7 Agile Estimation

AGILE ESTIMATION OVERVIEW

What's different about agile estimation?

BEFORE DISCUSSING AGILE ESTIMATION, it's important to have an understanding of what's different about the environment for an agile project, because there are some very significant differences that affect how you would do any kind of estimation.

A traditional plan-driven or waterfall project is usually based on somewhat of a contractual relationship between the business users and the project team. Typically, the business users agree to some fairly well-defined requirements and sign off on them prior to the start of the project. The project team then commits to a cost and delivery schedule to meet those requirements and any changes in the scope of the requirements are controlled from that point forward. In order to manage the reliability of the estimates, some kind of disciplined process for managing changes to the requirements is implemented, and significant changes are considered to be an exception rather than the norm. If a significant change does takes place, it normally triggers an assessment to determine the impact on the costs and schedule of the project and any initial estimates are recalculated and revised if necessary.

An agile project has a very different model:

- First, the requirements are typically not necessarily well-defined in detail up front; and, in addition, change is the norm rather than the exception. Change is expected and encouraged as the project is in progress. Naturally, any estimates can only be as good as the requirements are defined and any change in the requirements might invalidate any initial estimates. For that reason, some people say that it is futile to even attempt to estimate the costs and schedule of an agile project because the requirements are only going to change. I don't necessarily agree with that point of view—just because a project has some level of uncertainty, doesn't mean it is totally futile to estimate the costs and the schedule; but, of course the level of accuracy in the estimate needs to be understood in the context of the level of uncertainty associated with the project. Very few people get a blank check to do an agile project without any expectations of what the cost and schedule might be.

- An agile project is based on a much more collaborative partnership between the business users and the project team and there is a much higher level of transparency and sharing of information related to the project, which should lead to a much higher level of mutual understanding of the risks and uncertainties in the project as well as a shared understanding of the level of accuracy in any cost and schedule estimates.

The most important thing is to properly set the expectations of the customer regarding whatever estimates are made:

- A traditional plan-driven project estimate might have been based on what was thought of as a relatively static and well-defined model of what the requirements are and how the project will be managed (which might or might not have been realistic).
- An agile estimate is typically based on a high-level and dynamic view of what the requirements are that is understood to be not extremely accurate and likely to change.

Naturally, those are not discrete, binary alternatives and there are many variations in that involve a blend of those two approaches.

Table 7.1 shows a summary of the differences in these two environments.

TABLE 7.1 Estimation in Agile vs. Traditional Processes

	Typical Plan-driven or Waterfall Estimation Process	**Typical Agile Estimation Process**
Relationship of customer or business user to the project team	■ Arm's-length contract based on firmly-defined and signed-off requirements ■ Limited transparency and visibility of the user to project details and issues	■ Collaborative joint partnership model ■ High levels of transparency and sharing of information ■ Mutual understanding of the relationship of uncertainty and flexibility to estimates
Control over changes	Changes to requirements are controlled, and any significant changes are treated as an exceptional situation	Changes to requirements are encouraged and are the norm rather than an exception

Developing an estimation strategy

There are two common problems associated with estimation:

1. Analysis paralysis:
   - Spending too much time attempting to develop detailed estimates based on highly uncertain and unreliable information
   - Delaying making commitments or delaying making any decision at all until the information required to make the decision has a very high level of certainty about it and all of the risks associated with a project are well-known and completely understood
2. Cavalier approach:
   - Not worrying about managing uncertainty and risk at all and just starting to do a project with very little or no upfront planning and estimation
   - Assuming that whatever uncertainty and risk is inherent in the project will be discovered and somehow work itself out as the project proceeds

The right approach for a given project is somewhere in between those extremes and the approach selected needs to be consistent with the level of uncertainty in the project:

- The level of uncertainty in a project can vary widely, from a pure research and development initiative with very high levels of uncertainty to projects with very low levels of uncertainty (like building a bridge across a river).
- It is foolish to underestimate the level of uncertainty in a project and create unrealistic estimates based on very uncertain information.
- However, it can be equally foolish to not take advantage of information you do know and use it to make informed decisions.

The key is to honestly and objectively assess the level of uncertainty and risk in the project and reach agreement among all participants in the project (customer and project team) on an estimation approach that is consistent with that assessment.

