Measuring True Agility in Agile Software Development

Bob Moore
Senior Principal Process Engineer
www.biztransform.net

What is Agile Development?

- Agile development is about satisfying customer needs and wants faster.
- According to the Agile Manifesto (http://agilemanifesto.org/principles.html):
  - "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software. "

The Theme Of This Presentation

To be agile, development projects need to:

1. Deliver value to the customer quickly

AND

2. Deliver value to the customer while minimizing rework, waste, fear, and project turbulence.

How Do We Measure Agile Success?

- An (imaginary) typical agile developer would say that agile success is measured in terms of how early and often we deliver value to the customer.
- The agile developer would also ask, "How does measuring 'success' contribute to delivering value to the customer?"
  - The (sometimes) unspoken assertion being made is that if the act of measuring gets in the way of delivering value or does not contribute to value, then an agile project should not be doing it!

Good Agile Metrics

- In Refactoring to Agility, Carol Wellington suggests these rules for using metrics on agile projects:
  - **Metrics that fit on a napkin**: "Every metric should be simple enough that you can explain its computation on a napkin in the cafeteria."
  - **Integrated data collection**: As much as possible, gather data as part of other activities, not as a "metrics collection" practice.
  - **"Rough numbers are good enough"**: data being collected should be precise enough to support the analysis needed and no more.
  - **Metrics that lead to decisions or change**: metrics whose value is always known in advance, metrics that are always 0 or 1 (0% or 100%), and metrics that never change are not useful.

Measuring Speed When We Really Mean Agility

- Agile development projects are sometimes asked:
  - "How responsive are you to your customer?" or
  - "How fast do you meet your customers needs?"
- The answer given usually focuses on delivery *speed*.
  - Speed is the time from a customer need being identified to the need being satisfied by delivery of a feature.
- Speed might not be a *complete* answer.
  - After all , if customer needs are fixed and known at the start of the project, then a waterfall development life cycle will be faster than an agile life cycle—the waterfall project simply plans and implements!
  - Agile projects welcome change—which means we cannot just measure speed. We *also* must measure adaptability to change.

What is the Difference Between Speed and Agility?

- Speed: To go, move, or proceed quickly (http://www.thefreedictionary.com/speed)
- Agile: Characterized by quickness, lightness, and ease of movement (http://www.thefreedictionary.com/agile)
- In agile software development, these definitions might be translated as:
  - Speed: rapidly fulfilling requirements
  - Agile: adjusting to changes rapidly

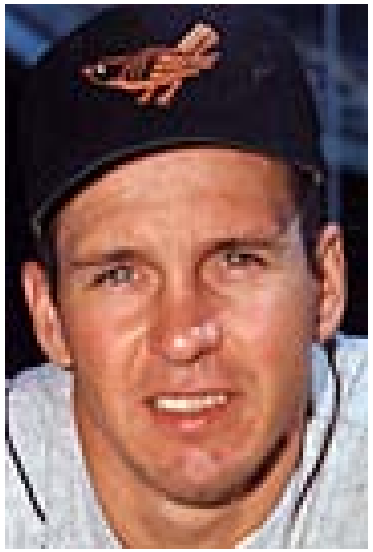
Why Should We Care About Agility?

- From the Agile Manifesto:
  - "(We) welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. "
- An agile development team needs to be able to nimbly adjust to changing customer needs and wants.
  - Requirements—customer needs and wants—change. **Period**.
  - Axiom from Experience: requirements change just when you have the software that implements the requirements completed.
  - Delivering value to the customer quickly is good.
  - Delivering value to the customer quickly while minimizing rework, waste, fear, and project turbulence is better.

Business Transformation Institute

Agile: A Baseball Example

An example of agile but not speedy—retired Baltimore Orioles third-baseman Brooks Robinson:



- Won more Gold Gloves at that position than any other third baseman (16) and tied for the most Gold Gloves at any position.
- Played in four World Series,
- Played on 18 consecutive All-Star teams,
- Beat out Mickey Mantle for the 1964 American League MVP,
- Set MLB records at third base for: seasons (23), fielding percentage (0.971), games (2,870), putouts (2,697), assists (6.205) and double plays (618).

Umpire Ed Hurley: "He plays third base like he came down from a higher league."

Holds the MLB record for batting into the most triple plays (four) in history.

Agile: Lean Concepts

- Agile development is not by itself in recognizing the value of agility.
- Lean methods provide some helpful insight into "agile":
  - Takt Time:
    - From Continuous Flow Manufacturing (CFM): "the maximum time per unit allowed to produce a product in order to meet demand "
    - Agile Development translation: "the time needed to implement features to deliver value to a customer"
  - Single-Minute Exchange of Die:
    - From CFM: change-over from manufacturing one product to another in less than 10 minutes.
    - Unfortunately, SMED is a goal to be achieved, not a measure to be analyzed.

