The Agile Critical Path

Ensure that the critical path is visible in an Agile world
A bit about me...Dave Browett

• I’ve been working in IT since 1985 and has held various positions in Project/Development Management since 1988.

• I’ve worked in a range of sectors (including Aerospace, Healthcare, Legal and Telecoms) and has worked with organisations and companies such as the European Union, FSA, BT and BAe.

• I’ve presented at Project Challenge (www.projchallenge.com) at the NEC in Birmingham and London Olympia.

• I’m currently a Project Manager working for Micro Focus in Newbury (www.microfocus.com).
Topics

- Intro – Critical path and Agile => Mindset changes
- Time, Dependencies, Buffer Mgt...
- MMF (Minimum Marketable Features) and MoSCoW
- Flow Mgt/Predictable Velocity (and unpredictable velocity...)
- Risk/Spiking
- Team workload
- Release balancing
- The Agile Critical Path and Portfolio Management
What is the critical path?

- A definition – “Longest sequence of activities in a project plan which must be completed on time for the project to complete on due date”.
- In a *non-agile world* the critical path is key because any delay with tasks on the critical path will result in a delay to the whole project...

<table>
<thead>
<tr>
<th>Task 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 2 (requires Task 1 to be completed)</td>
</tr>
<tr>
<td>Task 3 (requires Task 2 to be completed)</td>
</tr>
</tbody>
</table>
What about the Agile world?

- Purists will say that there isn’t a critical path, because
  - Stories are done in a priority order.
  - Iterations allow the team to take on a theme of related stories

- However, the agile ideal of “feature teams” where anyone can work on any story is often not the case and resource and skill bottlenecks arise as a result...
Mindset changes...

- **Classical project**
  - **Time** is measured in man-days
  - **Dependencies** can significantly influence a critical path by forcing the time and order in which things are done
  - **Buffer Mgt** is an amount of “contingency time” saved away in case of emergency
  - **Flow Mgt** – rarely understood, belief that a large amount of resource or heroic team can work around the clock to save a project
  - **Risks** are handled by having contingency and mitigation plans

- **Agile project**
  - **Time** is measured in iterations
  - **Dependencies** still have an impact, the backlog will rank the stories in priority order but individual skill bottleneck or team allocations can have an effect...
  - **Buffer Mgt** is a number of story points that is calculated based on a team’s velocity
  - **Flow Mgt** – flow is managed by teams achieving a sustained velocity
  - **Risks** are handled by doing Spikes. Mitigation is based on a “fail early” philosophy
Agile Delivery Parameters

- Scope - Story Points
- Time - iterations
- Velocity = Flow
- Errors in estimation
- Risks
- Dependencies
- Buffer
What is time in the Agile world?

- For non-Agile projects we may talk about “man-days”, “man-weeks” etc.
- For Agile/scrum based projects we need to think in terms of “iterations”
- An iteration is the amount of time that a Dev team require to work on a set of stories/features end-to-end – the end result is something that can be \textit{demonstrated} and is potentially \textit{releasable}.
Do dependencies exist in an Agile world?

- Purists would say “no” – a feature team can do anything.
- In reality skill dependencies exist – database, UI, graphic design – many niche areas exist where a particular resource may be required.
- So a story on an iteration or backlog may have skill dependencies (“only Fred can do this”) resulting in personal tasks or personal backlogs.
- If we ignore these then we leave ourselves open to these becoming bottlenecks that affect the team’s ability to deliver iterations.
Dependencies – loading iterations from the backlog

Ideal scenario – team allocate stories in priority order

<table>
<thead>
<tr>
<th>High Priority story</th>
<th>3sp</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Priority story</td>
<td>2sp</td>
</tr>
<tr>
<td>High Priority story</td>
<td>5sp</td>
</tr>
<tr>
<td>High Priority story</td>
<td>3sp</td>
</tr>
<tr>
<td>High Priority story</td>
<td>3sp</td>
</tr>
<tr>
<td>Medium Priority story</td>
<td>3sp</td>
</tr>
<tr>
<td>Medium Priority story</td>
<td>5sp</td>
</tr>
<tr>
<td>Medium Priority story</td>
<td>8sp</td>
</tr>
<tr>
<td>Medium Priority story</td>
<td>5sp</td>
</tr>
<tr>
<td>Medium Priority story</td>
<td>13sp</td>
</tr>
</tbody>
</table>

