
  
Deliver Better Software Faster

---

## Are We There Yet? An Agile Planning Workshop



**Coach: Paul Hodgetts, Agile Logic,**  
[www.AgileLogic.com](http://www.AgileLogic.com)

Rev 090715

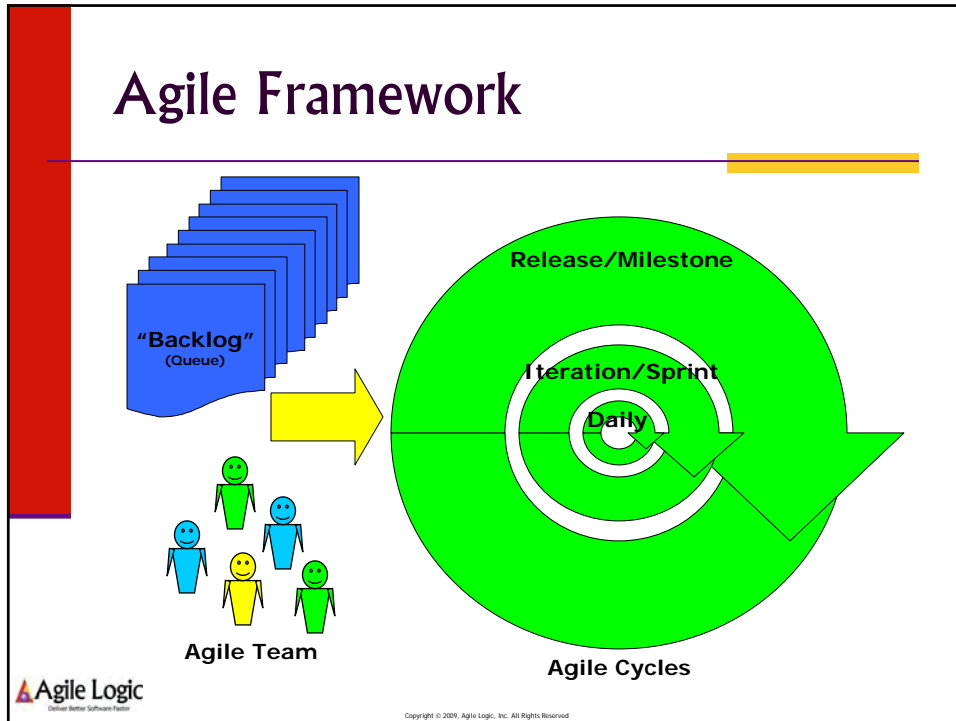
## Your Coach: Paul Hodgetts

- Team coach, trainer, consultant, developer
- Founder and CEO of Agile Logic (Fullerton)
- 26 years overall, 10 years agile experience
- Certified Scrum Trainer
- Worked with a lot of “enterprise” teams
- Author (Extreme Programming Perspectives)
- Speaker at conferences (Agile 200x, SD East/West, JavaOne)
- Active in Scrum Alliance, Agile Alliance (Program Director)
- Member of CSUF agile advisory board
- Contact info: [phodgetts@agilelogic.com](mailto:phodgetts@agilelogic.com)



  
Deliver Better Software Faster

Copyright © 2009, Agile Logic, Inc. All Rights Reserved



## “Predictive” Planning

- Predict the tasks required to deliver
- Usually combined with “waterfall” approach
  - Big batch, long cycle, phased activities
- Goal is to map out the plan, manage to it

**Project Development Schedule**

**Agile Logic**  
Deliver Better Software Faster

Copyright © 2009, Agile Logic, Inc. All Rights Reserved

## Why Predictive Planning Fails

### ■ Descartes Theory

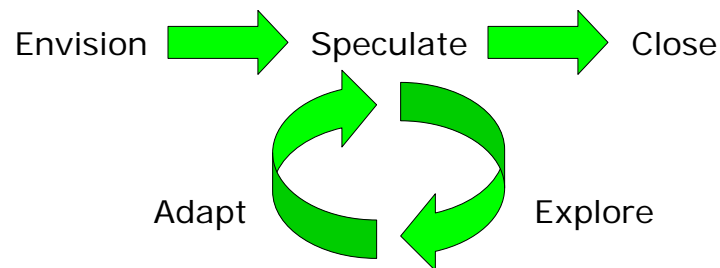
- Analysis leads to convergence on a single answer
- Assumes high degree of stability in environment
- Requires early choices without concrete feedback

