goto; conference aarhus

MEASURING PROGRESS AND PERFORMANCE IN LARGE AGILE DEVELOPMENTS

Andy Carmichael
OpenXprocess Ltd

INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE

Disclaimer

This presentation is a personal view and does not represent the views of any individual or organisation other than the presenter.

Any similarity to any person, event or institution, living or dead, is merely coincidental.

Why do clients move to agile?

- Client 1
  - Reduce **errors** being delivered to Integration Test by 50%
  - Focus on automated unit test and build in component teams
- Client 2
  - Reduce **cycle time** (concept-to-cash) from 18 months to 13 weeks
  - Focus on requirements and project initiation
- Client 3
  - Improve ability to **deliver to plan**
  - Focus on scrum practice and reporting…
  - … and measurement of progress

The client's always right… right?

Which of you, desiring to build a tower, does not first sit down and count the cost?

St Luke (1st Century)

Not everything that can be counted counts.
Not everything that counts can be counted

William Bruce Cameron (1963)

The most important figures that one needs for management are unknown or unknowable, but successful management must nevertheless take account of them.

W. Edwards Deming (1986)

You can’t control what you can’t measure.

Tom De Marco (1982)

Control is not the most important aspect of software projects… Manage people. Control cost and time.

Tom De Marco (2009)

Projects can be big!

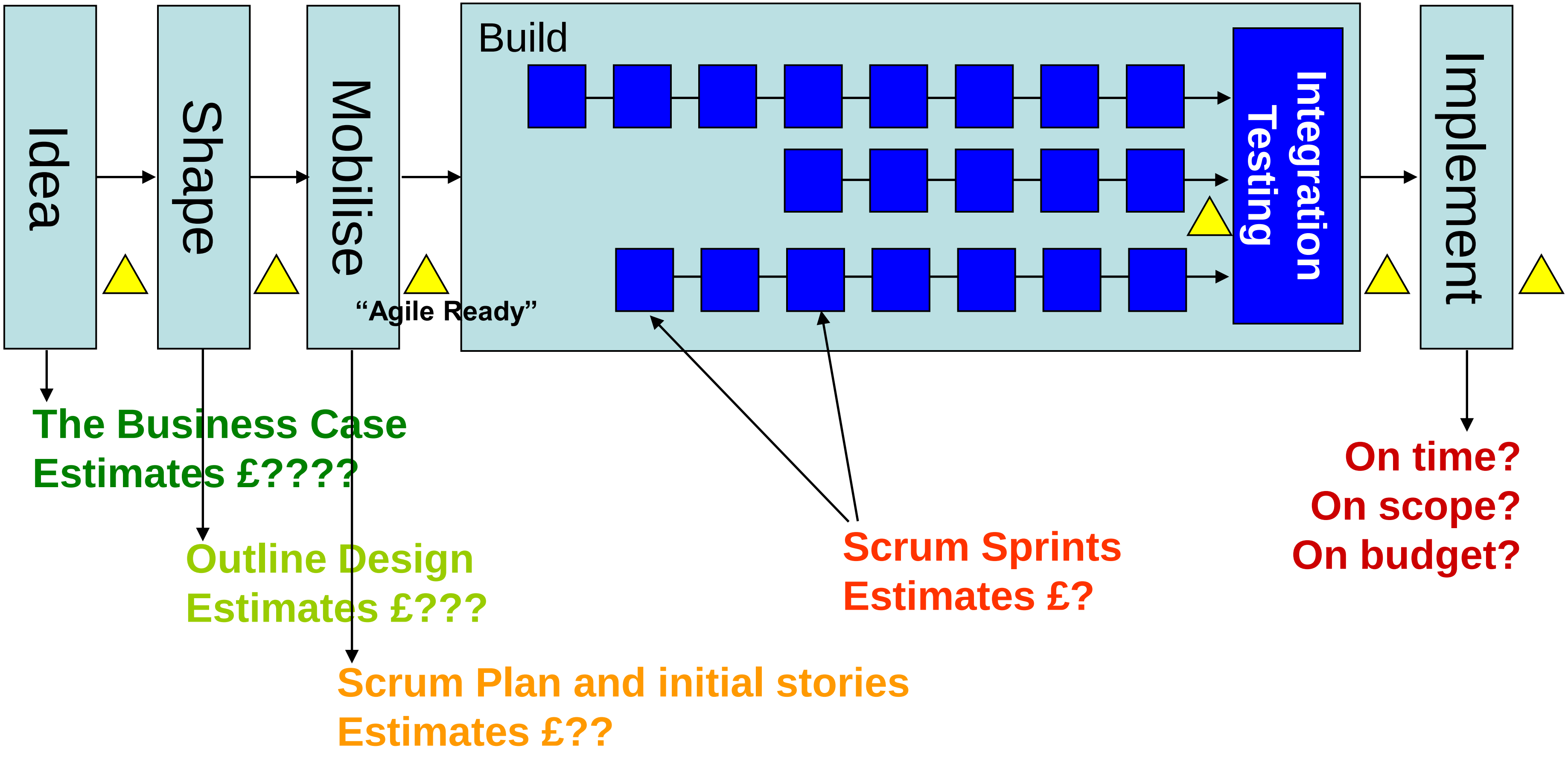
- A major energy company
- Over 100 scrums working in parallel
- SAP-based system(s)
- Managed through MSP, Prince II… and Scrum
  - ***"on scope, on time, on budget"***
  - *"maximise delivered value sprint by sprint"*
- Composite lifecycle (Scrum-Waterfall sandwich)
- Compressed early phases
  - specification, design and estimation of user stories is completed during scrum (Build) phase

Project Landscape

- Projects typically have between one and six scrum teams
- Projects are grouped in “Clusters”, typical one to four projects
- A Programme may have several Clusters
  - For example Catalyst has 6 Clusters and a total of around 60 scrums
  - Programme budgets may run to 9 figures (£) over several years
- The ability to deliver to initial estimates and control change is a key goal of management **???**

Hybrid Project Lifecycle

Compressed early Waterfall stages means specifications, estimates and design are *less* complete, allowing implementation to start earlier. This can accelerate delivery of business value but means scope is less defined and estimates are less accurate.



The Business Case
Estimates £????

Outline Design
Estimates £???

Scrum Plan and initial stories
Estimates £??

Scrum Sprints
Estimates £?

On time?
On scope?
On budget?

Goals of measurement

- show **progress** against a plan
  (to guide re-planning)
- show **process effectiveness**
  (to guide improvement)

5 Core Metrics*

1. **Time** (elapsed days / sprints)
2. **Budget** (£/€/$ or Man-Days)
3. **Scope** (normalised points)
4. **Quality**
5. **Value**

Ordered by difficulty of measuring accurately!

Measuring *Progress*

Focus on Essential Details First…

- **Scrum Teams** provide an agile plan with progress updated **every day** (less frequently if not automatically captured):
  - **Scope**
    - Current **backlog-size** estimated in points (*minimum / expected / possible* scope)
    - **Velocity**
      (Actual: previous sprints; Commitment: this sprint; Forecast: future sprints)
    - Forecast **% complete** at each milestone: versus *minimum / expected*
  - **Time**
    - Project Schedule (number and **dates** of sprints and milestones)
  - **Budget**
    - **Resource** Allocation / Cost profile
      (Actual and Forecast broken down by Test, Dev, SM, Other)

Goals for a single scrum team

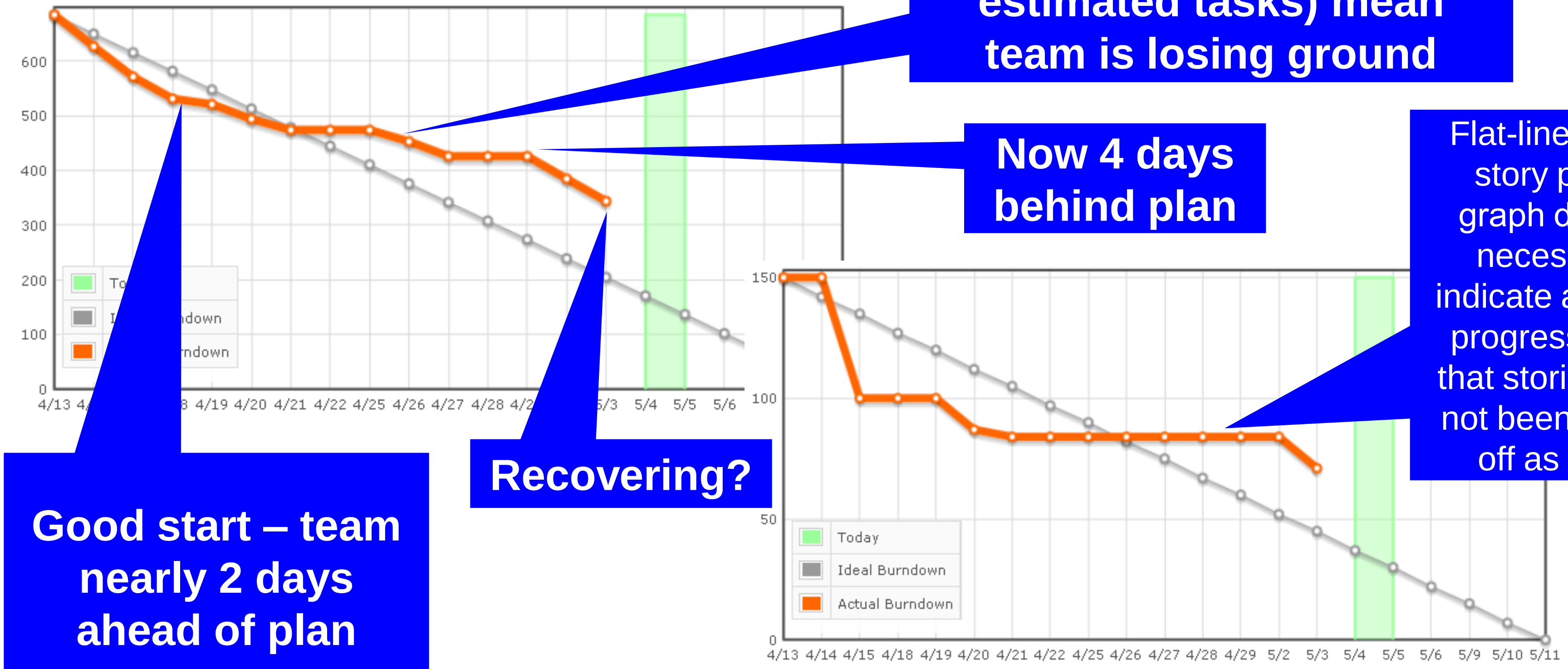
- Deliver *working software* as efficiently as possible
- Improve *process* wherever possible
- Be the *best team* to work in (and to have work for you)
- **Plan and forecast sprints well:**
  - **Achieve a roughly constant velocity as soon as possible**
  - **Be within 15% of the commitment**
  - **Beat the commitment *as often* as it beats you**
  - **Improve velocity whenever opportunities arise**

