

# Chapter 8. Visual Controls and Information Radiators for Enterprise Teams

---

*If you can't measure it, you can't improve it. —Anonymous.*

*If you can't see it, you really can't improve it. —Alan Shalloway*

*If it ain't visible, we ain't working on it. —Guy Beaver*

## In This Chapter

This chapter discusses several of the most important visual controls in the Lean-Agile toolkit. A visual control is used by teams and management to make progress visible to all, to help identify impediments to progress, and to keep everyone focused on delivering value. The purpose of each visual control is described in this chapter; however, the specifics about how to create and implement them are left to other resources. After exploring each visual control, we conclude with a thought about what makes a good control.

## Takeaways

Key insights to take away from this chapter include:

- Visual controls both help you see what is happening and help management assist teams.
- Visual controls should exist for all levels of the organization.
- If a visual control doesn't feel right to the team, they should modify it until it does.

## Visual Controls and Information Radiators

Many Agile methods, such as Scrum, use what are known as “information radiators” to convey information about the status of development to the team and management. Alastair Cockburn says an information radiator “displays information in a place where passersby can see it. With information radiators, the passersby don't need to ask questions, the information simply hits them as they pass.” (Cockburn 2001)

Common information radiators used in Scrum environments include:

- The product vision
- The product backlog / release plan
- The iteration backlog
- Burn-down and burn-up charts
- The impediment list

These are often organized on a team project board, a kind of “super” information radiator. Information radiators are a specialized type of what Lean calls a “visual control.” In Lean environments, visual controls are used to make it easier to control an activity or process through a variety of visual signals or cues. The visual control:

- Conveys its information visually
- Mirrors (at least some part of) the process being used by the team
- Describes the state of the work-in-process
- Is used to control the work-in-process
- Can be viewed by anyone

We prefer the term “visual control” to “information radiator” because it communicates more effectively the attitude and behavior we want. “Information radiator” indicates a one-way direction of information from the team to passersby, including management. It subtly reflects a belief that some Scrum practitioners (incorrectly) hold that the team only needs to provide information of results to management but does not need to provide management with information on how they work. Such an attitude hinders implementing Scrum at the enterprise level when such information is needed.

In Lean thinking, “visual control” is more inclusive. In addition to communicating information to all passersby (so that they don’t have to ask for status reports which would be disruptive to generate), it also reflects the intent that management is a participant in the team’s processes. Visual control invites management to help detect early when there are problems impeding progress toward the team’s goals. That is exactly when the team needs to be interrupted—so that they can stop and adjust while it still matters. Visual control increases the likelihood that management’s “interruptions” actually *add* value to the process.

## **Lean-Agile Visual Controls**

Visual controls should be present at all levels: business, management, and team. That is, they should help the business see how value is being created as well as assist management and the team in building software as effectively and efficiently as possible.

Let’s go into a little more detail about the visual controls used in Lean-Agile. The controls include

- Product Vision
- Product Backlog / Release Plan / Lean Portfolio
- Iteration Backlog – simple team, multiple teams
- Story Point Burn-Up Chart
- Iteration Burn-Down Chart
- Business Value Delivered Chart
- Impediment List

## Product Vision: Providing the Big Picture

Every Agile team should have a product vision. This provides the big picture for the product: what is motivating the development effort, what the current objectives are, and the key features. We have seen teams who were flailing about, trying to figure out what they were supposed to be doing, suddenly gain focus when the Product Champion produced a product vision statement. Creating a poster of the product vision, such as the one in Figure 8.1, helps the team stay focused.

Geoffrey Moore's Crossing the Chasm suggests template for creating product visions in a template format as follows:

**FOR** <target customer>  
**WHO** <statement of the need>,  
**THE** <product name> is a <product category>  
**THAT** <product key benefit, compelling reason to buy>.

**UNLIKE** < primary competitive alternative>,  
**OUR PRODUCT** <final statement of primary differentiation>


Of course, there are other ways to write a product vision as shown in Figure 8.1

### Product Vision Example

The Net Objectives course, *Agile Planning and Estimation with User Stories*, involves a project for a coffee shop with the following product vision:

**For** frequent customers  
**who** need to get in and out of our coffee shop quickly,  
**the** Coffee Kiosk is an ordering and payment system  
**that** allows customers to order their drinks quickly and easily.

**Unlike** any other coffee shop,  
**our product** provides superior and faster service than our competitors.



*Project tag line*  
**Project Name**  
Does *this* decision support the vision?  
Will the customer see this as valuable?

**Motivations**  
What is driving the business to want this? What is in it for them?  
Provide a customer quote  
What is required to make customers feel like they got value?