### ■ Chaos Theory

- Complexity creates convergence on probability
- Learning in complex environment spawns emergence
- Software development complex enough to be chaotic
- We will discard significant amount of predictive plans

## “Agile” Planning

- Set goals, continually steer to best outcome
- Progressive refinement based on feedback
- Low WIP, short nested cycles, iterate on plans
- Highsmith cycle:



## Levels of Agile Planning

### ■ From a time perspective:

- Business Cycle
- Product Cycle
- Product Release Cycle
- Development Milestone (integration event)
- Development Cycle (iteration / sprint)
- Construction Cycle (synch event, daily scrum)

## Levels of Agile Planning

### ■ From a granularity perspective:

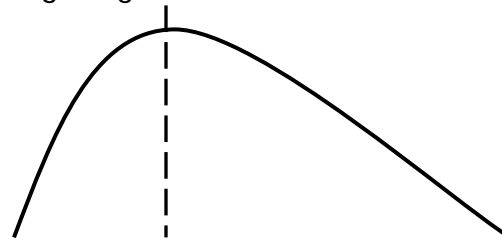
- Product Capability (features sets / areas)
- Feature (related capability cluster)
- Feature Capability (story)
- Work Element (task cluster)
- Work Task
- Work Episode

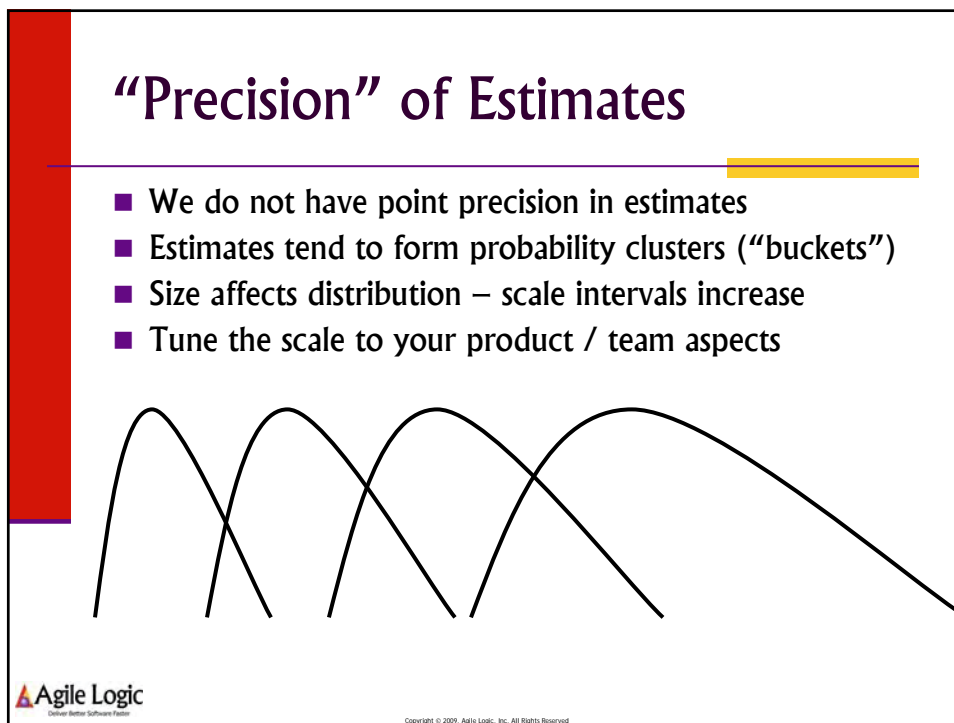
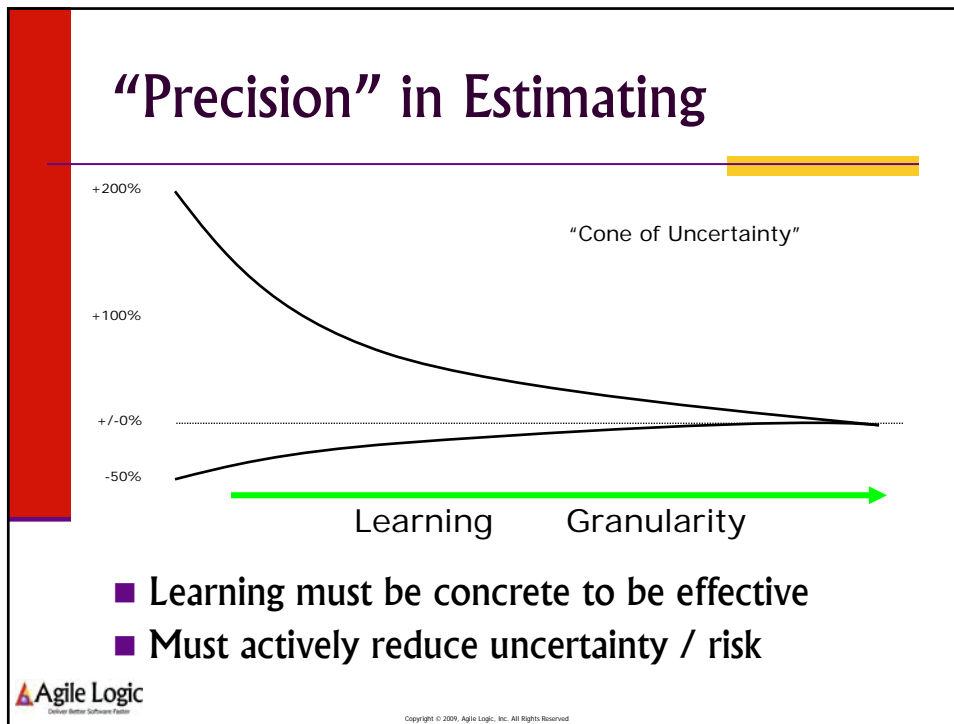
## Keys to Making Agile Planning Work