Management of uncertainty

Being able to deal with uncertainty effectively is a very critical skill for an agile project manager. You have to be able to accurately assess the risks and uncertainty in a given situation and make informed decisions based on the best information that you have, knowing that it might not be a perfect decision. That is not really a new skill with agile—it is a skill that all project managers should have; however, it is especially important in an agile environment where the level of uncertainty may be significantly higher.

This process of managing uncertainty is similar to making a decision when you're betting on a card game. Many people are familiar with the game of blackjack where players

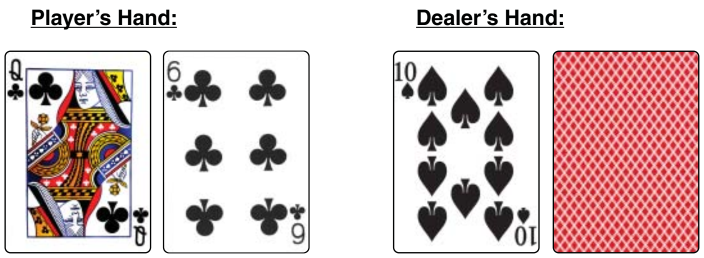

FIGURE 7.1 Example of a blackjack hand

compete against the dealer to get the highest value of cards without going over 21 points (see Figure 7.1):

- Face cards are worth 10 points each and aces can be worth either 1 or 11 points each.
- Players are each dealt two cards face up.
- The dealer is dealt two cards (one face up and one face down).
- The player has to decide to take more cards (called taking a hit), risk going over 21 points, and automatically losing; or to stay with his/her current cards and risk losing to the dealer because the dealer has a higher number of points.

Here's how I would go about analyzing this situation:

1. What information do I know for certain?
   - I have 16 points in my hand.
   - The dealer has more than 10 points in his/her hand.
2. What information is unknown or uncertain?
   - What is the dealer's second card?
   - What card will I be dealt next?
3. What is a reasonable assumption to make?
   - There is a high probability that the dealer's second card is another 10 or a face card, giving the dealer a total of 20.
   - If that is true, I will lose on 16 if I don't take a hit, so I should risk taking another card.

This is an example of making an informed decision based on incomplete information. It is something a gambler does all the time. It is a very important skill for an agile project manager. An agile project manager needs to be frequently making informed decisions in a very uncertain environment.

AGILE ESTIMATION PRACTICES

Levels of estimation

In an agile project, there is no prescribed way to do estimation, and the exact approach could range from doing very little or no estimation at all to having several levels of planning and estimation based on the scope and complexity of the project, as shown in Figure 7.2.

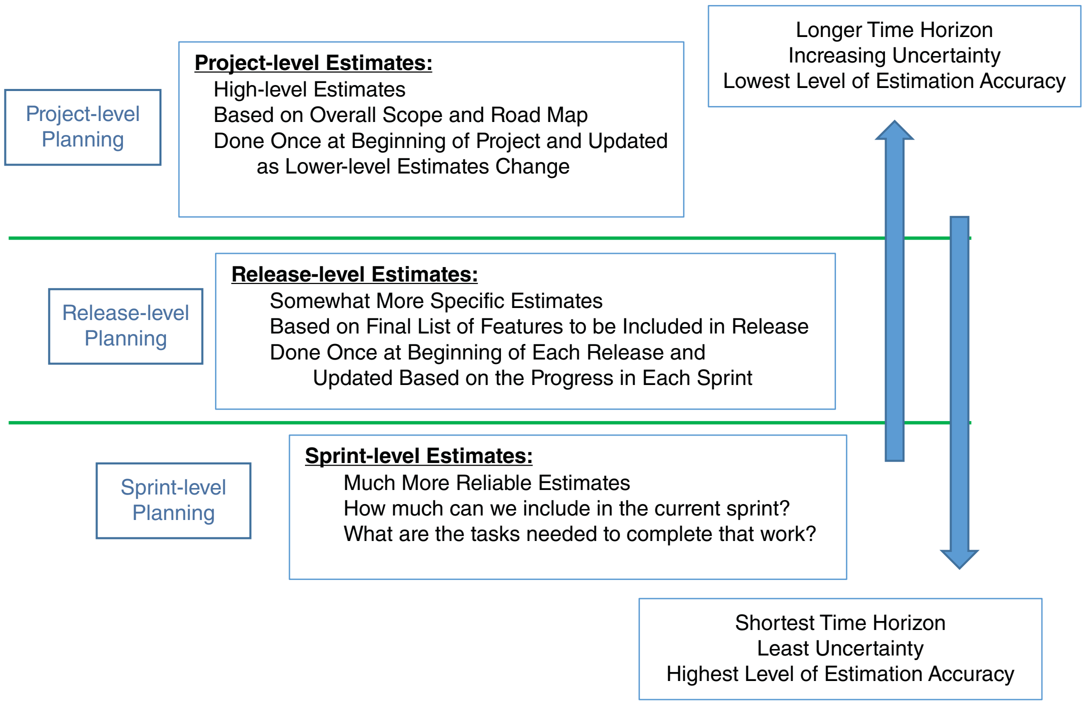
Project-level Planning
Project-level Estimates:
High-level Estimates
Based on Overall Scope and Road Map
Done Once at Beginning of Project and Updated as Lower-level Estimates Change
Longer Time Horizon
Increasing Uncertainty
Lowest Level of Estimation Accuracy
Release-level Planning
Release-level Estimates:
Somewhat More Specific Estimates
Based on Final List of Features to be Included in Release
Done Once at Beginning of Each Release and Updated Based on the Progress in Each Sprint
Sprint-level Planning
Sprint-level Estimates:
Much More Reliable Estimates
How much can we include in the current sprint?
What are the tasks needed to complete that work?
Shortest Time Horizon
Least Uncertainty
Highest Level of Estimation Accuracy

