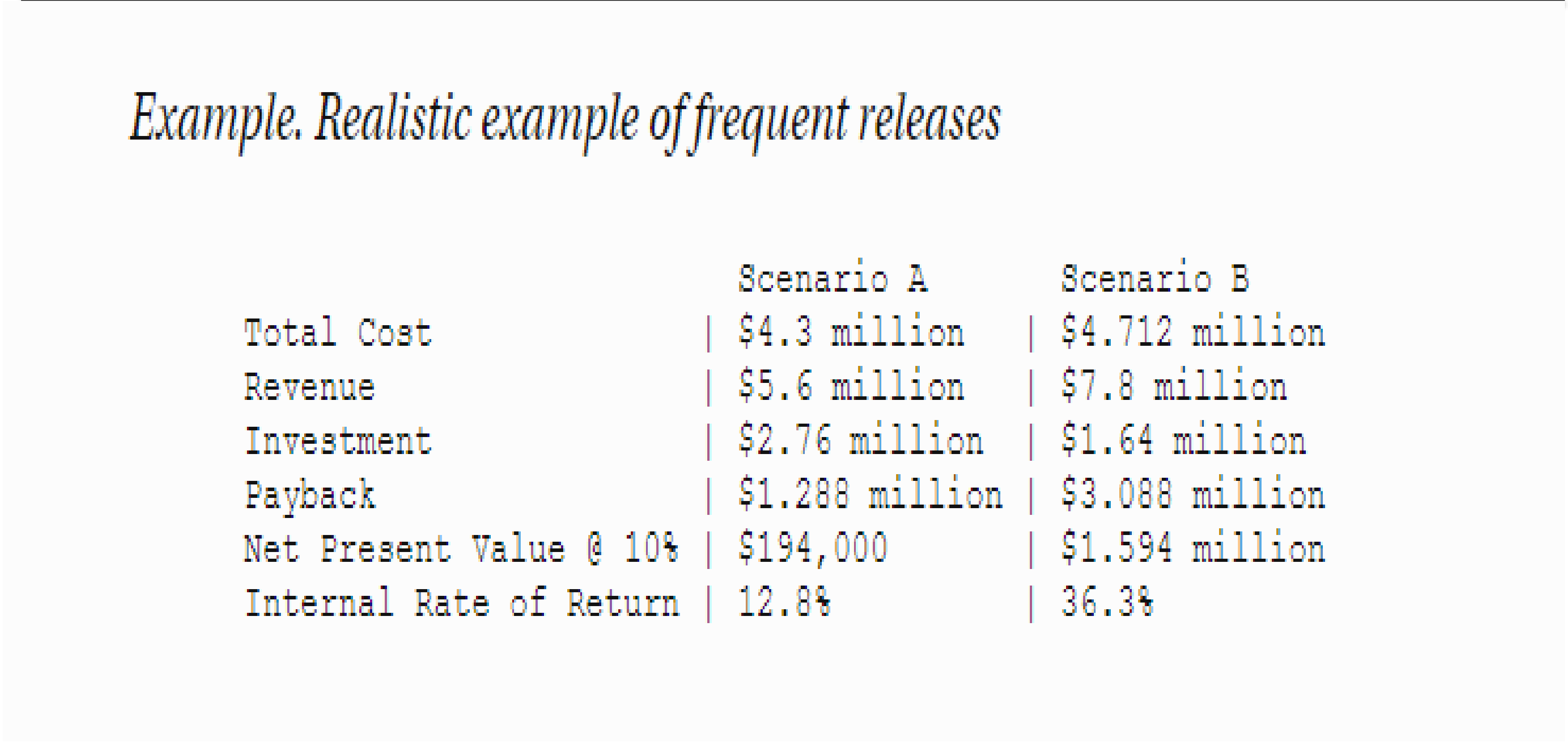


Figure 8-2. Effect of frequent releases on value



Example. Realistic example of frequent releases

	Scenario A	Scenario B
Total Cost	$4.3 million	$4.712 million
Revenue	$5.6 million	$7.8 million
Investment	$2.76 million	$1.64 million
Payback	$1.288 million	$3.088 million
Net Present Value @ 10%	$194,000	$1.594 million
Internal Rate of Return	12.8%	36.3%

How to Release Frequently

- Releasing frequently doesn’t mean setting aggressive deadlines.
- In fact, aggressive deadlines extend schedules rather than reducing them.
- Instead, release more often by including less in each release.
- Minimum marketable features (MMF) are an excellent tool for doing so.
- A minimum marketable feature, or MMF, is the smallest set of functionality that provides value to your market, whether that market is internal users (as with custom software) or external customers (as with commercial software).