1. Use probability and statistics in our favor
2. Know what we're delivering
3. Establish our knowledge base
4. Leverage team collaboration and expertise

## Estimates Are Just That...

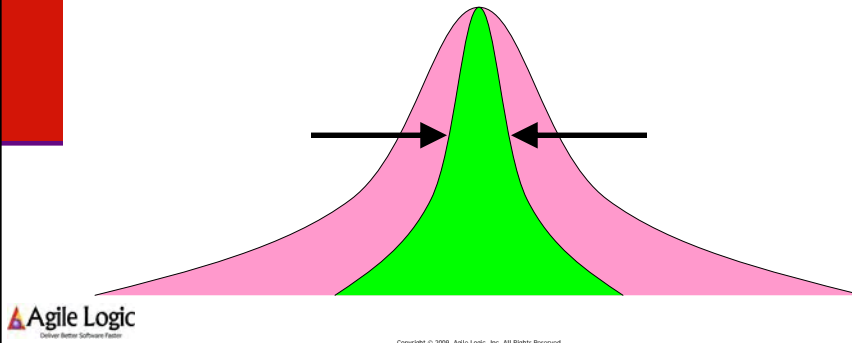
- Estimates are guesses, some better, some worse
- Each estimate has a probability of correctness
  - "Log Normal" probability
  - Complexity of software biases to it more often taking longer than less time