FIGURE 7.2 Levels of agile estimation

The level of emphasis that you would put on each of these levels will also vary significantly, based on the nature of the project. Doing sprint-level planning and estimation is most essential, but any of the levels of planning and estimation just described are optional, depending on the nature of the project. For example, an agile estimation approach could be as simple as having sprint-level estimates only without any project-level or release-level estimates at all.

The approach is totally dependent on the level of uncertainty in the project and the level of predictability and control that is desired. Of course, the level of predictability and control needs to be consistent with the level of uncertainty in the project. Attempting to achieve a very high level of predictability and control in a project that has very high levels of uncertainty may be futile.

Each of these levels of planning would naturally have a different planning horizon, different levels of uncertainty, and a different level of accuracy for any estimates associated with it. Naturally, the longer the planning horizon, the more the uncertainty will be and the less accurate the estimates will be.

Sprint-level planning is the lowest level, has the shortest planning horizon, and it should have the highest level of certainty and the highest accuracy in the estimates. An experienced agile team should be able to accurately predict the capacity that can be completed in a given sprint after the team has reached maturity and the velocity of the team has begun to stabilize.

Before the team starts a sprint, the features to be included in that sprint should be known, and they should be understood well enough to make a reasonably accurate estimate of the level of effort required to complete those features. The details associated with those features may be further elaborated during the sprint, but features should not normally be added or subtracted from the items to be included in the sprint while the sprint is in progress.

Project-level planning is the highest level of planning, has the longest time horizon, and has the highest level of uncertainty. Naturally, the estimates at this level can only be as accurate as the level of certainty in the requirements, so the estimates may not be very accurate. The level of accuracy depends on a number of factors such as the nature and complexity of the project, the level of uncertainty in the project requirements, and the importance of having schedule and cost estimates.

Release-level planning is an intermediate level of planning between project-level planning and sprint-level planning and is optional, based on the scope and complexity of the project. It can be very useful for large, complex projects to break up the project into releases.

What is a story point?

A *story point* is a metric commonly used for estimation in agile projects. It is a way of sizing the level of effort associated with a particular user story. A story point is normally not directly tied to any specific measure of time; it is simply a relative measure of the level of difficulty of a particular story relative to other stories, and a story point may not have the same meaning among different agile teams.

For example, as I was developing the course materials for the agile project management course based on this book, I used story points to estimate the level of effort required for each lesson. One story point was equivalent to one simple PowerPoint slide, and I used that as a reference point for sizing the level of effort associated with other slides and lessons, which is a relative sizing estimate.

Jeff Sutherland has written a great article to explain why story points are better than hours for agile estimation:

> Story points give more accurate estimates, they drastically reduce planning time, they more accurately predict release dates, and they help teams improve performance. Hours give worse estimates, introduce large amounts of waste into the system, handicap the Product Owner's release planning, and confuse the team about what process improvements really worked . . .
>
> The stability of a user story is critical for planning. A three point story today is three points next year and is a measurable part of the product release for a Product Owner. The hours to do a story depend on who is doing it and what day that person is doing it. This changes every day.[1]

Story points are also a more relevant metric to measure completion of work: "Hours completed tell the Product Owner nothing about how many features he can ship and when he can ship them."[2]

[1]Jeff Sutherland, "Story Points: Why are they better than hours?," posted on May 16, 2013, http://scrum.jeffsutherland.com/2010/04/story-points-why-are-they-better-than.html
[2]Ibid.