Adapt Your Plans

- ✧ If significant results are possible from frequent releases, imagine what you could accomplish if you could also increase the value of each release.
- ✧ After each release
  - collect stakeholder feedback
  - cancel work on features that turned out to be unimportant
  - put more effort into those features that stakeholders find most valuable

How to Create a Release Plan

- There are two basic types of plans
  - Scope boxed
  - Time boxed
- Scope boxed Plan
  - defines the features the team will build in advance, but the release date is uncertain.
- Time boxed plan
  - defines the release date in advance, but the specific features that release will include are uncertain.

Timeboxed plan

- Timeboxed plans are almost always better
  - They constrain the amount of work you can do
  - Force people to make difficult but important prioritization decisions.
  - This requires the team to identify cheaper, more valuable alternatives to some requests.
  - Without a timebox, your plan will include more low-value features

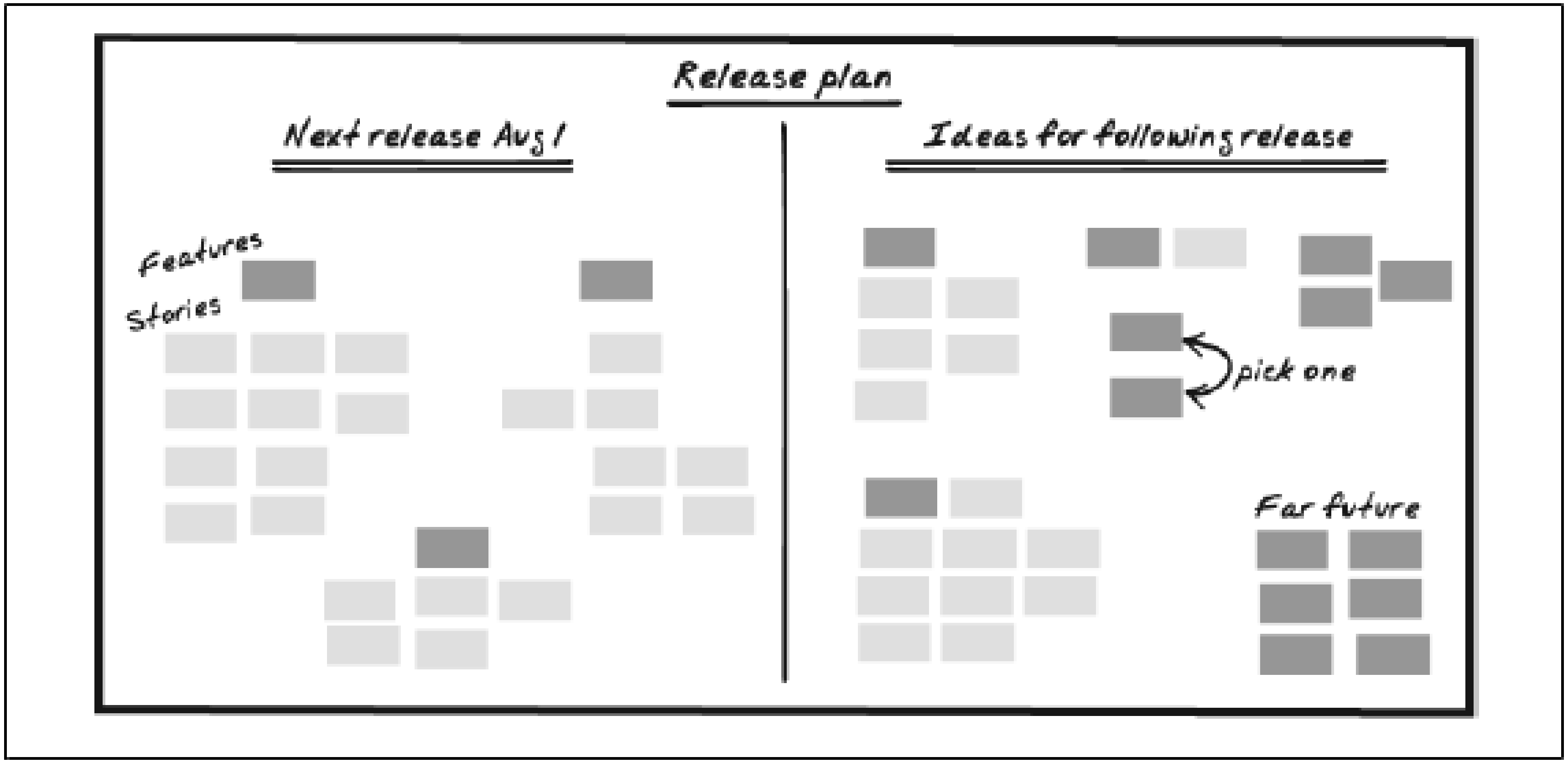


Figure 8-4. A release planning board

Adaptive Planning

- ✧ It takes a lot of time and effort to brainstorm stories, estimate them, and prioritize them
- ✧ If you’re adapting your plan as you go, some of that effort will be wasted
- ✧ To reduce waste, plan at the last responsible moment
- ✧ Your planning horizon determines how far you look into the future

Planning – Starting points

- Your planning horizon determines how far you look into the future
  - Define the *vision* for the entire project
  - Define the *release date* for the next two releases
  - Define the *minimum marketable features* for the current release, and start to place features that won't fit in this release into the next release
  - Define all the *stories* for the current feature and most of the current release. Place stories that don't fit into the next release.
  - *Estimate and prioritize* stories for the current iteration and the following three iterations
  - Determine *detailed requirements and customer tests* for the stories in the current iteration

Organizational Culture

- ✧ Work within your organization’s culture
  - No aspect of agile development challenges organizational culture more than the transition to adaptive planning.
  - It requires changes not only to the development team, but to reporting, evaluation, and executive oversight
- ✧ You can work within your organization’s culture to do adaptive planning under the radar.
- ✧ Use adaptive planning, but set your planning horizons to match the organization’s expectations