## Central Limit Theorem

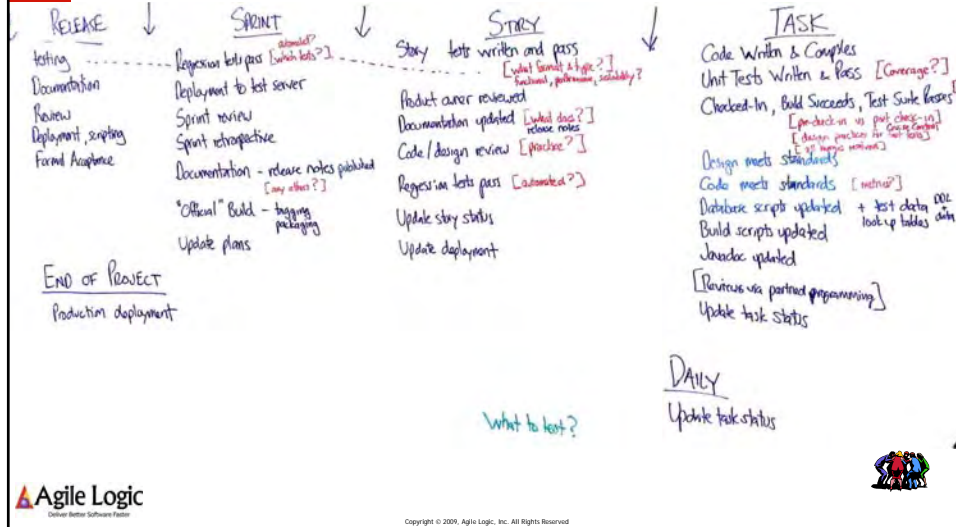
- The more samples we have, the more the overall result converges on an attractor
- The more items we estimate, our overall estimation variance will decrease and stabilize



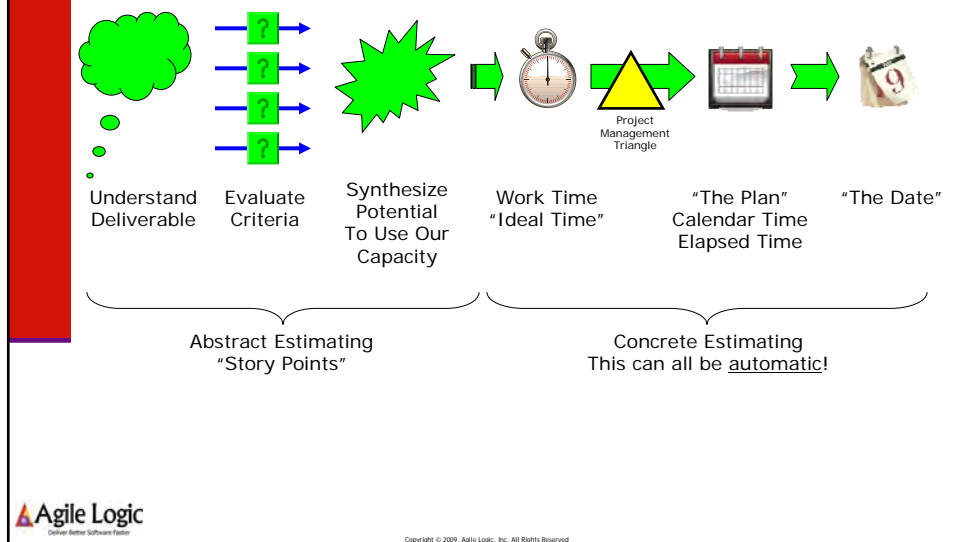
## 2. Know What We're Delivering

- We need a common understanding of “done”
- Consider each gating point:
  - Task Completion
  - Story Completion
  - Iteration / Sprint Completion
  - Release Completion
- Build a checklist of what's expected at each

## Example Completeness Criteria



## Abstract Estimating





## Evaluating Criteria

- Goal is to generate a rating of “potential to consume our capability”
  - Ratings must have relative size
- What’s needed to be effective?
  - Understanding of the item
  - Understanding of development strategy
- Some prep may be needed
  - Collaborative exploration (“meet and greet”)

## 3. Agile Estimating Knowledge Base

- Keep a team criteria check list
  - Draw on the broad team expertise
  - Use it for cross-learning
- Keep the criteria list relevant
  - Use retrospectives to explore exceptions
  - Metrics of estimates vs. actuals are not as relevant as root cause analysis



## 4. “Wide-Band” Estimating

- **Estimation by limited individuals is problematic:**
  - Smaller knowledge base to draw on
  - Less discussion, more potential for blind spots
  - Lack of buy-in from those responsible to deliver
- **We want wider involvement when estimating**
- **But larger group activities have issues:**
  - Hard to reach a group decision
  - Easy to drift off to tangent issues
  - Difficult to gain participation of everyone
- **We need a protocol to control the activity**

