Mindtree

Welcome to possible

Productivity Planning Considerations in Industrial Agile – A Vendor View

Avinash Rao and Mayand Singh

Mindtree UK

Agenda

- **Defining Context and Terms**
- **Measurement Setup**

- **Continuous improvement Targets**
- **Plan v/s Reality**
- **Investigations**
- **Productivity and Rework**

- **Integration Across the Application Boundary**
- **Transaction costs for Integration**

- **Solution**
- **Learning Summary**

Defining Context and Terms

- “Industrial Agile”

- ~200 people in Agile teams
- Development and Testing delivering a target output every iteration
- Complete, integrated product increments delivered every 2 weeks
- Each team’ output to meet defined exit quality criteria for line coverage and coding standards for every release
- Multi-vendor environment, integration with platform utilities and other vendor code being developed

Measurement Setup

- Output contracted in FPs (Mark II)
- Estimate delivered to the client lead planner for every iteration on Day 1, and actuals reported on the last day of the iteration
- Contracted measurements with penalties for non-performance applied for each quarter

- Primary measurement: Productivity Quotient

- Continuous improvement expected ever every two quarters

Contracted PQ

- Payments / penalties linked to PQ performance
- Specific contracted definitions created to permit Rework commissioned by the client
- Defect levels contracted by Severity
- PQ ratio for a team in an iteration =

  Iteration time available / (fresh work + client commissioned Rework)

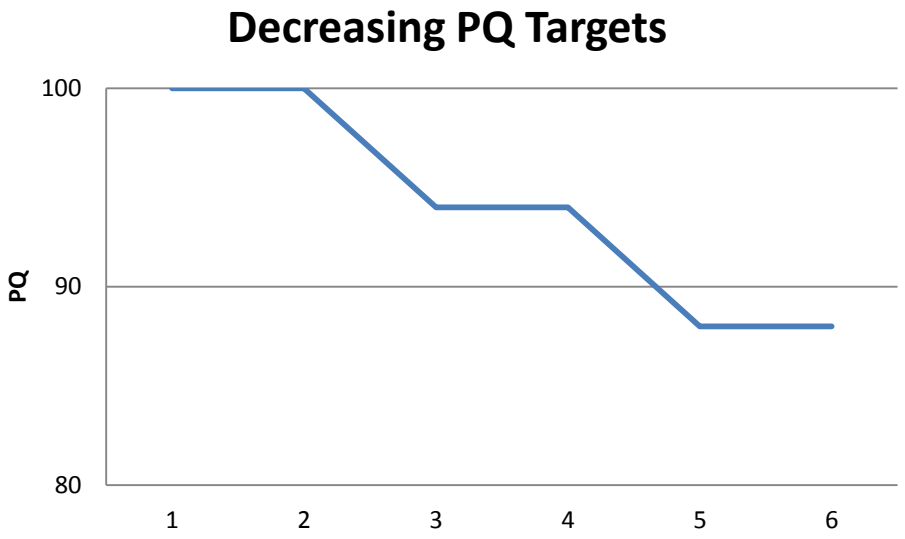
- Thus PQ not directly equated to product completion
- CQ expected to be tracked by the planners, no contracted level

A note on the data presented

- Normalized for output variations in teams, cumulative output
- Variations presented as a % change
- Absolute numbers are masked
- Rework is client- commissioned re-work, not including defect fixes or other development overhead

Continuous Improvement Target

- 6% improvement in PQ expected every 2 quarters
- Improvements expected as familiarity increases with the process and project domain and technology variants



Easy, right?