Team velocity = 15

- Iteration x
- Iteration x+1
- Iteration x+2
### Dependencies – impact of resource/skill bottleneck on how stories are chosen from the backlog

Reality bites – resource/skill bottleneck – “only Fred can do these stories”

<table>
<thead>
<tr>
<th></th>
<th>High Priority story</th>
<th>Medium Priority story</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>F</strong></td>
<td>3sp</td>
<td>13sp</td>
</tr>
<tr>
<td>High Priority</td>
<td>2sp</td>
<td>5sp</td>
</tr>
<tr>
<td>High Priority</td>
<td>5sp</td>
<td></td>
</tr>
<tr>
<td>High Priority</td>
<td>3sp</td>
<td></td>
</tr>
<tr>
<td>High Priority</td>
<td>3sp</td>
<td></td>
</tr>
</tbody>
</table>

- **Team velocity = 15**
  - **Iteration x**
  - **Iteration x+1**
  - **Iteration x+2**
Let’s scale things up...
Let’s have some teams – BLUE, GREEN, RED and YELLOW
Again, in an ideal world we can throw anything at these teams...
But in reality they may (intentionally) have skills that lean them towards being better at some features than others.
Dependencies - assessing backlog of Epics

Epic view – initial assessment

<table>
<thead>
<tr>
<th>Epic Type</th>
<th>Story Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Must Have Epic</td>
<td>20sp</td>
</tr>
<tr>
<td>Must Have Epic</td>
<td>50sp</td>
</tr>
<tr>
<td>Must Have Epic</td>
<td>30sp</td>
</tr>
<tr>
<td>Must Have Epic</td>
<td>50sp</td>
</tr>
<tr>
<td>Must Have Epic</td>
<td>80sp</td>
</tr>
<tr>
<td>Should Have Epic</td>
<td>20sp</td>
</tr>
<tr>
<td>Should Have Epic</td>
<td>50sp</td>
</tr>
</tbody>
</table>

At a high level we assemble the Epics that will be required. The nature of the Epics will often require particular knowledge/skills/resources of a particular team.

If the teams were all capable of the same velocity we can immediately see that the RED team are most heavily tasked with MUST HAVE items...

The GREEN team have the most work but current estimate is they have less MUST HAVE story points to achieve than the RED team.
Critical path analogy

Green Team
- Must Have Epic 50sp
- Must Have Epic 20sp

Red Team
- Must Have Epic 80sp

Blue Team
- Must Have Epic 50sp

Yellow Team
- Must Have Epic 30sp
Critical path analogy

- We’ll continue this example later – let’s first look more at Buffer and Flow mgt.
Buffer Mgt

- In non-Agile projects we’ll often store away some contingency for our projects
- We assess the amount of time required to do Task 1, 2 and 3 and then we add on some contingency as a buffer
Buffer Mgt (cont)

- If we don’t need the contingency then we can deliver sooner.
- If we do need the contingency then provided we don’t exceed it we can deliver as per agreed target.

Diagram:
- Task 1
- Task 2 (requires Task 1 to be completed)
- Task 3 (requires Task 3 to be completed)
- Contingency
- Possible earlier delivery date
- Target delivery date
- Star symbol
Buffer Mgt – what is the Agile equivalent?

• Let’s add an iteration...
• Two problems with this
  – Is one iteration enough?
  – We’ve increased the length of our time-box
Buffer Mgt – what is the Agile equivalent? (cont)

- For timely delivery let’s time-box our release

- Our time-box allows for 3 iterations
- After each iteration we will have something *releasable*
- After 3 iterations we aim to have *sufficient features to take to market - Minimum Marketable Features (MMF)*
MMF

• A definition (AgileBOK - www.agilebok.org)
  – A Minimal Marketable Feature (MMF) is a feature that is minimal, because if it was any smaller, it would not be marketable. A MMF is marketable, because when it is released as part of a product, people would use (or buy) the feature

• Other similar definitions
  – MVF – Minimum Viable Feature
  – MMR – Minimum Marketable Release
MMF and MoSCoW