Answers the 2 key management questions for each scrum:

1. **How is *this* sprint going?**
   - Burndown of planned tasks (in hours)
     - If we complete all these tasks we *should* finish the stories
   - Burndown of user stories (in points)
     - These are the story points we’ve *actually* “done”
2. **How are we progressing against the planned delivery?**
   - Burn-up of user story points against estimated size of the product backlog and the number of sprints budgeted

1. How is *this* sprint going?

Burndown of planned tasks (in hours)



Unplanned tasks (or under-estimated tasks) mean team is losing ground

Now 4 days behind plan

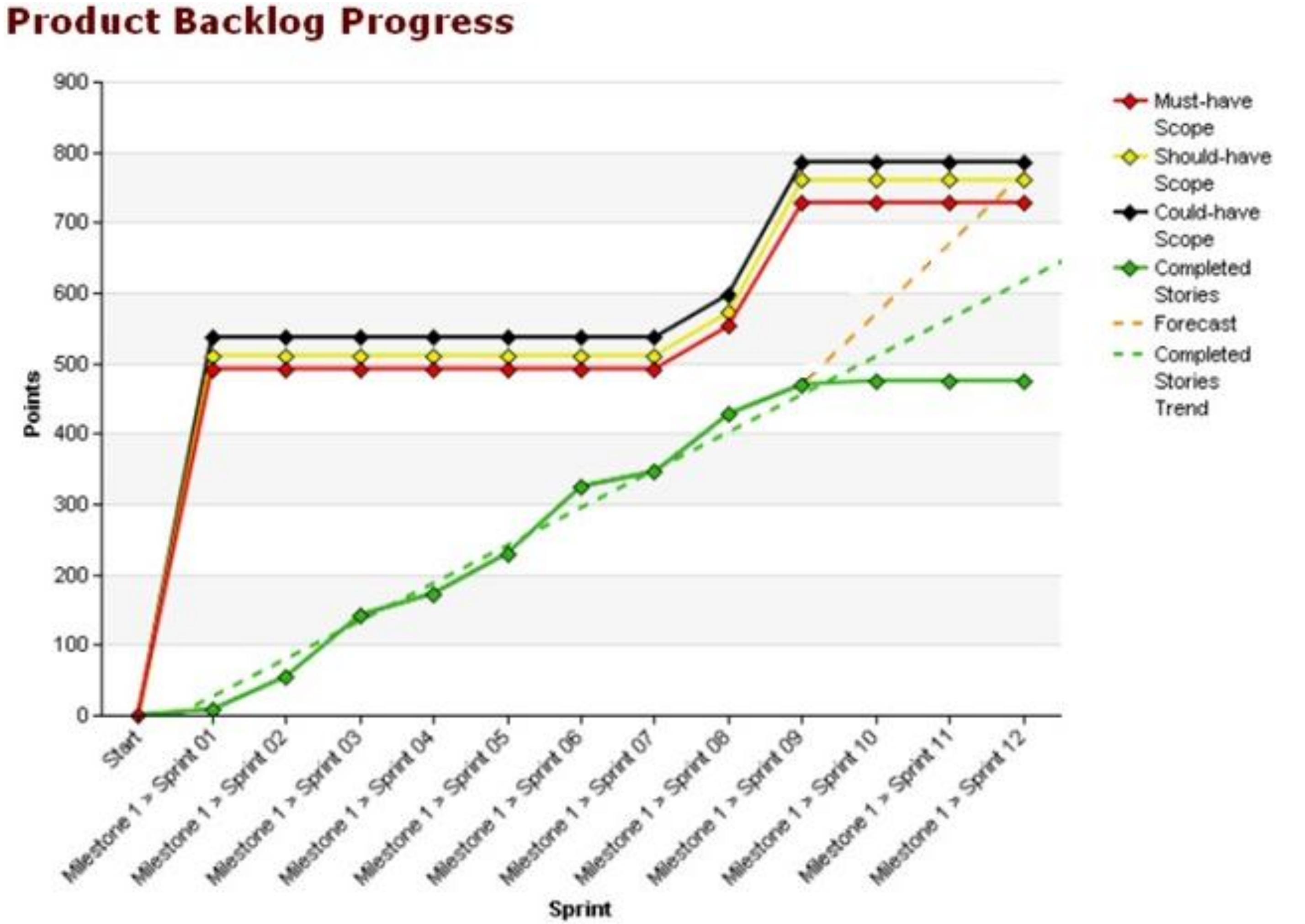
Flat-line on the story points graph doesn't necessarily indicate a lack of progress – just that stories have not been signed off as done

Good start – team nearly 2 days ahead of plan

Recovering?

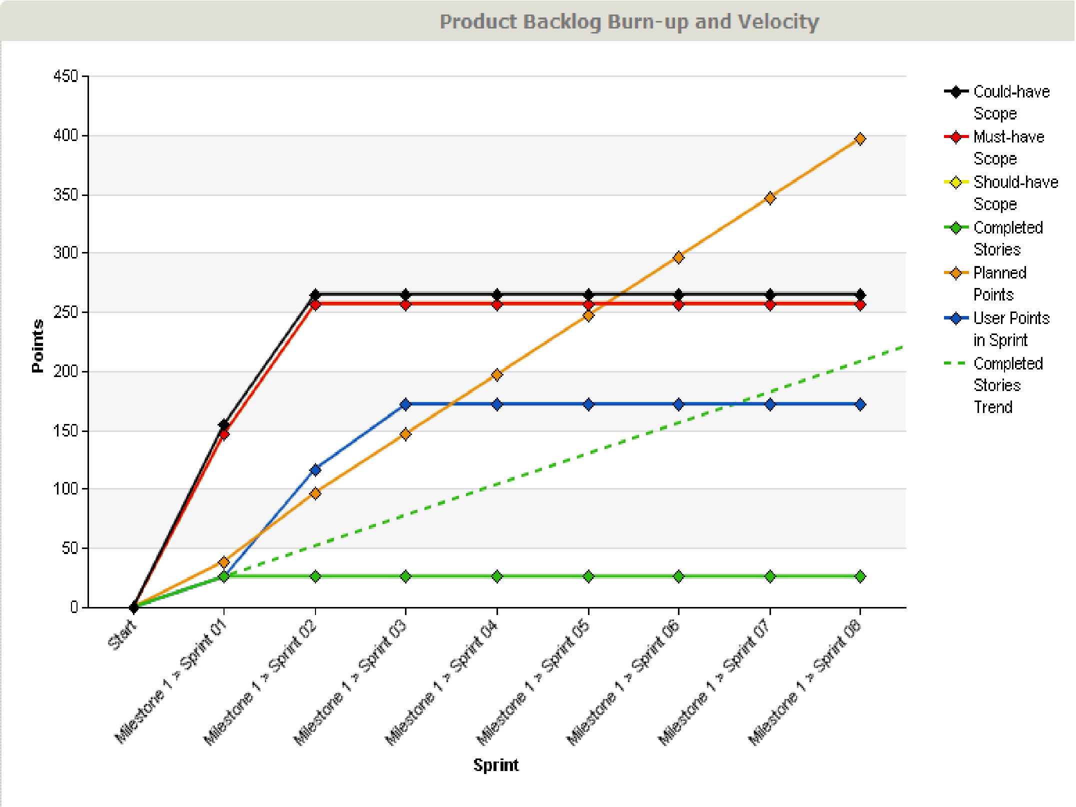
Burndown of user stories (in points)