## “Planning Poker”

- **“Wide-band” estimating technique**
- **Each team member gets a set of estimate cards**
- **Go through the backlog items, one-by one:**
  1. Discuss and understand each one
    - Ensure it's “good”
    - Ensure we have a strategy for implementing
    - Consider criteria, synthesize
  2. Each team member simultaneously reveals rating
  3. Discuss highs and lows, until alignment reached
- **More useful when abstraction level is higher**
- **Less useful when items are more specialized**



## Rate the “Dog-ness”



Labrador Retriever

## Rate the “Dog-ness”



Dachshund

## Rate the “Dog-ness”



Great Dane

## Rate the “Dog-ness”



Bulldog

## Rate the "Dog-ness"



Poodle

## Rate the "Dog-ness"



German Shepherd



## Rate the “Airplane-ness”

### ■ Criteria (in rough order of influence):

- Size
- Speed
- Passenger capacity
- “Cool-ness”

## Rate the “Airplane-ness”



Boeing 737

## Rate the “Airplane-ness”



Cessna 150

## Rate the “Airplane-ness”



C-5A Galaxy

## Rate the “Airplane-ness”



Airbus A-380

## Rate the “Airplane-ness”



Sopwith Camel



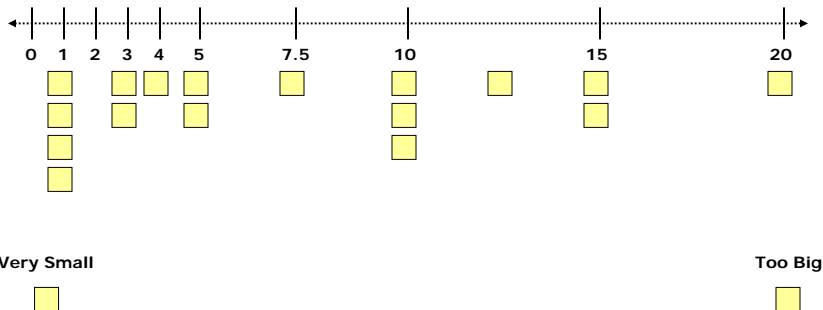
## Rate the "Airplane-ness"



SR-71 Blackbird



## "Wall of Wonder" Planning



- Make sure relative sizes feel right
- Set a cut-off size and stick to it
- Set a time-box for discussing each item

## Steps for Developing an Agile Plan

- **Generate goals and target items (backlog)**
- **Estimating size of items**
  - Abstract vs. concrete estimates
- **Estimating capacity to deliver**
  - Velocity (size units per time box)
- **Prioritizing items**
  - Rank ordering is the most concrete
- **Mapping to a plan**
  - Dates are mapped, not estimated!