- Stories are typically modelled in hierarchies
- Large stories are known as “Epics”
- Not all of the Epic may be Must Have for a release

- Epic
  - Must Have Story: Must Have for Release 1 (MMF)
  - Must Have Story: Must Have for Release 1 (MMF)
  - Should Have Story: May be in Release 1 – if not Must Have for Release 2
  - Should Have Story: May be in Release 1 – if not Must Have for Release 2
  - Could Have Story: Might be included in Release 1, might be in Release 2
Flow Mgt = Predictable Velocity

- Teams need to be able to deliver for each iteration a predictable number of story points.
- Obviously this number may vary from iteration to iteration depending on sickness/holiday etc but the key principle is that the team commit to and deliver a number of story points that is related to their performance in previous iterations.
Payload Calculation - predictable delivery

Best/worst case delivery range can be predicted within this zone.
Beware of “Iceberg Agile”!

Team can demo this part

Team are also working on this part BUT can’t demo it and expect it to Carry Over into future iterations
Beware of “Iceberg Agile”!

• A key aspect of Agile is transparency.
• Each story and the tasks within are updated to show a picture that represents the state of the iteration as accurate and "up to the minute" as possible.
• This transparency will build trust at all levels –
  – The Scrum team can show progress both in terms of achieved velocity and demonstrable features
  – Stakeholders/managers can take confidence from a regular cadence that provides demonstrable features
• But if your team is doing "Iceberg Agile" this breaks down - not all the planned stories will be completed and the reported velocity will be lower than expected or swing from low to high…
Beware of “Iceberg Agile”!

Agile teams should be able to demonstrate historical velocity over several iterations that is within a predictable range.

Whereas for Iceberg Agile teams, velocity will see-saw over a wider range and be less predictable.
Beware of “Iceberg Agile”!

- Teams that do "Iceberg Agile" will typically
  - Carry over several stories as common practice
  - Be unable to demonstrate all planned stories
  - Have a velocity that see-saws as the credit for carried over stories gets re-allocated one or two iterations down the line
- These teams will suffer from lack of transparency and consequently it is difficult to predict what they are capable of consistently achieving. Be aware and try to avoid your team doing "Iceberg Agile"!
Velocity calculations across multiple teams – Agile Purist Warning!

• Strictly you can’t simply “add-up” story points across teams (because each team is likely to have different measures)
• Then again – surely it doesn’t make sense to have wildly different sp measures across teams… (find an agile purist near you and discuss!)
• So – perhaps the pragmatic solution is to ensure that sp measures across teams are of the same order
• Assuming the principle above is held then these payload calculations can be used for an entire project across several teams – as a “high level indication”.
Bear in mind also…

- Estimation is always an inexact science!
- Beware of false precision, “37.5sp remaining”
The story (or epic!) so far...

- In Agile we measure time in iterations
- Skill/resource bottlenecks may still exist and may impact the critical path so need to be managed carefully
- At a higher level, team loading can also affect the critical path
- If we can estimate the MMF and roughly calculate velocities then we can attempt to predict best/worst case delivery ranges.
- So, our buffer can be described in terms of the variance of the number of iterations to achieve the MMF payload (+/- delta) and variance in velocity (V1 and V2)
- Flow is velocity – if V1 and V2 are too far apart then the ability to predict best/worse case delivery ranges becomes impaired.
Any Questions?
...Back to our example

• **Initial assessment** (assuming the teams all have similar velocities)
  
  => **RED team** are on the critical path

• This can change (of course!) whenever we
  – Evaluate Risks
  – Decompose Epics
  – Alter workload/resource

• Let’s examine what happens when we add another dimension – Risk
Iteration planning - assessing the backlog

Epic view – take risk into account

<table>
<thead>
<tr>
<th>Risk Level</th>
<th>Epic Type</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Risk</td>
<td>Must Have Epic</td>
<td>20sp</td>
</tr>
<tr>
<td></td>
<td>Must Have Epic</td>
<td>50sp</td>
</tr>
<tr>
<td></td>
<td>Must Have Epic</td>
<td>30sp</td>
</tr>
<tr>
<td></td>
<td>Must Have Epic</td>
<td>50sp</td>
</tr>
<tr>
<td></td>
<td>Should Have Epic</td>
<td>20sp</td>
</tr>
<tr>
<td></td>
<td>Should Have Epic</td>
<td>50sp</td>
</tr>
</tbody>
</table>

If we add another dimension – risk then we can re-assess the critical path...