Baseball and Agile Software Development

- Taking a cue from Brooks Robinson, let's look for agile opportunities in software development.
  - We are looking for places where agility (quickness of understanding and response) is important, not just speed.
  - Brooks Robinson's agility came from understanding and being able to react to the dynamics of pitching, batting, and the interaction of the ball with infield grass.
  - Brooks Robinson's success did not result from fleetness of foot!

Software Development Life Cycle Opportunities for Agility

- If insight and understanding are the key to agility, where in the software development life cycle might we increase our agility?
- A simple agile development life cycle:
  - Customer identifies need
  - *Team understands customer need (repeat as required)*
  - *Team plans work needed to implement customer need (repeat as required)*
  - Team develops, tests, and integrates software code to implement features satisfying customer need
  - Team demonstrates feature
  - *Team performs any rework needed to satisfy customer*
  - Customer accepts feature

Accentuating Agility in Development

- Anticipating changes and avoiding rework is the key to agility.
- In software development, our understanding of the problem comes from understanding the customer's needs and planning the work in response.
  - Project agility increases when:
    - We understand what the customer needs and can anticipate where the customer is going.
    - We make our plans focused enough to deliver but flexible enough to adapt.
  - Project agility decreases when:
    - We have to perform rework because we did not understand or anticipate the customer's needs.

How Do We Measure Agility? Terms

- Time stamp indicators:
  - T_{start} = time stamp when customer identifies need
  - T_{done} = time stamp when customer accepts the function(s) that satisfy the need
- Durations:
  - $T_{deliver}$ = total time spent from when the team accepts the customer requirement to when the function(s) corresponding to that requirement is delivered and accepted by the customer
  - T_{accept} = number of deliveries or demos to the customer needed for the customer to accept the function(s) corresponding to the requirement
  - $T_{life\ cycle}$ = standard duration of each agile delivery cycle

Business Transformation Institute

How Do We Measure Agility? Definition of ReCycle Time

- Standard delivery time = $T_{done} - T_{start}$
- ReCycle Time =

$$\left\lfloor \frac{T_{deliver}}{T_{life\ cycle}} \right\rfloor \times T_{accept}$$