2. How are we progressing against the planned delivery? (Scrum)



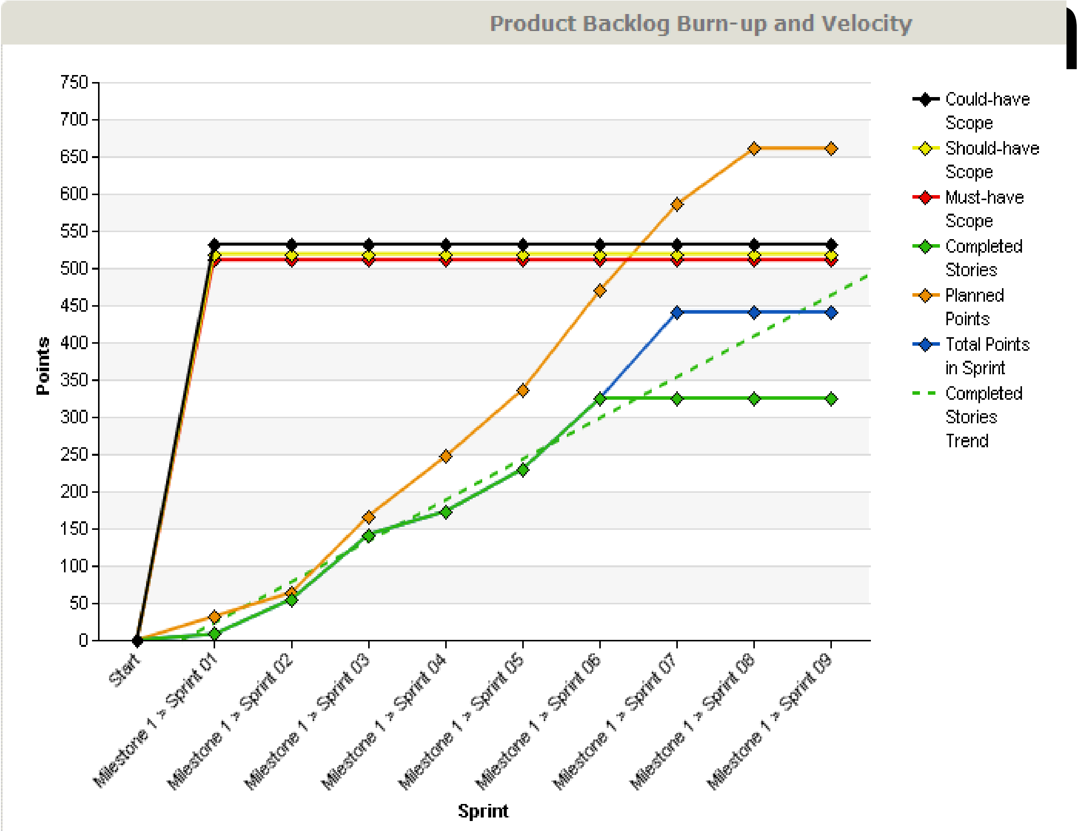
Examples – speed view!

New team (1 sprint complete)



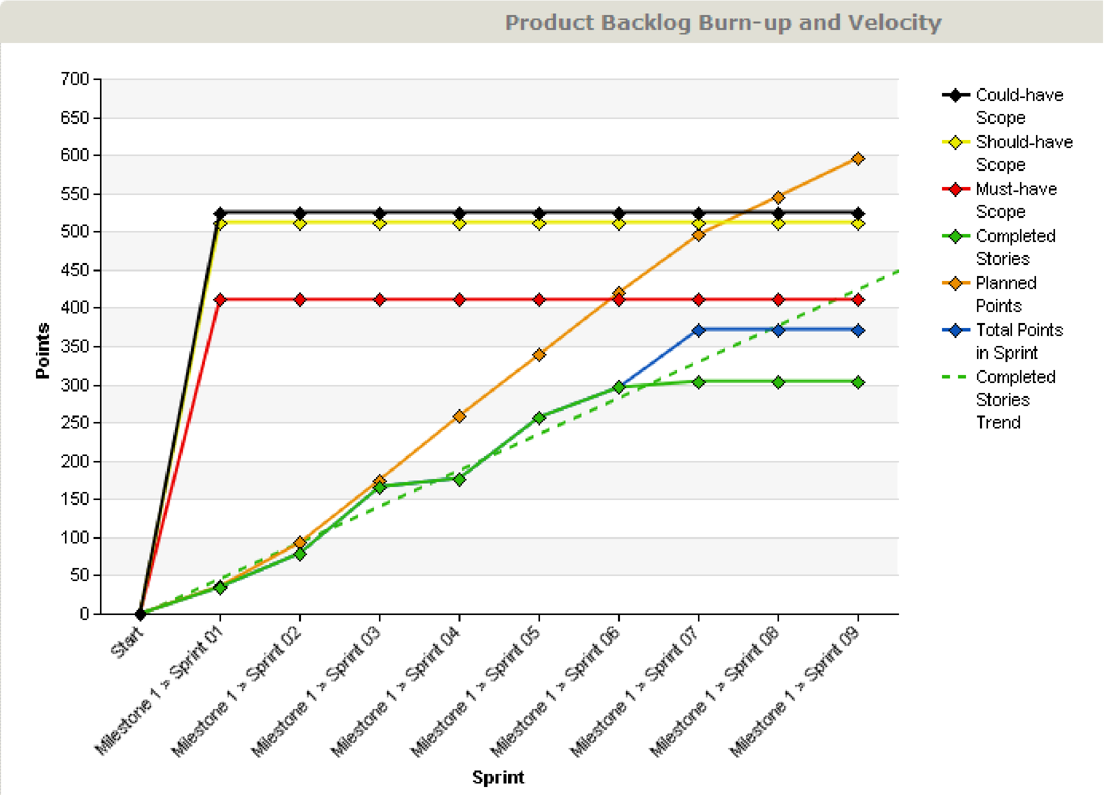
-Disappointing first sprint (green line) but forecast/commitment (orange line) indicates team expects to go faster

No scope slack – lower than forecast velocity necessitates a



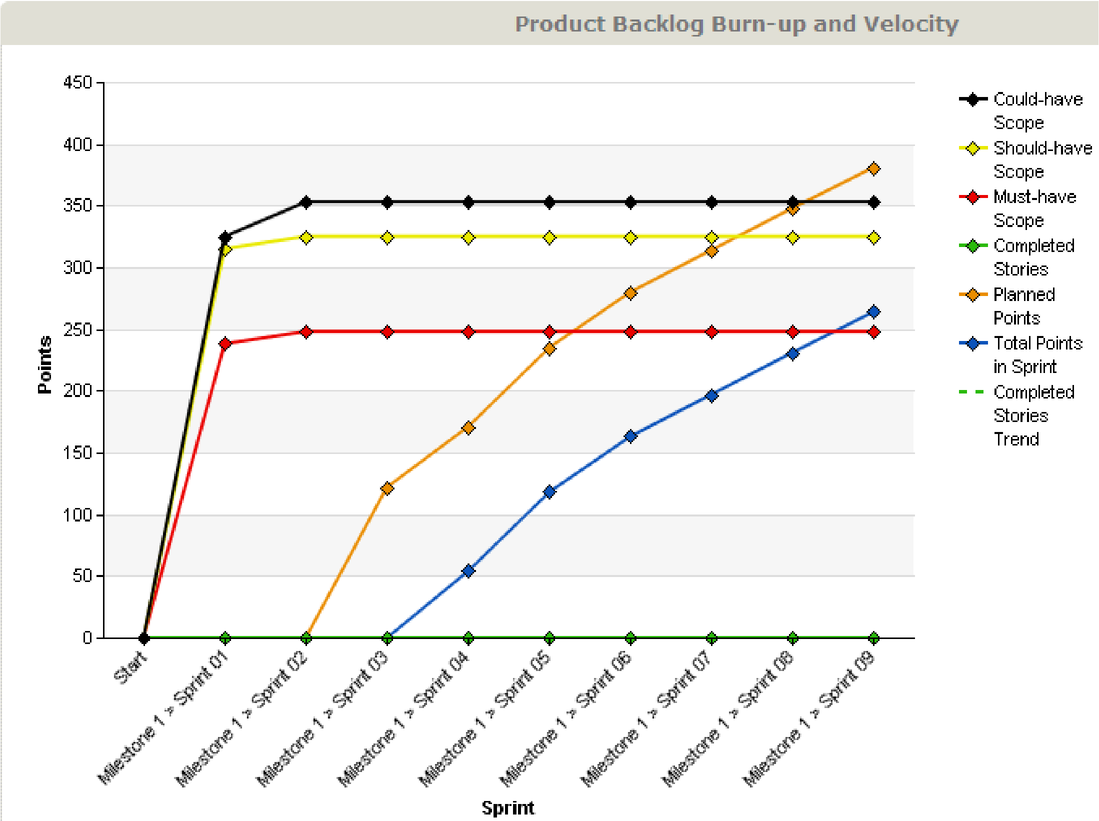
- Must-haves make up majority of scope
- Current velocity indicates that the planned scope will not be achieved
- No potential for de-scoping because of the low number of shoulds and coulds