**A Primary Objective**

- Elements of the objectives, described in business / customer terms
- *lorem ipsum dolor*

**Another Primary Objective**

- *lorem ipsum dolor*

**Key Features**

- A main feature that must be released (MM YYYY)
- A main feature that must be released (MM YYYY)

**Cautions**  
What are the risk factors?  
Remember to focus on value and customers, not (just) technology  
*lorem ipsum dolor*

Lean-Agile Software Development

Figure 8.1. Example of a product vision poster

## Product Backlog with Release Plan

A backlog is the accumulation of work that has to be done over time. In Lean-Agile, the product backlog describes that part of the product that is still to be developed. Before the first iteration, it shows every piece of information that is known about the product at that time, represented in terms of features and stories. As each iteration begins, some of these stories move from the product backlog to the iteration backlog. At the end of each iteration, the completed features move off of both backlogs.

During “Iteration 0,” the features in the product backlog are organized to reflect the priorities of the business: higher-priority features on the left and lower-priority features on the right, as described in chapter 4, *Lean Portfolio Management*.

Using this arrangement, it is straightforward to lay out the tentative release plan. The team has already estimated the size of features and stories. With experience, the team learns its capacity—the number of points it can sustainably deliver in a release (also known as the team’s velocity). Working left to right, the release points come when that capacity is met. Ideally, we never want to go beyond their capacity. Figure 8.2 shows release points based on team velocity.

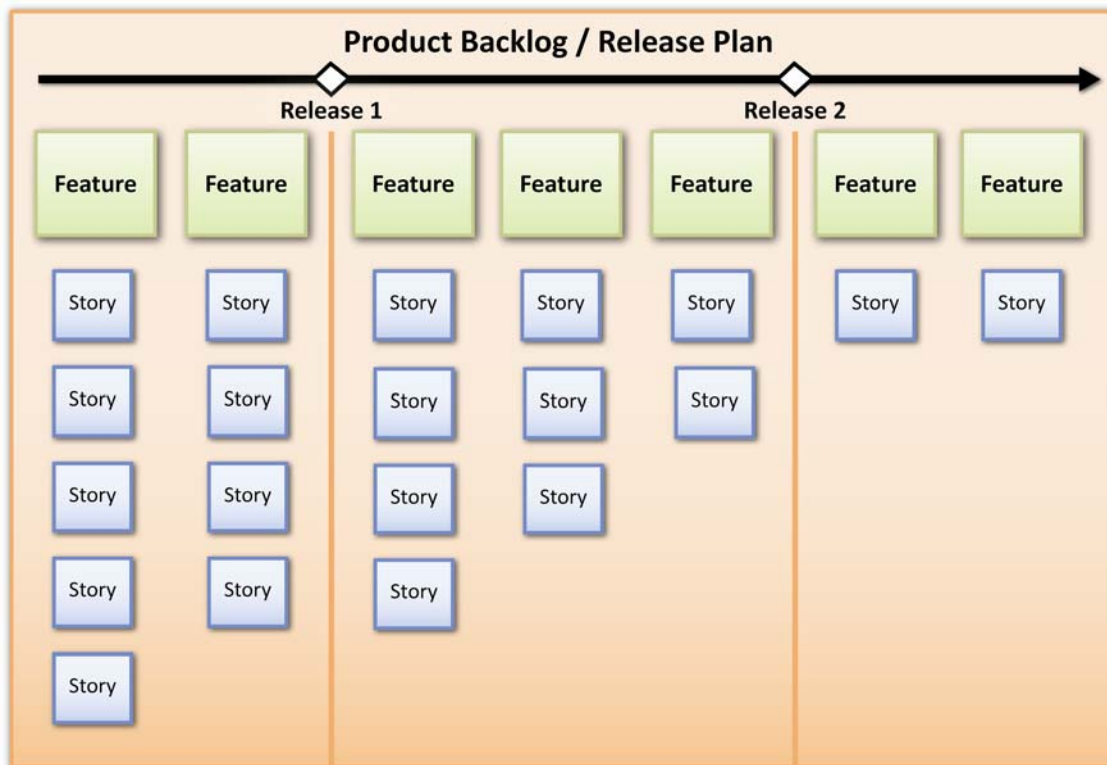


Figure 8.2. Example of a product backlog with release plan: releases align with team velocity

## Iteration Backlog

### The Basic Visual Control for Individual Teams

Iterations are managed with visual controls as well. Figure 8.3 illustrates a basic iteration backlog visual control. It is equivalent to a Scrum story board. The larger rectangles in the build column represent stories to be completed. The smaller rectangles under each of those are tasks. The stories are prioritized from left to right.