In reality, it is very difficult to start a new agile team using story points because they have no frame of reference to compare it to. In that case, I have sometimes made an arbitrary decision to let one story point be approximately one man-day of work but that is just an arbitrary decision to get started with. As the team gains more experience, they should be able to make relative comparisons among stories based on their past experience without a reference to any specific measure of time. An alternative to story points for teams who are not very experienced is *ideal days*. An ideal day is defined as follows:

> A unit for estimating the size of product backlog items based on how long an item would take to complete if it were the only work being performed, there were no interruptions, and all resources necessary to complete the work were immediately available.[3]

Once a team gains experience, story points become very easy and intuitive.

> A mature agile team intuitively knows what a story point means in terms of the relative size of a user story compared to other stories that it has sized in the past, but how does a new team that perhaps even has people who are new to agile get started with story points? As I mentioned in another blog post, Ideal Days is a story point estimation scale that blends size with effort and degrades the backlog sizing process into a drawn-out time estimation exercise.[4]

A Fibonacci sequence is often used for story point estimates because it provides an appropriate level of discrimination between large and small estimates without attempting to very accurately discriminate large estimates. A Fibonacci sequence is a series of numbers of the form of (1, 2, 3, 5, 8, 13, 21—where each number is the sum of the previous two numbers). T-shirt sizes can also be used to create estimates. Both story points and T-shirt sizes are designed to provide a very imprecise framework for making estimates that is appropriate to an agile environment.

For an estimation approach based on story points to work with agile teams, it is important for the teams to be stable. If people are rotated in and out of an agile team, it will destabilize the team and make the estimation process very difficult.

How are story points used?

Story points are used in different ways based on the level of planning and estimation being done.

1. High-level product backlog grooming:
   - In grooming the product backlog, stories are estimated in story points.
   - A voting technique is used to estimate the level of effort associated with each story in story points.

[3] Ideal day definition, http://www.innolution.com/resources/glossary/ideal-day.
[4] Blog archives, "Getting Started with Story Points," posted by Alec Hardy, Feb. 3, http://agilereflections.com/tag/ideal-days/.

- The relative size of the story points provides some information to the product owner to assess the business value against the level of effort.
- Based on that assessment, the product owner will rank order the items in the product backlog to try to maximize the business value to be gained.

2. Tactical sprint planning:

- The project team determines the team's velocity in story points based on the number of story points completed in each sprint.
- At the beginning of each sprint, the project team will make a determination of how many stories that they can take into the current sprint based on their previous velocity history.
- In sprint planning, a more detailed estimate of the level of effort required for the tasks associated with each story may be used to validate the story point estimates.

3. Project-level and release level planning:

- High-level story point estimates, combined with team velocity estimates, may also be used for developing project-level and release-level estimates.
- For example, to develop a rough estimate of the time required to complete a project, it would be necessary to make a high-level estimate of the story points required for each story to be included in the project or release, total those story points, and then divide that by the velocity of the team in story points per sprint to determine how many sprints it will take to complete the project or release.

What is planning poker?

Planning poker is a commonly used approach to come up with story point estimates:

- A story is presented to the team and is discussed to resolve any significant uncertainties or questions.
- Each member of the team chooses a card representing his or her estimate in story points but does not reveal the estimate to others.
- When everyone is ready to vote, all members of the team show their cards.
- The team then discusses why they voted differently to try to converge on a single estimate that everyone agrees to.

Planning poker is based on the Delphi method. The Delphi method is a structured communication technique, originally developed as a systematic, interactive forecasting method that relies on a panel of experts coming to consensus on an estimate:

- The experts answer questionnaires in two or more rounds.
- After each round, a facilitator provides an anonymous summary of the experts' forecasts from the previous round as well as the reasons provided for the facilitator's judgments.