The PQ settled down after teething troubles …

- Overcoming the teething problems in the first quarter, we settled down to a successful second quarter and looked set for further improvement

	Q1	Q2
Performance against target	20% over target PQ	0.3% under target

The PQ settled down after teething troubles …

- Overcoming the teething problems in the first quarter, we settled down to a successful second quarter and looked set for further improvement

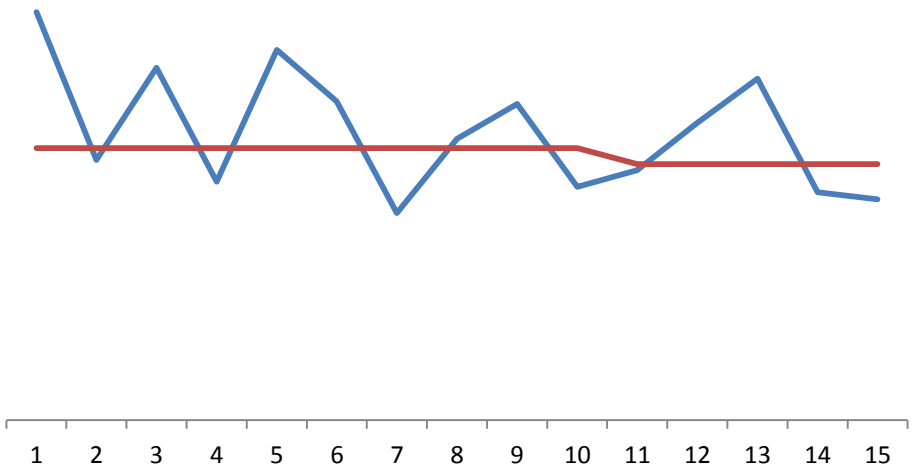
	Q1	Q2
Performance against target	20% over target PQ	0.3% under target

- However, with a reduced PQ target, no improvement in Q3 productivity

	Q1	Q2	Q3
Performance against target	20% over target PQ	0.3% under target	4% over target

… prompting us to take a closer look at the data

- Wide variability in output from individual iterations, plotted against target PQ



Investigating the variations

- Factors investigated to explain the variation included:
  - Complexity of work across iterations
  - Staff movement
  - Vacations and holiday patterns
  - Defects raised on output

- However, there appeared to be no difference in complexity of work handled, or in the teams that developed the functionality

Investigating the variations

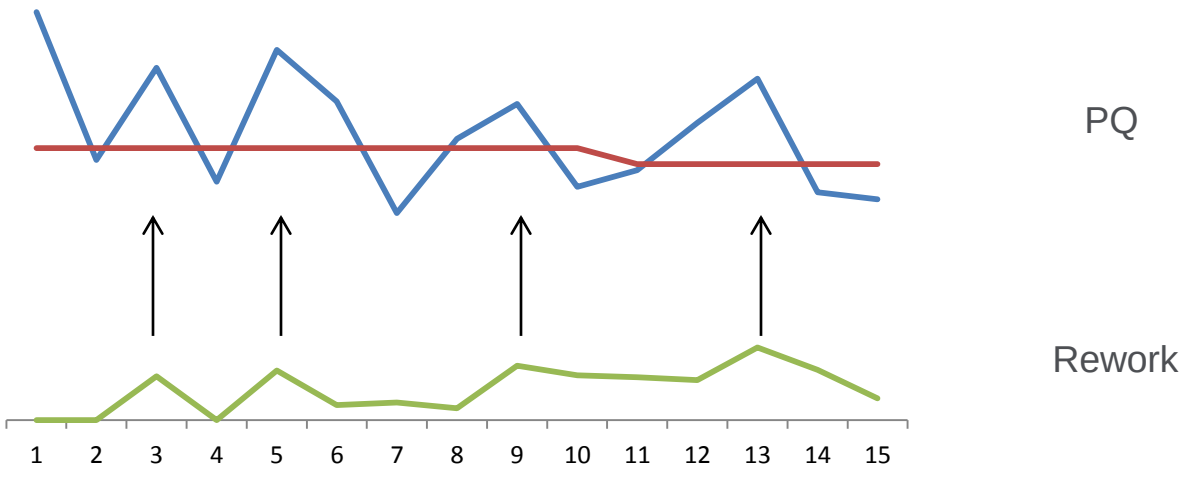
- Factors investigated to explain the variation included:
  - Complexity of work across iterations
  - Staff movement
  - Vacations and holiday patterns
  - Defects raised on output

- However, there appeared to be no difference in complexity of work handled, or in the teams that developed the functionality

- The scrum masters, however, complained that they had more trouble with iterations that touched code already written

Plotting Productivity against Rework

- Plotting PQ against Rework started providing clues to the cause of the variation



The Rework – Productivity link

- Why did Rework cause an increase in PQ?