Lower than forecast velocity but scope still feasible



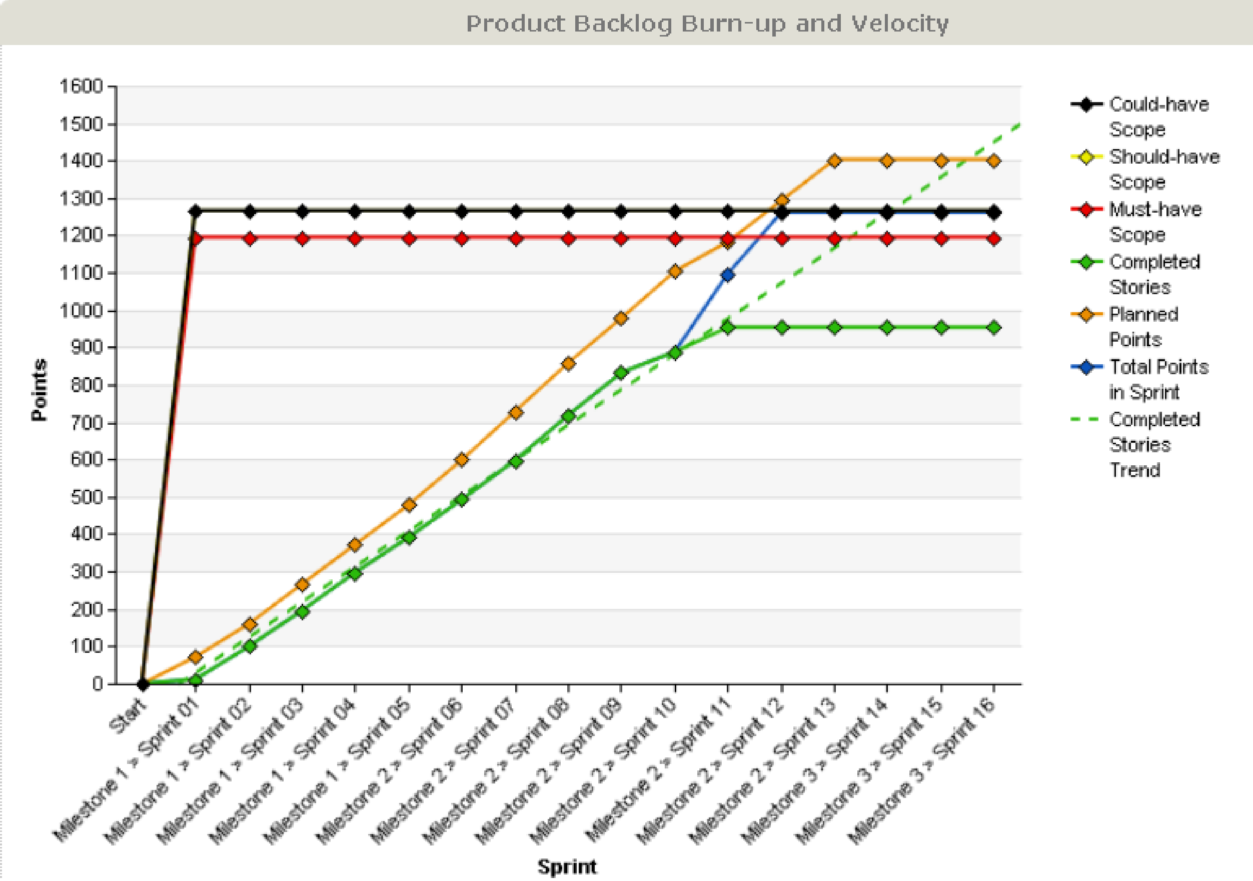
- Lower than forecast velocity shows that a large proportion of shoulds and coulds will not be delivered
- Good proportion of shoulds means re-planning not required

Flat-lining!



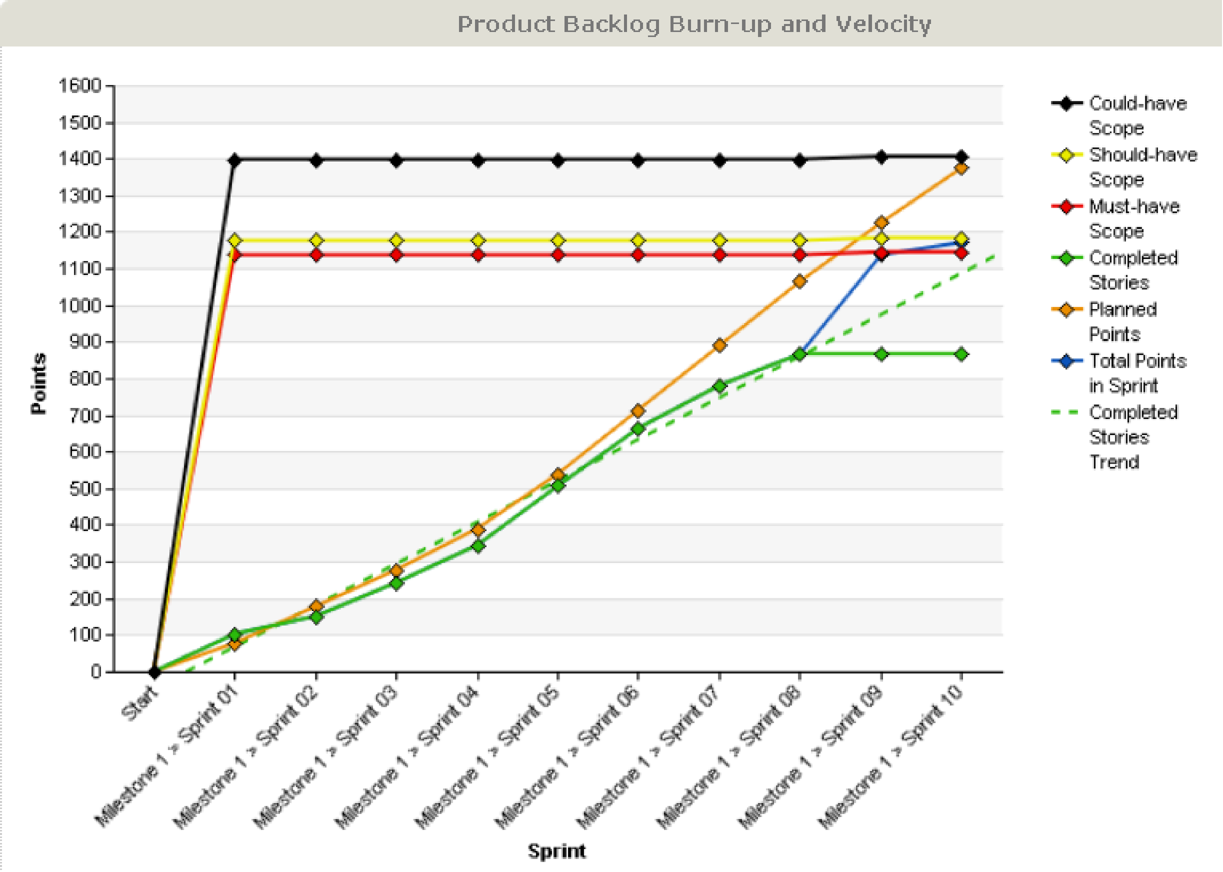
- Note the green line is on the axis!
- Velocity currently zero due to delay in availability of both dev and test environments
- Forecasts not based on actual velocities – nor can they be till environments available
- Environments were expected in Sprint 3 (hence commitment in that sprint) but is still not available in Sprint 4.

Steady velocity; Missed commitments



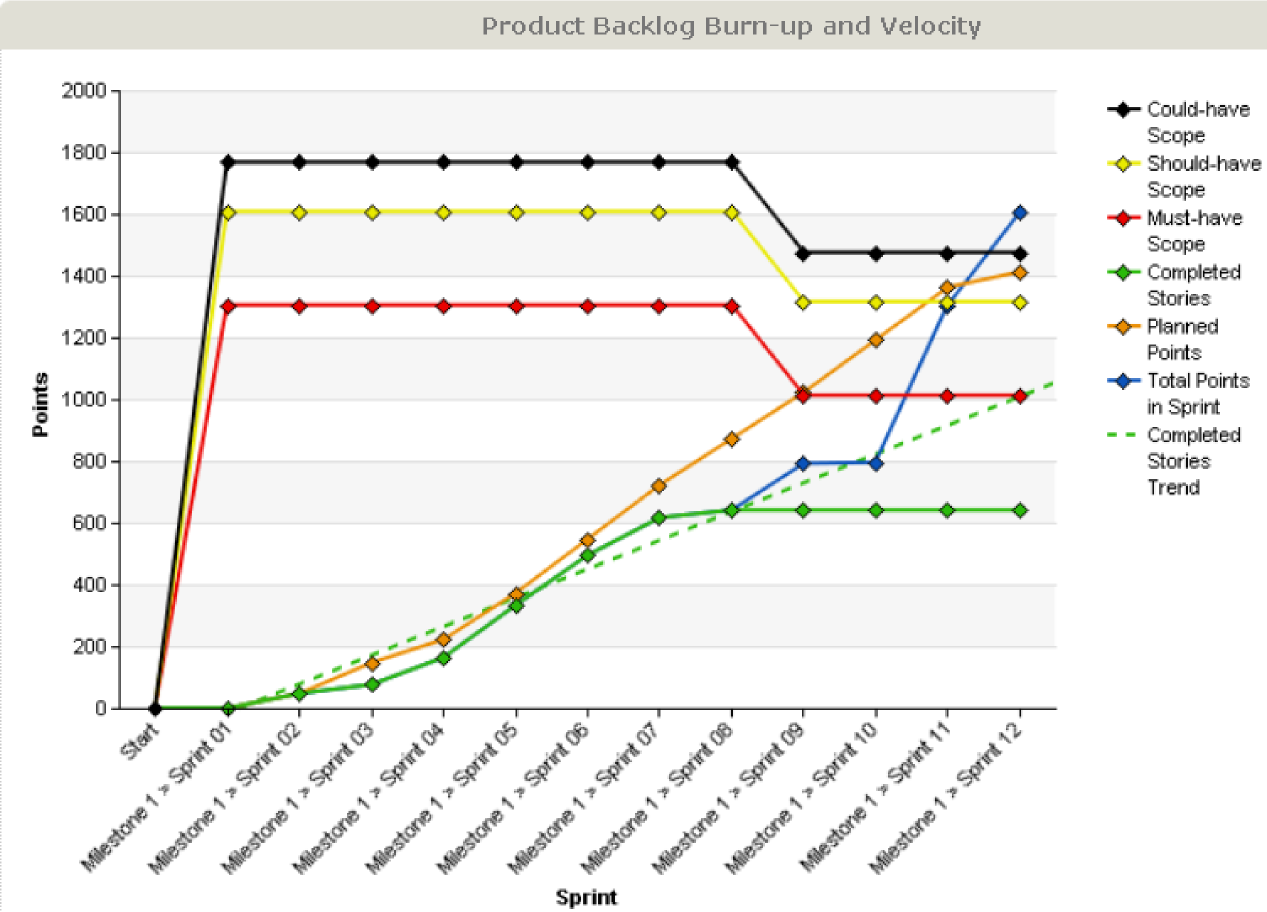
- Good number of sprints completed so forecasting more straightforward
- Cumulative effect of missed commitments means “planned” line is in the wrong position (dashed green line is more useful)