Both RED and GREEN teams have High Risk Epics...
Evaluate Risks

• Altering the expected impact/severity of a risk can affect your perception as to whether an Epic is on the critical path or not

• Let’s suppose that in our example we evaluate the risks and find that
  – The Green Team evaluate their High Risk 50sp Must Have Epic and decide that they need to do an additional 8sp spike.
  – The Red Team evaluate their High Risk 80sp Must Have Epic and decide that their approach covers them sufficiently

• We now have the Green and Red teams tying for who is on the critical path (see next slide)
Critical path – added spike to Green Team

Green Team

Red Team

Blue Team

Yellow Team
Analysis

- The Must Have workload of Red and Green teams are now **too close to call**

- The Red Team have 80sp and the Green Team $50+20+8 = 78sp$ – but we have to be careful as these are estimates and we can’t be that precise, a difference of 2sp out of 80sp is too small to be able to declare.
Decompose Epics

- Decomposing an Epic should result in getting a clearer understanding of
  - The scope
  - The importance/priority of features within the Epic
- Let’s look at how this can affect our view on the critical path...
Decompose Epics – Green Team

- Stories are typically modelled in hierarchies
- Large stories are known as “Epics”
- But – not all of the Epic may be Must Have for a release

---

Must Have Epic 50sp

- Must Have Spike 8sp
- Must Have Story 8sp
- Must Have Story 13sp
- Must Have Story 13sp
- Should Have Story 5sp
- Should Have Story 5sp
- Should Have Story 5sp
- Could Have Story 8sp

Revised Must Have total 8+8+13+13 =42sp
(let’s avoid false precision and say 40sp)

Should Have total 5+5 =10sp

Blue team could do this
Decompose Epics – Red Team

- Stories are typically modelled in hierarchies
- Large stories are known as “Epics”
- But – not all of the Epic may be Must Have for a release

Must Have Epic 80sp

- Must Have Story 8sp
- Must Have Story 13sp
- Must Have Story 13sp
- Must Have Story 21sp
- Must Have Story 21sp
- Should Have Story 2sp
- Should Have Story 3sp

Revised Must Have total
8+13+13+21+21 = 76sp
(let’s avoid false precision and keep with 80sp)

Should Have total 2+3 = 5sp
Critical path – added spike to Green Team

Green Team: Spike 8sp, Must Have Epic 40sp, Must Have Epic 20sp

Red Team: Must Have Epic 80sp

Blue Team: Must Have Epic 50sp

Yellow Team: Must Have Epic 30sp
Analysis

• The estimated Must Have payload for the Green team has reduced whereas that for the Red team has stayed stubbornly around 80sp.

• The Red team are back on the critical path and none of their stories can be allocated to other teams (whereas the Green team have a story that they can hand-over to the Blue team).

• Further sessions will be required to decompose the 21sp, 13sp stories otherwise they will be too large to be dealt with within a single iteration.
Velocity

• We have a “high level indication” of the critical path in story points.
• To make this an indicator in terms of time we need to factor in the teams velocities.
• We may already have historical evidence and be confident that the team composition and undertaking of work is sufficiently similar to be able to use this.
Calculate number of iterations - example

- Let’s say that both the Green and Red teams have 2 week iterations
- Let’s say that the Green team’s velocity is 20 and the Red team has a velocity of 15

- Green team’s Must Have payload is 60sp => 3 iterations
- Red team’s Must Have payload is 80sp => 6 iterations *
- So it looks like the Red team are on the critical path...

* we can’t have “part-iterations, 5 iterations = 75sp and 6 iterations is 90sp so to do 80sp will require 6 iterations
Our view of the Agile Critical Path
(Green and Red teams)

Green Team
Iteration 1 20sp  Iteration 2 20sp  Iteration 3 20sp
Need to do 60sp and have a $V_{\text{new features}}$ velocity of 20

Red Team
Iteration 1 15sp  Iteration 2 15sp  Iteration 3 15sp  Iteration 4 15sp  Iteration 5 15sp  Iteration 6 15sp
Need to do 80sp and have a $V_{\text{new features}}$ velocity of 15
Team workload categories

⇒ To maximise velocity on new features we need to reduce velocity on other areas
Warning - Alter Workload/resource

• Adding resource to a team is likely to *reduce* its velocity in the short-term and bringing in a new team is also likely to require ramp-up/familiarisation.