	Build
Stories and Tasks (not started)	
Tasks Started	
Stories/Tasks Needing Validation	
Tasks Impeded	
Stories Completed	

Figure 8.3. A basic iteration backlog visual control

As the team starts working on stories, they move tasks down to the Tasks Started row, as shown in Figure 8.4. The team must monitor how many stories are open and try to minimize that number (that is, keep WIP to a minimum). Since the stories themselves have an agreed-upon priority, the visual control clearly shows focus on that priority, since, as much as is reasonably possible, they open the stories in priority order. A good rule of thumb is that there should be fewer tasks open than there are people, which emphasizes collaboration and teamwork. There are three open stories in Figure 8.4. For each task, the control should show an estimate to completion, who is working on it, and a verification statement.

The following is an excerpt from Lean-Agile Software Development: Achieving Enterprise Agility by Shalloway, Beaver, and Trott. No portions may be reproduced without the express permission of Net Objectives, Inc.

	Build
Stories and Tasks (not started)	
Tasks Started	
Stories/Tasks Needing Validation	
Tasks Impeded	
Stories Completed	

Figure 8.4. An iteration backlog that shows tasks started

Once a task is completed, it is moved to the Completed row, as shown in Figure 8.5. Alternatively, if a task becomes impeded, it is moved to the Tasks Impeded row.

	Build
Stories and Tasks (not started)	
Tasks Started	
Stories/Tasks Needing Validation	
Tasks Impeded	
Stories Completed	

Figure 8.5. An iteration backlog that shows tasks started, completed, and impeded

When tasks and stories are done, but not yet validated, they are moved down into the Stories/Tasks Needing Validation row, as shown in Figure 8.6.

The following is an excerpt from *Lean-Agile Software Development: Achieving Enterprise Agility by Shalloway, Beaver, and Trott*. No portions may be reproduced without the express permission of Net Objectives, Inc.

	Build
Stories and Tasks (not started)	
Tasks Started	
Stories/Tasks Needing Validation	
Tasks Impeded	
Stories Completed	

Figure 8.6. An iteration backlog that shows tasks and stories completed

Once a story has been validated, it is moved to the bottom row, signifying its completion, as shown in Figure 8.7.

	Build
Stories and Tasks (not started)	
Tasks Started	
Stories/Tasks Needing Validation	
Tasks Impeded	
Stories Completed	

Figure 8.7. An iteration backlog that shows a completed story

### The Limitations of the Basic Visual Control

The basic iteration backlog visual control works well for individual teams on small projects; however, it has a serious limitation. It assumes that the analysis and estimation required for an iteration is accomplished at the beginning of each iteration. That is, it offers no support for looking a little bit ahead. Even when building in small pieces, it is still important to do a little pre-work analysis to facilitate building the next iteration.

Keep in mind, though, if you take this to an extreme, you get a Waterfall model. *So don't do that!* The intention is to look ahead just enough and no more. In other words, follow the Defer Commitment and Just-In-Time principles.

To build software that returns the greatest value to both our customers and the business, we must discover the business value, define how to create it, and then build it. This discover -> define -> build process may take more than one iteration due to the complexity of the problem or the limited availability of people with the necessary skills. Since “discover and define” is very much part of the process, we need visual controls (the iteration backlog in this case) to help us manage it.

## The Visual Control for Multiple Teams

In Lean-Agile, the work to be done is pulled through the value stream based on what delivers the most value to the business. There are three phases in this work: discover, define, and build. Table 8.1 highlights some of the important points from each phase.

Table 8.1. Three phases in Product Development

Phase	Question Considered	Answers pulled from...
<b>Discover</b>	What features does the business consider valuable? How can we release those incrementally? What are the priorities for those features?	The business, based on business needs and objectives
<b>Define</b>	What is the minimum set of features we need to build in the next release?	The backlog of features that has been discovered and prioritized
<b>Build</b>	How should we build those features?	The prioritized set of features that have been defined for this release

The basic iteration backlog visual control can easily accommodate this richer understanding of the workflow simply by adding two columns: Discover and Define Stories. This is shown in Figure 8.8.

- The Discover column holds stories and tasks while we analyze them. The goal is to discover and establish what is needed to implement the story so that it achieves the business goal. As a story is being worked on, is placed in the Story row; the task—such as analysis work required to get a better understanding of the story so we can work on it—is placed under it. This task will likely generate other stories as we break down the initial one. These will migrate over to the Define column, although it is possible that more discovery work will still be needed in the next iteration.

- The Define column contains stories and their associated tasks that require clearer definitions before they can be built. These tasks may include analysis, spikes<sup>1</sup>, and creating test cases. Tasks completed here ultimately generate stories and tasks in the Build column.

	Build	Define	Discover
Stories and Tasks (not started)			
Tasks Started			
Stories/Tasks Needing Validation			
Tasks Impeded			
Stories Completed			

Figure 8.8. A visual control for discovery, definition and building

The advantage of this visual control is that it gives full visibility on the build process. In addition, it gives a brief look ahead to the build. Clearly, the Define and Discover sections should contain no more than needed to avoid having too much WIP.