Expected acceleration did not occur



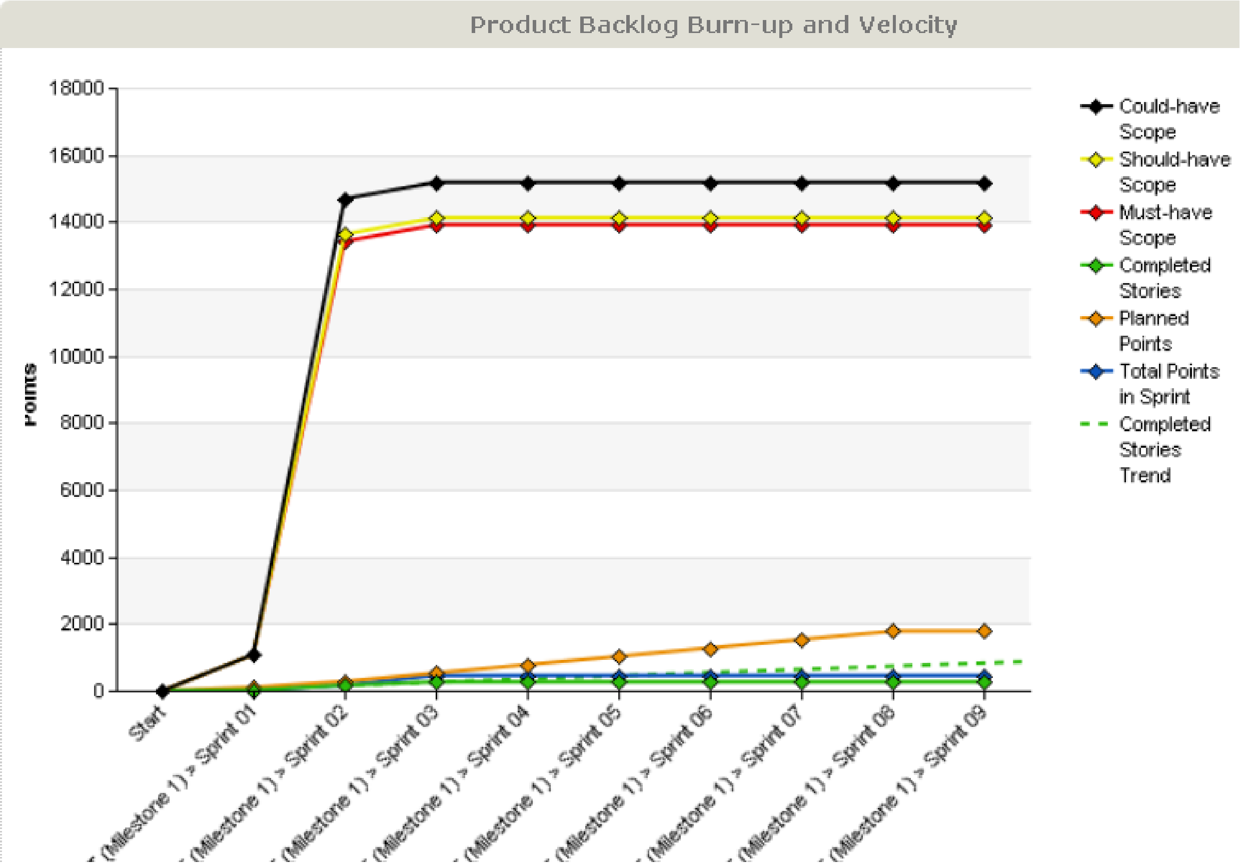
- Team expected to get faster sprint by sprint
- However recent sprints have in fact been slower: double whammie!

Backlog size change



- Scope reduced
- Means new scope is feasible in planned timescales
- Mingle reporting of backlog size changes is problematic though (usually this kind of change is not visible in current charts)

Infeasible Backlog



- Either the backlog size is wrong or this project won’t finish!

Velocity RAG status reporting

(different from typical meaning in Prince II)

- The RAG status indicates forecast of what will be delivered on the budgeted date:
  - Less than the *minimum acceptable scope* (Musts) - **RED**
  - Less than the *expected scope* (Shoulds) - **AMBER**
  - More than the *expected scope* (some Coulds) - **GREEN**
- If insufficient number of “Sho defined, Amber/Green bound outstanding “Musts” plus contingency*
- Highlights where *re-planning* is needed

Under-promise, over-deliver = Green

* typical contingency is 4

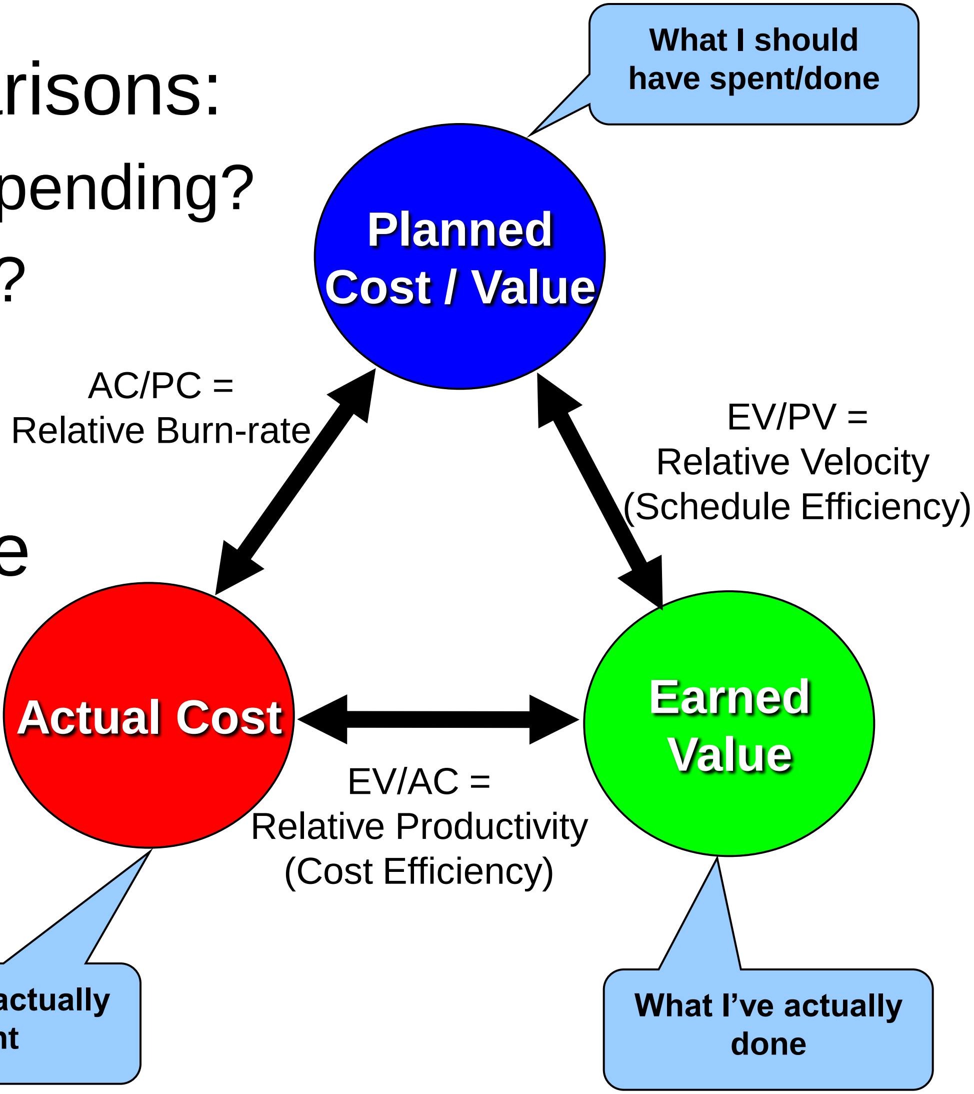
“**Amber** is the new **Green**!”

EVM – how close are you to the plan?