➢ The nature of Rework induced was a collection of small changes into functionality developed already

➢ The overheads in an iteration – planning day, testing (unit, regression, automation), code compliance and deployment is common irrespective of the amount of functionality developed

➢ An average increase in Rework of 25% led to an average increase in PQ of 29%

➢ An interesting observation was that the delta in rework mattered

➢ Further, at a team level, rework resulted in fragmented idle time that could not be used; rework teams also waited more for other teams to complete

Using the Rework – PQ link into estimation

- This allows us to translate the amount of Rework FPs into an expected additional overhead in terms of effort

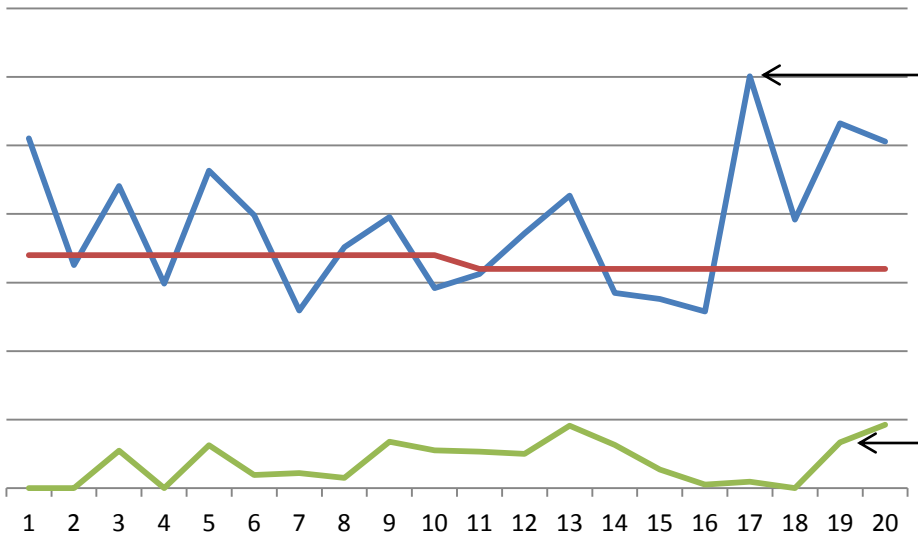
- **Learning: An ‘automatic’ penalty of is imposed on teams that perform fragments of Rework**
- In our case, the overhead was 20% for the teams that undertook rework

- We believe that any industrial Agile team will face this overhead for Rework; the exact amount will depend on the process and exit criteria used

- This effect has been consistent in the succeeding quarters

The next surprise

- Expanded view of the PQ to the following quarter:



But what is happening here?