- Thus, experts are encouraged to revise their earlier answers in light of the replies of other members of their panel.
- It is believed that during this process the range of the answers will decrease and the group will converge towards the 'correct' answer. Finally, the process is stopped after a predefined stop criterion (e.g., number of rounds, achievement of consensus, and stability of results) and the mean or median scores of the final rounds determine the results.[5]

An online tool is available for implementing planning poker at the following website: www.planningpoker.com.

VELOCITY AND BURN-DOWN/BURN-UP CHARTS

The estimation process in an agile project should be dynamic and continuously refined as the project progresses. An agile estimate is based primarily on the velocity of the team, projecting that velocity into the future, monitoring the actual velocity against the projected velocity, and then adjusting the estimate as necessary.

Velocity

The concept of velocity is essential to agile estimation. Velocity is defined as follows:

> Velocity is the measure of the throughput of an agile team per iteration. Since a user story, or story, represents something of value to the customer, velocity is actually the rate at which a team delivers value. More specifically, it is the number of estimated units (typically in story points) a team delivers in an iteration of a given length and in accordance with a given definition of "done."[6]

Here is an example of how velocity can be used to develop an estimate for completing future work:

- Assume that a team has completed an average of 60 story points of work over the past three sprints, and each sprint is two weeks long.
- Suppose there is a total of 300 story points' worth of work to be completed in the current release, and the product owner wants to estimate when the remainder of the release will be completed.
- An estimate for completing the remaining in the release would be 300/60, or five sprints, and since each sprint is two weeks long, it will take approximately 10 weeks to complete the work remaining in the release.

[5] "Delphi Method," http://en.wikipedia.org/wiki/Delphi_method.
[6] "Velocity," Agile Sherpa, http://www.agilesherpa.org/agile_coach/metrics/velocity/.

Burn-down charts

Of course, any estimate is only an estimate, and it should be continuously adjusted as the project progresses as more becomes known about the actual velocity of the team and the level of effort remaining to be completed. A way of doing that is to measure the ongoing actual velocity of the team against the projected velocity and adjust the projected velocity based on actual results. An important tool for doing that is a burn-down chart:

> A burn-down chart is a chart often used in scrum agile development to track work completed against time allowed. The *x*-axis is the time frame, and the *y*-axis is the amount of remaining work left that is labeled in story points and man hours, etc. The chart begins with the greatest amount of remaining work, which decreases during the project and slowly burns to nothing.[7]

Figure 7.3 shows an example of a burn-down chart. In this chart, the *y*-axis shows the total work remaining to be completed in story points, which is 120 units at the beginning of the sprint and is

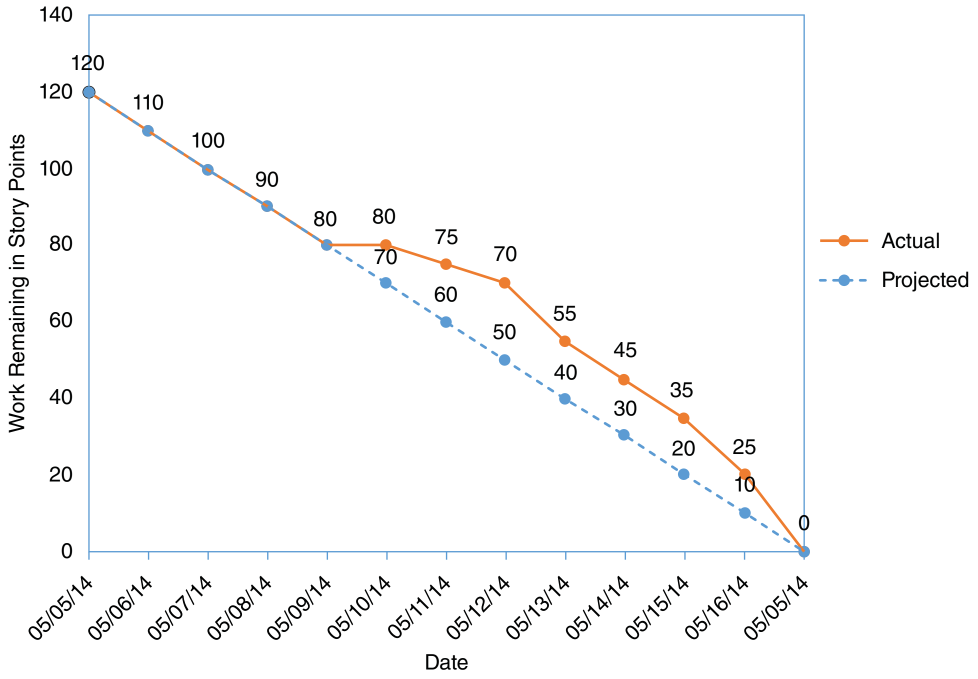

FIGURE 7.3 Example burn-down chart

[7] "What is a Burn-down Chart?" Techopedia, http://www.techopedia.com/definition/26294/burndown-chart.