$\lfloor a \rfloor$ = integer part of a

$\lfloor 3 \rfloor = 3.$ $\lfloor 4.5 \rfloor = 4.$ $\lfloor 0.5 \rfloor = 0.$

The Napkin Definition of ReCycle Time

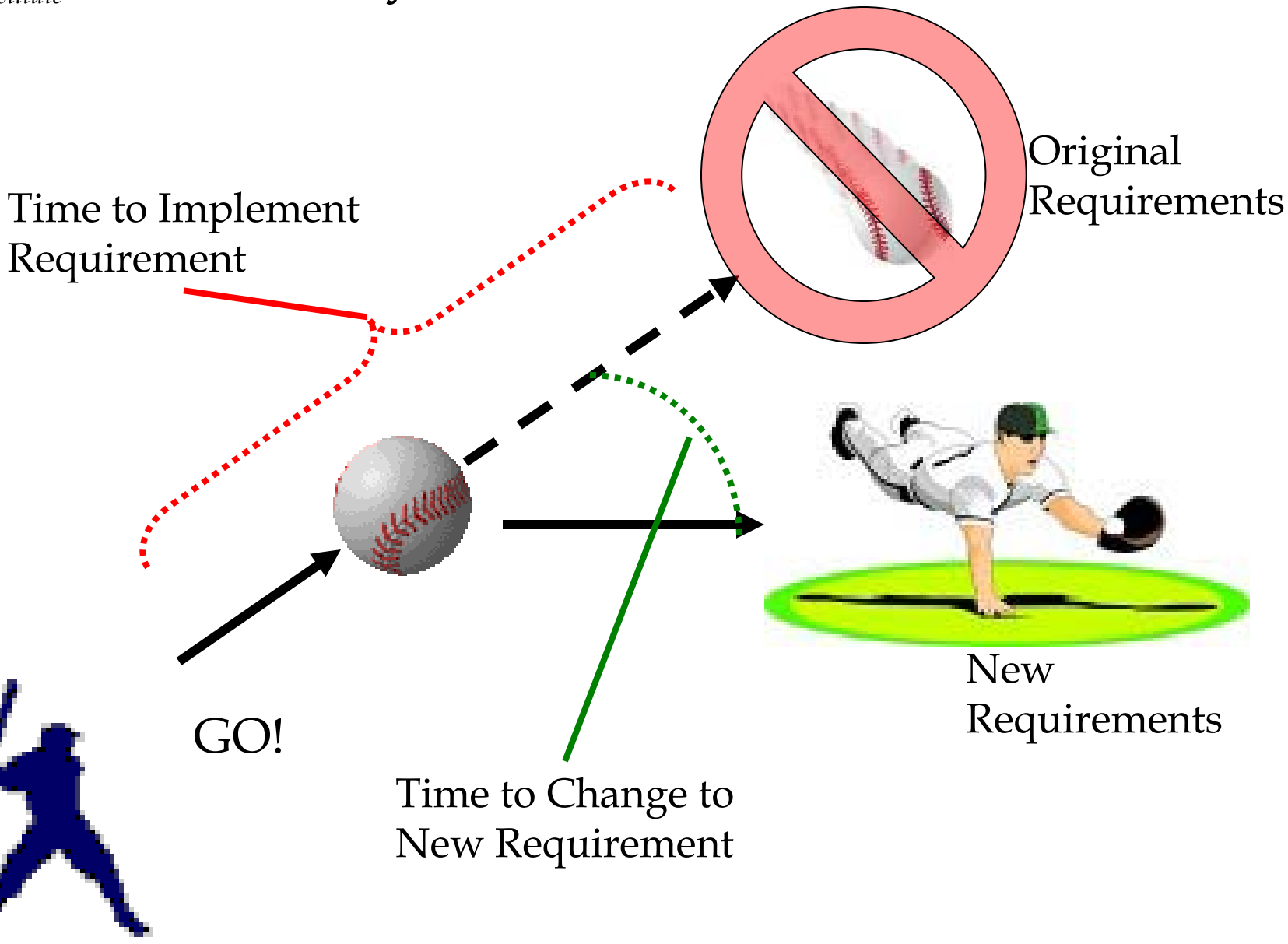
ReCycle Time =

- The number of *full* development cycles needed to deliver, measured from the time when you accepted the customer's requirement
  multiplied by
- The number of times you deliver what *you* think is the right solution before the customer thinks it is the right solution too.

- ReCycle Time is a dimensionless number.
  - A systems engineer would call it a "figure of merit".

