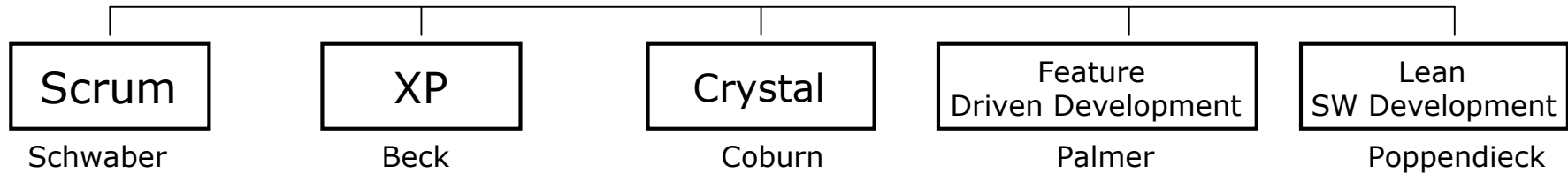
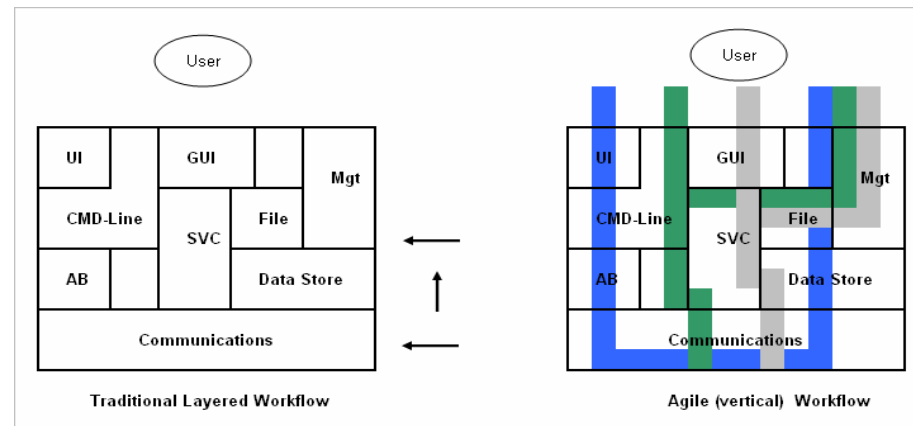


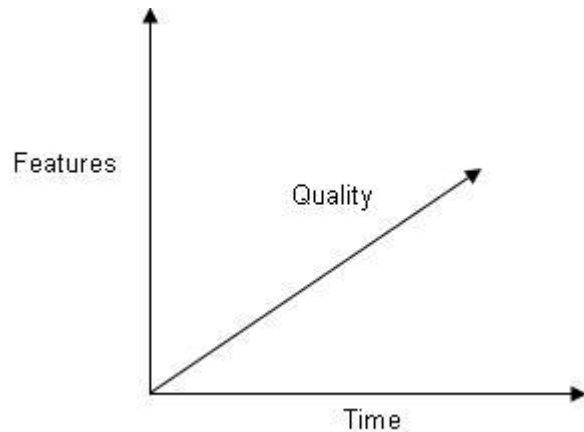
Agile Methodologies



Primary design distinction



Primary management distinction



In an Agile process,
time is fixed

Primary project distinction

- Continuous integration
- Continuous build and test

Topic Snippets

Introduction

Project planning

Setting expectations with executives

Functional implications beyond development

Project management

 Velocity

 Burn down charts

Initial implementation groundrules

Scrum mgmt

Sprint mgmt

Project retrospectives

QA

Assessing individual performance in agile

Refactoring management

“War stories”

Conflicts / similarities to CMMi, ISO-9000, etc.