projected to go down to zero by the end of the time interval. The time interval being measured could be a sprint, a release, or an entire project. The diagonal line between those two points shows the projected rate of completing work in the time interval. The darker solid line shows the actual performance of the team as the sprint is in progress:

- If the actual work remaining is above the projected line, work is proceeding slower than expected.
- If the actual work remaining is below the projected line, work is proceeding faster than expected.

In Figure 7.3 the work started out very close to the projected schedule, then started to lag behind and finally caught up with the projected schedule by the end of the sprint. Burn-down charts are very useful for monitoring the actual progress against projected progress to continuously adjust the estimated completion of an effort.

Burn-down charts are typically measured in either hours of work remaining or story points of work remaining. Each of those metrics provides different information:

- Hours of work remaining is difficult to track because it requires the people on the team to report time spent on tasks, as well as estimating the hours of work remaining, which can be difficult to do. Estimates of hours of work remaining may not be very accurate.
- Story points are easier to track, and it's an all-or-nothing metric—the team gets full credit for all of the story points associated with a given story if the story has been complete and meets the definition of "done," or no credit if it has not been completed. That's probably a more reliable and more accurate metric.

Burn-up charts

A burn-up chart is essentially the mirror image of a burn-down chart:

> A burn-up chart is a graphical representation that tracks progress over time by accumulating functionality as it is completed. The accumulated functionality can be compared to a goal such as a budget or release plan to provide the team and others with feedback. Graphically the X axis is time and the Y axis is accumulated functionality completed over that period of time. The burn-up chart like its cousin the burn-down chart provides a simple yet powerful tool to provide visibility into the sprint or program.[8]

Figure 7.4 shows an example of a burn-up chart. One of the advantages of a burn-up chart is that it provides a way to show a change in the scope of the projected work. In this example on May 13, 2014, there was an increase in the scope of work remaining from 80 to 100 story points, which raised the end goal from 120 to 140. (That shouldn't happen in the middle of a sprint, but it can

[8]Thomas F. Cagley, "Metrics Minute—Burn up Charts," May 9, 2011, http://tcagley.wordpress.com/2011/05/09/metrics-minute-burn-up-charts/.