- **Earned Value Management** is a *traditional* approach for reporting progress against plan
  - It designed to answer the questions:
    1. Is the project delivering functionality as **soon** as we expected? *(Schedule Efficiency, SE or SPI)*
    2. Is the project delivering functionality for the **cost** we expected? *(Cost Efficiency, CE or CPI)*
    3. Are we spending **cash** / man-days at the rate we expected? *(Relative Burn-Rate)*
- **Agile projects** are designed to cope with variable scope – this has to been taken into account when considering the applicability of EVM metrics

3D project tracking

- Three essential comparisons:
  - – AC/PC: over or under spending?
  - – EV/PV: faster or slower?
  - – EV/AC: more or less value for money?
- Units must be the same (e.g. £, $, € or Man-days)
- The value of a task (EV) is based on its estimated cost


What I should have spent/done
Planned Cost / Value
AC/PC = Relative Burn-rate
EV/PV = Relative Velocity (Schedule Efficiency)
Actual Cost
Earned Value
EV/AC = Relative Productivity (Cost Efficiency)
What I've actually spent
What I've actually done

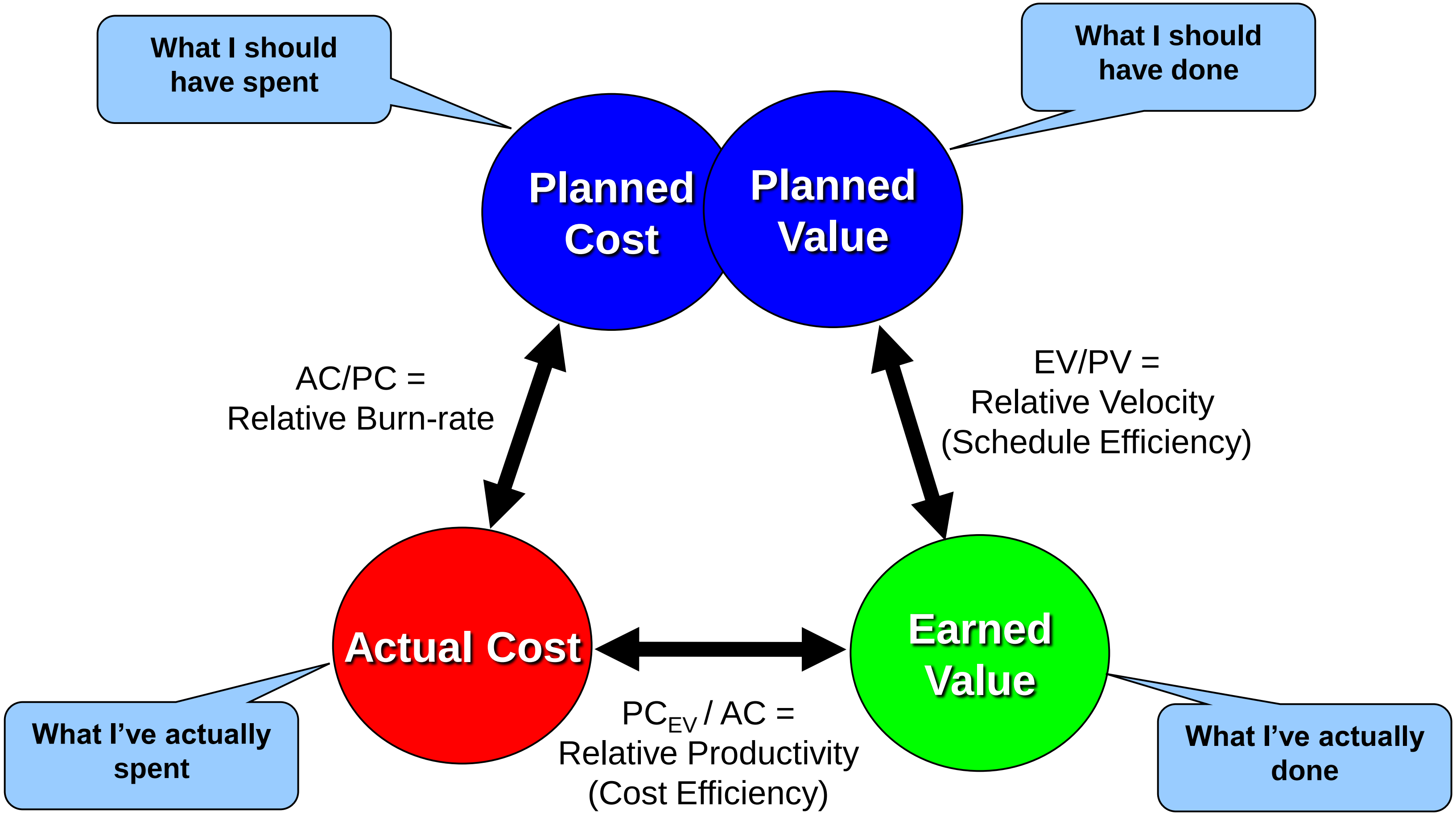
Differences in agile EVM

- All payload tasks (stories) are given an estimated size in points (standard agile practice)
- All overhead tasks (unplannable and administrative tasks) given a size of zero
- Planned Cost for stories is calculated on the basis of their relative size:

$$\textbf{Planned Cost} = \frac{\textbf{(Size of story) * (Planned Cost at completion)}}{\textbf{Total Backlog Size Estimate}}$$

- Forecasts of future EV and AC are based on historical velocity
- Scope changes supported as stories are substitutable because of the size estimate
- EV and Planned Cost may differ

The 4th dimension



What I should have spent

What I should have done

Planned Cost

Planned Value

AC/PC = Relative Burn-rate

EV/PV = Relative Velocity (Schedule Efficiency)

Actual Cost

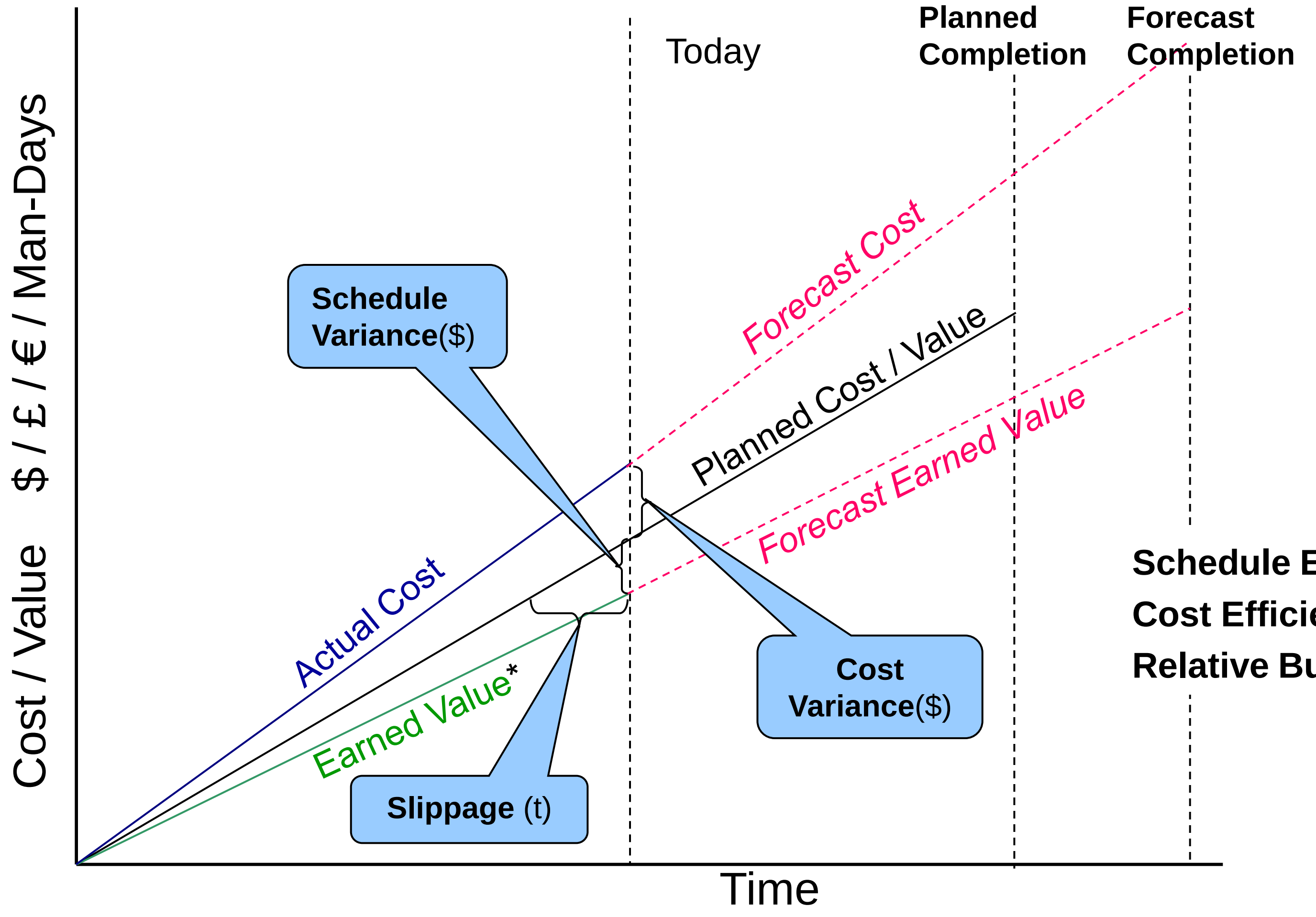
Earned Value

PC_{EV} / AC = Relative Productivity (Cost Efficiency)