One challenge with this control is showing priorities across the three columns. Although each column is prioritized from left to right, the relative priority of stories in the different columns is not clear. In practice, this is generally not a problem since people typically work on stories in different columns. But if it is, the control can be adapted easily by merging the Build, Define, and Discover columns into one section, denoting “define” and “discover” stories distinct from our normal “build” stories.

As Lean-Agile extends up through the enterprise, a visual control of the Lean Portfolio should emerge as the focus point for business decisions and planning. Ideally, Lean organizations are able to decouple business value from technical effort (cost) so that business decisions can be made at the smallest-marketable-feature level. This is typically an emergent trait that results when organizations learn the value of identifying minimum marketable features instead of the batch-and-queue approach of large requirement-gathering efforts.

<sup>1</sup> A spike is a short exploration into some aspect of the system. A developer typically writes code that will be discarded just to help understand the problem at hand.

## Establishing Clear Line of Sight

The Lean-Agile flow from product backlog to iteration backlog works best when the product team has clear sight of what the development team is implementing and the development team can see what is coming. This is illustrated in Figure 8.9.

The product- and iteration backlogs are two halves of the communication loop between the two teams. Manage them together and the result is continuous alignment between different teams: They will begin to act as one team! Following is an explanation of how this works.

The product backlog keeps the development team aware of what is coming from the product team. The product backlog shows estimates for features and stories, as described in chapter 4, Lean Portfolio Management. Since the development team is the source of these estimates, the size of the estimates helps reveal the story's readiness to be implemented. And the lack of an estimate is a cue that the development team has not yet seen a feature or story.

The iteration backlog keeps the product team aware of what is done. Stories in the iteration backlog are not closed—not done—until the Product Champion accepts them. Closed stories help the product team see the progress being made.

Managers should pay attention to visual controls—how they are organized and what they are saying. Consistency is critical. In fact, this is an area in Lean-Agile where absolute conformity is required. *Anyone, anywhere* in the organization, should be able to understand the status and progress for any team at *any* time.

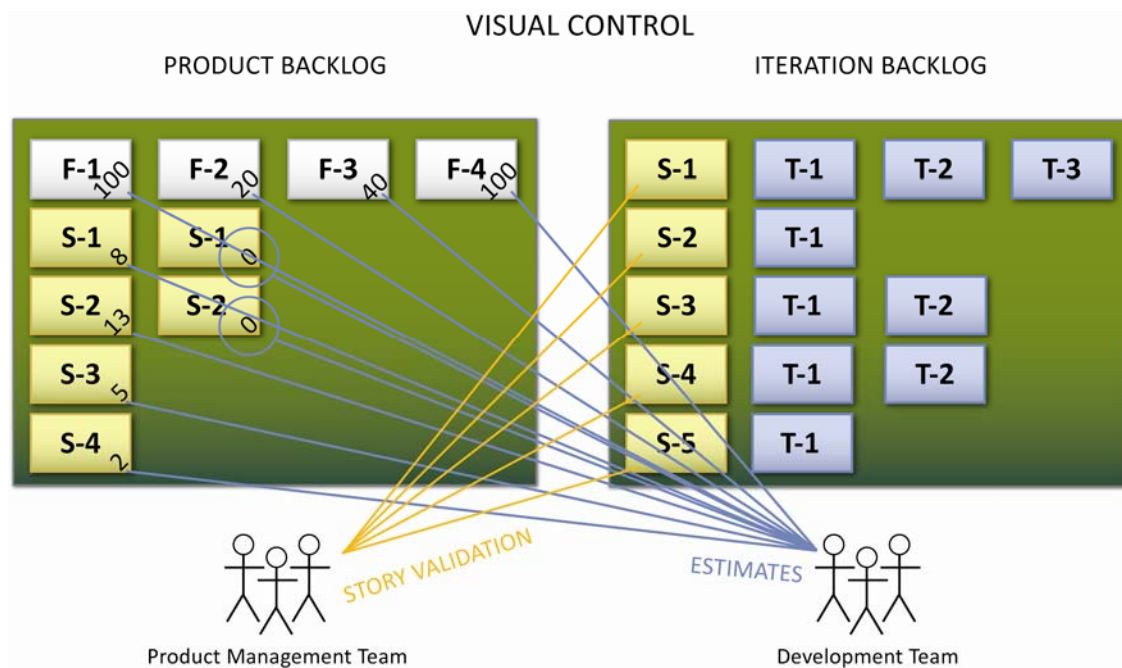
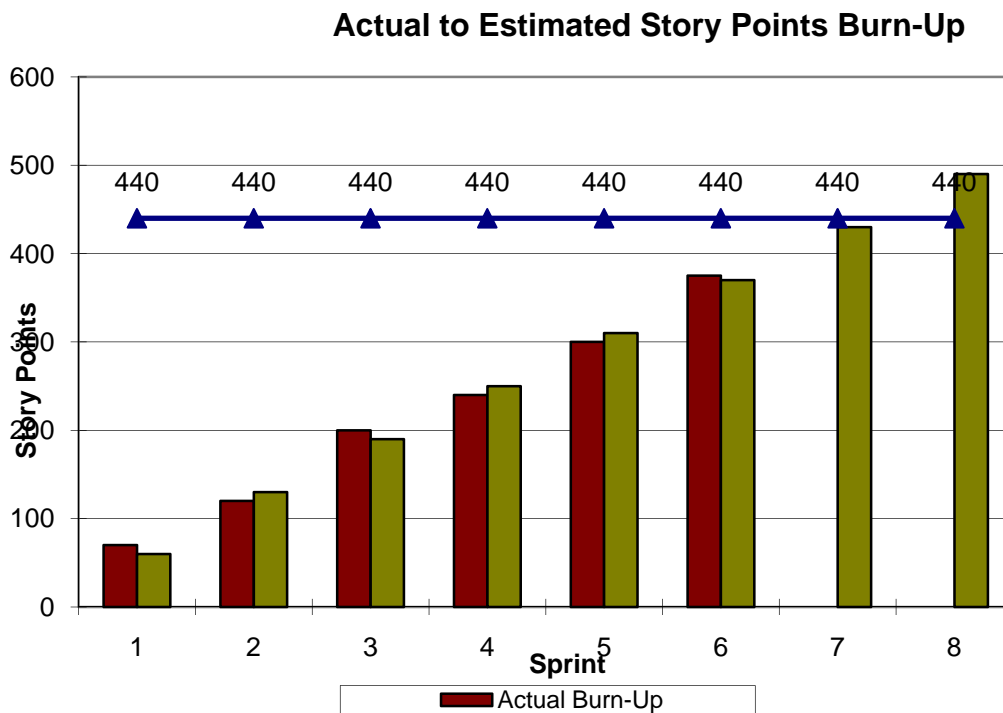