Burn out

Traditional Project Management vs Agile



A Continuum of Rigor

Get requirements nailed upfront

Structured process: throw away
code is indicator of failure

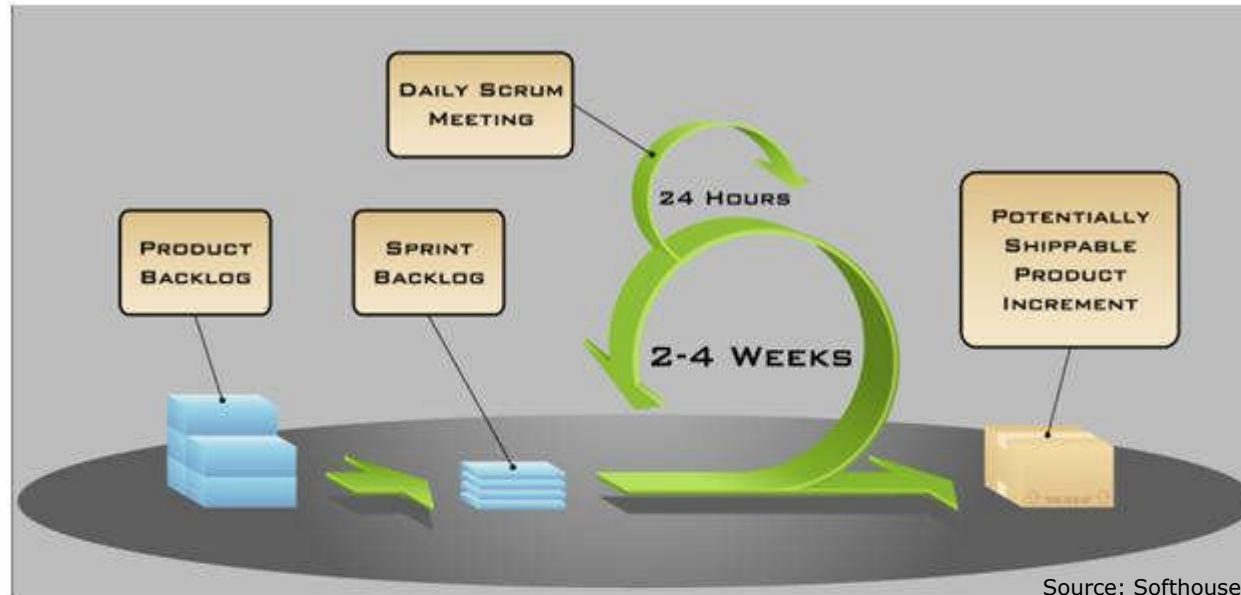
Documentation rich; key work
product

Requirements will change
along the way

Throw-away code OK

Reduced documentation in
favor of working code

Agile Development: Scrum Method



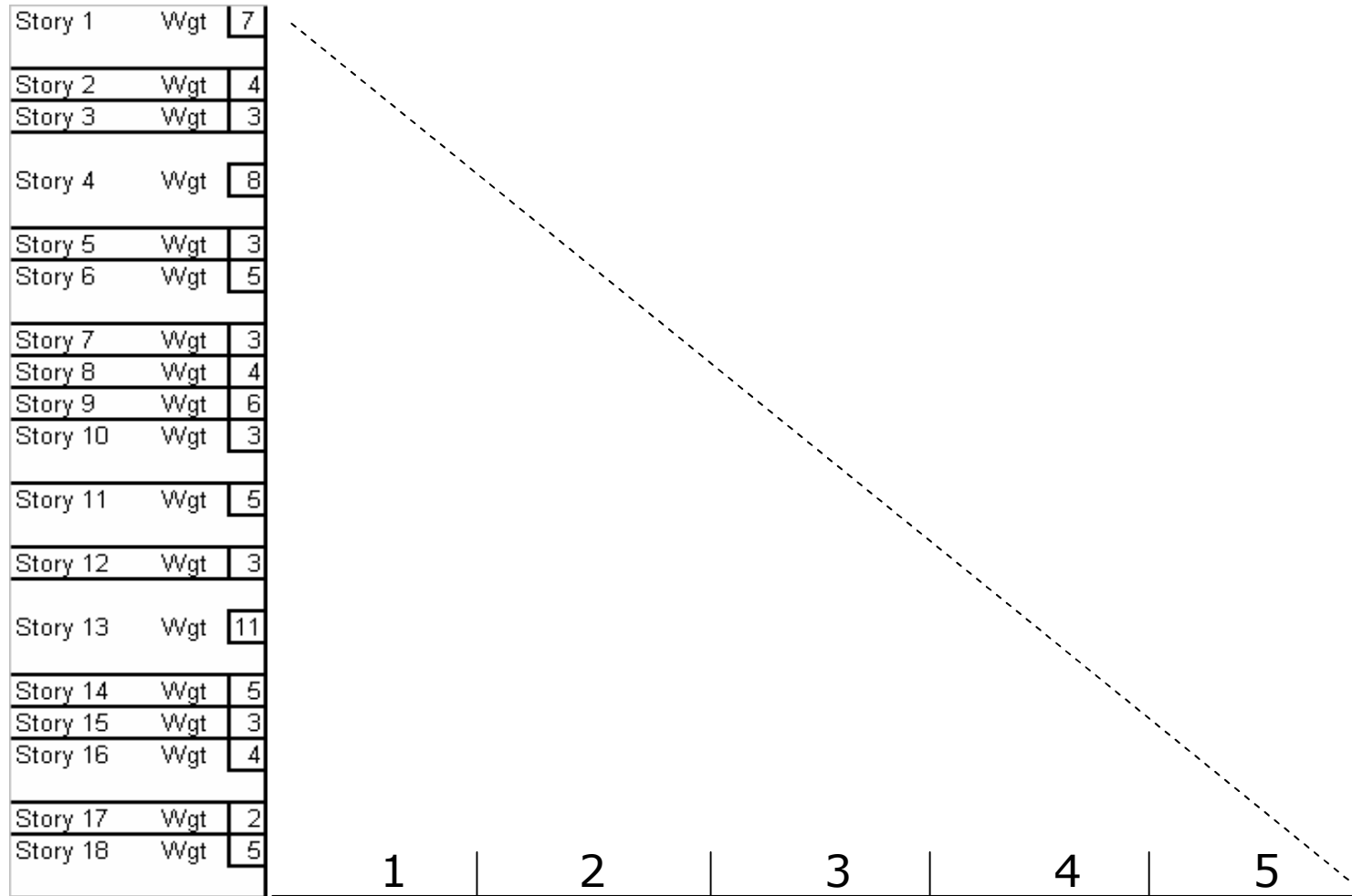
- Sprints and Scrums
- Velocity
- Story weights
- Continuous build and test

Sprint Management

- Sprint Planning Meeting
 - Part 1 run by Product Owner
 - Discussion of Top Product Backlog Items
 - Reviewed for clarity
 - Reviewed for correct prioritization based on needs of the business
 - Attended by developers and all interested stakeholders
 - Part 2 run by Scrum Master
 - Get agreement on which Product Backlog items will be accomplished in this Sprint
 - Create Sprint Backlog
 - List of specific Dev/QA/Doc tasks for each person with estimates
 - Task granularity should be > 2 hours and < 16 hours
 - Attended only by developers
- Sprint Review Meeting
 - Anyone and everyone invited to attend
 - Show off what was accomplished in Sprint
 - No slides / demo-ware; only working, tested code
 - Present as use cases / stories
- Ends with Sprint Retrospective Meeting