What I've actually spent

What I've actually done

Key terms in EVM



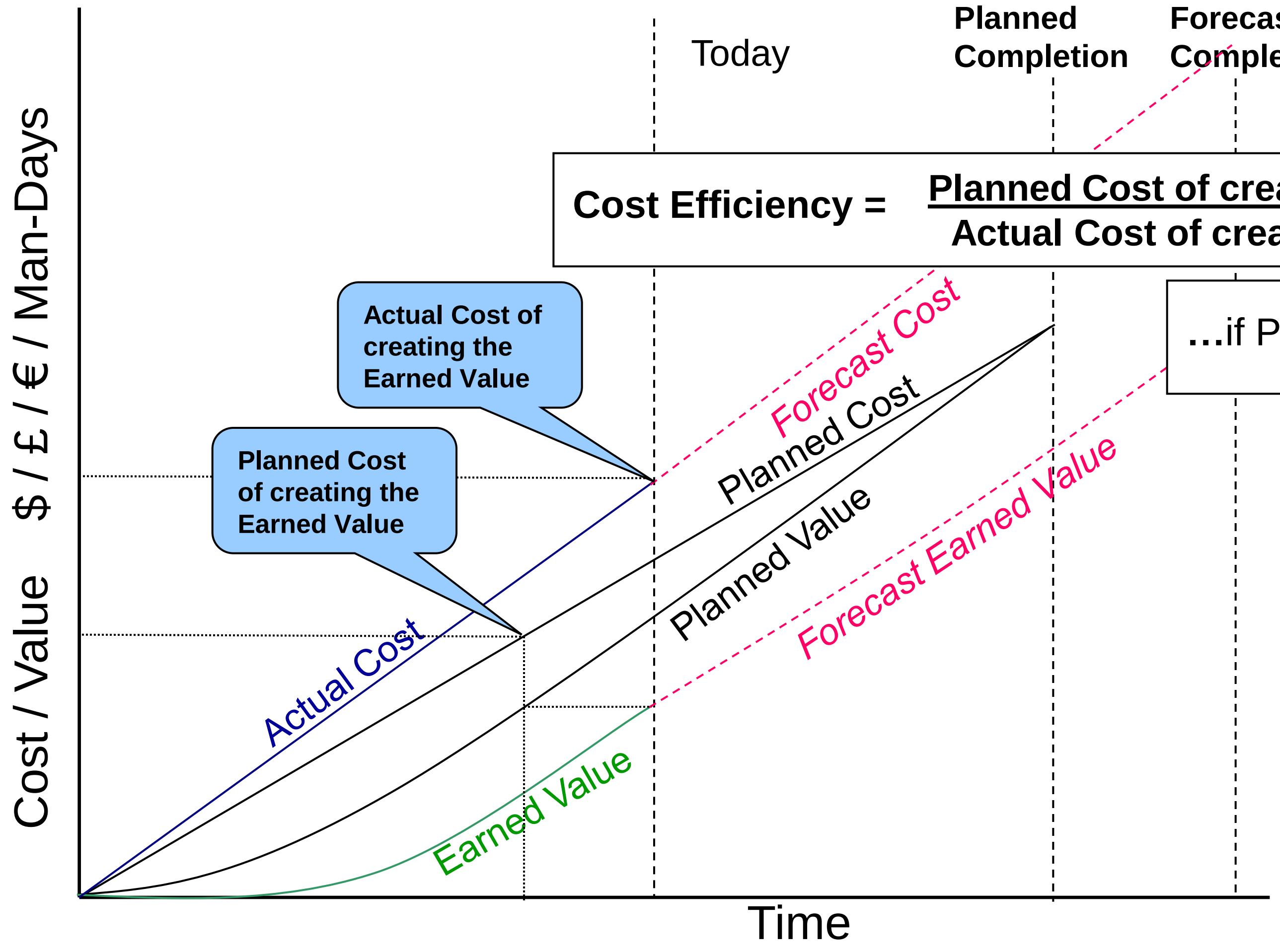
Schedule Efficiency = EV /PV

Cost Efficiency = EV / AC

Relative Burn-rate = AC/PC

**Note:* Earned Value = Budget Cost for Work Performed

Planned Cost and Planned Value may not be the same…

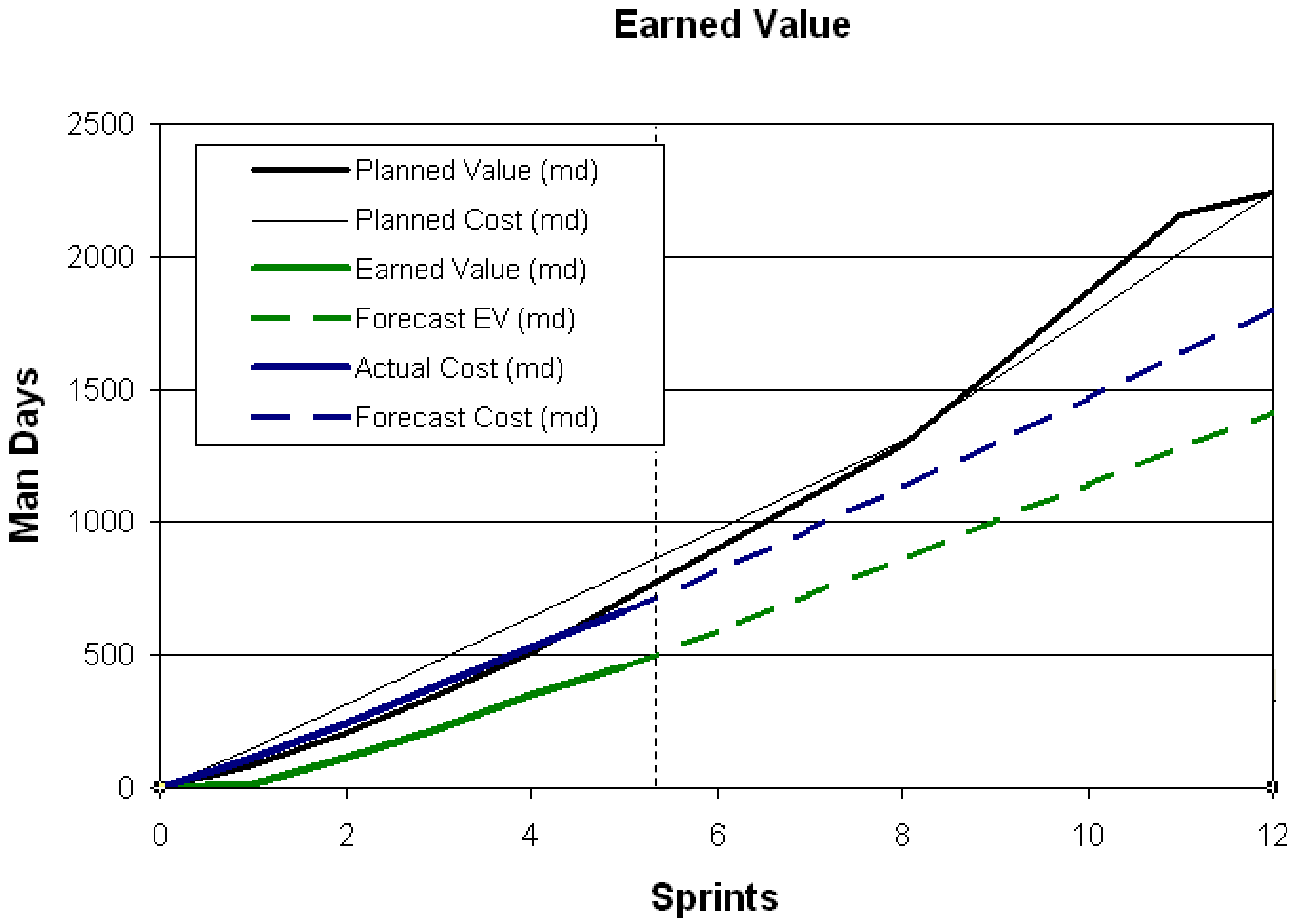


$$\textbf{Cost Efficiency} = \frac{\textbf{Planned Cost of creating the Earned Value}}{\textbf{Actual Cost of creating the Earned Value}}$$