• But there may be cases where this is desirable and achievable – remember
  – Allow sufficient time for ramp-up/familiarisation
  – Have sufficient team-team communication (scrum of scrums) where multiple teams working in a common area.
Calculate number of iterations - revisited

• If you recall we said that the Green team’s velocity is 20 and the Red team has a velocity of 15
• Let’s say that this was $V_{\text{new features}}$ and the total velocities are higher than this...let’s look more closely at the Red team
• Red team’s maintenance overhead is 25% => their total velocity is 20
• So, if we can agree to suspend maintenance for 4 iterations we can look to get the Red team Must Have payload done in 4 iterations ($20 \times 4 = 80\text{sp}$)
• So it still looks like the Red team are on the critical path but we’ve shortened the project by 2 iterations 😊
Our updated view of the Agile Critical Path

<table>
<thead>
<tr>
<th>Team</th>
<th>Iteration 1</th>
<th>Iteration 2</th>
<th>Iteration 3</th>
<th>Iteration 4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Green Team</strong></td>
<td>20sp</td>
<td>20sp</td>
<td>20sp</td>
<td></td>
</tr>
<tr>
<td>Need to do 60sp and have a $V_{\text{new features}}$ velocity of 20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Red Team</strong></td>
<td>20sp</td>
<td>20sp</td>
<td>20sp</td>
<td>20sp</td>
</tr>
<tr>
<td>Need to do 80sp and have a modified $V_{\text{new features}}$ velocity of 20 (agreed to suspend maintenance)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Blue Team</strong></td>
<td>16sp</td>
<td>16sp</td>
<td>16sp</td>
<td>16sp</td>
</tr>
<tr>
<td>Need to do 50sp and have a velocity of 16 (need to do 4 iterations or increase $V_{\text{new features}}$ somehow)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Yellow Team</strong></td>
<td>12sp</td>
<td>12sp</td>
<td>12sp</td>
<td></td>
</tr>
<tr>
<td>Need to do 30sp and have a $V_{\text{new features}}$ velocity of 12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Analysis

- Our high level view shows us that
  - The Red Team and Blue team are expected to take 4 iterations (the High Risk and 100% loading implies that the Red team are the main critical path concern).
  - Temporarily relaxing the maintenance overhead could allow the Blue team to finish in 3 iterations
  - Yellow and Green teams should be able to finish in 3 iterations without having to adjust their maintenance overhead
The assumptions that we have made:

- We’re doing these calculations as a “high level indication”
  - We’ll do them early and often to help spot any problems as early as possible
- Each team has a predictable (relatively consistent) velocity
- Each team is using story point estimates that are of the same order
- Teams can provide historical data as to their maintenance overhead
- The MMF for a product release can be determined
  - MMF Payload in story points
  - Plus/minus agreed (realistic) delta
How these assumptions can break down..

- We’re doing these calculations as a “high level indication”
- Estimates get nailed to a plan and are challenged the moment they increase...
- Each team has a predictable (relatively consistent) velocity
- Teams are unable to consistently commit to iterations
- Each team is using story point estimates that are of the same order
- Teams are unable to agree to story point sizes
- Teams can provide historical data as to their maintenance overhead
- The MMF for a product release can be determined
  - MMF Payload in story points
  - Plus/minus agreed (realistic) delta
- Product Owner cannot identify MMF (everything is MUST HAVE)
Let’s attempt to simplify things...