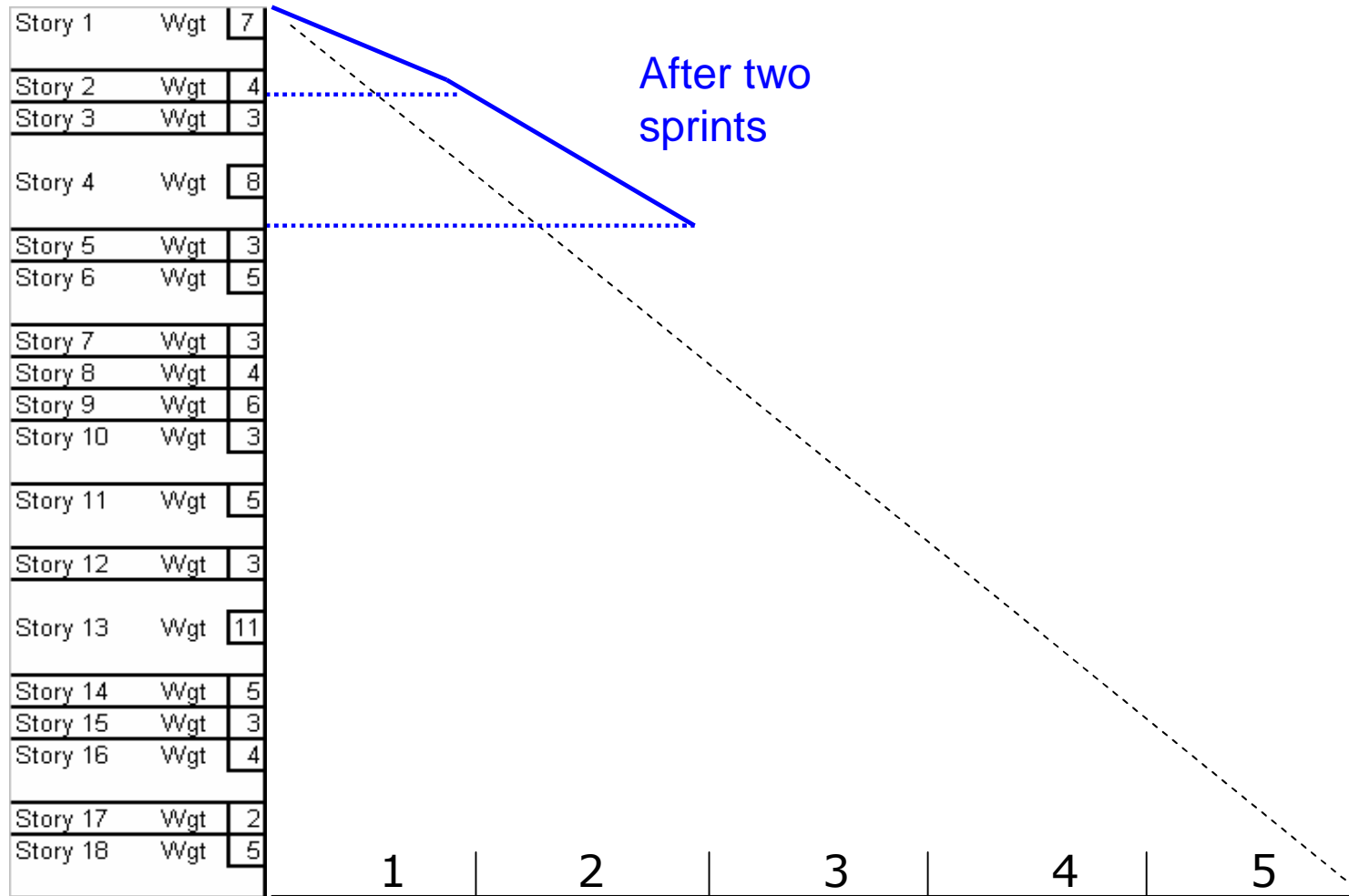
Agile Project Burn Down Chart

User Story - Weight



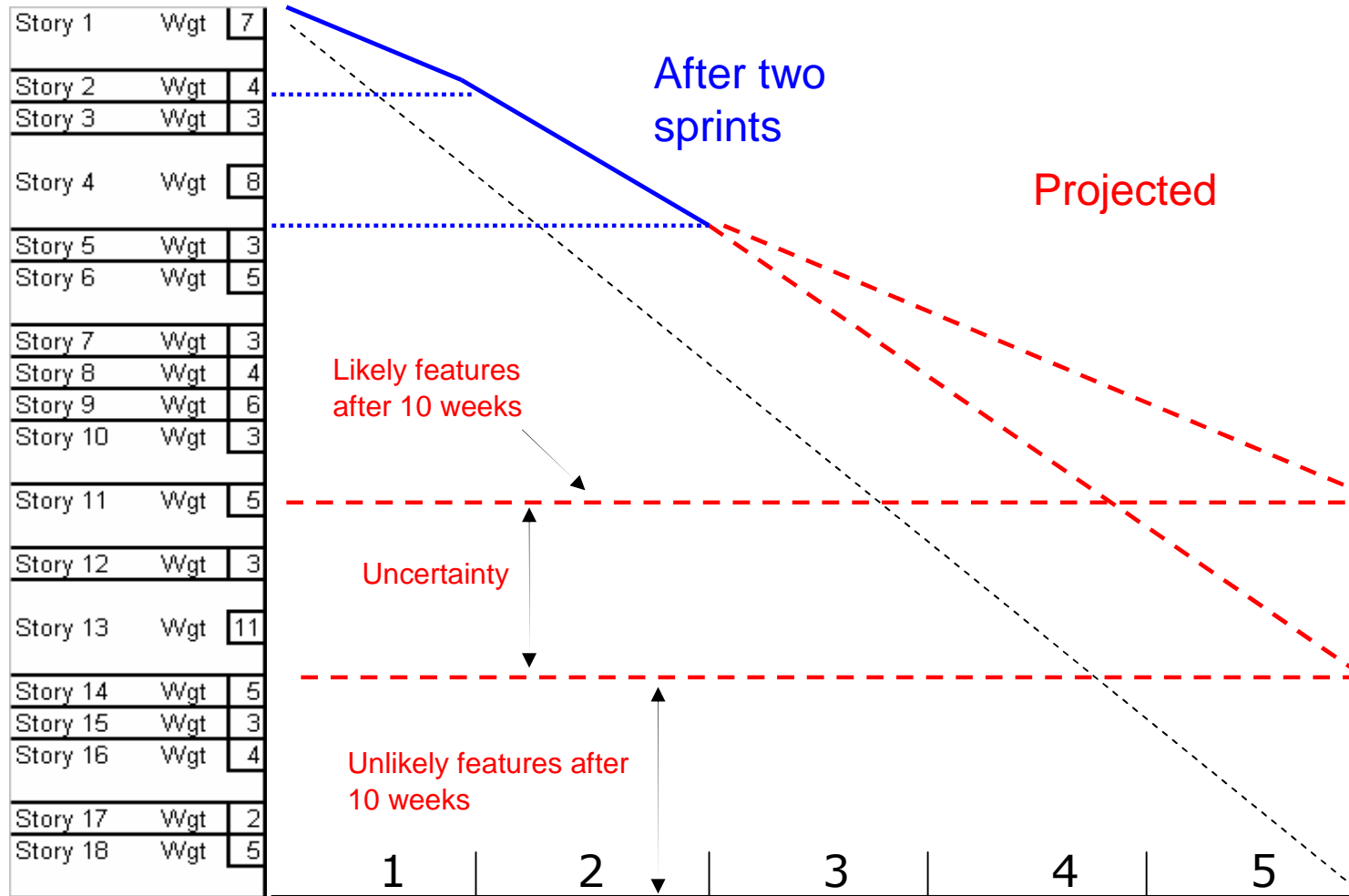
Five, two week sprints

Agile Project Burn Down Chart



Five, two week sprints

Agile Project Burn Down Chart



Five, two week sprints

Sprint Retrospective

- 1.5 – 2 hour meeting at end of the Sprint
- Attended by developers only
- Requires a bit of homework on each person's part
- Start off by reviewing last Sprint's Behavior Modifications to see how well they worked / were followed
- Go around the room twice
 - First, get 2 or 3 things from each person that worked well in the past Sprint
 - Second, get 2 or 3 things from each person that didn't work so well
 - Non-judgmental, no discussion, brainstorming rules
- Next gather ideas for Behavior Modifications
 - Tie them back to which item(s) they address
 - Discuss which ones are important, achievable, measurable
 - ***IMPORTANT*** Pick a small number to enact (2-3)
 - Picking too many or unachievable ones undermines the whole process
 - Assign owners, communicate them, and do followup!

Scrum Management

- Daily Scrum is key
- Developer attendance mandatory
- Others may attend, but they can't speak
- ~ 15 minutes in duration, standup format
- Starts exactly on time; latecomers fined
- Each developer answers the following three questions:
 - What did you accomplish yesterday?
 - What will you accomplish today?
 - What, if anything, is blocking you?
- Limited interaction between team allowed/encouraged
 - 1-on-1 or “next level” discussions discussed post Daily Scrum
- The Scrum Master sets the tone
- ***IMPORTANT*** the team is not reporting status to the Scrum Master
 - He / she is just the facilitator / issue unblocker
 - Get the team into the mindset that they are reporting status to the team (not to mgmt)
 - Make it a safe environment to report problems/issues (surprises == blown Sprints!)