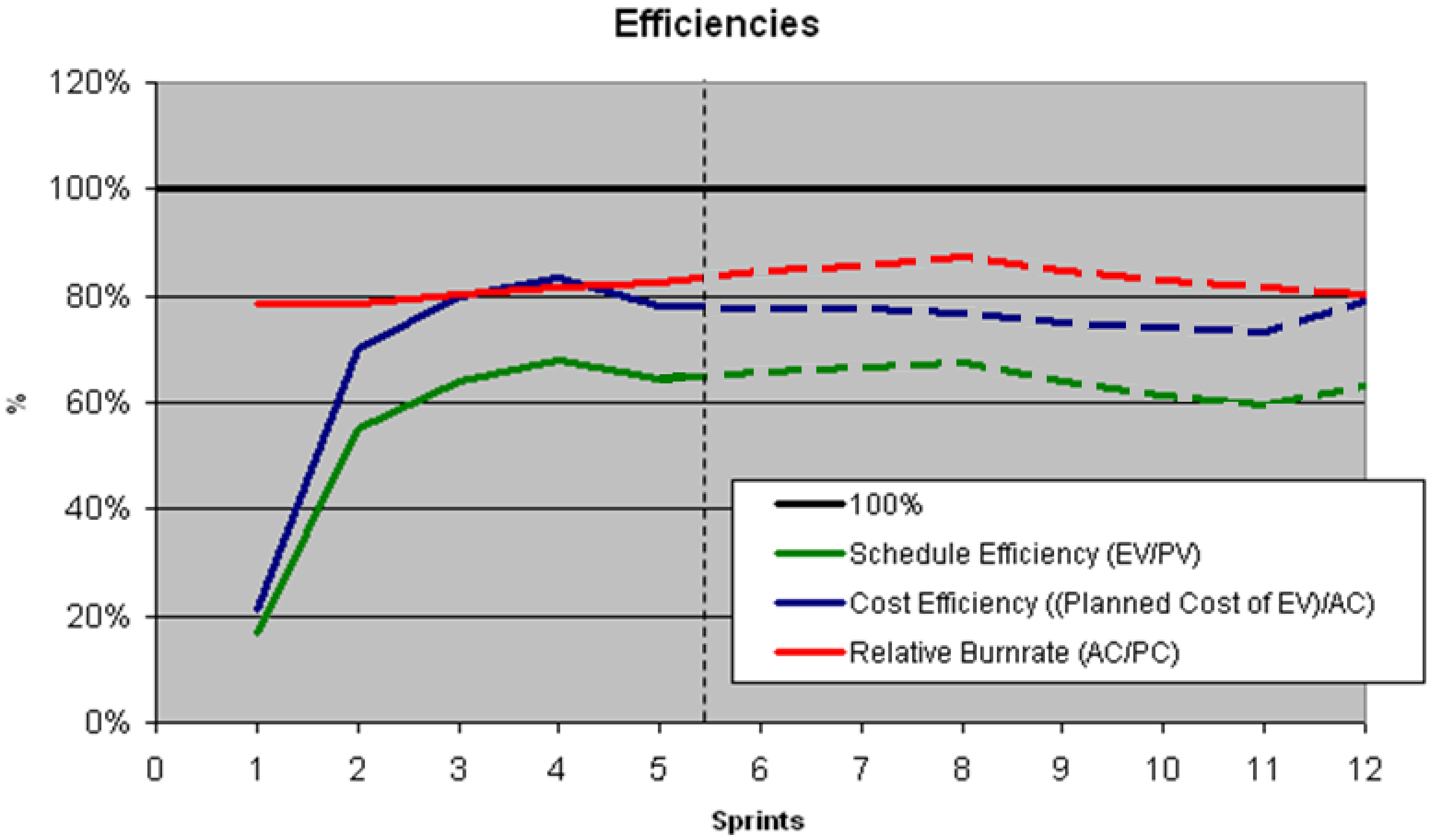
…if PV=PC then $CE = \frac{EV}{AC}$

…because overhead tasks can exist which contribute to *cost* but not to *value*.

Actual Project Data (1 Scrum)



"Efficiencies" (closeness to plan)



Earned Value vs Business Value

- Earned Value is about **Cost** ***not*** **Value**
- Business Value is a measure of the relative importance of **Minimum Marketable Features** (MMFs)
- The Business must rate relative importance at each level of a hierarchical breakdown
- Monitoring delivery of business value means **diminishing returns** can be detected once the most important Epics/MMFs have been delivered

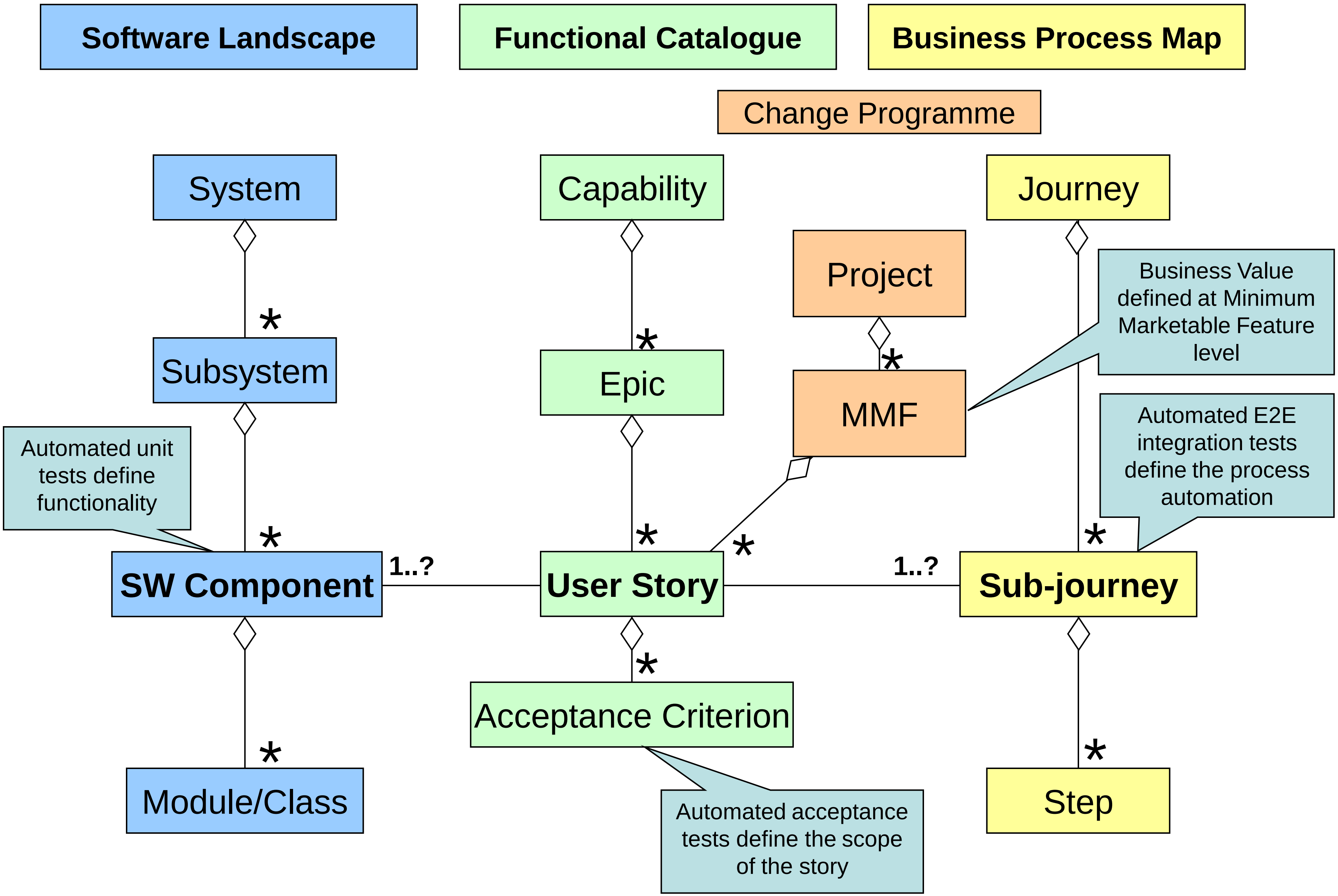
Business Value

“The most important figures… unknown or unknowable…”

- Approach taken
  - *Project* value estimated in Business Case
  - Assignment to hierarchical story breakdown by *relative* values
  - Scheduling of stories based on highest value first
  - Measurement of Business Value of project in “warranty period” post-delivery
- Problems
  - Dependencies: Does the value (or dis-value) depend on just this feature/project?
  - Quantification: How to estimate (then measure) future growth, resilience to competition, retention of staff, reduction of staff?

“…successful management must take account of them.”

Projects change an existing Software / Process Structure


Software Landscape
Functional Catalogue
Business Process Map
Change Programme
System
Capability
Journey
Project
*
*
*
Subsystem
Epic
MMF
Business Value defined at Minimum Marketable Feature level
Automated unit tests define functionality
Automated E2E integration tests define the process automation
*
*
*
*
SW Component
1..?
User Story
1..?
Sub-journey
*
Acceptance Criterion
*
*
Module/Class
Step
Automated acceptance tests define the scope of the story

Measuring *Performance*

- What we'd like to know (but can only infer at best)
  - Business Value for cost
  - Productivity
  - Reliability (of product)
  - Reliability (of estimate)
  - Ability to innovate
  - Ability to improve
  - Ability to forecast
  - Etc…
- But… attempts at measurement can adversely affect outcome. Avoid:
  - non-team based metrics
  - unbalanced measured (e.g. velocity ignoring DoD, quality measures; acceleration ignoring "technical debt"; accuracy of forecast ignoring velocity)
  - using metrics that the teams themselves don't use
  - drawing conclusions from a simple premise (e.g. "high velocity is good"; "high focus factor is good"; "deceleration is bad"; "increasing team size will increase velocity")
  - using data from a tool without validating its accuracy/applicability

Conclusions

- **Measuring progress against plan** is essential in large programmes
  - *"on-time-on-scope-on-budget"* is *not* a Holy Grail
- **Measure** and record history of the (easier) **progress metrics**: Cost, Time, Scope (normalised points), Quality (defect rates, user satisfaction, code metrics)
- **Estimate** and record change of the (harder) **performance metrics**: Business Value, Productivity, Team Performance
- **Invest** in the collection and analysis of standard, simple metrics - why? - to **improve**

Conclusions (cont.)

- Automate *unit* tests and build
- Automate acceptance tests
- Define the *ownership* of code base and design
- Simplify the lines of accountability
  - Project
  - Community of practice
- Reduce the *size* of projects…
- and reduce/eliminate *dependencies* between projects
  - by good design / software engineering practice
  - by “Feature Teams” supported by “Component Teams”