- We need some measure of the MMF
- We need some indication of the teams capability
  - Assuming they continue to work on maintenance
  - Assuming they drop everything else
- To scale this up we need teams capability (measured in story points) to be something that we can add up to provide a total project capability
- If we can do these (and have teams that commit and deliver consistently each iteration) then we can adopt the following approach to “release balancing”...
Managing Payload – a balanced release

- Maximum Story Points achievable based on average total velocity
- Maximum Story Points achievable taking into account maintenance

- Payload is in the AT RISK region
- Payload is in the CAUTION region
- Payload is not threatened
Example 1 - Well balanced Release

Maximum Story Points achievable based on average total velocity

Maximum Story Points achievable taking into account maintenance

Key
- **Estimated MUST HAVE payload**
- **Estimated Non MUST HAVE payload**
- **AT RISK payload**
Example 2 – Under committed Release (slack)

Maximum Story Points achievable based on average total velocity

Maximum Story Points achievable taking into account maintenance

Key

- **Estimated MUST HAVE payload**
- **Estimated Non MUST HAVE payload**
- **AT RISK payload**
Example 3 – Over committed Release

Maximum Story Points achievable based on average total velocity

Maximum Story Points achievable taking into account maintenance

Key
- Estimated MUST HAVE payload
- Estimated Non MUST HAVE payload
- AT RISK payload
Example 4 – Release with non MUST HAVE at Risk with additional Caution indicator

Key

- Estimated MUST HAVE payload
- Non MUST HAVE payload
- CAUTION payload
- AT RISK payload

Maximum Story Points achievable based on average total velocity

- Maximum Story Points achievable taking into account maintenance
Release Balancing

• The best way to use this is “early and often” – do it as soon as you have the key data and then keep checking that your profile doesn’t change from being “well balanced” to “over committed”

• This approach is taking into account
  • Time, Buffer Mgt and Flow Mgt

• This approach is not helping with risk or dependency mgt but adopting an early and often approach may be a good way to assess impact if the MUST HAVE payload increases owing to a risk coming home to roost...
Any Questions?
The Agile Critical Path and Portfolio Management
The Agile Critical Path and Portfolio Management

- **TIME** - We need to think in terms of iterations – report against days/weeks but make iteration start/end dates clearly visible
- **DEPENDENCIES** – these need to be clearly visible and ability to perform “what if” analysis
- **BUFFER MGT** – We need to make the MUST HAVE payload clearly visible and so use the NON MUST HAVE payload as a buffer
- **FLOW MGT** – We need to have some indication as to the rate at which the payload is being completed – we could then have what-if scenarios and buffer iteration markers to provide powerful insight
- **RISKS** – associated risks need to be made clearly visible, again “what if” scenarios would be a powerful feature
More concisely...

- Use historical data to predict the future by making the MUST HAVE payload and the teams delivery capability clearly visible (ideally predicting the effect of future iterations)
- Indicate clearly any dependencies within and across teams.
- Indicate clearly any risks and their likely impact.
Talking points...

• Should we make all teams within a project/portfolio align to the same iterations?
• Should we expect all teams to estimate to the same order of story points?
• Should we attempt to create a “super backlog” or should we accept that we may have several backlogs that make up the total picture?
Where is the source of truth?

- More often than not teams will use more than one application
  - One application to manage releases/stories
  - Another application to manage defects
- Effectively a team has multiple backlogs
  - Project/release backlogs
  - Defect backlog(s)
- A portfolio management tool should be able to access all of these backlogs so as to be able to provide a complete view.
Logical backlogs...

- A product backlog may often be comprised of several smaller backlogs...

Blue Team backlog

Green Team backlog

Yellow Team backlog

Product Payload

Bug tracker list
Provenance, priority and associated information...

Provenance: Green Team Backlog Item 3
Priority: MUST HAVE (item 1)

Provenance: Blue Team Backlog Item 1
Priority: MUST HAVE (item 2)
Assoc info: Blue Team assess as HIGH RISK and have dependency with YELLOW TEAM (item 8)
Utopia...

• An Agile portfolio management tool should make “the critical path visible in an Agile world”
• It should measure time in iterations and take this into account when scheduling and calculating impacts arising from dependencies
• It should be able to compile a payload by being able to link multiple logical backlogs whilst understanding their relative priorities and provenance
• It should also understand the relative throughput for each of these workstreams, be able to make predictions based on historical data and perform “what-if” scenarios based on available options to manage the workload
• It should also be able to take relative risk impacts into account and provide “what if” scenarios based on the available mitigation options
Thank you – any questions?

My blog on WordPress -
http://davebrowettagile.wordpress.com/

David.Browett@microfocus.com