The Planning Game

- ✧ Our plans take advantage of both business and technology expertise.
- ✧ You may know when and what to release, but how do you actually construct your release plan?
- ✧ That’s where the planning game comes in.

The Planning Game

- In economics,
  - a game is something in which “players select actions and the payoffs depend on the actions of all players.”
  - The study of these games “deals with strategies for maximizing gains and minimizing losses
- That describes the planning game perfectly.
- It’s a structured approach to creating the best possible plan given the information available.
- The planning game is most notable for the way it maximizes the amount of information contributed to the plan.

How to Play

- XP assumes that customers have the most information about value
  - what best serves the organization.
- Programmers have the most information about costs:
  - what it will take to implement and maintain those features.
- To be successful, the team needs to
  - maximize value while minimizing costs.
- A successful plan needs to take into account information from both groups.

How to Play - Contd

- The planning game requires the participation of both customers and programmers.
- It’s a cooperative game
  - the team as a whole wins or loses, not individual players.
- Programmers have the most information about costs
  - Programmers estimate: most qualified to say how long it will take to implement a story
- Customers have the most information about value
  - Customers prioritize: most qualified to say what is important

How to Play - Contd

- Both groups (programmers & customers) come together, each with their areas of expertise, and play the planning game:
    - Anyone creates a story or selects an unplanned story.
    - Programmers estimate the story.
    - Customers place the story into the plan in order of its relative priority.
    - The steps are repeated until all stories have been estimated and placed into the plan.
- The result of the planning game is a plan: a single list of stories in priority order

The Planning Game - Results

- When you play the planning game well
  - Both customers and programmers feel that they have contributed to the plan.

Risk Management

- Stakeholders need schedule commitments that they can rely upon.
- Risk management allows you to make and meet these commitments.

A Generic Risk Management Plan

- ✧ Every project faces a set of common risks
  - Turnover, New requirements, Work disruption, and so forth.
  - These risks act as a multiplier on your estimates, doubling or tripling the amount of time it takes to finish your work.