Case Study Results

User Stories	Description	W/O Open Source Sprints #	Complexity Points	Complexity Points	With Open Source Sprints #
User Profile	Create User Profile	1	50	50	1
	Edit User Profile		25	25	
	View User Profile		15	15	
Material Master	Delete User Profile	2	25	25	2
	Create Material Master		35	35	
	Edit Material Master		25	25	
Storage Location	View Material Master	3	15	15	3
	Delete Material Master		25	25	
	Create Storage Location		35	35	
Goods Receipt	Edit Storage Location	4	25	25	4
	View Storage Location		15	15	
	Delete Storage Location		20	20	
Goods Issue	Create Goods Receipt	5	30	30	5
	Edit Goods Receipt		20	20	
	View Goods Receipt		15	15	
Goods Issue	Delete Goods Receipt	6	20	20	6
	Generate Inventory Report by Goods Receipt		35	35	
	Create Goods Issue		35	35	
View Inventory	Edit Goods Issue	7	20	20	7
	View Goods Issue		15	15	
	Delete Goods Issue		20	20	
Optimize Goods Movement	Generate Inventory report by Goods Issue	7	20	20	7
	View Inventory Overview		15	15	
	View Stock Levels by Storage Location		15	15	
Material Reservation	View Inventory Value by Storage Location	6	15	15	6
	Generate Inventory report by Storage Location		20	20	
	Optimize Inventory levels by Storage Location		50	50	
Adjust Inventory	Create Goods Movement	7	35	35	7
	Edit Goods Movement		20	20	
	View Goods Movement		15	15	
Availability	Delete Goods Movement	7	20	20	7
	Send Email Alert on Goods Movement		10	10	
	Create Material Reservation		20	20	
Adjust Inventory	Edit Material Reservation	6	15	15	6
	View Material Reservation		15	15	
	Delete Material Reservation		15	15	
Availability	Adjust inventory Level for Material	6	15	15	6
	Adjust Inventory Level for Storage Location		20	20	
	View Inventory Availability by Material Master		20	20	
Availability	View Inventory Availability by Date	6	15	15	6
	View Stock in Transit		10	10	
	View Inventory Availability by Date		10	10	
Complexity Points Completed			570	755	
Velocity (complexity units per sprint)			81	108	
Increase in Velocity:				32%	

Case Study:

Increasing Velocity Reusing Open Source

- Application
 - Inventory management system for office supplies
- Scope and iterations
 - 12 week release cycle
 - Planned for four 3-week sprints per release
- Team size
 - 8 developers
 - 2 quality engineers
 - 1 build release engineer
 - Scrum master
 - Product owner
- Assumptions
 - Team velocity from previous project was 80 complexity points per sprint not using open source software

User Stories for Inventory Management Application

Functional Areas

- User Profile
- Material Master
- Storage Location
- Goods Receipt
- Goods Issue
- View Inventory
- Optimize
- Goods Movement
- Material Reservation
- Adjust Inventory
- Inventory Availability

Open Source Components

- Acegi – user authorization/authentication
- Java Mail - notification
- Yahoo UI library
- Silk Icon Library
- JGAP – optimization algorithms
- BIRT - reporting
- Hibernate – object relational mapping
- Spring – application framework
- log4j - logging
- Xerces – XML parser
- Apache Axis - an implementation of SOAP
- Apache Collections - collection handling
- AspectJ AOP – aspect oriented programming
- JAXB – XML binding
- PostgreSQL - Object-Relational Database
- Tomcat – Java servlet container

QA

- Need heavy investment in unit tests and automation
 - Establish goals e.g. 75% code coverage minimum and 85% expected
 - Measured and reported with each build
 - Dropping below the threshold results in a build failure
- Live by Red, Green, Refactor
 - Write tests first (they will be red because they won't pass)
 - Write the functionality to make them pass
 - Refactor the code to make it right
- Automation dilemma: How to automatically test code that's being developed during the Sprint (sometimes in the last few days)?
 - Develop automation for things developed in the last Sprint
 - Write lots of unit tests for things developed in this Sprint
 - Manual test (use the whole team) for things non-automated (usually just the stuff developed in this Sprint)
- Nancy's analogy: Keep the snow in front of the plow from building up!