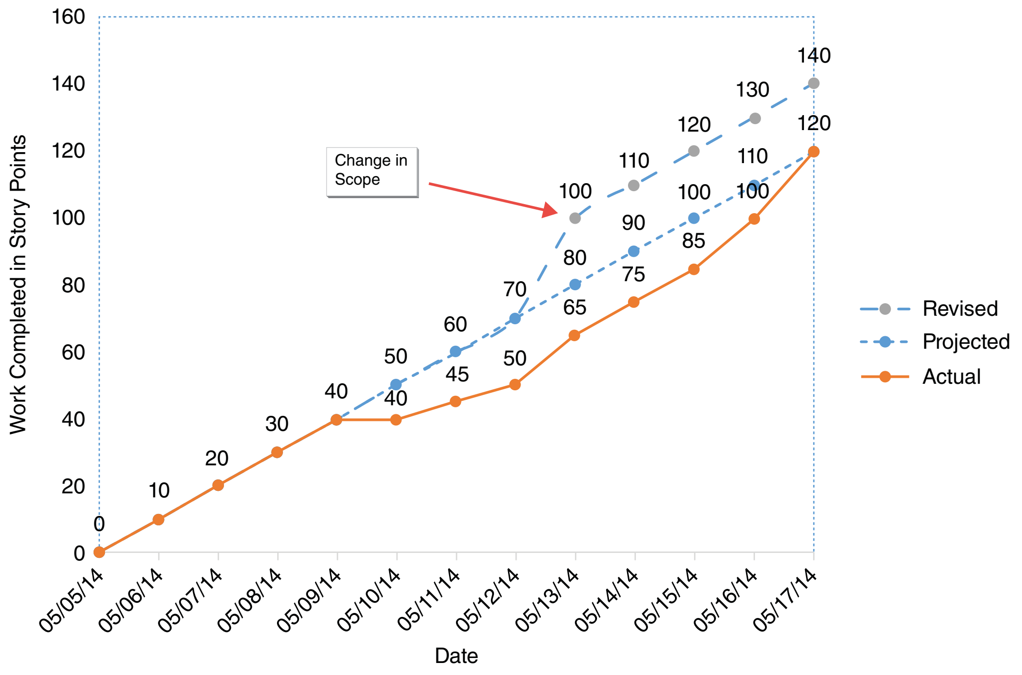

FIGURE 7.4 Example burn-up chart

easily happen in the middle of a release or project.) The chart shows that the team finished the sprint on the original goal of 120 but failed to meet the revised goal of 140.

SUMMARY OF KEY POINTS

1. Agile Estimation

Agile estimation is based on a very different environment than traditional project estimation approaches.

- A traditional model is based on somewhat of a contractual relationship between the customer and the project team and the project estimates are expected to be reasonably accurate to support that kind of relationship.
- Any agile project estimate can only be as accurate as the requirements are known and there typically is lot of uncertainty in the requirements that would limit the accuracy of any estimate. For that reason, it is essential for an agile project to be based more on a collaborative partnership between the business users and the project team as well as a shared understanding of the level of accuracy in any cost and schedule estimates.
- Management of uncertainty is an important skill for an agile project manager—it is something that all project managers should have, but it is especially important in an agile environment where the level of uncertainty may be significantly higher.

2. Agile Estimation Practices

There is no prescribed way to do estimation in an agile project and the exact approach could range from doing little or no estimation at all to having several levels of planning and estimation based on the scope and complexity of the project. Whatever level of estimation is selected, the approach should be consistent with the level of uncertainty in the project and mutually understood by all participants in the project.

Agile estimation practices are based on an understanding of velocity and flow and require breaking up requirements into small functional elements typically in the form of user stories that can be sized by the project team in terms of story points to estimate their difficulty.

A *story point* is a metric commonly used for estimation in agile projects. It is a way of sizing the level of effort associated with a particular user story.

3. Velocity and Burn-down/Burn-up Charts

An agile estimate is based primarily on the velocity of the team, projecting that velocity into the future, monitoring the actual velocity against the projected velocity, and then adjusting the estimate as necessary.

Burn-down charts are typically measured in either hours of work remaining or story points of work remaining. Each of those metrics provides different information. A burn-up chart is a graphical representation that tracks progress over time by accumulating functionality as it is completed.

DISCUSSION TOPICS

Agile Estimation

1. How would you go about determining the appropriate estimation strategy for a project?

2. Discuss an example of a real world project and how you might have done it differently with a more agile estimation approach.

3. What is the advantage of story points over estimating in hours?

4. Complete the agile estimation exercise on the following page and answer the following questions:

- Select two to three of the items in the list and write a user story to describe each one.
- If the work is divided into two-week sprints and I have a velocity of completing 20 story points of work in each two-week sprint, how long will it take to complete this work? (Note that the items in the hierarchy are summed at different levels.)

Agile Estimation Exercise

5. I need to develop an estimate for the schedule required to develop course materials to support this book. I have identified the following tasks and estimated the work associated with each task in story points.

List of Tasks for Estimation

Title	Estimate
1. Introduction, Course Objectives, and Agile Overview	8.00
Introduction and Course Objectives	2.00
Introductions	1.00
Course Objectives	1.00
Agile Overview	6.00
What Is Agile?	3.00
Agile Perception versus Reality	3.00
2. Agile Fundamentals	17.00
Agile History, Values, and Principles	11.00
Agile Manifesto Values	3.00
Agile Manifesto Principles	8.00
Agile Benefits and Obstacles to Becoming Agile	6.00
Agile Benefits	3.00
Obstacles to Becoming Agile	3.00
3. Scrum Overview	34.00
Scrum Roles	3.00
Scrum Master Role	1.00
Product Owner Role	1.00
Team Role	1.00
Kanban Process Overview	10.00
What Is Kanban?	1.00
Differences Between Push and Pull Processes	2.00
Differences Between Kanban and Scrum	2.00
WIP Limits in Kanban	1.00
Theory of Constraints	2.00
Kanban Boards	2.00
Scrum Methodology	8.00
Time-Boxing	3.00
General Scrum/Agile Principles	10.00