Figure 8.9. Visual controls give line of sight between the product-management and development teams.

The entire enterprise (business, management, and development teams) also need line of sight to velocity (points/time), which, after the first few stories are done, should begin to represent the velocity of business solutions delivered. This is a clear representation of the value stream (from flow of ideas to completed work) and is mapped to the number of points that teams can sustainably deliver in a release. We use two visual controls working as a pair to give management a dashboard-type view of work (see Figure 8.10). The *release* burn-up chart tracks cumulative points delivered with each iteration. This can be broken out or rolled up based on program, stakeholder, and so on. The *feature* burn-up chart is created by using the initial high-level estimate to calculate the percent complete after each iteration and plotting this with all features currently identified in the release plan.

This view gives the enterprise a clear visual control of current priorities, along with what work is actually in process. A Lean enterprise continuously watches out for too many features being worked on at any time—this indicates process problems and potential thrashing.



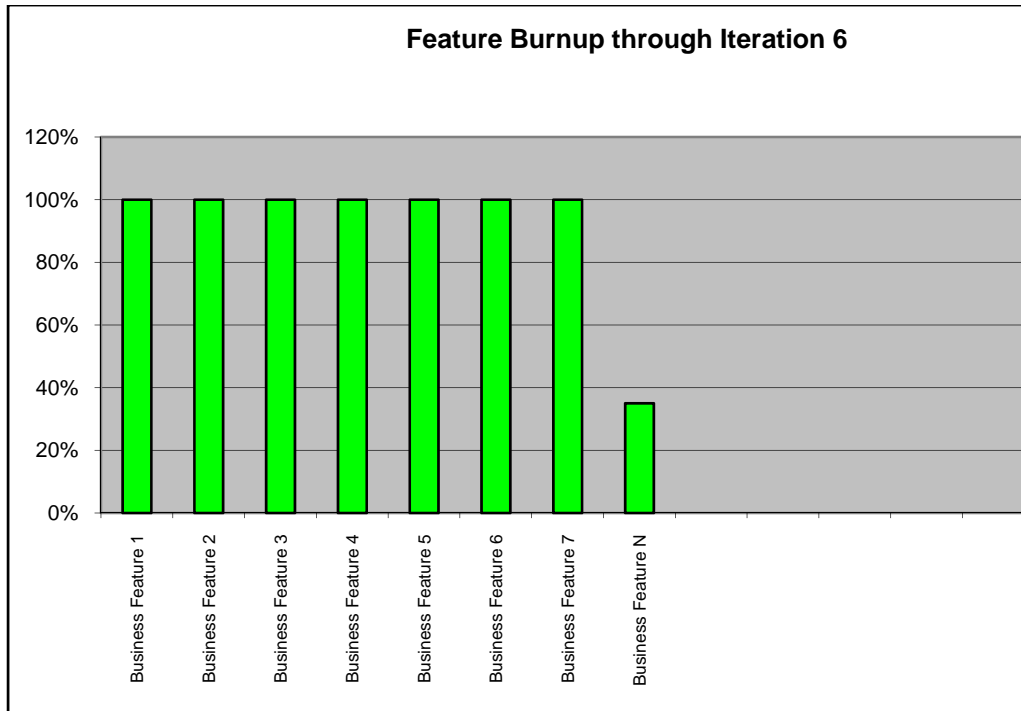


Figure 8.10. The release- and feature burn-up charts together provide a business-driven view of enterprise WIP.

## Managing Dependencies with Visual Controls

As discussed in chapter 4, Lean Portfolio Management, the release plan lays out the priorities and sequence of implementation planned by the team. The approach has been to sequence rows of estimated features and divide them into potential iterations derived from the team's velocity (story points per iteration). Not only is this visual presentation a great approach for seeing the sequence, it facilitates discussions between multiple dependent teams (for example, enterprise data, ETL, UI Design, mid-tier services, and so on), as are typically found in large enterprises.

It is easy to adapt this visual presentation to illustrate dependencies by using simple colored dots. For example, a team can use yellow dots on features and stories to indicate a known future dependency on the enterprise data team; green, some other team. The colors chosen to represent dependencies don't matter; however, we highly urge consistency, particularly across the enterprise. Teach those who are represented by the dots how to read the color-coded release plan. Once managers in those dependent organizations can do that, they should be able to break down the work and deliver incremental pieces (best case) into iterations leading up to the release plan or (worst case) deliver complete pieces when needed.<sup>2</sup>

---

<sup>2</sup> Another approach is to use one color for the dependent item and another color the item it is dependent on (writing the same number on each to show the relationship).