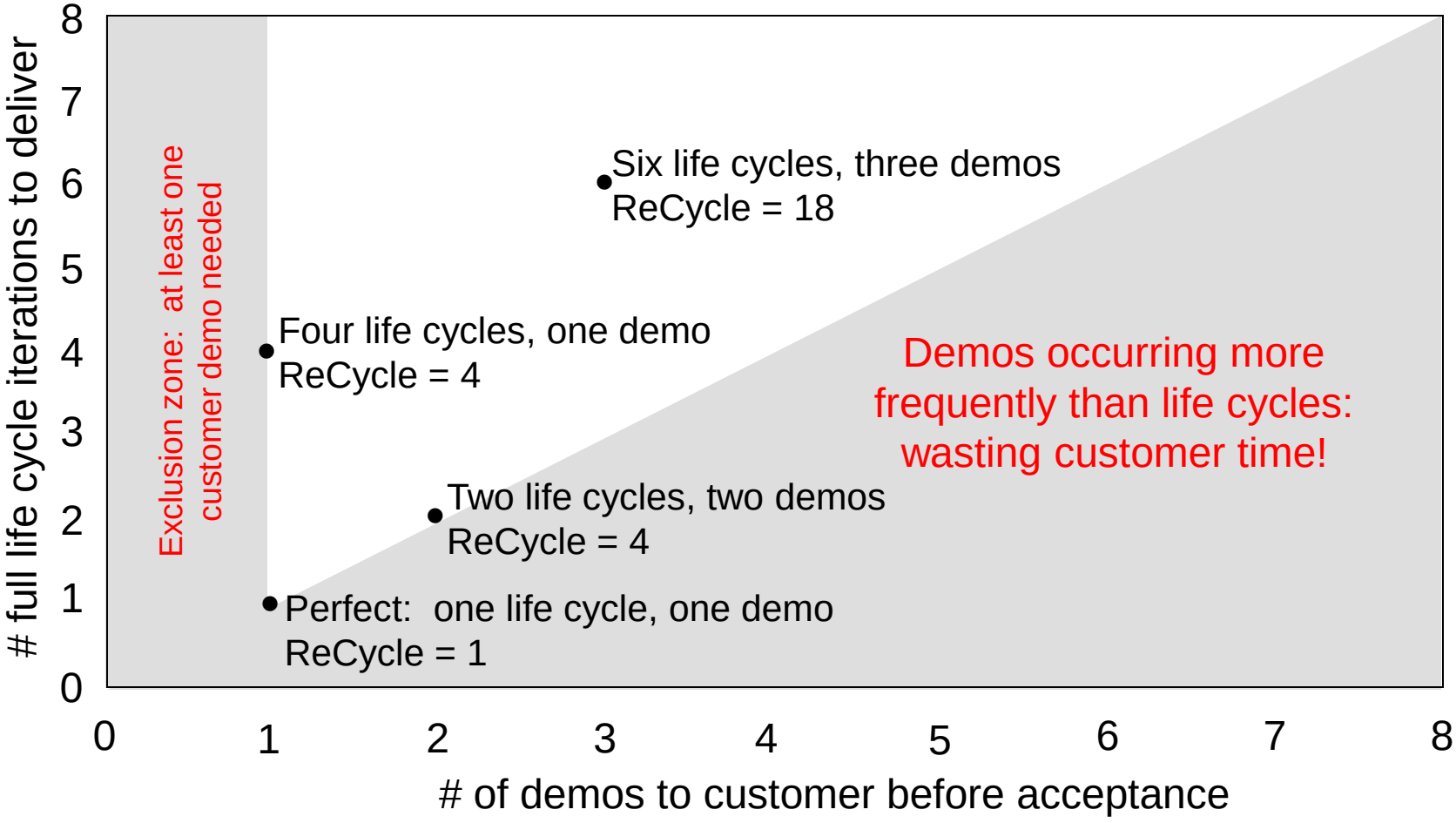
Business Transformation Institute

ReCycle Time Illustrated



Business Transformation Institute

How ReCycle Works



Notes on ReCycle Implementation (1)

- The ReCycle Time clock for either life cycle iterations or number of demos does ***not*** restart when the customer corrects (or changes) the statement of an existing requirement.
  - The focus of ReCycle time is *agility* – the customer is expected to change requirements.
  - Customer changes may increase the number of demos or the number of life cycle iterations needed—the team has to be agile in dealing with these changes.

Notes on ReCycle Implementation (2)

- ReCycle time presumes the ability to track customers' individual requirements from acceptance to completion.
  - Only tracking how often new functions are delivered without linkage to requirements is not sufficient.
  - ReCycle time depends on an honest accounting of when the team accepts a requirement and when the customer accepts the implementation.

Notes on ReCycle Implementation (3)

- Both team development efficiency and communications with the customer during development are important:
  - Team efficiency drives the number of implementation life cycle iterations that are needed.
  - Customer communications during development drives the number of customer demos that will be successful the first time.
  - Customer communications also drives anticipation of customer needs—increasing team agility.

A Case Study (1)

- An agile development organization changed their delivery life cycle from quarterly feature delivery to bi-weekly delivery.
  - The focus was reducing time to market.
  - The data presented is from a real organization whose identity is withheld.
- The organization measured time from the start of development to delivery to the customer for each delivery cycle:
  - Delivery A: 4 months
  - Delivery B: 5 months
  - Delivery C: 8 months
  - Delivery D: 6 months
  - Delivery E: 5 months
  - Delivery F: 6 weeks
  - Delivery G: 5 weeks

Case Study (2)

- On first glance, the organization seems to be delivering with faster time to market.
  - Looking at average completion time for individual requirements paints a different picture!
- Time from requirement accepted to feature delivery and acceptance:
  - Delivery A: 89 days
  - Delivery B: 88 days
  - Delivery C: 92 days
  - Delivery D: 89 days
  - Delivery E: 87 days
  - Delivery F: 90 days
  - Delivery G: 89 days

Case Study (3)

- How did the organization do on ReCycle time?
  - Although the organization made more frequent deliveries, requirements persisted across many development cycles.
  - The rejections by customer and resulting rework went up slightly.
  - ReCycle:
    - Delivery A: 1
    - Delivery B: 1
    - Delivery C: 2
    - Delivery D: 2
    - Delivery E: 1
    - Delivery F: 6
    - Delivery G: 4

Case Study Conclusions

- The organization made more frequent deliveries.
  - This may have benefited the customer by providing a more frequent flow of new features.
- Individual requirements were not fulfilled any faster.
  - No benefit: any given customer requirement was not fulfilled any sooner or later.
- Customer rejection and developer rework increased.
  - The organization was less likely to satisfy customer requirements the first time.

Techniques for Increasing Agility

- The following suggestions for increasing agility are made without proof, based on the cited reference:
- **Lean Software Development**, Mary and Tom Poppendiek:
  - Share partially complete design information.
  - Organize for direct, worker-to-worker collaboration.
  - Develop a sense of how to absorb changes:
    - Use modules
    - Use interfaces
    - Use parameters
    - Use abstractions
    - Avoid sequential programming
    - Beware of custom tool building

Business Transformation Institute

Contact Information

Bob Moore

Business Transformation Institute, Inc.

Phone: 410-997-1237

Email: rlmoore@biztransform.net