## Release Planning

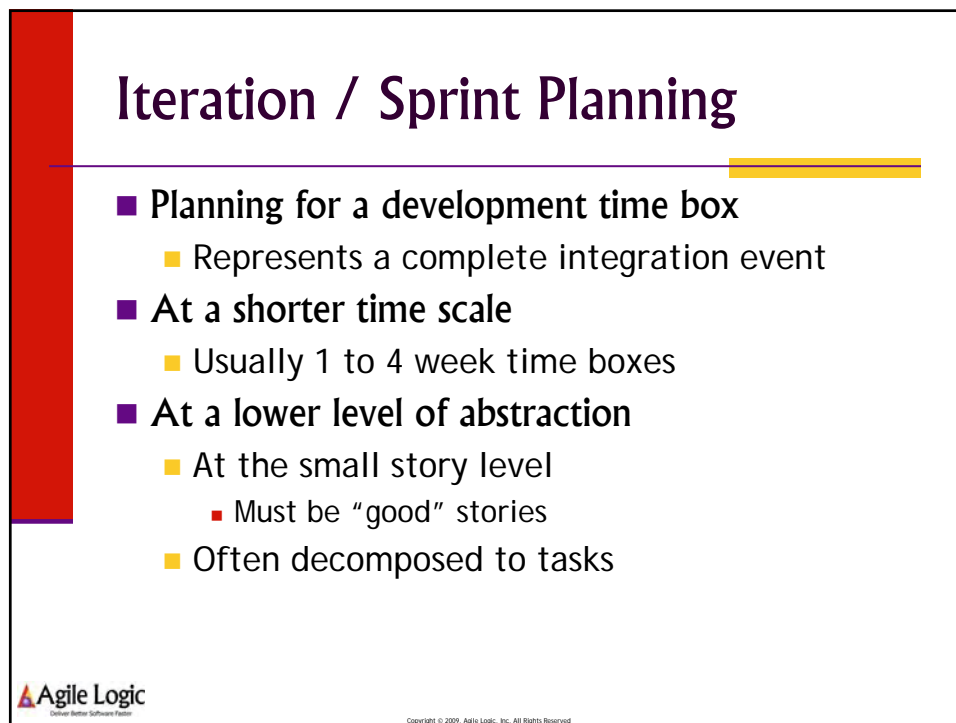
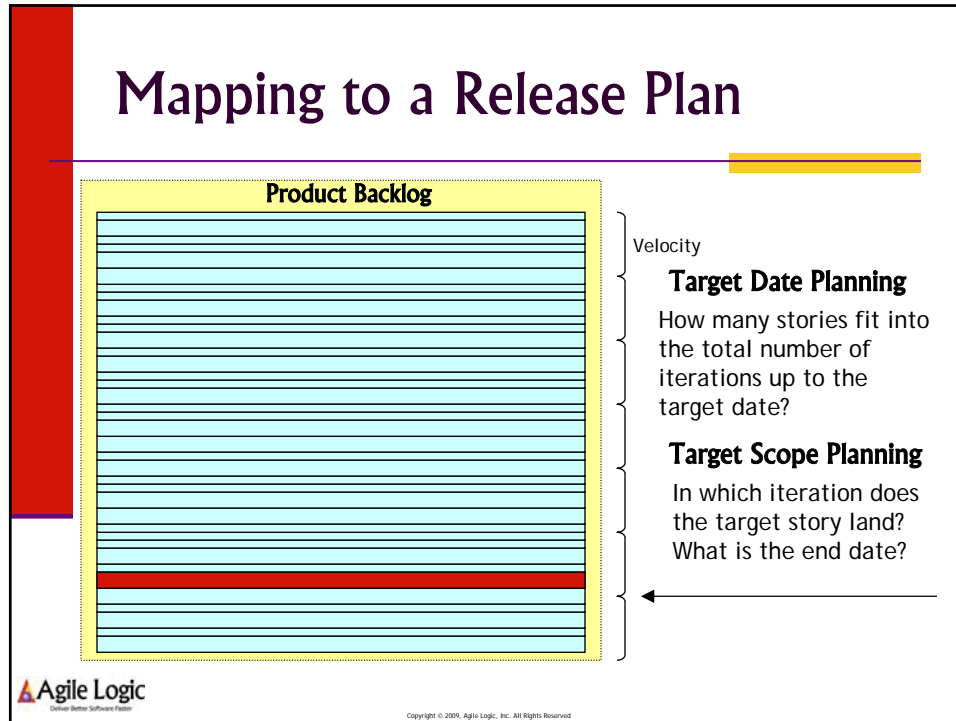
- **Planning for release milestones from the team**
  - Could be either internal or external release
- **At a longer time scale**
  - Usually multiple iterations / sprints
- **At a higher level of abstraction**
  - At the feature (epic story) / story level