A Generic Risk Management Plan

Generic Risk Multipliers

Process approach			
Percent chance	Rigorous	Risky	Description
10%	x 1	x 1	Almost impossible (“ignore”)
50%	x 1.4	x 2	50-50 chance (“stretch goal”)
90%	x 1.8	x 4	Virtually certain (“commit”)

These multipliers show your chances of meeting various schedules

A Generic Risk Management Plan - contd

- ✧ In a “Risky” approach,
  - You have a 10 percent chance of finishing according to your estimated schedule.
  - Doubling your estimates gives you a 50 percent chance of on-time completion, and
  - To be virtually certain of meeting your schedule, you haveto quadruple your estimates

A Generic Risk Management Plan - contd

- If you use the XP practices - in particular
  - If you're strict about being "done done" every iteration
  - Your velocity is stable, and
  - You fix all your bugs each iteration,
  - Then your risk is lowered
  - Use the risk multiplier in the "Rigorous" column.
- On the other hand,
  - If you're not strict about being "done done" every iteration
  - If your velocity is unstable, or
  - If you postpone bugs and other work for future iterations
  - Then - use the risk multiplier in the "Risky" column

Project-Specific Risks

- Create a risk census
  - A list of the risks your project faces that focuses on your project’s unique risks.
    - What about the project keeps you up at night?
    - Imagine it’s a year after the project’s disastrous failure and you’re being interviewed about what went wrong. What happened?
    - Imagine your best dreams for the project, then write down the opposite.
    - How could the project fail without anyone being at fault?
    - How could the project fail if it were the stakeholders’ faults? The customers’ faults? Testers? Programmers? Management? Your fault? Etc.
    - How could the project succeed but leave one specific stakeholder unsatisfied or angry?

Project-Specific Risks – Contd.

- ✧ Once you have your list of catastrophes, brainstorm scenarios that could lead to those catastrophes.
- ✧ From those scenarios, imagine possible root causes.
- ✧ These root causes are your risks: the causes of scenarios that will lead to catastrophic results.

Project-Specific Risks – Contd.

- For the risks you decide to handle, determine
  - Transition Indicators
  - Mitigation Activities
  - Contingency Activities
  - Risk Exposure

Project-Specific Risks – Contd.

- ✧ Transition Indicators
  - Tell you when the risk will come true
- ✧ Mitigation Activities
  - Reduce the impact of the risk
  - Mitigation happens in advance, regardless of whether the risk comes to pass.
- ✧ Contingency Activities
  - Also reduce the impact of the risk, but they are only necessary if the risk occurs.
- ✧ Risk Exposure
  - Reflects how much time or money you should set aside to contain the risk.

Iteration Planning

- ✧ Iterations are the heartbeat of an XP project.
- ✧ When an iteration starts, stories flow in to the team as they select the most valuable stories from the release plan.
- ✧ Over the course of the iteration, the team breathes those stories to life.
- ✧ By the end of the iteration, they've pumped out working, tested software for each story and are ready to begin the cycle again.

The Iteration Timebox

- ✧ Iterations allow you to avoid surprises.
- ✧ Iterations are exactly one week long and have a strictly defined completion time.
- ✧ This is a timebox: work ends at a particular time regardless of how much you’ve finished.
- ✧ Although the iteration timebox doesn’t prevent problems, it reveals them, which gives you the opportunity to correct the situation.
- ✧ In XP, the iteration demo marks the end of the iteration.

The Iteration Schedule

- Iterations follow a consistent, unchanging schedule:
  - Demonstrate previous iteration (up to half an hour)
  - Hold retrospective on previous iteration (one hour)
  - Plan iteration (half an hour to four hours)
  - Commit to delivering stories (five minutes)
  - Develop stories (remainder of iteration)
  - Prepare release (less than 10 minutes)
- Choose an iteration start time that works for your team, and stick with it.

Planning an Iteration

- ✧ After the iteration demo and retrospective are complete, iteration planning begins.
- ✧ Start by measuring the velocity of the previous iteration.
- ✧ Take all the stories that are “done done” and add up their original estimates.
- ✧ This number is the amount of story points you can reasonably expect to complete in the upcoming iteration

Terminologies

- ✧ Your velocity is the number of story points you can complete in an iteration.
- ✧ Story points are the ideal engineering days
  - ▪ Number of days a task would take if you focused on it entirely and experienced no interruptions.

Estimation

- ✧ Tips for Accurate Estimates
- ✧ You can have accurate estimates if you
  - Estimate in terms of ideal engineering days (story points), not calendar time
  - Use velocity to determine how many story points the team can finish in an iteration
  - Use iteration slack to smooth over surprises and deliver on time every iteration
  - Use risk management to adjust for risks to the overall release plan