Imagine the insight this simple visual control provides to the enterprise. It brings multiple release plans together, with colors indicating shared service dependencies required to make each planned release. For example, if yellow dots with “ED” represent dependencies on every backlog for enterprise data to deliver something, we should be able to recognize quickly whether that organization has the capacity to keep multiple release plans on track. The mature Lean enterprise has a corresponding visual control for the development team to ensure that they do indeed have the velocity to keep up with the demands of their internal clients.

The key to this dependency management is being able to give enough lead time to support or service groups so that the Agile release plan can successfully accept and integrate the deliveries by those organizations. We track those deliveries with stories that capture the impact on our team and the effort required to integrate the dependent deliveries.

One approach is to use stories that “look ahead” to future features. These are sometimes referred to by the number of iterations required to complete the dependent delivery. For example, if we have to lead enterprise data’s delivery by two iterations, we would call its stories “N-2,” indicating the dependent data team needs a two-iteration lead time to deliver our requested data change. Figure 8.11 shows what these dependencies might look like in a large organization.

User Story 1		User Story 2		User Story 3	
N, N-1	Requirements	N, N-1	Requirements	N, N-1	Requirements
N-1	UXG	N-1	UXG	N-1	UXG
N, N-1	UCD/Web Services	N, N-1	UCD/Web Services	N, N-1	UCD/Web Services
N	UI Dev/MT	N	UI Dev/MT	N	UI Dev/MT
N	MT Dev	N	MT Dev	N	MT Dev
N	Data Development	N	Data Development	N	Data Development
N	SAT	N	SAT	N	SAT
N	CAT	N	CAT	N	CAT
N-1	Batch/Heat	N-1	Batch/Heat	N-1	Batch/Heat
N-2	Data Discovery	N-2	Data Discovery	N-2	Data Discovery
N-1	Data Design	N-1	Data Design	N-1	Data Design
N-1	Data Define/Deploy	N-1	Data Define/Deploy	N-1	Data Define/Deploy

Figure 8.11. Example of dependency management (N-1 means the story requires lead time of one iteration)

A Lean-Agile best practice for dependency management is to include dependent representatives in release-planning discussions and then encourage the same representatives to begin attending daily stand-up sessions as their delivery date approaches. This way they can update status and work with the team to test and validate the integration of the dependent delivery piece.

## Burn-Down and Burn-Up Charts

The purpose of the task burn-down chart (Figure 8.12) is to illustrate the effort spent to date and the expected effort remaining. The story point burn-up chart (Figure 8.13) shows the amount of work-in-process and the amount of completed work; it gives credit for a complete story only after it has been validated. They both report effort expended.