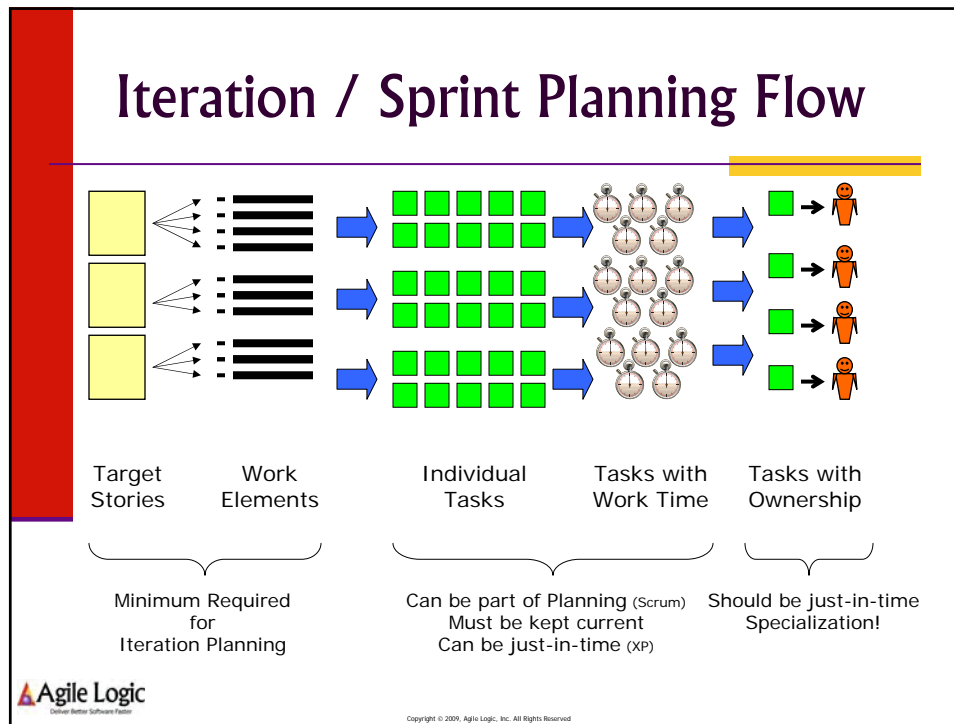
## Estimating Velocity

- First velocity estimate is a guess
- Gain additional feedback from first iteration / sprint planning session
- Actual iteration / sprint results establish actual velocity feedback
- For ongoing sprints:
  - Use “yesterday’s weather”
  - Moving average of prior actual velocities
  - Velocity should stabilize
    - But not if team or project is changing

## Prioritizing Product Backlog

- We need to break the “all or nothing” mindset
- Priority is based on many factors:
  - Value of the capability to stakeholders
  - Relative cost vs. value (ROI)
  - Risks and unknowns
  - Dependencies from sagas or epics
- Mixed priorities may require splitting
- Backlog items are sorted by priority
- Sequencing reflects incremental strategy





## Task Breakdown and Sizing

- **Target tasks to deliver something concrete**
  - Consider milestones toward story complete
  - Specialization forces specific breakdowns
- **Make tasks useful for tracking**
  - Tasks should be completed in about a day
  - Breakdown larger task clusters
- **Task estimates typically in “ideal hours”**
  - Consider completeness criteria
  - Limited wide-band estimating can be useful

Agile Logic  
Deliver Better Software Faster  
Copyright © 2009, Agile Logic, Inc. All Rights Reserved

## Estimating Capacity

- **Capacity to deliver is not just available hours!**
  - Velocity should drive committed work
  - But hour capacity data can be useful
- **Create a matrix (spreadsheet):**
  - Each team member's available days
  - Standard hours per day (consider overhead)
  - Adjustment for shared / part-timers
  - Calculate each team members hours, total
  - Specialization forces additional calculations

## Sample Capacity Matrix

Team Member	Available Days	Hours per Day	% Dedicated	Total Hours
Fred	10.00	6.50	100%	65.00
Sanjiv	10.00	6.50	100%	65.00
Carla	10.00	6.50	50%	32.50
Jose	8.00	6.50	100%	52.00
Wu	10.00	6.50	100%	65.00
Bob	10.00	6.50	100%	65.00
Dimitri	10.00	6.50	33%	21.45
				<b>365.95</b>