Other data points confirm the rework penalty

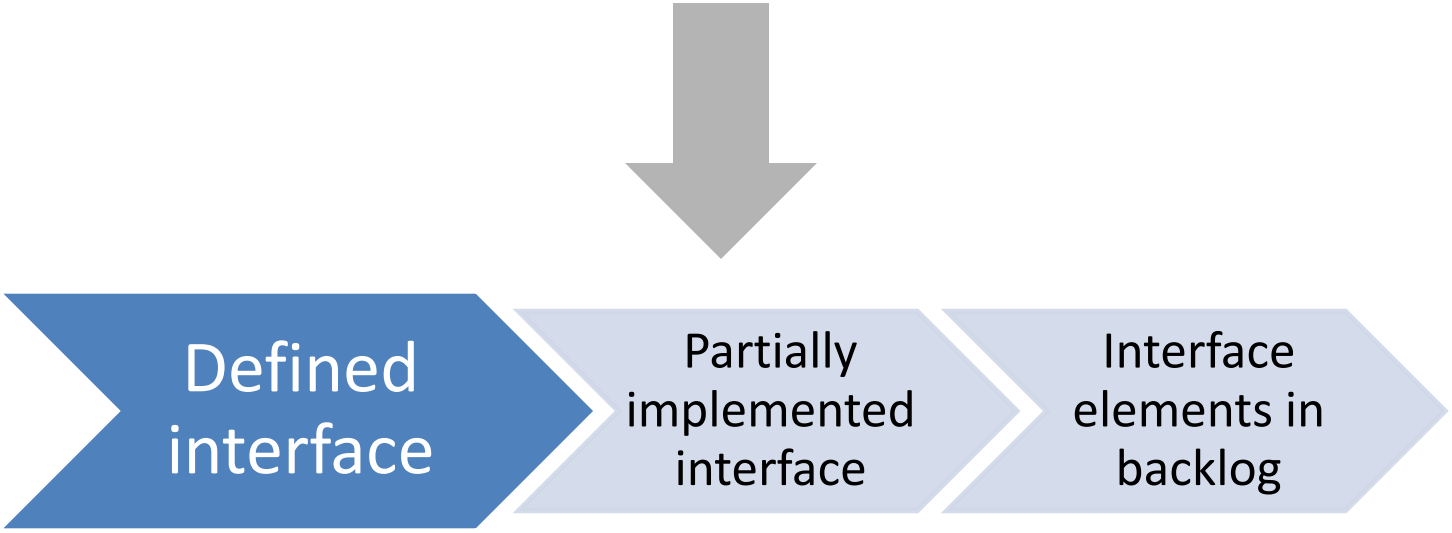
Integration Across the Application Boundary

- Fortunately, the cause of the spike was easier to identify
- The program involves integrations to another significant module; this iteration involved a significant change to an integration
- Per Mark II FP counting practices, both modules individually get credit for development; however, the integration itself provides 2 FPs
- What if the integration is across vendors?

Transaction costs for 2 FPs

- Integration across vendors (with both Applications evolving interfaces and interface data) incurs significant transaction costs
- This is compounded in Agile, where both vendors may develop only a part of the interface in each iteration
- Teams spend time in understanding the other vendor's code, testing and (finding and) fixing defects
- Re-releasing patches causes time loss on both sides

2 FPs are counted for the integration; however, the integration typically undergoes multiple rounds of changes to reach steady state
Especially in Agile, effort incurred is disproportionate to the FPs!



An ‘Agile’ integration?

- We believe that an Agile, iterative approach compounds the problem of cross application integration, if progress is measured in FPs

- Cross vendor ownership compounds the transaction costs

- An additional wrinkle is added by defects found mid-iteration on the other vendor’s code leading to spiraling effort without moving the product forward

Solution

- Agreement with the client on a ‘credit’ of ~ 20% when fragmented rework is undertaken

- Cross application integration carefully rationed, with upgrades only when there are significant upsides; backward compatibility provided
    - Still, we see defects and costs to maintain integration. This will stabilize as both parties stabilize interface design and code
- Credit for time spent in integration

Learning Summary

- In Industrial Agile, while measuring product progress in FPs:

✓ Rework imposes a overhead that causes reduction in FP output from Agile development teams

✓ This overhead will vary based on process and exit criteria followed

✓ Wait times are introduced as teams are not completely loaded with fragmented rework

✓ Transaction costs in terms of effort are incurred dis-proportionate to the FPs are incurrent in Agile integrations, especially cross-vendor

✓ If contract terms are based on FPs delivered, a mechanism to account for these effects should be agreed in advance!

Thank you!

Welcome to possible