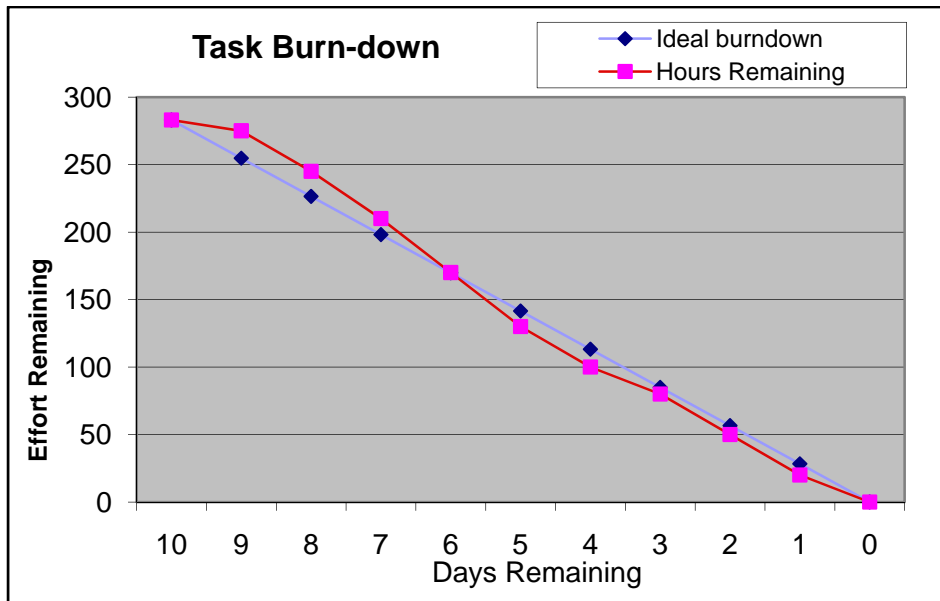


Figure 8.12. Example of a task burn-down chart for an iteration

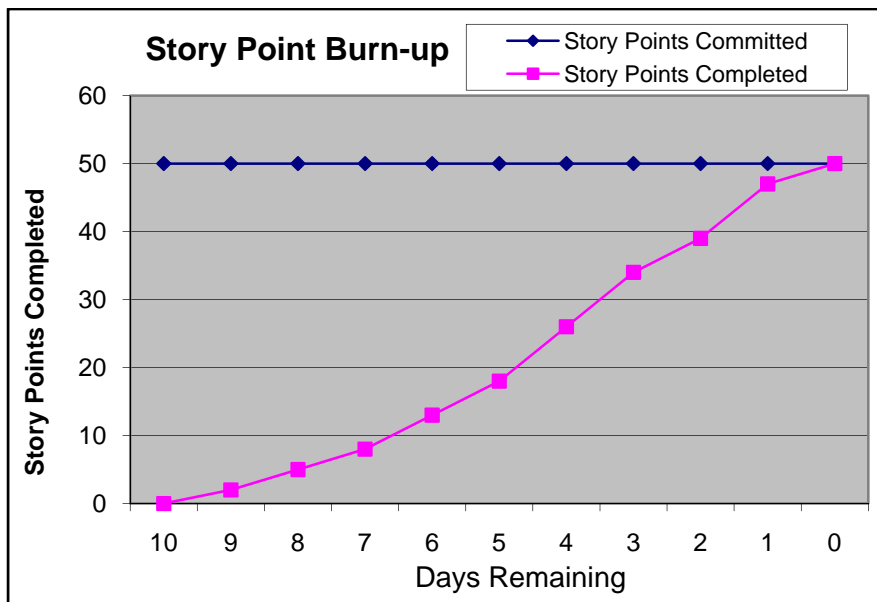


Figure 8.13. Example of a story burn-up chart for an iteration

## The Impediment List

The Impediment List is the final visual control we will consider here. A foundation of both Scrum and Lean-Agile is that continuous improvement includes continually removing impediments. One of the purposes of the daily meeting is to expose impediments, to make them explicit. The Scrum Master or Agile project manager must maintain a list of current impediments so that progress on resolving them can be visible to all. Entries on this list should include:

- Date entered on list
- Description of impediment
- Severity of impediment (what its impact is)
- Who it affects
- Actions being taken to remove it

The impediment list should be maintained on an ongoing basis as impediments are removed, added, or modified in some way.

## How to Tell If You have a Good Visual Control

In general, there is no right visual control to use. However, a particular visual control can be more or less suited to a particular team. Good visual controls have the following characteristics:

- Require little overhead work to use
- Illustrate to the team what to do next
- Illustrate to management how and what the team is doing

If the visual control a team is using does not have these characteristics, then it needs to be improved. In particular, teams should assess whether they like their visual controls. Do they help them get their work done? Are the controls themselves an impediment? If the visual controls do not indicate clearly what the team needs to do next, they should look for ways to improve their controls.

If a team complains about the visual controls they are using then *something is wrong*. Either the team doesn't understand the purpose of visual controls or the visual control itself needs to be improved.