## Prioritization Within Iterations

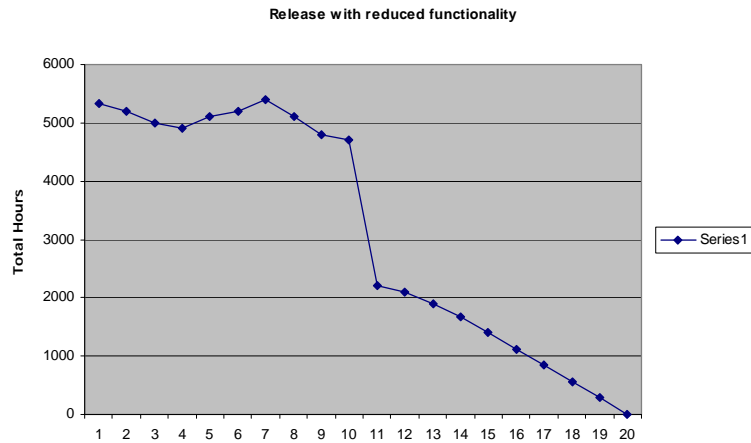
- **Prioritization should follow story priority**
  - Team may optimize ordering within iteration
- **But be careful of risking completing stories**
  - Watch out for “fan-out” (“Scrummerfalls”)
  - Team should “swarm” on each story in turn
  - Variance should be limited to one story



## Incorporating Feedback

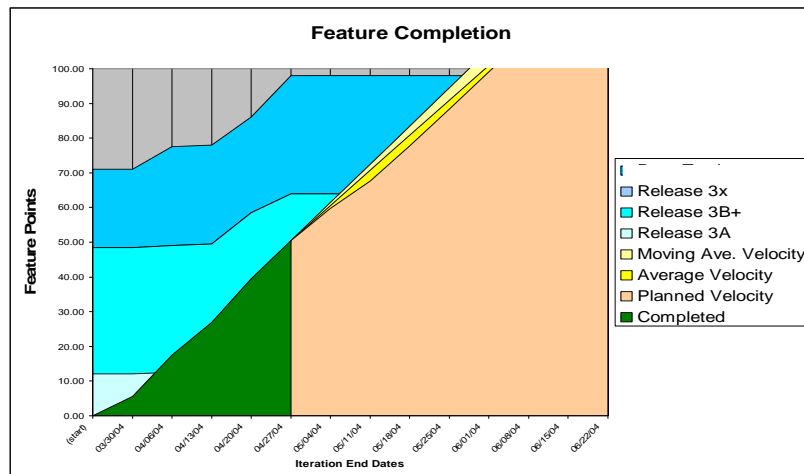
- **Iteration / Sprint Plan Feedback**
  - Stories completed vs. stories committed
  - Task completion vs. estimated tasks
  - Actual team member availability
- **Release Plan Feedback**
  - Actual velocity for each sprint
  - Actual story size vs. estimated size
  - Changes to other stories from feedback

## Tracking: Basic Burndown Chart



Copyright © 2009, Agile Logic, Inc. All Rights Reserved

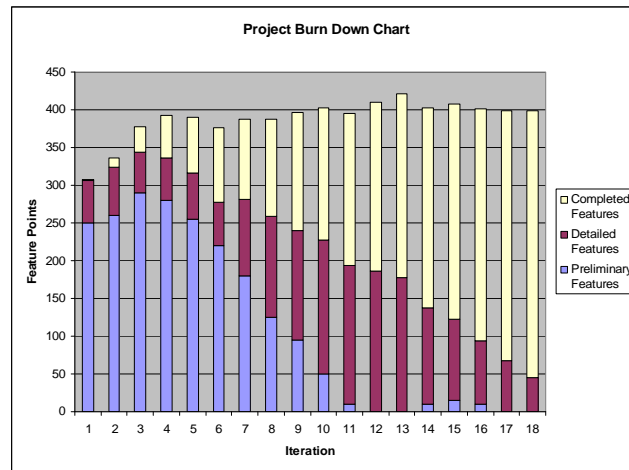
## Tracking: Burn-Up Chart w/Trends



Copyright © 2009, Agile Logic, Inc. All Rights Reserved



## Tracking: Burndown w/Status



# Thank You For Attending!

Presentation materials will be available in the Resources section of [www.agilelogic.com](http://www.agilelogic.com)

Paul Hodgetts  
Agile Logic  
[www.agilelogic.com](http://www.agilelogic.com)  
[phodgetts@agilelogic.com](mailto:phodgetts@agilelogic.com)  
(714) 577-5795