## Summary

Visual controls are a key component for managing Lean-Agile software development efforts. They provide a low-cost method for the team to see where they are. Just as important, they create visibility so that for management and customers can see the progress of the development efforts. By providing status information to all of these parties, the visual controls enable them to work together to solve whatever problems the team faces in getting the highest value, highest quality products out the door.

Visual controls provide a mechanism for teams and management to work together. They help management monitor the team's progress on the outcomes management has corrected. Command and control is replaced with visibility, support, and direction.

The visual controls in this chapter are not intended to be a complete list. They are rather a description of some of the more fundamental ones required for a project. Teams should add additional visual controls as they see fit.

## Try This

These exercises are best done as a conversation with someone in your organization. After each exercise, ask each other if there are any actions either of you can take to improve your situation.

- What visual controls do you have that don't seem to add value?
- What aspect of work are you not measuring that you think would be valuable to measure?
- What is management asking for that is not in a visual control?
  - What would the cost be to put it in a visual control?
  - Would this help you?

## Recommended Reading

The following works offer helpful insights into the topics of this chapter.

Cockburn, Alistair. *Agile Software Development: The Cooperative Game*. Boston, MA: Addison-Wesley Professional, 2001.

Mann, David. *Creating a Lean Culture: Tools to Sustain Lean Conversions*. New York: Productivity Press, 2005.

Reinertsen, Donald G. *Managing the Design Factory*. New York: Free Press, 1997.

## NET OBJECTIVES LEAN-AGILE APPROACH

### INTEGRATED AND COHESIVE

All of our trainers, consultants, and coaches follow a consistent Lean-Agile approach to sustainable product development. By providing services at all of these levels, we provide you teams and management with a consistent message.

### PROCESS EXECUTION

Net Objectives helps you initiate Agile adoption across teams and management with process training and follow-on coaching to accelerate and ease the transition to Lean-Agile practices.

### SKILLS & COMPETENCIES

Both technical and process skills and competencies are essential for effective Agile software development. Net Objectives provides your teams with the knowledge and understanding required to build the right functionality in the right way to provide the greatest value and build a sustainable development environment.

### ENTERPRISE STRATEGIES

Enterprise Agility requires a perspective of software development that embraces Lean principles as well as Agile methodologies. Our experienced consultants can help you develop a realistic strategy to leverage the benefits of Agile development within your organization.



Contact Us:  
[sales@netobjectives.com](mailto:sales@netobjectives.com)

I-888-LEAN-244  
(1-888-532-6244)

*We deliver unique solutions  
that lead to tangible improvements in software development  
for your business, organization and teams.*

## SERVICES OVERVIEW

### TRAINING FOR AGILE DEVELOPERS AND MANAGERS

Net Objectives provides essential Lean-Agile technical and process training to organizations, teams and individuals through in-house course delivery worldwide and public course offerings across the US.

### CURRICULA — CUSTOM COURSES AND PROGRAMS

Our Lean-Agile Core Curriculum provides the foundation for Agile Teams to succeed.

Lean Software Development

- Implementing Scrum for Your Team
- Agile Enterprise Release Planning
- Sustainable Test-Driven Development
- Agile Estimation with User Stories
- Design Patterns

In addition, we offer the most comprehensive technical and process training for Agile professionals in the industry as well as our own Certifications for Scrum Master and Product Champion.

### PROCESS AND TECHNICAL TEAM COACHING

Our coaches facilitate your teams with their experience and wisdom by providing guidance, direction and motivation to quickly put their newly acquired competencies to work. Coaching ensures immediate knowledge transfer while working on your problem domain.

### LEAN-AGILE ASSESSMENTS

Understand what Agility means to your organization and how best to implement your initiative by utilizing our Assessment services that include value mapping, strategic planning and execution. Our consultants will define an actionable plan that best fits your needs.

### LEAN-AGILE CONSULTING

Seasoned Lean-Agile consultants provide you with an outside view to see what structural and cultural changes need to be made in order to create an organization that fosters effective Agile development that best serves your business and deliver value to your customers.

### FREE INFORMATION

#### CONTACT US FOR A FREE CONSULTATION

Receive a free no-obligation consultation to discuss your needs, requirements and objectives. Learn about our courses, curricula, coaching and consulting services. We will arrange a free consultation with instructors or consultants most qualified to answer all your questions.

Call toll free at 1-888-LEAN-244 (1-888-532-6244) or email [sales@netobjectives.com](mailto:sales@netobjectives.com)

### REGISTER PROFESSIONAL LEAN-AGILE RESOURCES

Visit our website and register for access to professional Lean-Agile resources for management and developers. Enjoy access to webinars, podcasts, blogs, whitepapers, articles and more to help you become more Agile. Register at <http://www.netobjectives.com/user/register>.